AIMS Lecture Notes 2006

Peter J. Olver

OBSAH

AIMS Lecture Notes 2006

Peter J. Olver

1. Computer Arithmetic

The purpose of computing is insight, not numbers.
— R.W. Hamming, [**23**]

The main goal of *numerical analysis* is to develop efficient algorithms for computing precise numerical values of mathematical quantities, including functions, integrals, solutions of algebraic equations, solutions of differential equations (both ordinary and partial), solutions of minimization problems, and so on. The objects of interest typically (but not exclusively) arise in applications, which seek not only their qualitative properties, but also quantitative numerical data. The goal of this course of lectures is to introduce some of the most important and basic numerical algorithms that are used in practical computations. Beyond merely learning the basic techniques, it is crucial that an informed practitioner develop a thorough understanding of how the algorithms are constructed, why they work, and what their limitations are.

In any applied numerical computation, there are four key sources of error:

(*i*) Inexactness of the mathematical model for the underlying physical phenomenon.
(*ii*) Errors in measurements of parameters entering the model.
(*iii*) Round-off errors in computer arithmetic.
(*iv*) Approximations used to solve the full mathematical system.

Of these, (*i*) is the domain of mathematical modeling, and will not concern us here. Neither will (*ii*), which is the domain of the experimentalists. (*iii*) arises due to the finite numerical precision imposed by the computer. (*iv*) is the true domain of *numerical analysis*, and refers to the fact that most systems of equations are too complicated to solve explicitly, or, even in cases when an analytic solution formula is known, directly obtaining the precise numerical values may be difficult.

There are two principal ways of quantifying computational errors.

Definition 1.1. Let x be a real number and let $x^\star$ be an approximation. The *absolute error* in the approximation $x^\star \approx x$ is defined as $|\, x^\star - x\,|$. The *relative error* is defined as the ratio of the absolute error to the size of x, i.e., $\dfrac{|\, x^\star - x\,|}{|\, x\,|}$, which assumes $x \neq 0$; otherwise relative error is not defined.

For example, 1000001 is an approximation to 1000000 with an absolute error of 1 and a relative error of 10^{-6}, while 2 is an approximation to 1 with an absolute error of 1 and a relative error of 1. Typically, relative error is more intuitive and the preferred determiner of the size of the error. The present convention is that errors are always ≥ 0, and are $= 0$ if and only if the approximation is exact. We will say that an approximation $x^\star$ has k *significant decimal digits* if its relative error is $< 5 \times 10^{-k-1}$. This means that the first k digits of $x^\star$ following its first nonzero digit are the same as those of x.

Ultimately, numerical algorithms must be performed on a computer. In the old days, "computer" referred to a person or an analog machine, but nowadays inevitably means a digital, electronic machine. A most unfortunate fact of life is that all digital computers, no matter how "super", can only store finitely many quantities. Thus, there is no way that a computer can represent the (discrete) infinity of all integers or all rational numbers, let alone the (continuous) infinity of all real or all complex numbers. So the decision of how to approximate more general numbers using only the finitely many that the computer can handle becomes of critical importance.

Each number in a computer is assigned a location or *word*, consisting of a specified number of binary digits or *bits*. A k bit word can store a total of $N = 2^k$ different numbers. For example, the standard single precision computer uses 32 bit arithmetic, for a total of $N = 2^{32} \approx 4.3 \times 10^9$ different numbers, while double precision uses 64 bits, with $N = 2^{64} \approx 1.84 \times 10^{19}$ different numbers. The question is how to distribute the N exactly representable numbers over the real line for maximum efficiency and accuracy in practical computations.

One evident choice is to distribute them evenly, leading to *fixed point arithmetic*. For example, one can use the first bit in a word to represent a sign, and treat the remaining bits as integers, thereby representing the integers from $1 - \frac{1}{2}N = 1 - 2^{k-1}$ to $\frac{1}{2}N = 2^{k-1}$ exactly. Of course, this does a poor job approximating most non-integral numbers. Another option is to space the numbers closely together, say with uniform gap of 2^{-n}, and so distribute our N numbers uniformly over the interval $-2^{-n-1}N < x \leq 2^{-n-1}N$. Real numbers lying between the gaps are represented by either *rounding*, meaning the closest exact representative, or *chopping*, meaning the exact representative immediately below (or above if negative) the number. Numbers lying beyond the range must be represented by the largest (or largest negative) representable number, which thus becomes a symbol for overflow. When processing speed is a significant bottleneck, the use of such fixed point representations is an attractive and faster alternative to the more cumbersome floating point arithmetic most commonly used in practice.

The most common non-uniform distribution of our N numbers is the *floating point* system, which is based on scientific notation. If x is any real number it can be written in the form

$$x = \pm\, .d_1 d_2 d_3 d_4 \ \ldots \ \times 2^n,$$

where $d_\nu = 0$ or 1, and $n \in \mathbb{Z}$ is an integer. We call $d_1 d_2 d_3 d_4 \ \ldots$ the *mantissa* and n the exponent. If $x \neq 0$, then we can uniquely choose n so that $d_1 = 1$. In our computer, we approximate x by a finite number of mantissa digits

$$x^\star = \pm\, .d_1 d_2 d_3 d_4 \ \ldots \ d_{k-1} \widehat{d}_k \times 2^n,$$

by either chopping or rounding the final digit. The exponent is also restricted to a finite range of integers $n_\star \leq N \leq N^\star$. Very small numbers, lying in the gap between $0 < |\,x\,| < 2^{n_\star}$, are said to cause *underflow*.

- In *single precision* floating point arithmetic, the sign is 1 bit, the exponent is 7 bits, and the mantissa is 24 bits. The resulting nonzero numbers lie in the range

$$2^{-127} \approx 10^{-38} \leq |\,x\,| \leq 2^{127} \approx 10^{38},$$

and allow one to accurately represent numbers with approximately 7 significant decimal digits of real numbers lying in this range.

- In *double precision* floating point arithmetic, the sign is 1 bit, the exponent is 10 bits, and the mantissa is 53 bits, leading to floating point representations for a total of 1.84×10^{19} different numbers which, apart from 0. The resulting nonzero numbers lie in the range

$$2^{-1023} \approx 10^{-308} \leq |\,x\,| \leq 2^{1023} \approx 10^{308}.$$

In double precision, one can accurately represent approximately 15 decimal digits. Keep in mind floating point numbers are *not uniformly spaced*! Moreover, when passing from $.111111\ldots \times 2^n$ to $.100000\ldots \times 2^{n+1}$, the inter-number spacing suddenly jumps by a factor of 2. The non-smoothly varying spacing inherent in the floating point system can cause subtle, undesirable numerical artifacts in high precision computations.

Remark: Although they are overwhelmingly the most prevalent, fixed and floating point are not the only number systems that have been proposed. See [**9**] for another intriguing possibility.

In the course of a calculation, intermediate errors interact in a complex fashion, and result in a final total error that is not just the sum of the individual errors. If $x^\star$ is an approximation to x, and $y^\star$ is an approximation to y, then, instead of arithmetic operations $+, -, \times, /$ and so on, the computer uses a "pseudo-operations" to combine them. For instance, to approximate the difference $x - y$ of two real numbers, the computer begins by replacing each by its floating point approximation $x^\star$ and $y^\star$. It the subtracts these, and replaces the exact result $x^\star - y^\star$ by its nearest floating point representative, namely $(x^\star - y^\star)^\star$. As the following example makes clear, this is *not* necessarily the same as the floating point approximation to $x - y$, i.e., in general $(x^\star - y^\star)^\star \neq (x - y)^\star$.

Example 1.2. . Lets see what happens if we subtract the rational numbers

$$x = \frac{301}{2000} \approx .15050000\ldots, \qquad y = \frac{301}{2001} \approx .150424787\ldots.$$

The exact answer is

$$x - y = \frac{301}{4002000} \approx .00007521239\ldots.$$

However, if we use rounding to approximate with 4 significant digits[†], then

$$x = \frac{301}{2000} \approx .1505, \qquad y = \frac{301}{2001} \approx .1504 \qquad \text{and so} \qquad x - y \approx .0001,$$

[†] To aid comprehension, we are using base 10 instead of base 2 arithmetic in this example.

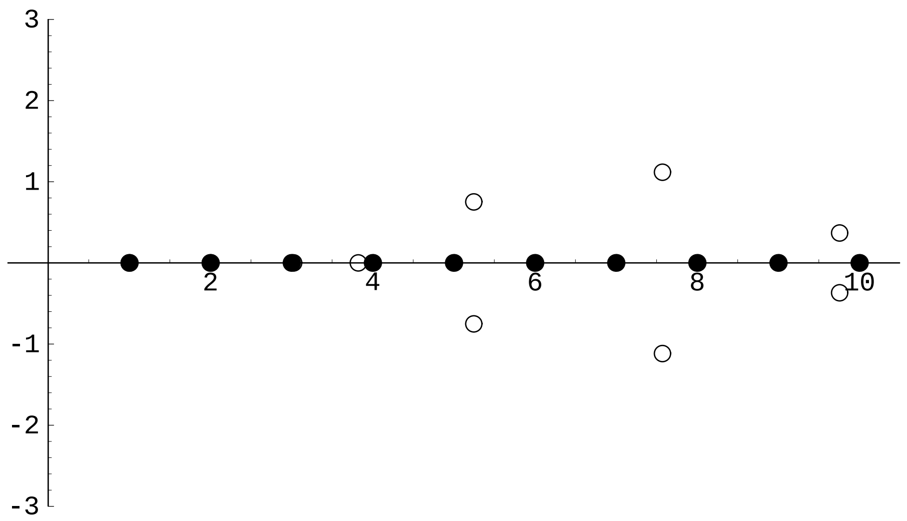


Figure 1.1. Roots of Polynomials.

which has *no* significant digits in common with the actual answer. On the other hand, if we evaluate the difference using the alternative formula

$$
\begin{aligned}
x - y &= \frac{301 \times 2001 - 301 \times 2000}{4002000} = \frac{602301 - 602000}{4002000} \\
&\approx \frac{6.023 \times 10^5 - 6.020 \times 10^5}{4.002 \times 10^6} = \frac{.003 \times 10^5}{4.002 \times 10^6} \approx .00007496,
\end{aligned}
$$

we at least reproduce the first two significant digits. Thus, the result of a floating point computation *depends on the order of the arithmetic operations*! In particular, the associative and distributive laws *are not valid in floating point arithmetic*! In the development of numerical analysis, one tends to not pay attention to this “minor detail', although its effects must always be kept in the back of one's mind when evaluating the results of any serious numerical computation.

Just in case you are getting a little complacent, thinking “gee, a few tiny round off errors can't really make that big a difference”, let us close with two cautionary examples.

Example 1.3. Consider the pair of degree 10 polynomials

$$p(x) = (x-1)(x-2)(x-3)(x-4)(x-5)(x-6)(x-7)(x-8)(x-9)(x-10)$$

and

$$q(x) = p(x) + x^5.$$

They only differ in the value of the coefficient of their middle term, which, by a direct expansion, is $-902055x^5$ in $p(x)$, and $-902054x^5$ in $q(x)$; all other terms are the same. The relative error between the two coefficients is roughly one-thousandth of one percent.

Our task is to compute the roots of these two polynomials, i.e., the solutions to $p(x) = 0$ and $q(x) = 0$. Those of $p(x)$ are obvious. One might expect the roots of $q(x)$ to

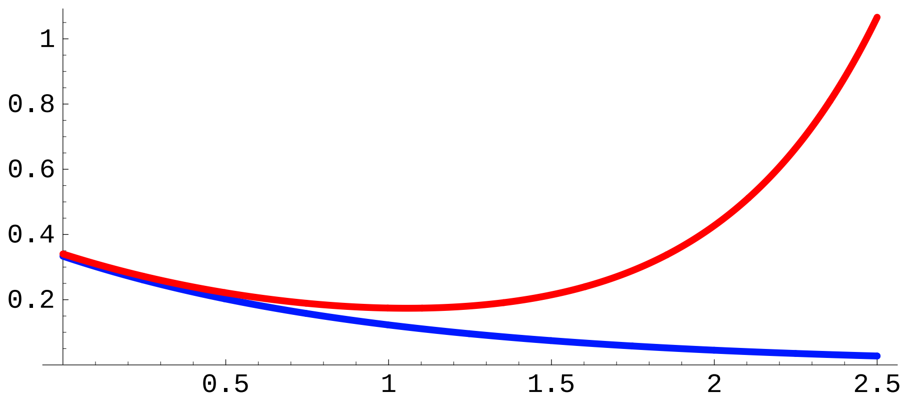


Figure 1.2. Sensitive Dependence on Initial Data.

be rather close to the integers $1, 2, \ldots, 10$. However, their approximate values are

$$\begin{array}{llll} 1.0000027558, & 1.99921, & 3.02591, & 3.82275, \\ 5.24676 \pm 0.751485\, \mathrm{i}\,, & 7.57271 \pm 1.11728\, \mathrm{i}\,, & 9.75659 \pm 0.368389\, \mathrm{i}\,. \end{array}$$

Surprisingly, only the smallest two roots are relatively unchanged; the third and fourth roots differ by roughly 1% and 27%, while the final six roots have mysteriously metamorphosed into three complex conjugate pairs of roots of $q(x)$. The two sets of roots are plotted in Figure 1.1; those of $p(x)$ are indicated by solid disks, while those of $q(x)$ are indicated by open circles.

We have thus learned that an almost negligible change in a single coefficient of a real polynomial can have dramatic and unanticipated effects on its roots. Needless to say, this indicates that finding accurate numerical values for the roots of high degree polynomials is a very challenging problem.

Example 1.4. Consider the initial value problem

$$\frac{du}{dt} - 2\,u = -\,e^{-\,t}, \qquad u(0) = \tfrac{1}{3}.$$

The solution is easily found:

$$u(t) = \tfrac{1}{3}\, e^{-\,t},$$

and is exponentially decaying as $t \to \infty$.

However, in a floating point computer, we are not able to represent the initial condition $\frac{1}{3}$ exactly, and make some small round-off error (depending on the precision of the computer). Let $\varepsilon \neq 0$ represent the error in the initial condition. The solution to the perturbed initial value problem

$$\frac{dv}{dt} - 2\,v = -\,e^{-\,t}, \qquad v(0) = \tfrac{1}{3} + \varepsilon,$$

is

$$v(t) = \tfrac{1}{3}\, e^{-\,t} + \varepsilon\, e^{2\,t},$$

which is exponentially growing as t increases. As sketched in Figure 1.2, the two solutions remain close only for a very short time interval, the duration of which depends on the size in the initial error, but then they eventually diverge far away from each other. As a consequence, a tiny error in the initial data can have a dramatic effect on the solution. This phenomenon is referred to as *sensitive dependence on initial conditions*.

The numerical computation of the exponentially decaying exact solution in the face of round-off errors is an extreme challenge. Even the tiniest of error will immediately introduce an exponentially growing mode which will eventually swamp the true solution. Furthermore, in many important applications, the physically relevant solution is the one that decays to zero at large distances, and is usually distinguished from the vast majority of solutions that grow at large distances. So the computational problem is both important and very difficult.

Examples 1.3 and 1.4 are known as *ill-conditioned problems* meaning that tiny changes in the data have dramatic effects on the solutions. Simple numerical methods work as advertised on well-conditioned problems, but all have their limitations and a sufficiently ill-conditioned problem will test the limits of the algorithm and/or computer, and, in many instances, require revamping a standard numerical solution scheme to adapt to the ill-conditioning. Some problems are so ill-conditioned as to defy even the most powerful computers and sophisticated algorithms. For this reason, numerical analysis will forever remain a vital and vibrant area of mathematical research.

So, numerical analysis cannot be viewed in isolation as a black box, and left in the hands of the mathematical experts. Every practitioner will, sooner or later, confront a problem that tests the limitations of standard algorithms and software. Without a proper understanding of the mathematical principles involved in constructing basic numerical algorithms, one is ill-equipped, if not hopelessly handicapped, when dealing with such problems. The purpose of this series of lectures is to give you the proper mathematical grounding in modern numerical analysis.

AIMS Lecture Notes 2006

Peter J. Olver

2. Numerical Solution of Scalar Equations

Most numerical solution methods are based on some form of iteration. The basic idea is that repeated application of the algorithm will produce closer and closer approximations to the desired solution. To analyze such algorithms, our first task is to understand general iterative processes.

2.1. Iteration of Functions.

Iteration, meaning repeated application of a function, can be viewed as a *discrete dynamical system* in which the continuous time variable has been "quantized" to assume integer values. Even iterating a very simple quadratic scalar function can lead to an amazing variety of dynamical phenomena, including multiply-periodic solutions and genuine chaos. Nonlinear iterative systems arise not just in mathematics, but also underlie the growth and decay of biological populations, predator-prey interactions, spread of communicable diseases such as AIDS, and host of other natural phenomena. Moreover, many numerical solution methods — for systems of algebraic equations, ordinary differential equations, partial differential equations, and so on — rely on iteration, and so the theory underlies the analysis of convergence and efficiency of such numerical approximation schemes.

In general, an iterative system has the form

$$\mathbf{u}^{(k+1)} = \mathbf{g}(\mathbf{u}^{(k)}), \tag{2.1}$$

where $\mathbf{g}\colon \mathbb{R}^n \to \mathbb{R}^n$ is a real vector-valued function. (One can similarly treat iteration of complex-valued functions $\mathbf{g}\colon \mathbb{C}^n \to \mathbb{C}^n$, but, for simplicity, we only deal with real systems here.) A solution is a discrete collection of points[†] $\mathbf{u}^{(k)} \in \mathbb{R}^n$, in which the index $k = 0, 1, 2, 3, \ldots$ takes on non-negative integer values.

Once we specify the initial iterate,

$$\mathbf{u}^{(0)} = \mathbf{c}, \tag{2.2}$$

then the resulting solution to the discrete dynamical system (2.1) is easily computed:

$$\mathbf{u}^{(1)} = \mathbf{g}(\mathbf{u}^{(0)}) = \mathbf{g}(\mathbf{c}), \quad \mathbf{u}^{(2)} = \mathbf{g}(\mathbf{u}^{(1)}) = \mathbf{g}(\mathbf{g}(\mathbf{c})), \quad \mathbf{u}^{(3)} = \mathbf{g}(\mathbf{u}^{(2)}) = \mathbf{g}(\mathbf{g}(\mathbf{g}(\mathbf{c}))), \quad \ldots$$

[†] The superscripts on $\mathbf{u}^{(k)}$ refer to the iteration number, and do *not* denote derivatives.

and so on. Thus, unlike continuous dynamical systems, the existence and uniqueness of solutions is not an issue. As long as each successive iterate $\mathbf{u}^{(k)}$ lies in the domain of definition of $\mathbf{g}$ one merely repeats the process to produce the solution,

$$\mathbf{u}^{(k)} = \overbrace{\mathbf{g} \circ \cdots \circ \mathbf{g}}^{k \text{ times}}(\mathbf{c}), \qquad k = 0, 1, 2, \ldots, \tag{2.3}$$

which is obtained by composing the function $\mathbf{g}$ with itself a total of k times. In other words, the solution to a discrete dynamical system corresponds to repeatedly pushing the $\mathbf{g}$ key on your calculator. For example, entering 0 and then repeatedly hitting the `cos` key corresponds to solving the iterative system

$$u^{(k+1)} = \cos u^{(k)}, \qquad u^{(0)} = 0. \tag{2.4}$$

The first 10 iterates are displayed in the following table:

k	0	1	2	3	4	5	6	7	8	9
$u^{(k)}$	0	1	.540302	.857553	.65429	.79348	.701369	.76396	.722102	.750418

For simplicity, we shall always assume that the vector-valued function $\mathbf{g}\colon \mathbb{R}^n \to \mathbb{R}^n$ is defined on all of $\mathbb{R}^n$; otherwise, we must always be careful that the successive iterates $\mathbf{u}^{(k)}$ never leave its domain of definition, thereby causing the iteration to break down. To avoid technical complications, we will also assume that $\mathbf{g}$ is at least continuous; later results rely on additional smoothness requirements, e.g., continuity of its first and second order partial derivatives.

While the solution to a discrete dynamical system is essentially trivial, understanding its behavior is definitely not. Sometimes the solution converges to a particular value — the key requirement for numerical solution methods. Sometimes it goes off to ∞, or, more precisely, the norms of the iterates are unbounded: $\| \mathbf{u}^{(k)} \| \to \infty$ as $k \to \infty$. Sometimes the solution repeats itself after a while. And sometimes the iterates behave in a seemingly random, chaotic manner — all depending on the function $\mathbf{g}$ and, at times, the initial condition $\mathbf{c}$. Although all of these cases may arise in real-world applications, we shall mostly concentrate upon understanding convergence.

Definition 2.1. A *fixed point* or *equilibrium* of a discrete dynamical system (2.1) is a vector $\mathbf{u}^\star \in \mathbb{R}^n$ such that

$$\mathbf{g}(\mathbf{u}^\star) = \mathbf{u}^\star. \tag{2.5}$$

We easily see that every fixed point provides a constant solution to the discrete dynamical system, namely $\mathbf{u}^{(k)} = \mathbf{u}^\star$ for all k. Moreover, it is not hard to prove that any convergent solution necessarily converges to a fixed point.

Proposition 2.2. *If a solution to a discrete dynamical system converges,*

$$\lim_{k \to \infty} \mathbf{u}^{(k)} = \mathbf{u}^\star,$$

then the limit $\mathbf{u}^\star$ is a fixed point.

Proof: This is a simple consequence of the continuity of $\mathbf{g}$. We have

$$\mathbf{u}^\star = \lim_{k\to\infty} \mathbf{u}^{(k+1)} = \lim_{k\to\infty} \mathbf{g}(\mathbf{u}^{(k)}) = \mathbf{g}\left(\lim_{k\to\infty} \mathbf{u}^{(k)}\right) = \mathbf{g}(\mathbf{u}^\star),$$

the last two equalities following from the continuity of $\mathbf{g}$. *Q.E.D.*

For example, continuing the cosine iteration (2.4), we find that the iterates gradually converge to the value $u^\star \approx .739085$, which is the unique solution to the fixed point equation

$$\cos u = u.$$

Later we will see how to rigorously prove this observed behavior.

Of course, not every solution to a discrete dynamical system will necessarily converge, but Proposition 2.2 says that if it does, then it must converge to a fixed point. Thus, a key goal is to understand when a solution converges, and, if so, to which fixed point — if there is more than one. Fixed points are divided into three classes:

- *asymptotically stable*, with the property that all nearby solutions converge to it,
- *stable*, with the property that all nearby solutions stay nearby, and
- *unstable*, almost all of whose nearby solutions diverge away from the fixed point.

Thus, from a practical standpoint, convergence of the iterates of a discrete dynamical system requires asymptotic stability of the fixed point. Examples will appear in abundance in the following sections.

Scalar Functions

As always, the first step is to thoroughly understand the scalar case, and so we begin with a discrete dynamical system

$$u^{(k+1)} = g(u^{(k)}), \qquad u^{(0)} = c, \tag{2.6}$$

in which $g\colon \mathbb{R} \to \mathbb{R}$ is a continuous, scalar-valued function. As noted above, we will assume, for simplicity, that g is defined everywhere, and so we do not need to worry about whether the iterates $u^{(0)}, u^{(1)}, u^{(2)}, \ldots$ are all well-defined.

As usual, to study systems one begins with an in-depth analysis of the scalar version. Consider the iterative equation

$$u^{(k+1)} = a\, u^{(k)}, \qquad u^{(0)} = c. \tag{2.7}$$

The general solution to (2.7) is easily found:

$$u^{(1)} = a\, u^{(0)} = a\, c, \qquad u^{(2)} = a\, u^{(1)} = a^2\, c, \qquad u^{(3)} = a\, u^{(2)} = a^3\, c,$$

and, in general,

$$u^{(k)} = a^k\, c. \tag{2.8}$$

If the initial condition is $a = 0$, then the solution $u^{(k)} \equiv 0$ is constant. Therefore, 0 is a *fixed point* or *equilibrium solution* for the iterative system.

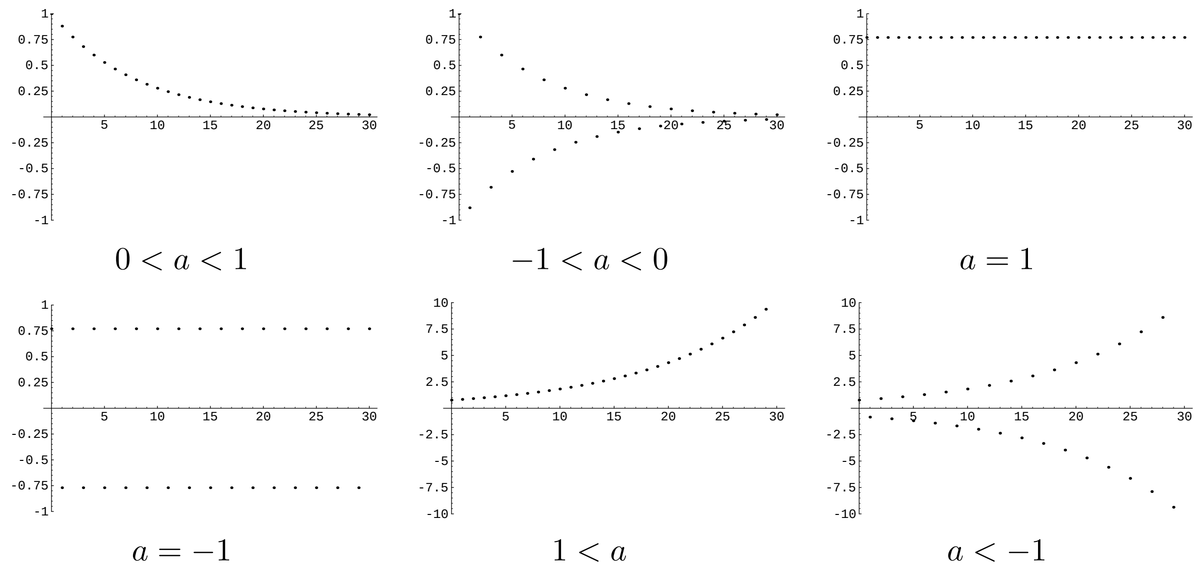


Figure 2.1. One Dimensional Real Linear Iterative Systems.

Example 2.3. Banks add interest to a savings account at discrete time intervals. For example, if the bank offers 5% interest compounded yearly, this means that the account balance will increase by 5% each year. Thus, assuming no deposits or withdrawals, the balance $u^{(k)}$ after k years will satisfy the iterative equation (2.7) with $a = 1 + r$ where $r = .05$ is the interest rate, and the 1 indicates that all the money remains in the account. Thus, after k years, your account balance is

$$u^{(k)} = (1+r)^k c, \qquad \text{where} \qquad c = u^{(0)} \tag{2.9}$$

is your initial deposit. For example, if $c = \$1,000$, after 1 year your account has $u^{(1)} = \$1,050$, after 10 years $u^{(10)} = \$1,628.89$, after 50 years $u^{(50)} = \$11,467.40$, and after 200 years $u^{(200)} = \$17,292,580.82$.

When the interest is compounded monthly, the rate is still quoted on a yearly basis, and so you receive $\frac{1}{12}$ of the interest each month. If $\hat{u}^{(k)}$ denotes the balance after k months, then, after n years, the account balance is $\hat{u}^{(12n)} = \left(1 + \frac{1}{12} r\right)^{12n} c$. Thus, when the interest rate of 5% is compounded monthly, your account balance is $\hat{u}^{(12)} = \$1,051.16$ after 1 year, $\hat{u}^{(120)} = \$1,647.01$ after 10 years, $\hat{u}^{(600)} = \$12,119.38$ after 50 years, and $\hat{u}^{(2400)} = \$21,573,572.66$ dollars after 200 years. So, if you wait sufficiently long, compounding will have a dramatic effect. Similarly, daily compounding replaces 12 by 365.25, the number of days in a year. After 200 years, the balance is $\$22,011,396.03$.

Let us analyze the solutions of scalar iterative equations, starting with the case when $a \in \mathbb{R}$ is a real constant. Aside from the equilibrium solution $u^{(k)} \equiv 0$, the iterates exhibit five qualitatively different behaviors, depending on the size of the coefficient a.

(*a*) If $a = 0$, the solution immediately becomes zero, and stays there, so $u^{(k)} = 0$ for all $k \geq 1$.

(*b*) If $0 < a < 1$, then the solution is of one sign, and tends monotonically to zero, so $u^{(k)} \to 0$ as $k \to \infty$.

(c) If $-1 < a < 0$, then the solution tends to zero: $u^{(k)} \to 0$ as $k \to \infty$. Successive iterates have alternating signs.

(d) If $a = 1$, the solution is constant: $u^{(k)} = a$, for all $k \geq 0$.

(e) If $a = -1$, the solution switches back and forth between two values; $u^{(k)} = (-1)^k c$.

(f) If $1 < a < \infty$, then the iterates $u^{(k)}$ become unbounded. If $c > 0$, they go monotonically to $+\infty$; if $c < 0$, to $-\infty$.

(g) If $-\infty < a < -1$, then the iterates $u^{(k)}$ also become unbounded, with alternating signs.

In Figure 2.1 we exhibit representative *scatter plots* for the nontrivial cases (b – g). The horizontal axis indicates the index k and the vertical axis the solution value u. Each dot in the scatter plot represents an iterate $u^{(k)}$.

To describe the different scenarios, we adopt a terminology that already appeared in the continuous realm. In the first three cases, the fixed point $u = 0$ is said to be *globally asymptotically stable* since all solutions tend to 0 as $k \to \infty$. In cases (d) and (e), the zero solution is *stable*, since solutions with nearby initial data, $|\, c\,| \ll 1$, remain nearby. In the final two cases, the zero solution is *unstable*; any nonzero initial data $a \neq 0$ — no matter how small — will give rise to a solution that eventually goes arbitrarily far away from equilibrium.

Let us also analyze complex scalar iterative systems. The coefficient a and the initial datum c in (2.7) are allowed to be complex numbers. The solution is the same, (2.8), but now we need to know what happens when we raise a complex number a to a high power. The secret is to write $a = r\, e^{\,\mathrm{i}\,\theta}$ in polar form, where $r = |\, a\,|$ is its modulus and $\theta = \operatorname{ph} a$ its angle or phase. Then $a^k = r^k\, e^{\,\mathrm{i}\,k\,\theta}$. Since $|\, e^{\,\mathrm{i}\,k\,\theta}\,| = 1$, we have $|\, a^k\,| = |\, a\,|^k$, and so the solutions (2.8) have modulus $|\, u^{(k)}\,| = |\, a^k\, a\,| = |\, a\,|^k\, |\, a\,|$. As a result, $u^{(k)}$ will remain bounded if and only if $|\, a\,| \leq 1$, and will tend to zero as $k \to \infty$ if and only if $|\, a\,| < 1$.

We have thus established the basic stability criteria for scalar, linear systems.

Theorem 2.4. *The zero solution to a* (*real or complex*) *scalar iterative system* $u^{(k+1)} = a\, u^{(k)}$ *is*

(a) asymptotically stable *if and only if* $|\, a\,| < 1$,

(b) stable *if and only if* $|\, a\,| \leq 1$,

(c) unstable *if and only if* $|\, a\,| > 1$.

Nonlinear Scalar Iteration

The simplest "nonlinear" case is that of an affine function

$$g(u) = a\, u + b, \tag{2.10}$$

leading to an *affine discrete dynamical system*

$$u^{(k+1)} = a\, u^{(k)} + b. \tag{2.11}$$

The only fixed point is the solution to

$$u^\star = g(u^\star) = a\, u^\star + b, \qquad \text{namely,} \qquad u^\star = \frac{b}{1-a}\,. \tag{2.12}$$

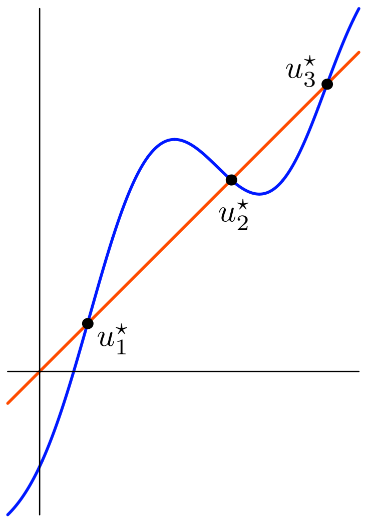


Figure 2.2. Fixed Points.

The formula for $u^\star$ requires that $a \neq 1$, and, indeed, the case $a = 1$ has no fixed point, as the reader can easily confirm.

Since we already know the value of $u^\star$, we can readily analyze the differences

$$e^{(k)} = u^{(k)} - u^\star, \tag{2.13}$$

between successive iterates and the fixed point. Observe that, the smaller $e^{(k)}$ is, the closer $u^{(k)}$ is to the desired fixed point. In many applications, the iterate $u^{(k)}$ is viewed as an approximation to the fixed point $u^\star$, and so $e^{(k)}$ is interpreted as the *error* in the k^{th} iterate. Subtracting the fixed point equation (2.12) from the iteration equation (2.11), we find

$$u^{(k+1)} - u^\star = a\,(u^{(k)} - u^\star).$$

Therefore the errors $e^{(k)}$ are related by a *linear iteration*

$$e^{(k+1)} = a\,e^{(k)}, \qquad \text{and hence} \qquad e^{(k)} = a^k e^{(0)}. \tag{2.14}$$

Therefore, the solutions to this scalar linear iteration converge:

$$e^{(k)} \longrightarrow 0 \qquad \text{and hence} \qquad u^{(k)} \longrightarrow u^\star, \qquad \text{if and only if} \qquad |\,a\,| < 1.$$

This is the criterion for *asymptotic stability* of the fixed point, or, equivalently, convergence of the affine iterative system (2.11). The magnitude of a determines the rate of convergence, and the closer it is to 0, the faster the iterates approach the fixed point.

Example 2.5. The affine function

$$g(u) = \tfrac{1}{4}\,u + 2$$

leads to the iterative scheme

$$u^{(k+1)} = \tfrac{1}{4}\,u^{(k)} + 2.$$

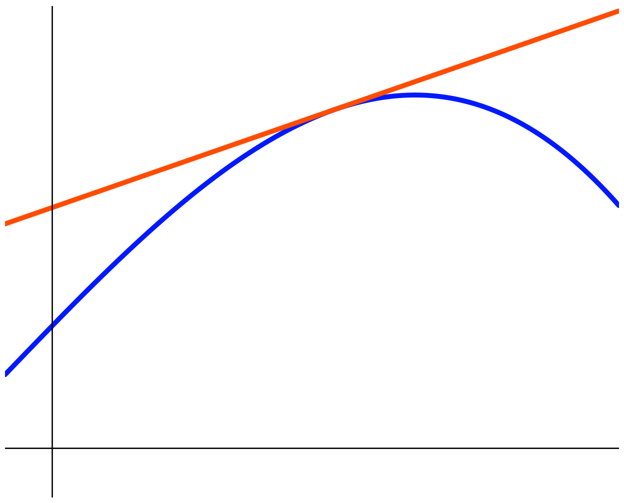


Figure 2.3. Tangent Line Approximation.

Starting with the initial condition $u^{(0)} = 0$, the ensuing values are

k	1	2	3	4	5	6	7	8
$u^{(k)}$	2.0	2.5	2.625	2.6562	2.6641	2.6660	2.6665	2.6666

Thus, after 8 iterations, the iterates have produced the fixed point $u^\star = \frac{8}{3}$ to 4 decimal places. The rate of convergence is $\frac{1}{4}$, and indeed

$$|\, e^{(k)} \,| = |\, u^{(k)} - u^\star \,| = \left(\tfrac{1}{4}\right)^k |\, u^{(0)} - u^\star \,| = \tfrac{8}{3} \left(\tfrac{1}{4}\right)^k \longrightarrow 0 \qquad \text{as} \qquad k \longrightarrow \infty.$$

Let us now turn to the fully nonlinear case. First note that the fixed points of $g(u)$ correspond to the intersections of its graph with the graph of the function $i(u) = u$. For instance Figure 2.2 shows the graph of a function that has 3 fixed points, labeled $u_1^\star, u_2^\star, u_3^\star$.

In general, near any point in its domain, a (smooth) nonlinear function can be well approximated by its tangent line, which repre4sents the graph of an affine function; see Figure 2.3. Therefore, if we are close to a fixed point $u^\star$, then we might expect the iterative system based on the nonlinear function $g(u)$ to behave very much like that of its affine tangent line approximation. And, indeed, this intuition turns out to be essentially correct. This result forms our first concrete example of *linearization*, in which the analysis of a nonlinear system is based on its linear (or, more precisely, affine) approximation.

The explicit formula for the tangent line to $g(u)$ near the fixed point $u = u^\star = g(u^\star)$ is

$$g(u) \approx g(u^\star) + g'(u^\star)(u - u^\star) \equiv a\,u + b, \tag{2.15}$$

where

$$a = g'(u^\star), \qquad\qquad b = g(u^\star) - g'(u^\star)\,u^\star = \big(\,1 - g'(u^\star)\,\big)\,u^\star.$$

Note that $u^\star = b\,/(1-a)$ remains a fixed point for the affine approximation: $a\,u^\star + b = u^\star$. According to the preceding discussion, the convergence of the iterates for the affine approximation is governed by the size of the coefficient $a = g'(u^\star)$. This observation inspires the basic stability criterion for fixed points of scalar iterative systems.

Theorem 2.6. *Let $g(u)$ be a continuously differentiable scalar function. Suppose $u^\star = g(u^\star)$ is a fixed point. If $|\, g'(u^\star) \,| < 1$, then $u^\star$ is an asymptotically stable fixed point, and hence any sequence of iterates $u^{(k)}$ which starts out sufficiently close to $u^\star$ will converge to $u^\star$. On the other hand, if $|\, g'(u^\star) \,| > 1$, then $u^\star$ is an unstable fixed point, and the only iterates which converge to it are those that land exactly on it, i.e., $u^{(k)} = u^\star$ for some $k \geq 0$.*

Proof: The goal is to prove that the errors $e^{(k)} = u^{(k)} - u^\star$ between the iterates and the fixed point tend to 0 as $k \to \infty$. To this end, we try to estimate $e^{(k+1)}$ in terms of $e^{(k)}$. According to (2.6) and the Mean Value Theorem from calculus,

$$e^{(k+1)} = u^{(k+1)} - u^\star = g(u^{(k)}) - g(u^\star) = g'(v)\,(u^{(k)} - u^\star) = g'(v)\,e^{(k)}, \tag{2.16}$$

for some v lying between $u^{(k)}$ and $u^\star$. By continuity, if $|\, g'(u^\star) \,| < 1$ at the fixed point, then we can choose $\delta > 0$ and $|\, g'(u^\star) \,| < \sigma < 1$ such that the estimate

$$|\, g'(v) \,| \leq \sigma < 1 \qquad \text{whenever} \qquad |\, v - u^\star \,| < \delta \tag{2.17}$$

holds in a (perhaps small) interval surrounding the fixed point. Suppose

$$|\, e^{(k)} \,| = |\, u^{(k)} - u^\star \,| < \delta.$$

Then the point v in (2.16), which is closer to $u^\star$ than $u^{(k)}$, satisfies (2.17). Therefore,

$$|\, u^{(k+1)} - u^\star \,| \;\leq\; \sigma\, |\, u^{(k)} - u^\star \,|, \qquad \text{and hence} \qquad |\, e^{(k+1)} \,| \;\leq\; \sigma\, |\, e^{(k)} \,|. \tag{2.18}$$

In particular, since $\sigma < 1$, we have $|\, u^{(k+1)} - u^\star \,| < \delta$, and hence the subsequent iterate $u^{(k+1)}$ also lies in the interval where (2.17) holds. Repeating the argument, we conclude that, provided the initial iterate satisfies

$$|\, e^{(0)} \,| = |\, u^{(0)} - u^\star \,| < \delta,$$

the subsequent errors are bounded by

$$e^{(k)} \;\leq\; \sigma^k\, e^{(0)}, \qquad \text{and hence} \qquad e^{(k)} = |\, u^{(k)} - u^\star \,| \;\longrightarrow\; 0 \quad \text{as} \quad k \to \infty,$$

which completes the proof of the theorem in the stable case.

The proof in unstable case is left as an exercise for the reader. *Q.E.D.*

Remark: The constant σ governs the rate of convergence of the iterates to the fixed point. The closer the iterates are to the fixed point, the smaller we can choose δ in (2.17), and hence the closer we can choose σ to $|\, g'(u^\star) \,|$. Thus, roughly speaking, $|\, g'(u^\star) \,|$ governs the speed of convergence, once the iterates get close to the fixed point. This observation will be developed more fully in the following subsection.

Remark: The cases when $g'(u^\star) = \pm 1$ are *not* covered by the theorem. For a linear system, such fixed points are stable, but not asymptotically stable. For nonlinear systems, more detailed knowledge of the nonlinear terms is required in order to resolve the status — stable or unstable — of the fixed point. Despite their importance in certain applications, we will not try to analyze such borderline cases any further here.

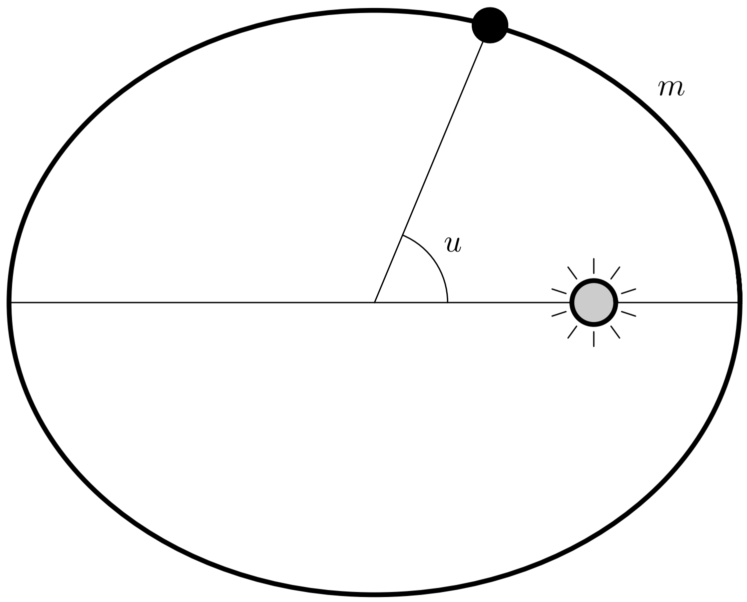


Figure 2.4. Planetary Orbit.

Example 2.7. Given constants ϵ, m, the trigonometric equation

$$u = m + \epsilon \sin u \tag{2.19}$$

is known as *Kepler's equation*. It arises in the study of planetary motion, in which $0 < \epsilon < 1$ represents the *eccentricity* of an elliptical planetary orbit, u is the *eccentric anomaly*, defined as the angle formed at the center of the ellipse by the planet and the major axis, and $m = 2\pi t/T$ is its *mean anomaly*, which is the time, measured in units of $T/(2\pi)$ where T is the period of the orbit, i.e., the length of the planet's year, since perihelion or point of closest approach to the sun; see Figure 2.4.

The solutions to Kepler's equation are the fixed points of the discrete dynamical system based on the function

$$g(u) = m + \epsilon \sin u.$$

Note that

$$|\, g'(u) \,| = |\, \epsilon \cos u \,| = |\, \epsilon \,| < 1, \tag{2.20}$$

which automatically implies that the as yet unknown fixed point is stable. Indeed, condition (2.20) is enough to prove the existence of a unique stable fixed point; see the remark after Theorem 9.7. In the particular case $m = \epsilon = \frac{1}{2}$, the result of iterating $u^{(k+1)} = \frac{1}{2} + \frac{1}{2} \sin u^{(k)}$ starting with $u^{(0)} = 0$ is

k	1	2	3	4	5	6	7	8	9
$u^{(k)}$	.5	.7397	.8370	.8713	.8826	.8862	.8873	.8877	.8878

After 13 iterations, we have converged sufficiently close to the solution (fixed point) $u^\star = .887862$ to have computed its value to 6 decimal places.

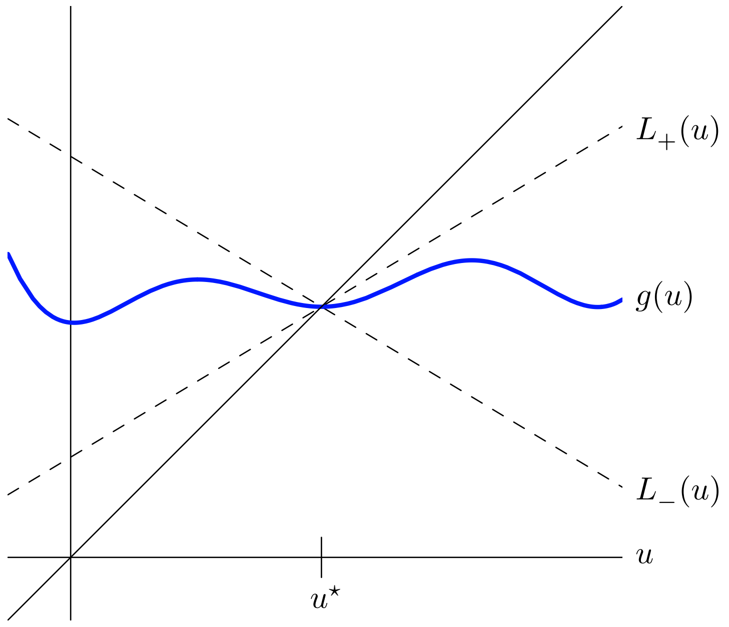


Figure 2.5. Graph of a Contraction.

Inspection of the proof of Theorem 2.6 reveals that we never really used the differentiability of g, except to verify the inequality

$$|\, g(u) - g(u^\star)\,| \le \sigma\,|\, u - u^\star\,| \qquad \text{for some fixed } \sigma < 1. \tag{2.21}$$

A function that satisfies (2.21) for all nearby u is called a *contraction* at the point $u^\star$. *Any* function $g(u)$ whose graph lies between the two lines

$$L_\pm(u) = g(u^\star) \pm \sigma\,(u - u^\star) \qquad \text{for some} \quad \sigma < 1,$$

for all u sufficiently close to $u^\star$, i.e., such that $|\, u - u^\star\,| < \delta$ for some $\delta > 0$, defines a contraction, and hence fixed point iteration starting with $|\, u^{(0)} - u^\star\,| < \delta$ will converge to $u^\star$; see Figure 2.5. In particular, any function that is differentiable at $u^\star$ with $|\, g'(u^\star)\,| < 1$ defines a contraction at $u^\star$.

Example 2.8. The simplest truly nonlinear example is a quadratic polynomial. The most important case is the so-called *logistic map*

$$g(u) = \lambda\, u(1-u), \tag{2.22}$$

where $\lambda \ne 0$ is a fixed non-zero parameter. (The case $\lambda = 0$ is completely trivial. Why?) In fact, an elementary change of variables can make any quadratic iterative system into one involving a logistic map.

The fixed points of the logistic map are the solutions to the quadratic equation

$$u = \lambda\, u(1-u), \qquad \text{or} \qquad \lambda\, u^2 - \lambda\, u + 1 = 0.$$

Using the quadratic formula, we conclude that $g(u)$ has two fixed points:

$$u_1^\star = 0, \qquad\qquad u_2^\star = 1 - \frac{1}{\lambda}\,.$$

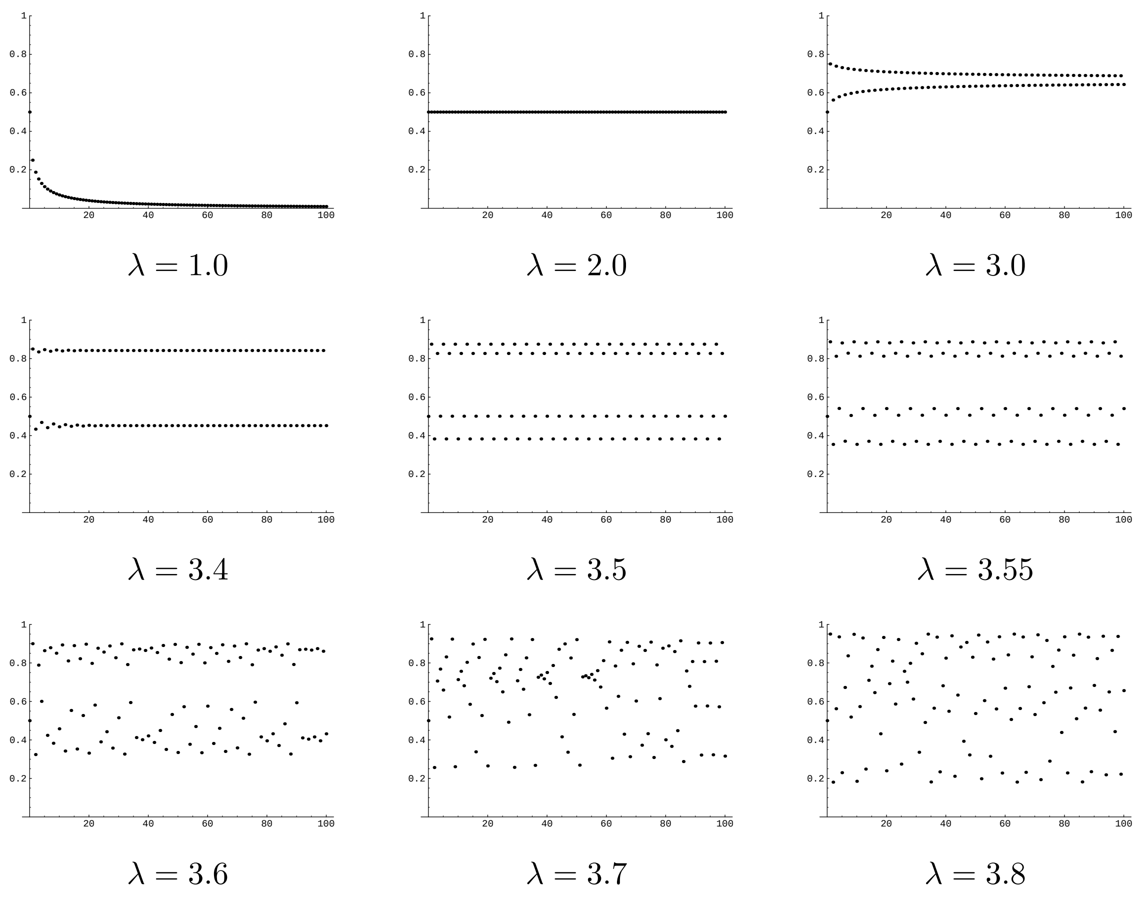


Figure 2.6. Logistic Iterates.

Let us apply Theorem 2.6 to determine their stability. The derivative is

$$g'(u) = \lambda - 2\,\lambda\, u, \qquad \text{and so} \qquad g'(u_1^\star) = \lambda, \qquad g'(u_2^\star) = 2 - \lambda.$$

Therefore, if $|\,\lambda\,| < 1$, the first fixed point is stable, while if $1 < \lambda < 3$, the second fixed point is stable. For $\lambda < -1$ or $\lambda > 3$ neither fixed point is stable, and we expect the iterates to not converge at all.

Numerical experiments with this example show that it is the source of an amazingly diverse range of behavior, depending upon the value of the parameter λ. In the accompanying Figure 2.6, we display the results of iteration starting with initial point $u^{(0)} = .5$ for several different values of λ; in each plot, the horizontal axis indicates the iterate number k and the vertical axis the iterate valoue $u^{(k)}$ for $k = 0, \ldots, 100$. As expected from Theorem 2.6, the iterates converge to one of the fixed points in the range $-1 < \lambda < 3$, except when $\lambda = 1$. For λ a little bit larger than $\lambda_1 = 3$, the iterates do not converge to a fixed point. But it does not take long for them to settle down, switching back and forth between two particular values. This behavior indicates the exitence of a (stable) *period* 2 *orbit* for the discrete dynamical system, in accordance with the following definition.

Definition 2.9. A *period k orbit* of a discrete dynamical system is a solution that satisfies $u^{(n+k)} = u^{(n)}$ for all $n = 0, 1, 2, \ldots$. The (*minimal*) *period* is the smallest positive value of k for which this condition holds.

Thus, a fixed point
$$u^{(0)} = u^{(1)} = u^{(2)} = \cdots$$
is a period 1 orbit. A period 2 orbit satisfies
$$u^{(0)} = u^{(2)} = u^{(4)} = \cdots \qquad \text{and} \qquad u^{(1)} = u^{(3)} = u^{(5)} = \cdots ,$$
but $u^{(0)} \neq u^{(1)}$, as otherwise the minimal period would be 1. Similarly, a period 3 orbit has
$$u^{(0)} = u^{(3)} = u^{(6)} = \cdots , \qquad u^{(1)} = u^{(4)} = u^{(7)} = \cdots , \qquad u^{(2)} = u^{(5)} = u^{(8)} = \cdots ,$$
with $u^{(0)}, u^{(1)}, u^{(2)}$ distinct. Stability of a period k orbit implies that nearby iterates converge to this periodic solution.

For the logistic map, the period 2 orbit persists until $\lambda = \lambda_2 \approx 3.4495$, after which the iterates alternate between four values — a period 4 orbit. This again changes at $\lambda = \lambda_3 \approx 3.5441$, after which the iterates end up alternating between eight values. In fact, there is an increasing sequence of values
$$3 = \lambda_1 < \lambda_2 < \lambda_3 < \lambda_4 < \cdots ,$$
where, for any $\lambda_n < \lambda \leq \lambda_{n+1}$, the iterates eventually follow a period 2^n orbit. Thus, as λ passes through each value λ_n the period of the orbit goes from 2^n to $2 \cdot 2^n = 2^{n+1}$, and the discrete dynamical system experiences a *bifurcation*. The bifurcation values λ_n are packed closer and closer together as n increases, piling up on an eventual limiting value
$$\lambda_\star = \lim_{n \to \infty} \lambda_n \approx 3.5699,$$
at which point the orbit's period has, so to speak, become infinitely large. The entire phenomena is known as a *period doubling cascade*.

Interestingly, the ratios of the distances between successive bifurcation points approaches a well-defined limit,
$$\frac{\lambda_{n+2} - \lambda_{n+1}}{\lambda_{n+1} - \lambda_n} \quad \longrightarrow \quad 4.6692\ldots , \tag{2.23}$$
known as *Feigenbaum's constant*. In the 1970's, the American physicist Mitchell Feigenbaum, [**16**], discovered that similar period doubling cascades appear in a broad range of discrete dynamical systems. Even more remarkably, in almost all cases, the corresponding ratios of distances between bifurcation points has the *same* limiting value. Feigenbaum's experimental observations were rigorously proved by Oscar Lanford in 1982, [**32**].

After λ passes the limiting value $\lambda_\star$, all hell breaks loose. The iterates become completely chaotic[†], moving at random over the interval $[0, 1]$. But this is not the end of the

[†] The term "chaotic" does have a precise mathematical definition, but the reader can take it more figuratively for the purposes of this elementary exposition.

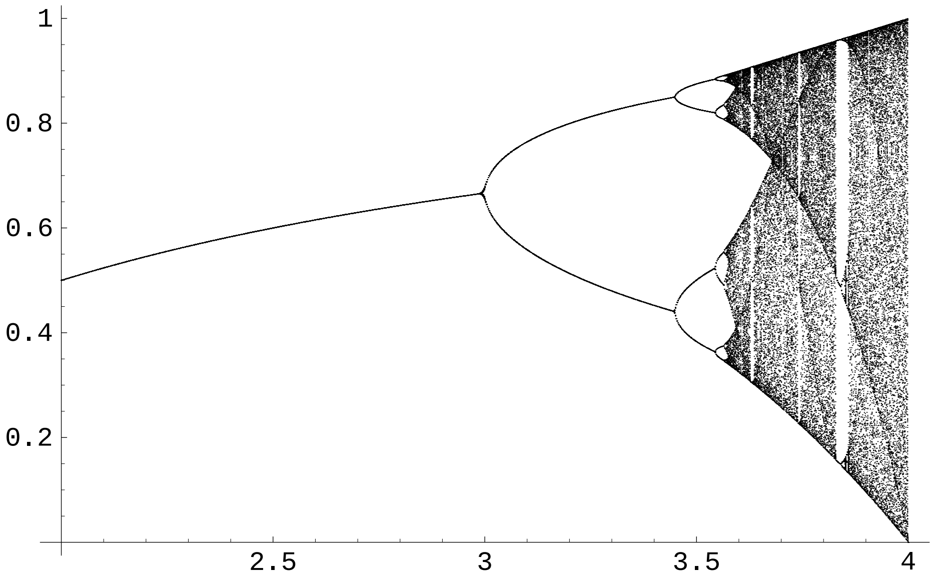


Figure 2.7. The Logistic Map.

story. Embedded within this chaotic regime are certain small ranges of λ where the system settles down to a stable orbit, whose period is no longer necessarily a power of 2. In fact, there exist values of λ for which the iterates settle down to a stable orbit of period k for *any* positive integer k. For instance, as λ increases past $\lambda_{3,\star} \approx 3.83$, a period 3 orbit appears over a small range of values, after which, as λ increses slightly further, there is a period doubling cascade where period 6, 12, 24, ... orbits successively appear, each persisting on a shorter and shorter range of parameter values, until λ passes yet another critical value where chaos breaks out yet again. There is a well-prescribed order in which the periodic orbits make their successive appearance, and each odd period k orbit is followed by a very closely spaced sequence of period doubling bifurcations, of periods $2^n k$ for $n = 1, 2, 3, \ldots$, after which the iterates revert to completely chaotic behavior until the next periodic case emerges. The ratios of distances between bifurcation points always have the same Feigenbaum limit (2.23). Finally, these periodic and chaotic windows all pile up on the ultimate parameter value $\lambda_\star^\star = 4$. And then, when $\lambda > 4$, all the iterates go off to ∞, and the system ceases to be interesting.

The reader is encouraged to write a simple computer program and perform some numerical experiments. In particular, Figure 2.7 shows the asymptotic behavior of the iterates for values of the parameter in the interesting range $2 < \lambda < 4$. The horizontal axis is λ, and the marked points show the ultimate fate of the iteration for the given value of λ. For instance, each point the single curve lying above the smaller values of λ represents a stable fixed point; this bifurcates into a pair of curves representing stable period 2 orbits, which then bifurcates into 4 curves representing period 4 orbits, and so on. Chaotic behavior is indicated by a somewhat random pattern of points lying above the value of λ. To plot this figure, we ran the logistic iteration $u^{(n)}$ for $0 \leq n \leq 100$, discarded the first 50 points, and then plotted the next 50 iterates $u^{(51)}, \ldots, u^{(100)}$. Investigation of the fine detailed structure of the logistic map requires yet more iterations with increased numerical accuracy. In addition one should discard more of the initial iterates so as to give the system enough time to settle down to a stable periodic orbit or, alternatively, continue in a chaotic manner.

Remark: So far, we have only looked at real scalar iterative systems. Complex discrete dynamical systems display yet more remarkable and fascinating behavior. The complex version of the logistic iteration equation leads to the justly famous Julia and Mandelbrot sets, [**33**], with their stunning, psychedelic fractal structure, [**42**].

The rich range of phenomena in evidence, even in such extremely simple nonlinear iterative systems, is astounding. While intimations first appeared in the late nineteenth century research of the influential French mathematician Henri Poincaré, serious investigations were delayed until the advent of the computer era, which precipitated an explosion of research activity in the area of dynamical systems. Similar period doubling cascades and chaos are found in a broad range of nonlinear systems, [**1**], and are often encountered in physical applications, [**35**]. A modern explanation of fluid turbulence is that it is a (very complicated) form of chaos, [**1**].

Quadratic Convergence

Let us now return to the more mundane case when the iterates converge to a stable fixed point of the discrete dynamical system. In applications, we use the iterates to compute a precise[†] numerical value for the fixed point, and hence the efficiency of the algorithm depends on the speed of convergence of the iterates.

According to the remark following the proof Theorem 2.6, the convergence rate of an iterative system is essentially governed by the magnitude of the derivative $|\, g'(u^\star)\,|$ at the fixed point. The basic inequality (2.18) for the errors $e^{(k)} = u^{(k)} - u^\star$, namely

$$|\, e^{(k+1)}\,| \le \sigma\, |\, e^{(k)}\,|,$$

is known as a *linear convergence estimate*. It means that, once the iterates are close to the fixed point, the error decreases by a factor of (at least) $\sigma \approx |\, g'(u^\star)\,|$ at each step. If the kth iterate $u^{(k)}$ approximates the fixed point $u^\star$ correctly to m decimal places, so its error is bounded by

$$|\, e^{(k)}\,| < .5 \times 10^{-m},$$

then the $(k+1)^{\rm st}$ iterate satisfies the error bound

$$|\, e^{(k+1)}\,| \le \sigma\, |\, e^{(k)}\,| < .5 \times 10^{-m}\,\sigma = .5 \times 10^{-m+\log_{10}\sigma}.$$

More generally, for any $j > 0$,

$$|\, e^{(k+j)}\,| \le \sigma^j\, |\, e^{(k)}\,| < .5 \times 10^{-m}\,\sigma^j = .5 \times 10^{-m+j\,\log_{10}\sigma},$$

which means that the $(k+j)^{\rm th}$ iterate $u^{(k+j)}$ has at least[‡]

$$m - j\,\log_{10}\sigma = m + j\,\log_{10}\sigma^{-1}$$

[†] The degree of precision is to be specified by the user and the application.

[‡] Note that since $\sigma < 1$, the logarithm $\log_{10}\sigma^{-1} = -\log_{10}\sigma > 0$ is positive.

correct decimal places. For instance, if $\sigma = .1$ then each new iterate produces one new decimal place of accuracy (at least), while if $\sigma = .9$ then it typically takes $22 \approx -1/\log_{10} .9$ iterates to produce just one additional accurate digit!

This means that there is a huge advantage — particularly in the application of iterative methods to the numerical solution of equations — to arrange that $|\, g'(u^\star) \,|$ be as small as possible. The fastest convergence rate of all will occur when $g'(u^\star) = 0$. In fact, in such a happy situation, the rate of convergence is not just slightly, but dramatically faster than linear.

Theorem 2.10. *Suppose that $g \in \mathrm{C}^2$, and $u^\star = g(u^\star)$ is a fixed point such that $g'(u^\star) = 0$. Then, for all iterates $u^{(k)}$ sufficiently close to $u^\star$, the errors $e^{(k)} = u^{(k)} - u^\star$ satisfy the* quadratic convergence estimate

$$|\, e^{(k+1)} \,| \;\leq\; \tau \,|\, e^{(k)} \,|^2 \tag{2.24}$$

for some constant $\tau > 0$.

Proof: Just as that of the linear convergence estimate (2.18), the proof relies on approximating $g(u)$ by a simpler function near the fixed point. For linear convergence, an affine approximation sufficed, but here we require a higher order approximation. Thus, we replace the mean value formula (2.16) by the first order Taylor expansion

$$g(u) = g(u^\star) + g'(u^\star)\,(u - u^\star) + \tfrac{1}{2}\, g''(w)\,(u - u^\star)^2, \tag{2.25}$$

where the final error term depends on an (unknown) point w that lies between u and $u^\star$. At a fixed point, the constant term is $g(u^\star) = u^\star$. Furthermore, under our hypothesis $g'(u^\star) = 0$, and so (2.25) reduces to

$$g(u) - u^\star = \tfrac{1}{2}\, g''(w)\,(u - u^\star)^2.$$

Therefore,

$$|\, g(u) - u^\star \,| \leq \tau \,|\, u - u^\star \,|^2, \tag{2.26}$$

where τ is chosen so that

$$\tfrac{1}{2}\,|\, g''(w) \,| \leq \tau \tag{2.27}$$

for all w sufficiently close to $u^\star$. Therefore, the magnitude of τ is governed by the size of the *second derivative* of the iterative function $g(u)$ near the fixed point. We use the inequality (2.26) to estimate the error

$$|\, e^{(k+1)} \,| = |\, u^{(k+1)} - u^\star \,| = |\, g(u^{(k)}) - g(u^\star) \,| \;\leq\; \tau \,|\, u^{(k)} - u^\star \,|^2 = \tau \,|\, e^{(k)} \,|^2,$$

which establishes the quadratic convergence estimate (2.24). *Q.E.D.*

Let us see how the quadratic estimate (2.24) speeds up the convergence rate. Following our earlier argument, suppose $u^{(k)}$ is correct to m decimal places, so

$$|\, e^{(k)} \,| < .5 \times 10^{-m}.$$

Then (2.24) implies that

$$| e^{(k+1)} | < .5 \times (10^{-m})^2 \tau = .5 \times 10^{-2m + \log_{10} \tau},$$

and so $u^{(k+1)}$ has $2m - \log_{10} \tau$ accurate decimal places. If $\tau \approx | g''(u^\star) |$ is of moderate size, we have essentially *doubled* the number of accurate decimal places in just a single iterate! A second iteration will double the number of accurate digits yet again. Thus, the convergence of a quadratic iteration scheme is *extremely* rapid, and, barring round-off errors, one can produce any desired number of digits of accuracy in a very short time. For example, if we start with an initial guess that is accurate in the first decimal digit, then a linear iteration with $\sigma = .1$ will require 49 iterations to obtain 50 decimal place accuracy, whereas a quadratic iteration (with $\tau = 1$) will only require 6 iterations to obtain $2^6 = 64$ decimal places of accuracy!

Example 2.11. Consider the function

$$g(u) = \frac{2u^3 + 3}{3u^2 + 3}.$$

There is a unique (real) fixed point $u^\star = g(u^\star)$, which is the real solution to the cubic equation

$$\tfrac{1}{3} u^3 + u - 1 = 0.$$

Note that

$$g'(u) = \frac{2u^4 + 6u^2 - 6u}{3(u^2+1)^2} = \frac{6u\left(\frac{1}{3}u^3 + u - 1\right)}{3(u^2+1)^2},$$

and hence $g'(u^\star) = 0$ vanishes at the fixed point. Theorem 2.10 implies that the iterations will exhibit quadratic convergence to the root. Indeed, we find, starting with $u^{(0)} = 0$, the following values:

k	1	2	3
$u^{(k)}$	1.00000000000000	.833333333333333	.817850637522769

	4	5	6
	.817731680821982	.817731673886824	.817731673886824

The convergence rate is dramatic: after only 5 iterations, we have produced the first 15 decimal places of the fixed point. In contrast, the linearly convergent scheme based on $\widetilde{g}(u) = 1 - \frac{1}{3}u^3$ takes 29 iterations just to produce the first 5 decimal places of the same solution.

In practice, the appearance of a quadratically convergent fixed point is a matter of luck. The construction of quadratically convergent iterative methods for solving equations will be the focus of the following Section.

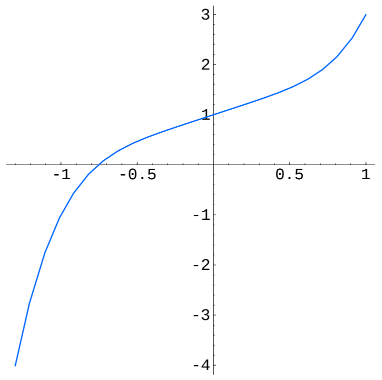


Figure 2.8. Graph of $u^5 + u + 1$.

2.2. Numerical Solution of Equations.

Solving nonlinear equations and systems of equations is, of course, a problem of utmost importance in mathematics and its manifold applications. We begin by studying the scalar case. Thus, we are given a real-valued function $f: \mathbb{R} \to \mathbb{R}$, and seek its *roots*, i.e., the real solution(s) to the scalar equation

$$f(u) = 0. \tag{2.28}$$

Here are some prototypical examples:

(a) Find the roots of the quintic polynomial equation

$$u^5 + u + 1 = 0. \tag{2.29}$$

Graphing the left hand side of the equation, as in Figure 2.8, convinces us that there is just one real root, lying somewhere between -1 and $-.5$. While there are explicit algebraic formulas for the roots of quadratic, cubic, and quartic polynomials, a famous theorem[†] due to the Norwegian mathematician Nils Henrik Abel in the early 1800's states that there is *no* such formula for generic fifth order polynomial equations.

(b) Any fixed point equation $u = g(u)$ has the form (2.28) where $f(u) = u - g(u)$. For example, the trigonometric Kepler equation

$$u - \epsilon \sin u = m$$

arises in the study of planetary motion, cf. Example 2.7. Here ϵ, m are fixed constants, and we seek a corresponding solution u.

(c) Suppose we are given chemical compounds A, B, C that react to produce a fourth compound D according to

$$2A + B \longleftrightarrow D, \qquad\qquad A + 3C \longleftrightarrow D.$$

Let a, b, c be the initial concentrations of the reagents A, B, C injected into the reaction chamber. If u denotes the concentration of D produced by the first reaction, and v that

[†] A modern proof of this fact relies on Galois theory, [**19**].

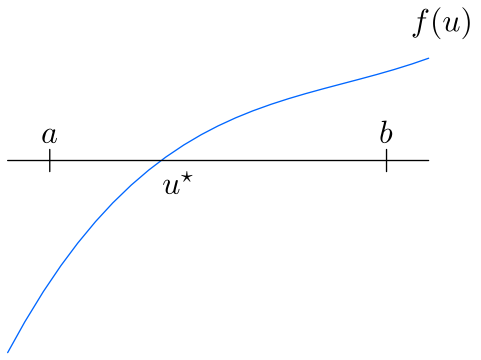


Figure 2.9. Intermediate Value Theorem.

by the second reaction, then the final equilibrium concentrations

$$a_\star = a - 2u - v, \qquad b_\star = b - u, \qquad c_\star = c - 3v, \qquad d_\star = u + v,$$

of the reagents will be determined by solving the nonlinear system

$$(a - 2u - v)^2 (b - u) = \alpha (u + v), \qquad (a - 2u - v)(c - 3v)^3 = \beta (u + v), \tag{2.30}$$

where α, β are the known equilibrium constants of the two reactions.

Our immediate goal is to develop numerical algorithms for solving such nonlinear scalar equations.

The Bisection Method

The most primitive algorithm, and *the only one that is guaranteed to work in all cases*, is the Bisection Method. While it has an iterative flavor, it cannot be properly classed as a method governed by functional iteration as defined in the preceding section, and so must be studied directly in its own right.

The starting point is the Intermediate Value Theorem, which we state in simplified form. See Figure 2.9 for an illustration, and [**2**] for a proof.

Lemma 2.12. *Let $f(u)$ be a continuous scalar function. Suppose we can find two points $a < b$ where the values of $f(a)$ and $f(b)$ take* opposite *signs, so either $f(a) < 0$ and $f(b) > 0$, or $f(a) > 0$ and $f(b) < 0$. Then there exists at least one point $a < u^\star < b$ where $f(u^\star) = 0$.*

Note that if $f(a) = 0$ or $f(b) = 0$, then finding a root is trivial. If $f(a)$ and $f(b)$ have the same sign, then there may or may not be a root in between. Figure 2.10 plots the functions $u^2 + 1$, u^2 and $u^2 - 1$, on the interval $-2 \leq u \leq 2$. The first has two simple roots; the second has a single double root, while the third has no root. We also note that continuity of the function on the entire interval $[a, b]$ is an essential hypothesis. For example, the function $f(u) = 1/u$ satisfies $f(-1) = -1$ and $f(1) = 1$, but there is no root to the equation $1/u = 0$.

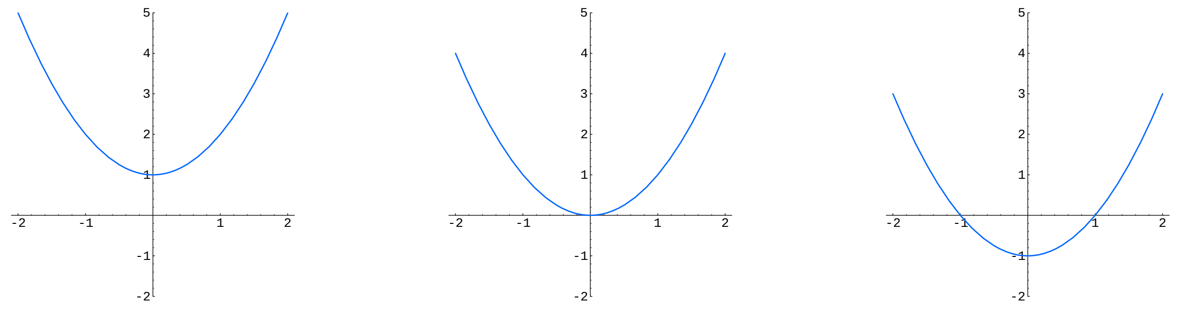


Figure 2.10. Roots of Quadratic Functions.

Note carefully that the Lemma 2.12 does *not* say there is a unique root between a and b. There may be many roots, or even, in pathological examples, infinitely many. All the theorem guarantees is that, under the stated hypotheses, there is at least one root.

Once we are assured that a root exists, bisection relies on a "divide and conquer" strategy. The goal is to locate a root $a < u^\star < b$ between the endpoints. Lacking any additional evidence, one tactic would be to try the midpoint $c = \frac{1}{2}(a+b)$ as a first guess for the root. If, by some miracle, $f(c) = 0$, then we are done, since we have found a solution! Otherwise (and typically) we look at the sign of $f(c)$. There are two possibilities. If $f(a)$ and $f(c)$ are of opposite signs, then the Intermediate Value Theorem tells us that there is a root $u^\star$ lying between $a < u^\star < c$. Otherwise, $f(c)$ and $f(b)$ must have opposite signs, and so there is a root $c < u^\star < b$. In either event, we apply the same method to the interval in which we are assured a root lies, and repeat the procedure. Each iteration halves the length of the interval, and chooses the half in which a root is sure to lie. (There may, of course, be a root in the other half interval, but as we cannot be sure, we discard it from further consideration.) The root we home in on lies trapped in intervals of smaller and smaller width, and so convergence of the method is guaranteed.

Example 2.13. The roots of the quadratic equation

$$f(u) = u^2 + u - 3 = 0$$

can be computed exactly by the quadratic formula:

$$u_1^\star = \frac{-1+\sqrt{13}}{2} \approx 1.302775\ldots\,, \qquad u_2^\star = \frac{-1-\sqrt{13}}{2} \approx -2.302775\ldots\,.$$

Let us see how one might approximate them by applying the Bisection Algorithm. We start the procedure by choosing the points $a = u^{(0)} = 1,\ \ b = v^{(0)} = 2$, noting that $f(1) = -1$ and $f(2) = 3$ have opposite signs and hence we are guaranteed that there is at least one root between 1 and 2. In the first step we look at the midpoint of the interval $[1,2]$, which is 1.5, and evaluate $f(1.5) = .75$. Since $f(1) = -1$ and $f(1.5) = .75$ have opposite signs, we know that there is a root lying between 1 and 1.5. Thus, we take $u^{(1)} = 1$ and $v^{(1)} = 1.5$ as the endpoints of the next interval, and continue. The next midpoint is at 1.25, where $f(1.25) = -.1875$ has the opposite sign to $f(1.5) = .75$, and so a root lies between $u^{(2)} = 1.25$ and $v^{(2)} = 1.5$. The process is then iterated as long as desired — or, more practically, as long as your computer's precision does not become an issue.

k	$u^{(k)}$	$v^{(k)}$	$w^{(k)} = \frac{1}{2}(u^{(k)} + v^{(k)})$	$f(w^{(k)})$
0	1	2	1.5	.75
1	1	1.5	1.25	−.1875
2	1.25	1.5	1.375	.2656
3	1.25	1.375	1.3125	.0352
4	1.25	1.3125	1.2813	−.0771
5	1.2813	1.3125	1.2969	−.0212
6	1.2969	1.3125	1.3047	.0069
7	1.2969	1.3047	1.3008	−.0072
8	1.3008	1.3047	1.3027	−.0002
9	1.3027	1.3047	1.3037	.0034
10	1.3027	1.3037	1.3032	.0016
11	1.3027	1.3032	1.3030	.0007
12	1.3027	1.3030	1.3029	.0003
13	1.3027	1.3029	1.3028	.0001
14	1.3027	1.3028	1.3028	−.0000

The table displays the result of the algorithm, rounded off to four decimal places. After 14 iterations, the Bisection Method has correctly computed the first four decimal digits of the positive root $u_1^\star$. A similar bisection starting with the interval from $u^{(1)} = -3$ to $v^{(1)} = -2$ will produce the negative root.

A formal implementation of the Bisection Algorithm appears in the accompanying pseudocode program. The endpoints of the kth interval are denoted by $u^{(k)}$ and $v^{(k)}$. The midpoint is $w^{(k)} = \frac{1}{2}\big(u^{(k)} + v^{(k)}\big)$, and the key decision is whether $w^{(k)}$ should be the right or left hand endpoint of the next interval. The integer n, governing the number of iterations, is to be prescribed in accordance with how accurately we wish to approximate the root $u^\star$.

The algorithm produces two sequences of approximations $u^{(k)}$ and $v^{(k)}$ that both converge monotonically to $u^\star$, one from below and the other from above:

$$a = u^{(0)} \le u^{(1)} \le u^{(2)} \le \;\cdots\; \le u^{(k)} \;\longrightarrow\; u^\star \;\longleftarrow\; v^{(k)} \le \;\cdots\; \le v^{(2)} \le v^{(1)} \le v^{(0)} = b.$$

and $u^\star$ is trapped between the two. Thus, the root is trapped inside a sequence of intervals $[u^{(k)}, v^{(k)}]$ of progressively shorter and shorter length. Indeed, the length of each interval is exactly half that of its predecessor:

$$v^{(k)} - u^{(k)} = \tfrac{1}{2}\,(v^{(k-1)} - u^{(k-1)}).$$

Iterating this formula, we conclude that

$$v^{(n)} - u^{(n)} = \left(\tfrac{1}{2}\right)^n (v^{(0)} - u^{(0)}) = \left(\tfrac{1}{2}\right)^n (b - a) \;\longrightarrow\; 0 \qquad \text{as} \qquad n \;\longrightarrow\; \infty.$$

The Bisection Method

```
start
    if $f(a)\,f(b) < 0$ set $u^{(0)} = a$, $v^{(0)} = b$
      else print “Bisection Method not applicable”
    for $k = 0$ to $n - 1$
        set $w^{(k)} = \frac{1}{2}(u^{(k)} + v^{(k)})$
        if $f(w^{(k)}) = 0$, stop; print $u^\star = w^{(k)}$
        if $f(u^{(k)})\,f(w^{(k)}) < 0$, set $u^{(k+1)} = u^{(k)}$, $v^{(k+1)} = w^{(k)}$
                        else    set $u^{(k+1)} = w^{(k)}$, $v^{(k+1)} = v^{(k)}$
    next $k$
    print $u^\star = w^{(n)} = \frac{1}{2}(u^{(n)} + v^{(n)})$
end
```

The midpoint

$$w^{(n)} = \tfrac{1}{2}\,(u^{(n)} + v^{(n)})$$

lies within a distance

$$|\, w^{(n)} - u^\star \,| \le \tfrac{1}{2}\,(v^{(n)} - u^{(n)}) = \left(\tfrac{1}{2}\right)^{n+1}(b-a)$$

of the root. Consequently, if we desire to approximate the root within a prescribed tolerance ε, we should choose the number of iterations n so that

$$\left(\tfrac{1}{2}\right)^{n+1}(b-a) < \varepsilon, \qquad \text{or} \qquad n > \log_2 \frac{b-a}{\varepsilon} - 1\,. \tag{2.31}$$

Summarizing:

Theorem 2.14. *If $f(u)$ is a continuous function, with $f(a)\,f(b) < 0$, then the Bisection Method starting with $u^{(0)} = a$, $v^{(0)} = b$, will converge to a solution $u^\star$ to the equation $f(u) = 0$ lying between a and b. After n steps, the midpoint $w^{(n)} = \frac{1}{2}\,(u^{(n)} + v^{(n)})$ will be within a distance of $\varepsilon = 2^{-n-1}(b-a)$ from the solution.*

For example, in the case of the quadratic equation in Example 2.13, after 14 iterations, we have approximated the positive root to within

$$\varepsilon = \left(\tfrac{1}{2}\right)^{15}(2-1) \;\approx\; 3.052 \times 10^{-5},$$

reconfirming our observation that we have accurately computed its first four decimal places. If we are in need of 10 decimal places, we set our tolerance to $\varepsilon = .5 \times 10^{-10}$, and so, according to (2.31), must perform $n = 34 > 33.22 \approx \log_2 2 \times 10^{10} - 1$ successive bisections[†].

[†] This assumes we have sufficient precision on the computer to avoid round-off errors.

Example 2.15. As noted at the beginning of this section, the quintic equation

$$f(u) = u^5 + u + 1 = 0$$

has one real root, whose value can be readily computed by bisection. We start the algorithm with the initial points $u^{(0)} = -1,\ v^{(0)} = 0$, noting that $f(-1) = -1 < 0$ while $f(0) = 1 > 0$ are of opposite signs. In order to compute the root to 6 decimal places, we set $\varepsilon = .5 \times 10^{-6}$ in (2.31), and so need to perform $n = 20 > 19.93 \approx \log_2 2 \times 10^6 - 1$ bisections. Indeed, the algorithm produces the approximation $u^\star \approx -.754878$ to the root, and the displayed digits are guaranteed to be accurate.

Fixed Point Methods

The Bisection Method has an ironclad guarantee to converge to a root of the function — provided it can be properly started by locating two points where the function takes opposite signs. This may be tricky if the function has two very closely spaced roots and is, say, negative only for a very small interval between them, and may be impossible for multiple roots, e.g., the root $u^\star = 0$ of the quadratic function $f(u) = u^2$. When applicable, its convergence rate is completely predictable, but not especially fast. Worse, it has no immediately apparent extension to systems of equations, since there is *no* obvious counterpart to the Intermediate Value Theorem for vector-valued functions.

Most other numerical schemes for solving equations rely on some form of fixed point iteration. Thus, we seek to replace the system of equations $\mathbf{f}(\mathbf{u}) = \mathbf{0}$ with a fixed point system $\mathbf{u} = \mathbf{g}(\mathbf{u})$, that leads to the iterative solution scheme $\mathbf{u}^{(k+1)} = \mathbf{g}(\mathbf{u}^{(k)})$. For this to work, there are two key requirements:

(*a*) The solution $\mathbf{u}^\star$ to the equation $\mathbf{f}(\mathbf{u}) = \mathbf{0}$ is also a fixed point for $\mathbf{g}(\mathbf{u})$, and

(*b*) $\mathbf{u}^\star$ is, in fact a stable fixed point, meaning that the Jacobian $\mathbf{g}'(\mathbf{u}^\star)$ is a convergent matrix, or, slightly more restrictively, $\|\, \mathbf{g}'(\mathbf{u}^\star) \,\| < 1$ for a prescribed matrix norm.

If both conditions hold, then, *provided we choose the initial iterate* $\mathbf{u}^{(0)} = \mathbf{c}$ *sufficiently close to* $\mathbf{u}^\star$, the iterates $\mathbf{u}^{(k)} \to \mathbf{u}^\star$ will converge to the desired solution. Thus, the key to the practical use of functional iteration for solving equations is the proper design of an iterative system — coupled with a reasonably good initial guess for the solution. Before implementing general procedures, let us discuss a naïve example.

Example 2.16. To solve the cubic equation

$$f(u) = u^3 - u - 1 = 0 \tag{2.32}$$

we note that $f(1) = -1$ while $f(2) = 5$, and so there is a root between 1 and 2. Indeed, the Bisection Method leads to the approximate value $u^\star \approx 1.3247$ after 17 iterations.

Let us try to find the same root by fixed point iteration. As a first, naïve, guess, we rewrite the cubic equation in fixed point form

$$u = u^3 - 1 = \widetilde{g}(u).$$

Starting with the initial guess $u^{(0)} = 1.5$, successive approximations to the solution are found by iterating

$$u^{(k+1)} = \widetilde{g}(u^{(k)}) = (u^{(k)})^3 - 1, \qquad k = 0, 1, 2, \ldots .$$

However, their values

$$\begin{aligned} &u^{(0)} = 1.5, && u^{(1)} = 2.375, && u^{(2)} = 12.396, \\ &u^{(3)} = 1904, && u^{(4)} = 6.9024 \times 10^{9}, && u^{(5)} = 3.2886 \times 10^{29}, \quad \ldots \end{aligned}$$

rapidly become unbounded, and so fail to converge. This could, in fact, have been predicted by the convergence criterion in Theorem 2.6. Indeed, $\widetilde{g}'(u) = -3u^2$ and so $|\,\widetilde{g}'(u)\,| > 3$ for all $u \geq 1$, including the root $u^\star$. This means that $u^\star$ is an unstable fixed point, and the iterates cannot converge to it.

On the other hand, we can rewrite the equation (2.32) in the alternative iterative form

$$u = \sqrt[3]{1+u} = g(u).$$

In this case

$$0 \;\leq\; g'(u) \;=\; \frac{1}{3(1+u)^{2/3}} \;\leq\; \frac{1}{3} \qquad \text{for} \qquad u > 0.$$

Thus, the stability condition (2.17) is satisfied, and we anticipate convergence at a rate of at least $\frac{1}{3}$. (The Bisection Method converges more slowly, at rate $\frac{1}{2}$.) Indeed, the first few iterates $u^{(k+1)} = \sqrt[3]{1+u^{(k)}}$ are

$$1.5, \qquad 1.35721, \qquad 1.33086, \qquad 1.32588, \qquad 1.32494, \qquad 1.32476, \qquad 1.32473,$$

and we have converged to the root, correct to four decimal places, in only 6 iterations.

Newton's Method

Our immediate goal is to design an efficient iterative scheme $u^{(k+1)} = g(u^{(k)})$ whose iterates converge rapidly to the solution of the given scalar equation $f(u) = 0$. As we learned in Section 2.1, the convergence of the iteration is governed by the magnitude of its derivative at the fixed point. At the very least, we should impose the stability criterion $|\,g'(u^\star)\,| < 1$, and the smaller this quantity can be made, the faster the iterative scheme converges. if we are able to arrange that $g'(u^\star) = 0$, then the iterates will converge quadratically fast, leading, as noted in the discussion following Theorem 2.10, to a dramatic improvement in speed and efficiency.

Now, the first condition requires that $g(u) = u$ whenever $f(u) = 0$. A little thought will convince you that the iterative function should take the form

$$g(u) = u - h(u)\, f(u), \tag{2.33}$$

where $h(u)$ is a reasonably nice function. If $f(u^\star) = 0$, then clearly $u^\star = g(u^\star)$, and so $u^\star$ is a fixed point. The converse holds provided $h(u) \neq 0$ is never zero.

For quadratic convergence, the key requirement is that the derivative of $g(u)$ be zero at the fixed point solutions. We compute

$$g'(u) = 1 - h'(u)\, f(u) - h(u)\, f'(u).$$

Thus, $g'(u^\star) = 0$ at a solution to $f(u^\star) = 0$ if and only if

$$0 = 1 - h'(u^\star)\, f(u^\star) - h(u^\star)\, f'(u^\star) = 1 - h(u^\star)\, f'(u^\star).$$

Consequently, we should require that

$$h(u^\star) = \frac{1}{f'(u^\star)} \tag{2.34}$$

to ensure a quadratically convergent iterative scheme. This assumes that $f'(u^\star) \neq 0$, which means that $u^\star$ is a *simple root* of f. For here on, we leave aside multiple roots, which require a different approach.

Of course, there are many functions $h(u)$ that satisfy (2.34), since we only need to specify its value at a single point. The problem is that we do not know $u^\star$ — after all this is what we are trying to compute — and so cannot compute the value of the derivative of f there. However, we can circumvent this apparent difficulty by a simple device: we impose equation (2.34) at all points, setting

$$h(u) = \frac{1}{f'(u)}\,, \tag{2.35}$$

which certainly guarantees that it holds at the solution $u^\star$. The result is the function

$$g(u) = u \, - \, \frac{f(u)}{f'(u)}\,, \tag{2.36}$$

and the resulting iteration scheme is known as *Newton's Method*, which, as the name suggests, dates back to the founder of the calculus. To this day, Newton's Method remains *the* most important general purpose algorithm for solving equations. It starts with an initial guess $u^{(0)}$ to be supplied by the user, and then successively computes

$$u^{(k+1)} = u^{(k)} - \frac{f(u^{(k)})}{f'(u^{(k)})}\,. \tag{2.37}$$

As long as the initial guess is sufficiently close, the iterates $u^{(k)}$ are guaranteed to converge, quadratically fast, to the (simple) root $u^\star$ of the equation $f(u) = 0$.

Theorem 2.17. *Suppose $f(u) \in \mathrm{C}^2$ is twice continuously differentiable. Let $u^\star$ be a solution to the equation $f(u^\star) = 0$ such that $f'(u^\star) \neq 0$. Given an initial guess $u^{(0)}$ sufficiently close to $u^\star$, the Newton iteration scheme* (2.37) *converges at a quadratic rate to the solution $u^\star$.*

Proof: By continuity, if $f'(u^\star) \neq 0$, then $f'(u) \neq 0$ for all u sufficiently close to $u^\star$, and hence the Newton iterative function (2.36) is well defined and continuously differentiable near $u^\star$. Since $g'(u) = f(u)\, f''(u)/f'(u)^2$, we have $g'(u^\star) = 0$ when $f(u^\star) = 0$, as promised by our construction. The quadratic convergence of the resulting iterative scheme is an immediate consequence of Theorem 2.10. *Q.E.D.*

Example 2.18. Consider the cubic equation

$$f(u) = u^3 - u - 1 = 0,$$

that we already solved in Example 2.16. The function used in the Newton iteration is

$$g(u) = u - \frac{f(u)}{f'(u)} = u - \frac{u^3 - u - 1}{3\,u^2 - 1}\,,$$

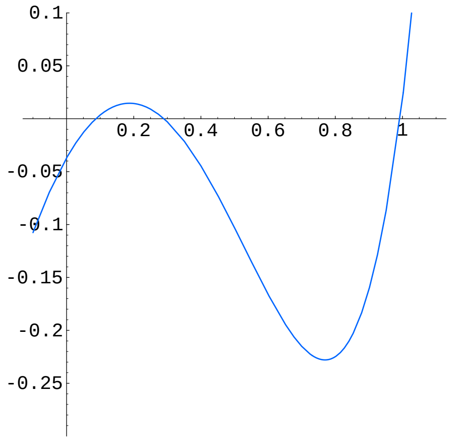


Figure 2.11. The function $f(u) = u^3 - \frac{3}{2}u^2 + \frac{5}{9}u - \frac{1}{27}$.

which is well-defined as long as $u \neq \pm \frac{1}{\sqrt{3}}$. We will try to avoid these singular points. The iterative procedure

$$u^{(k+1)} = g(u^{(k)}) = u^{(k)} - \frac{(u^{(k)})^3 - u^{(k)} - 1}{3\,(u^{(k)})^2 - 1}$$

with initial guess $u^{(0)} = 1.5$ produces the following values:

$$1.5, \qquad 1.34783, \qquad 1.32520, \qquad 1.32472,$$

and we have computed the root to 5 decimal places after only three iterations. The quadratic convergence of Newton's Method implies that, roughly, each new iterate doubles the number of correct decimal places. Thus, to compute the root accurately to 40 decimal places would only require 3 further iterations[†]. This underscores the tremendous advantage that the Newton algorithm offers over competing methods.

Example 2.19. Consider the cubic polynomial equation

$$f(u) = u^3 - \tfrac{3}{2}u^2 + \tfrac{5}{9}u - \tfrac{1}{27} = 0.$$

Since

$$f(0) = -\tfrac{1}{27}\,, \qquad f\left(\tfrac{1}{3}\right) = \tfrac{1}{54}\,, \qquad f\left(\tfrac{2}{3}\right) = -\tfrac{1}{27}\,, \qquad f(1) = \tfrac{1}{54}\,,$$

the Intermediate Value Lemma 2.12 guarantees that there are three roots on the interval $[0,1]$: one between 0 and $\frac{1}{3}$, the second between $\frac{1}{3}$ and $\frac{2}{3}$, and the third between $\frac{2}{3}$ and 1. The graph in Figure 2.11 reconfirms this observation. Since we are dealing with a cubic polynomial, there are no other roots. (Why?)

[†] This assumes we are working in a sufficiently high precision arithmetic so as to avoid round-off errors.

It takes sixteen iterations of the Bisection Method starting with the three subintervals $\left[0, \frac{1}{3}\right]$, $\left[\frac{1}{3}, \frac{2}{3}\right]$ and $\left[\frac{2}{3}, 1\right]$, to produce the roots to six decimal places:

$$u_1^\star \approx .085119, \qquad u_2^\star \approx .451805, \qquad u_3^\star \approx .963076.$$

Incidentally, if we start with the interval $[0,1]$ and apply bisection, we converge (perhaps surprisingly) to the largest root $u_3^\star$ in 17 iterations.

Fixed point iteration based on the formulation

$$u = g(u) = -u^3 + \tfrac{3}{2}u^2 + \tfrac{4}{9}u + \tfrac{1}{27}$$

can be used to find the first and third roots, but not the second root. For instance, starting with $u^{(0)} = 0$ produces $u_1^\star$ to 5 decimal places after 23 iterations, whereas starting with $u^{(0)} = 1$ produces $u_3^\star$ to 5 decimal places after 14 iterations. The reason we cannot produce $u_2^\star$ is due to the magnitude of the derivative

$$g'(u) = -3u^2 + 3u + \tfrac{4}{9}$$

at the roots, which is

$$g'(u_1^\star) \approx 0.678065, \qquad g'(u_2^\star) \approx 1.18748, \qquad g'(u_3^\star) \approx 0.551126.$$

Thus, $u_1^\star$ and $u_3^\star$ are stable fixed points, but $u_2^\star$ is unstable. However, because $g'(u_1^\star)$ and $g'(u_3^\star)$ are both bigger than .5, this iterative algorithm actually converges *slower* than ordinary bisection!

Finally, Newton's Method is based upon iteration of the rational function

$$g(u) = u - \frac{f(u)}{f'(u)} = u - \frac{u^3 - \frac{3}{2}u^2 + \frac{5}{9}u - \frac{1}{27}}{3u^2 - 3u + \frac{5}{9}}\,.$$

Starting with an initial guess of $u^{(0)} = 0$, the method computes $u_1^\star$ to 6 decimal places after only 4 iterations; starting with $u^{(0)} = .5$, it produces $u_2^\star$ to similar accuracy after 2 iterations; while starting with $u^{(0)} = 1$ produces $u_3^\star$ after 3 iterations — a dramatic speed up over the other two methods.

Newton's Method has a very pretty graphical interpretation, that helps us understand what is going on and why it converges so fast. Given the equation $f(u) = 0$, suppose we know an approximate value $u = u^{(k)}$ for a solution. Nearby $u^{(k)}$, we can approximate the nonlinear function $f(u)$ by its tangent line

$$y = f(u^{(k)}) + f'(u^{(k)})(u - u^{(k)}). \tag{2.38}$$

As long as the tangent line is not horizontal — which requires $f'(u^{(k)}) \neq 0$ — it crosses the axis at

$$u^{(k+1)} = u^{(k)} - \frac{f(u^{(k)})}{f'(u^{(k)})}\,,$$

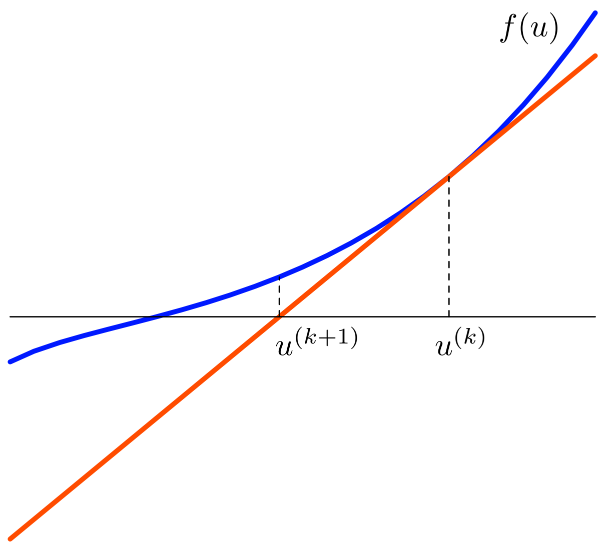


Figure 2.12. Newton's Method.

which represents a new, and, presumably more accurate, approximation to the desired root. The procedure is illustrated pictorially in Figure 2.12. Note that the passage from $u^{(k)}$ to $u^{(k+1)}$ is exactly the Newton iteration step (2.37). Thus, Newtonian iteration is the same as the approximation of function's root by those of its successive tangent lines.

Given a sufficiently accurate initial guess, Newton's Method will rapidly produce highly accurate values for the simple roots to the equation in question. In practice, barring some kind of special exploitable structure, Newton's Method is the root-finding algorithm of choice. The one caveat is that we need to start the process reasonably close to the root we are seeking. Otherwise, there is no guarantee that a particular set of iterates will converge, although if they do, the limiting value is necessarily a root of our equation. The behavior of Newton's Method as we change parameters and vary the initial guess is very similar to the simpler logistic map that we studied in Section 2.1, including period doubling bifurcations and chaotic behavior. The reader is invited to experiment with simple examples; further details can be found in [**42**].

Example 2.20. For fixed values of the eccentricity ϵ, Kepler's equation

$$u - \epsilon \sin u = m \tag{2.39}$$

can be viewed as a implicit equation defining the eccentric anomaly u as a function of the mean anomaly m. To solve Kepler's equation by Newton's Method, we introduce the iterative function

$$g(u) \;=\; u \;-\; \frac{u - \epsilon \sin u - m}{1 - \epsilon \cos u}\,.$$

Notice that when $|\,\epsilon\,| < 1$, the denominator never vanishes and so the iteration remains well-defined everywhere. Starting with a sufficiently close initial guess $u^{(0)}$, we are assured that the method will quickly converge to the solution.

Fixing the eccentricity ϵ, we can employ tghe method of *continuation* to determine how the solution $u^{\star} = h(m)$ depends upon the mean anomaly m. Namely, we start at

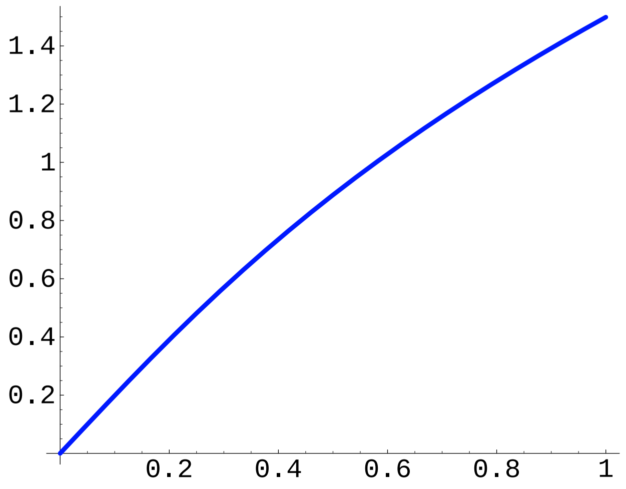


Figure 2.13. The Solution to the Kepler Equation for Eccentricity $\epsilon = .5$.

$m = m_0 = 0$ with the obvious solution $u^\star = h(0) = 0$. Then, to compute the solution at successive closely spaced values $0 < m_1 < m_2 < m_3 < \cdots$, we use the previously computed value as an initial guess $u^{(0)} = h(m_k)$ for the value of the solution at the next mesh point m_{k+1}, and run the Newton scheme until it converges to a sufficiently accurate approximation to the value $u^\star = h(m_{k+1})$. As long as m_{k+1} is reasonably close to m_k, Newton's Method will converge to the solution quite quickly.

The continuation method will quickly produce the values of u at the sample points. Intermediate values can either be determined by an interpolation scheme, e.g., a cubic spline fit of the data, or by running the Newton scheme using the closest known value as an initial condition. A plot for $0 \le m \le 1$ using the value $\epsilon = .5$ appears in Figure 2.13.

AIMS Lecture Notes 2006

Peter J. Olver

3. Review of Matrix Algebra

Vectors and matrices are essential for modern analysis of systems of equations — algebrai, differential, functional, etc. In this part, we will review the most basic facts of matrix arithmetic. See [**38**] for full details.

3.1. Matrices and Vectors.

A *matrix* is a rectangular array of numbers. Thus,

$$\begin{pmatrix} 1 & 0 & 3 \\ -2 & 4 & 1 \end{pmatrix}, \qquad \begin{pmatrix} \pi & 0 \\ e & \frac{1}{2} \\ -1 & .83 \\ \sqrt{5} & -\frac{4}{7} \end{pmatrix}, \qquad (\,.2 \quad -1.6 \quad .32\,), \qquad \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \begin{pmatrix} 1 & 3 \\ -2 & 5 \end{pmatrix},$$

are all examples of matrices. We use the notation

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix} \tag{3.1}$$

for a general matrix of size $m \times n$ (read "m by n"), where m denotes the number of *rows* in A and n denotes the number of *columns*. Thus, the preceding examples of matrices have respective sizes 2×3, 4×2, 1×3, 2×1, and 2×2. A matrix is *square* if $m = n$, i.e., it has the same number of rows as columns. A *column vector* is a $m \times 1$ matrix, while a *row vector* is a $1 \times n$ matrix. As we shall see, column vectors are by far the more important of the two, and the term "vector" without qualification will always mean "column vector". A 1×1 matrix, which has but a single entry, is both a row and a column vector.

The number that lies in the i^{th} row and the j^{th} column of A is called the (i,j) *entry* of A, and is denoted by a_{ij}. The row index always appears first and the column index second. Two matrices are equal, $A = B$, if and only if they have the same size, and *all* their entries are the same: $a_{ij} = b_{ij}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$.

A general linear system of m equations in n unknowns will take the form

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \;\cdots\; + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \;\cdots\; + a_{2n}x_n &= b_2, \\ \vdots \qquad\qquad \vdots \qquad\qquad & \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \;\cdots\; + a_{mn}x_n &= b_m. \end{aligned} \tag{3.2}$$

As such, it is composed of three basic ingredients: the $m \times n$ *coefficient matrix* A, with entries a_{ij} as in (3.1), the column vector $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ containing the *unknowns*, and the column vector $\mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$ containing *right hand sides*. As an example, consider the linear system

$$\begin{aligned} x + 2y + z &= 2, \\ 2y + z &= 7, \\ x + y + 4z &= 3, \end{aligned}$$

The coefficient matrix $A = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & 1 \\ 1 & 1 & 4 \end{pmatrix}$ can be filled in, entry by entry, from the coefficients of the variables appearing in the equations. (Don't forget to put a zero when a avariable doesn't appear in an equation!) The vector $\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ lists the variables, while the entries of $\mathbf{b} = \begin{pmatrix} 2 \\ 7 \\ 3 \end{pmatrix}$ are the right hand sides of the equations.

Remark: We will consistently use bold face lower case letters to denote vectors, and ordinary capital letters to denote general matrices.

Matrix Arithmetic

Matrix arithmetic involves three basic operations: *matrix addition*, *scalar multiplication*, and *matrix multiplication*. First we define *addition* of matrices. You are only allowed to add two matrices of the *same size*, and matrix addition is performed entry by entry. For example,

$$\begin{pmatrix} 1 & 2 \\ -1 & 0 \end{pmatrix} + \begin{pmatrix} 3 & -5 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 4 & -3 \\ 1 & 1 \end{pmatrix}.$$

Therefore, if A and B are $m \times n$ matrices, their sum $C = A + B$ is the $m \times n$ matrix whose entries are given by $c_{ij} = a_{ij} + b_{ij}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. When defined, matrix addition is commutative, $A + B = B + A$, and associative, $A + (B + C) = (A + B) + C$, just like ordinary addition.

A *scalar* is a fancy name for an ordinary number — the term merely distinguishes it from a vector or a matrix. For the time being, we will restrict our attention to real scalars and matrices with real entries, but eventually complex scalars and complex matrices must be dealt with. We will consistently identify a scalar $c \in \mathbb{R}$ with the 1×1 matrix (c) in which it is the sole entry, and so will omit the redundant parentheses in the latter case. *Scalar multiplication* takes a scalar c and an $m \times n$ matrix A and computes the $m \times n$ matrix $B = c\,A$ by multiplying each entry of A by c. For example,

$$3\begin{pmatrix} 1 & 2 \\ -1 & 0 \end{pmatrix} = \begin{pmatrix} 3 & 6 \\ -3 & 0 \end{pmatrix}.$$

In general, $b_{ij} = c\,a_{ij}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. Basic properties of scalar multiplication are summarized at the end of this section.

Finally, we define *matrix multiplication*. First, the product between a row vector $\mathbf{a}$ and a column vector $\mathbf{x}$ having the *same* number of entries is the *scalar* or 1×1 matrix defined by the following rule:

$$\mathbf{a}\,\mathbf{x} = (\,a_1\ a_2\ \ldots\ a_n\,)\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = a_1 x_1 + a_2 x_2 + \ \cdots\ + a_n x_n = \sum_{k=1}^{n} a_k\,x_k. \tag{3.3}$$

More generally, if A is an $m \times n$ matrix and B is an $n \times p$ matrix, so that the number of *columns* in A equals the number of *rows* in B, then the matrix product $C = A\,B$ is defined as the $m \times p$ matrix whose (i, j) entry equals the vector product of the i^{th} row of A and the j^{th} column of B. Therefore,

$$c_{ij} = \sum_{k=1}^{n} a_{ik}\,b_{kj}. \tag{3.4}$$

Note that our restriction on the sizes of A and B guarantees that the relevant row and column vectors will have the same number of entries, and so their product is defined.

For example, the product of the coefficient matrix A and vector of unknowns $\mathbf{x}$ for our original system (4.1) is given by

$$A\,\mathbf{x} = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 4 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x + 2y + z \\ 2x + 6y + z \\ x + y + 4z \end{pmatrix}.$$

The result is a column vector whose entries reproduce the left hand sides of the original linear system! As a result, we can rewrite the system

$$A\,\mathbf{x} = \mathbf{b} \tag{3.5}$$

as an equality between two column vectors. This result is general; a linear system (3.2) consisting of m equations in n unknowns can be written in the matrix form (3.5) where A

is the $m \times n$ coefficient matrix (3.1), $\mathbf{x}$ is the $n \times 1$ column vector of unknowns, and $\mathbf{b}$ is the $m \times 1$ column vector containing the right hand sides. This is one of the principal reasons for the non-evident definition of matrix multiplication. Component-wise multiplication of matrix entries turns out to be almost completely useless in applications.

Now, the bad news. Matrix multiplication is *not* commutative — that is, BA is not necessarily equal to AB. For example, BA may not be defined even when AB is. Even if both are defined, they may be different sized matrices. For example the product $s = \mathbf{r}\,\mathbf{c}$ of a row vector $\mathbf{r}$, a $1 \times n$ matrix, and a column vector $\mathbf{c}$, an $n \times 1$ matrix with the same number of entries, is a 1×1 matrix or scalar, whereas the reversed product $C = \mathbf{c}\,\mathbf{r}$ is an $n \times n$ matrix. For instance,

$$\begin{pmatrix} 1 & 2 \end{pmatrix}\begin{pmatrix} 3 \\ 0 \end{pmatrix} = 3, \qquad \text{whereas} \qquad \begin{pmatrix} 3 \\ 0 \end{pmatrix}\begin{pmatrix} 1 & 2 \end{pmatrix} = \begin{pmatrix} 3 & 6 \\ 0 & 0 \end{pmatrix}.$$

In computing the latter product, don't forget that we multiply the *rows* of the first matrix by the *columns* of the second. Moreover, even if the matrix products AB and BA have the same size, which requires both A and B to be square matrices, we may still have $AB \neq BA$. For example,

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ -1 & 2 \end{pmatrix} = \begin{pmatrix} -2 & 5 \\ -4 & 11 \end{pmatrix} \neq \begin{pmatrix} 3 & 4 \\ 5 & 6 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 2 \end{pmatrix}\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}.$$

On the other hand, matrix multiplication is associative, so $A(BC) = (AB)C$ whenever A has size $m \times n$, B has size $n \times p$, and C has size $p \times q$; the result is a matrix of size $m \times q$. The proof of associativity is a tedious computation based on the definition of matrix multiplication that, for brevity, we omit. Consequently, the one difference between matrix algebra and ordinary algebra is that you need to be careful not to change the order of multiplicative factors without proper justification.

Since matrix multiplication acts by multiplying rows by columns, one can compute the columns in a matrix product AB by multiplying the matrix A and the individual columns of B. For example, the two columns of the matrix product

$$\begin{pmatrix} 1 & -1 & 2 \\ 2 & 0 & -2 \end{pmatrix}\begin{pmatrix} 3 & 4 \\ 0 & 2 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 4 \\ 8 & 6 \end{pmatrix}$$

are obtained by multiplying the first matrix with the individual columns of the second:

$$\begin{pmatrix} 1 & -1 & 2 \\ 2 & 0 & -2 \end{pmatrix}\begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 8 \end{pmatrix}, \qquad \begin{pmatrix} 1 & -1 & 2 \\ 2 & 0 & -2 \end{pmatrix}\begin{pmatrix} 4 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \end{pmatrix}.$$

In general, if we use $\mathbf{b}_k$ to denote the k^{th} column of B, then

$$AB = A\begin{pmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \ldots & \mathbf{b}_p \end{pmatrix} = \begin{pmatrix} A\mathbf{b}_1 & A\mathbf{b}_2 & \ldots & A\mathbf{b}_p \end{pmatrix}, \tag{3.6}$$

indicating that the k^{th} column of their matrix product is $A\mathbf{b}_k$.

There are two special matrices. The first is the *zero matrix*, all of whose entries are 0. We use $\mathrm{O}_{m\times n}$ to denote the $m \times n$ zero matrix, often written as just O if the size is clear from the context. The zero matrix is the additive unit, so $A + \mathrm{O} = A = \mathrm{O} + A$ when O has the same size as A. In particular, we will use a bold face $\mathbf{0}$ to denote a column vector with all zero entries.

The role of the multiplicative unit is played by the square *identity matrix*

$$\mathrm{I} = \mathrm{I}_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$$

of size $n \times n$. The entries along the *main diagonal* (which runs from top left to bottom right) are equal to 1, while the *off-diagonal* entries are all 0. As you can check, if A is any $m \times n$ matrix, then $\mathrm{I}_m A = A = A\,\mathrm{I}_n$. We will sometimes write the preceding equation as just $\mathrm{I}\,A = A = A\,\mathrm{I}$, since each matrix product is well-defined for exactly one size of identity matrix.

The identity matrix is a particular example of a *diagonal matrix*. In general, a square matrix A is diagonal if all its off-diagonal entries are zero: $a_{ij} = 0$ for all $i \neq j$. We will sometimes write $D = \operatorname{diag}(c_1, \ldots, c_n)$ for the $n \times n$ diagonal matrix with diagonal entries $d_{ii} = c_i$. Thus, $\operatorname{diag}(1, 3, 0)$ refers to the diagonal matrix $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, while the $n \times n$ identity matrix can be written as $\mathrm{I}_n = \operatorname{diag}(1, 1, \ldots, 1)$.

Let us conclude this section by summarizing the basic properties of matrix arithmetic. In the accompanying table, A, B, C are matrices; c, d are scalars; O is a zero matrix; and I is an identity matrix. All matrices are assumed to have the correct sizes so that the indicated operations are defined.

Matrix Inverses

The inverse of a matrix is analogous to the reciprocal $a^{-1} = 1/a$ of a scalar. We already encountered the inverses of matrices corresponding to elementary row operations. In this section, we will study inverses of general square matrices. We begin with the formal definition.

Definition 3.1. Let A be a square matrix of size $n \times n$. An $n \times n$ matrix X is called the *inverse* of A if it satisfies

$$X A = \mathrm{I} = A X, \tag{3.7}$$

where $\mathrm{I} = \mathrm{I}_n$ is the $n \times n$ identity matrix. The inverse is commonly denoted by $X = A^{-1}$.

Remark: Noncommutativity of matrix multiplication requires that we impose both conditions in (3.7) in order to properly define an inverse to the matrix A. The first condition, $X A = \mathrm{I}$, says that X is a *left inverse*, while the second, $A X = \mathrm{I}$, requires that X also be a *right inverse*. Rectangular matrices might have either a left inverse or a right inverse, but, as we shall see, *only* square matrices have both, and so only square matrices

Basic Matrix Arithmetic

Matrix Addition:	Commutativity	$A+B=B+A$
	Associativity	$(A+B)+C=A+(B+C)$
	Zero Matrix	$A+\mathrm{O}=A=\mathrm{O}+A$
	Inverse	$A+(-A)=\mathrm{O}, \quad -A=(-1)A$
Scalar Multiplication:	Associativity	$c(dA)=(cd)A$
	Distributivity	$c(A+B)=(cA)+(cB)$ $(c+d)A=(cA)+(dA)$
	Unit	$1A=A$
	Zero	$0A=\mathrm{O}$
Matrix Multiplication:	Associativity	$(AB)C=A(BC)$
	Distributivity	$A(B+C)=AB+AC,$ $(A+B)C=AC+BC,$
	Identity Matrix	$A\,\mathrm{I}=A=\mathrm{I}\,A$
	Zero Matrix	$A\,\mathrm{O}=\mathrm{O}, \quad \mathrm{O}\,A=\mathrm{O}$

can have full-fledged inverses. However, not every square matrix has an inverse. Indeed, not every scalar has an inverse: $0^{-1}=1/0$ is not defined since the equation $0x=1$ has no solution.

Example 3.2. Since

$$\begin{pmatrix} 1 & 2 & -1 \\ -3 & 1 & 2 \\ -2 & 2 & 1 \end{pmatrix}\begin{pmatrix} 3 & 4 & -5 \\ 1 & 1 & -1 \\ 4 & 6 & -7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 4 & -5 \\ 1 & 1 & -1 \\ 4 & 6 & -7 \end{pmatrix}\begin{pmatrix} 1 & 2 & -1 \\ -3 & 1 & 2 \\ -2 & 2 & 1 \end{pmatrix},$$

we conclude that when $A=\begin{pmatrix} 1 & 2 & -1 \\ -3 & 1 & 2 \\ -2 & 2 & 1 \end{pmatrix}$, then $A^{-1}=\begin{pmatrix} 3 & 4 & -5 \\ 1 & 1 & -1 \\ 4 & 6 & -7 \end{pmatrix}$. Observe that there is no obvious way to anticipate the entries of A^{-1} from the entries of A.

Example 3.3. Let us compute the inverse $X=\begin{pmatrix} x & y \\ z & w \end{pmatrix}$, when it exists, of a general 2×2 matrix $A=\begin{pmatrix} a & b \\ c & d \end{pmatrix}$. The right inverse condition

$$A\,X=\begin{pmatrix} a\,x+b\,z & a\,y+b\,w \\ c\,x+d\,z & c\,y+d\,w \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \mathrm{I}$$

holds if and only if x, y, z, w satisfy the linear system

$$\begin{aligned} a\,x + b\,z &= 1, & a\,y + b\,w &= 0, \\ c\,x + d\,z &= 0, & c\,y + d\,w &= 1. \end{aligned}$$

Solving by Gaussian Elimination (or directly), we find

$$x = \frac{d}{a\,d - b\,c}\,, \qquad y = -\,\frac{b}{a\,d - b\,c}\,, \qquad z = -\,\frac{c}{a\,d - b\,c}\,, \qquad w = \frac{a}{a\,d - b\,c}\,,$$

provided the common denominator $a\,d - b\,c \neq 0$ does not vanish. Therefore, the matrix

$$X = \frac{1}{a\,d - b\,c} \begin{pmatrix} d & -\,b \\ -\,c & a \end{pmatrix}$$

forms a right inverse to A. However, a short computation shows that it also defines a left inverse:

$$X\,A = \begin{pmatrix} x\,a + y\,c & x\,b + y\,d \\ z\,a + w\,c & z\,b + w\,d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \mathrm{I}\,,$$

and hence $X = A^{-1}$ is the inverse to A.

The denominator appearing in the preceding formulae has a special name; it is called the *determinant* of the 2×2 matrix A, and denoted

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = a\,d - b\,c. \tag{3.8}$$

Thus, the determinant of a 2×2 matrix is the product of the diagonal entries minus the product of the off-diagonal entries. Thus, the 2×2 matrix A is invertible, with

$$A^{-1} = \frac{1}{a\,d - b\,c} \begin{pmatrix} d & -\,b \\ -\,c & a \end{pmatrix}, \tag{3.9}$$

if and only if $\det A \neq 0$. For example, if $A = \begin{pmatrix} 1 & 3 \\ -2 & -4 \end{pmatrix}$, then $\det A = 2 \neq 0$. We conclude that A has an inverse, which, by (3.9), is $A^{-1} = \frac{1}{2}\begin{pmatrix} -4 & -3 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} -\,2 & -\frac{3}{2} \\ 1 & \frac{1}{2} \end{pmatrix}$.

Lemma 3.4. *The inverse of a square matrix, if it exists, is unique.*

Proof: Suppose both X and Y satisfy (3.7), so $X\,A = \mathrm{I} = A\,X$ and $Y\,A = \mathrm{I} = A\,Y$. Then, by associativity, $X = X\,\mathrm{I} = X(A\,Y) = (X\,A)\,Y = \mathrm{I}\,Y = Y$, and hence $X = Y$. *Q.E.D.*

Inverting a matrix twice brings us back to where we started.

Lemma 3.5. *If A is invertible, then A^{-1} is also invertible and $(A^{-1})^{-1} = A$.*

Proof: The matrix inverse equations $A^{-1}\,A = \mathrm{I} = A\,A^{-1}$ are sufficient to prove that A is the inverse of A^{-1}. *Q.E.D.*

Lemma 3.6. *If A and B are invertible matrices of the same size, then their product, AB, is invertible, and*

$$(AB)^{-1} = B^{-1}A^{-1}. \tag{3.10}$$

Note that the order of the factors is reversed under inversion.

Proof: Let $X = B^{-1}A^{-1}$. Then, by associativity,

$$X(AB) = B^{-1}A^{-1}AB = B^{-1}B = \mathrm{I}, \qquad (AB)X = ABB^{-1}A^{-1} = AA^{-1} = \mathrm{I}.$$

Thus X is both a left and a right inverse for the product matrix AB and the result follows. *Q.E.D.*

Example 3.7. One verifies, directly, that the inverse of $A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$ is $A^{-1} = \begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix}$, while the inverse of $B = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ is $B^{-1} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$. Therefore, the inverse of their product $C = AB = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = \begin{pmatrix} -2 & 1 \\ -1 & 0 \end{pmatrix}$ is given by $C^{-1} = B^{-1}A^{-1} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & -2 \end{pmatrix}$.

We can straightforwardly generalize the preceding result. The inverse of a k-fold product of invertible matrices is the product of their inverses, *in the reverse order*:

$$(A_1A_2\cdots A_{k-1}A_k)^{-1} = A_k^{-1}A_{k-1}^{-1}\cdots A_2^{-1}A_1^{-1}. \tag{3.11}$$

Warning: In general, $(A+B)^{-1} \neq A^{-1} + B^{-1}$. This equation is not even true for scalars (1×1 matrices)!

Transposes and Symmetric Matrices

Another basic operation on matrices is to interchange their rows and columns. If A is an $m \times n$ matrix, then its *transpose*, denoted A^T, is the $n \times m$ matrix whose (i,j) entry equals the (j,i) entry of A; thus

$$B = A^T \qquad \text{means that} \qquad b_{ij} = a_{ji}.$$

For example, if

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}, \qquad \text{then} \qquad A^T = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}.$$

Observe that the rows of A become the columns of A^T and vice versa. In particular, the transpose of a row vector is a column vector, while the transpose of a column vector is a row vector; if $\mathbf{v} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$, then $\mathbf{v}^T = (\,1 \ \ 2 \ \ 3\,)$. The transpose of a scalar, considered as a 1×1 matrix, is itself: $c^T = c$.

Remark: Most vectors appearing in applied mathematics are column vectors. To conserve vertical space in this text, we will often use the transpose notation, e.g., $\mathbf{v} = (v_1, v_2, v_3)^T$, as a compact way of writing column vectors.

In the square case, transposition can be viewed as "reflecting" the matrix entries across the main diagonal. For example,

$$\begin{pmatrix} 1 & 2 & -1 \\ 3 & 0 & 5 \\ -2 & -4 & 8 \end{pmatrix}^T = \begin{pmatrix} 1 & 3 & -2 \\ 2 & 0 & -4 \\ -1 & 5 & 8 \end{pmatrix}.$$

In particular, the transpose of a lower triangular matrix is upper triangular and vice-versa.

Transposing twice returns you to where you started:

$$(A^T)^T = A. \tag{3.12}$$

Unlike inversion, transposition *is* compatible with matrix addition and scalar multiplication:

$$(A+B)^T = A^T + B^T, \qquad (cA)^T = c\,A^T. \tag{3.13}$$

Transposition is also compatible with matrix multiplication, but with a twist. Like the inverse, the transpose *reverses* the order of multiplication:

$$(A\,B)^T = B^T A^T. \tag{3.14}$$

Indeed, if A has size $m \times n$ and B has size $n \times p$, so they can be multiplied, then A^T has size $n \times m$ and B^T has size $p \times n$, and so, in general, one has no choice but to multiply $B^T A^T$ in that order. Formula (3.14) is a straightforward consequence of the basic laws of matrix multiplication. An important special case is the product between a row vector $\mathbf{v}^T$ and a column vector $\mathbf{w}$ with the same number of entries. In this case,

$$\mathbf{v}^T\mathbf{w} = (\mathbf{v}^T\mathbf{w})^T = \mathbf{w}^T\mathbf{v}, \tag{3.15}$$

because their product is a scalar and so, as noted above, equals its own transpose.

Lemma 3.8. *If A is a nonsingular matrix, so is A^T, and its inverse is denoted*

$$A^{-T} = (A^T)^{-1} = (A^{-1})^T. \tag{3.16}$$

Thus, transposing a matrix and then inverting yields the same result as first inverting and then transposing.

Proof: Let $X = (A^{-1})^T$. Then, according to (3.14),

$$X\,A^T = (A^{-1})^T A^T = (A\,A^{-1})^T = \mathrm{I}^T = \mathrm{I}\,.$$

The proof that $A^T X = \mathrm{I}$ is similar, and so we conclude that $X = (A^T)^{-1}$. *Q.E.D.*

A particularly important class of square matrices is those that are unchanged by the transpose operation.

Definition 3.9. A square matrix is called *symmetric* if it equals its own transpose: $A = A^T$.

Thus, A is symmetric if and only if its entries satisfy $a_{ji} = a_{ij}$ for all i, j. In other words, entries lying in "mirror image" positions relative to the main diagonal must be equal. For example, the most general symmetric 3×3 matrix has the form

$$A = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}.$$

Note that any diagonal matrix, including the identity, is symmetric. A lower or upper triangular matrix is symmetric if and only if it is, in fact, a diagonal matrix.

AIMS Lecture Notes 2006

Peter J. Olver

4. Gaussian Elimination

In this part, our focus will be on the most basic method for solving linear algebraic systems, known as *Gaussian Elimination* in honor of one of the all-time mathematical greats — the early nineteenth century German mathematician Carl Friedrich Gauss. As the father of linear algebra, his name will occur repeatedly throughout this text. Gaussian Elimination is quite elementary, but remains one of *the* most important algorithms in applied (as well as theoretical) mathematics. Our initial focus will be on the most important class of systems: those involving the same number of equations as unknowns — although we will eventually develop techniques for handling completely general linear systems. While the former typically have a unique solution, general linear systems may have either no solutions or infinitely many solutions. Since physical models require existence and uniqueness of their solution, the systems arising in applications often (but not always) involve the same number of equations as unknowns. Nevertheless, the ability to confidently handle all types of linear systems is a basic prerequisite for further progress in the subject. In contemporary applications, particularly those arising in numerical solutions of differential equations, in signal and image processing, and elsewhere, the governing linear systems can be huge, sometimes involving millions of equations in millions of unknowns, challenging even the most powerful supercomputer. So, a systematic and careful development of solution techniques is essential. Section 4.5 discusses some of the practical issues and limitations in computer implementations of the Gaussian Elimination method for large systems arising in applications.

4.1. Solution of Linear Systems.

Gaussian Elimination is a simple, systematic algorithm to solve systems of linear equations. It is the workhorse of linear algebra, and, as such, of absolutely fundamental importance in applied mathematics. In this section, we review the method in the most important case, in which there are the same number of equations as unknowns.

To illustrate, consider an elementary system of three linear equations

$$\begin{aligned} x + 2y + z &= 2, \\ 2x + 6y + z &= 7, \\ x + y + 4z &= 3, \end{aligned} \tag{4.1}$$

in three unknowns x, y, z. Linearity refers to the fact that the unknowns only appear to the first power, and there are no product terms like $x\,y$ or $x\,y\,z$. The basic solution method is to systematically employ the following fundamental operation:

Linear System Operation #1: Add a multiple of one equation to another equation.

Before continuing, you might try to convince yourself that this operation doesn't change the solutions to the system. Our goal is to judiciously apply the operation and so be led to a much simpler linear system that is easy to solve, and, moreover has the same solutions as the original. Any linear system that is derived from the original system by successive application of such operations will be called an *equivalent system*. By the preceding remark, *equivalent linear systems have the same solutions*.

The systematic feature is that we successively eliminate the variables in our equations in order of appearance. We begin by eliminating the first variable, x, from the second equation. To this end, we subtract twice the first equation from the second, leading to

$$\begin{aligned} x + 2y + z &= 2, \\ 2y - z &= 3, \\ x + y + 4z &= 3. \end{aligned} \tag{4.2}$$

Next, we eliminate x from the third equation by subtracting the first equation from it:

$$\begin{aligned} x + 2y + z &= 2, \\ 2y - z &= 3, \\ -y + 3z &= 1. \end{aligned} \tag{4.3}$$

The equivalent system (4.3) is already simpler than the original (4.1). Notice that the second and third equations do not involve x (by design) and so constitute a system of two linear equations for two unknowns. Moreover, once we have solved this subsystem for y and z, we can substitute the answer into the first equation, and we need only solve a single linear equation for x.

We continue on in this fashion, the next phase being the elimination of the second variable, y, from the third equation by adding $\frac{1}{2}$ the second equation to it. The result is

$$\begin{aligned} x + 2y + z &= 2, \\ 2y - z &= 3, \\ \tfrac{5}{2}\,z &= \tfrac{5}{2}, \end{aligned} \tag{4.4}$$

which is the simple system we are after. It is in what is called *triangular form*, which means that, while the first equation involves all three variables, the second equation only involves the second and third variables, and the last equation only involves the last variable.

Any triangular system can be straightforwardly solved by the method of *Back Substitution*. As the name suggests, we work backwards, solving the last equation first, which requires that $z = 1$. We substitute this result back into the penultimate equation, which becomes $2y - 1 = 3$, with solution $y = 2$. We finally substitute these two values for y and z into the first equation, which becomes $x + 5 = 2$, and so the solution to the triangular system (4.4) is

$$x = -3, \qquad y = 2, \qquad z = 1. \tag{4.5}$$

Moreover, since we only used our basic linear system operation to pass from (4.1) to the triangular system (4.4), this is also the solution to the original system of linear equations, as you can check. We note that the system (4.1) has a unique — meaning one and only one — solution, namely (4.5).

And that, barring a few minor complications that can crop up from time to time, is all that there is to the method of Gaussian Elimination! It is extraordinarily simple, but its importance cannot be overemphasized. Before exploring the relevant issues, it will help to reformulate our method in a more convenient matrix notation.

4.2. Gaussian Elimination — Regular Case.

With the basic matrix arithmetic operations in hand, let us now return to our primary task. The goal is to develop a systematic method for solving linear systems of equations. While we could continue to work directly with the equations, matrices provide a convenient alternative that begins by merely shortening the amount of writing, but ultimately leads to profound insight into the structure of linear systems and their solutions.

We begin by replacing the system (3.2) by its matrix constituents. It is convenient to ignore the vector of unknowns, and form the *augmented matrix*

$$M = \begin{pmatrix} A \mid \mathbf{b} \end{pmatrix} = \left(\begin{array}{cccc|c} a_{11} & a_{12} & \ldots & a_{1n} & b_1 \\ a_{21} & a_{22} & \ldots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} & b_m \end{array}\right) \tag{4.6}$$

which is an $m \times (n+1)$ matrix obtained by tacking the right hand side vector onto the original coefficient matrix. The extra vertical line is included just to remind us that the last column of this matrix plays a special role. For example, the augmented matrix for the system (4.1), i.e.,

$$\begin{aligned} x + 2y + z &= 2, \\ 2x + 6y + z &= 7, \\ x + y + 4z &= 3, \end{aligned} \qquad \text{is} \qquad M = \left(\begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 2 & 6 & 1 & 7 \\ 1 & 1 & 4 & 3 \end{array}\right). \tag{4.7}$$

Note that one can immediately recover the equations in the original linear system from the augmented matrix. Since operations on equations also affect their right hand sides, keeping track of everything is most easily done through the augmented matrix.

For the time being, we will concentrate our efforts on linear systems that have the same number, n, of equations as unknowns. The associated coefficient matrix A is square, of size $n \times n$. The corresponding augmented matrix $M = \begin{pmatrix} A \mid \mathbf{b} \end{pmatrix}$ then has size $n \times (n+1)$.

The matrix operation that assumes the role of Linear System Operation #1 is:

Elementary Row Operation #1:

Add a scalar multiple of one row of the augmented matrix to another row.

For example, if we add -2 times the first row of the augmented matrix (4.7) to the second row, the result is the row vector

$$-2\,(\,1 \quad 2 \quad 1 \quad 2\,) \;+\; (\,2 \quad 6 \quad 1 \quad 7\,) = (\,0 \quad 2 \quad -1 \quad 3\,).$$

The result can be recognized as the second row of the modified augmented matrix

$$\left(\begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 0 & 2 & -1 & 3 \\ 1 & 1 & 4 & 3 \end{array}\right) \tag{4.8}$$

that corresponds to the first equivalent system (4.2). When elementary row operation #1 is performed, it is critical that the result replaces the row being added to — *not* the row being multiplied by the scalar. Notice that the elimination of a variable in an equation — in this case, the first variable in the second equation — amounts to making its entry in the coefficient matrix equal to zero.

We shall call the $(1,1)$ entry of the coefficient matrix the *first pivot*. The precise definition of pivot will become clear as we continue; the one key requirement is that a pivot be *nonzero*. Eliminating the first variable x from the second and third equations amounts to making all the matrix entries in the column below the pivot equal to zero. We have already done this with the $(2,1)$ entry in (4.8). To make the $(3,1)$ entry equal to zero, we subtract (that is, add -1 times) the first row from the last row. The resulting augmented matrix is

$$\left(\begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 0 & 2 & -1 & 3 \\ 0 & -1 & 3 & 1 \end{array}\right),$$

which corresponds to the system (4.3). The *second pivot* is the $(2,2)$ entry of this matrix, which is 2, and is the coefficient of the second variable in the second equation. Again, the pivot must be nonzero. We use the elementary row operation of adding $\frac{1}{2}$ of the second row to the third row to make the entry below the second pivot equal to 0; the result is the augmented matrix

$$N = \left(\begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 0 & 2 & -1 & 3 \\ 0 & 0 & \frac{5}{2} & \frac{5}{2} \end{array}\right)$$

that corresponds to the triangular system (4.4). We write the final augmented matrix as

$$N = \bigl(U \mid \mathbf{c} \bigr), \qquad \text{where} \qquad U = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & -1 \\ 0 & 0 & \frac{5}{2} \end{pmatrix}, \qquad \mathbf{c} = \begin{pmatrix} 2 \\ 3 \\ \frac{5}{2} \end{pmatrix}.$$

The corresponding linear system has vector form

$$U\mathbf{x} = \mathbf{c}. \tag{4.9}$$

Its coefficient matrix U is *upper triangular*, which means that all its entries below the main diagonal are zero: $u_{ij} = 0$ whenever $i > j$. The three nonzero entries on its diagonal, $1, 2, \frac{5}{2}$, including the last one in the $(3,3)$ slot, are the three pivots. Once the system has been reduced to triangular form (4.9), we can easily solve it by Back Substitution, as before.

Gaussian Elimination — Regular Case

```
start
    for $j = 1$ to $n$
        if $m_{jj} = 0$, stop; print "$A$ is not regular"
        else for $i = j + 1$ to $n$
            set $l_{ij} = m_{ij} / m_{jj}$
            add $- l_{ij}$ times row $j$ of $M$ to row $i$ of $M$
        next $i$
    next $j$
end
```

The preceding algorithm for solving a linear system of n equations in n unknowns is known as *regular Gaussian Elimination*. A square matrix A will be called *regular*[†] if the algorithm successfully reduces it to upper triangular form U with all non-zero pivots on the diagonal. In other words, for regular matrices, as the algorithm proceeds, each successive pivot appearing on the diagonal must be nonzero; otherwise, the matrix is not regular. We then use the pivot row to make all the entries lying in the column below the pivot equal to zero through elementary row operations. The solution is found by applying Back Substitution to the resulting triangular system.

Let us state this algorithm in the form of a program, written in a general "pseudocode" that can be easily translated into any specific language, e.g., C++, FORTRAN, JAVA, MAPLE, MATHEMATICA or MATLAB. By convention, the same letter $M = (m_{ij})$ will be used to denote the current augmented matrix at each stage in the computation, keeping in mind that its entries will change as the algorithm progresses. We initialize $M = \left(A \mid \mathbf{b} \right)$. The final output of the program, assuming A is regular, is the augmented matrix $M = \left(U \mid \mathbf{c} \right)$, where U is the upper triangular matrix whose diagonal entries are the pivots, while $\mathbf{c}$ is the resulting vector of right hand sides in the triangular system $U\mathbf{x} = \mathbf{c}$.

Elementary Matrices

A key observation is that elementary row operations can, in fact, be realized by matrix multiplication. To this end, we introduce the first type of "elementary matrix". (Later we will meet two other types of elementary matrix, corresponding to two other kinds of elementary row operation.)

Definition 4.1. The *elementary matrix E* associated with an elementary row operation for m–rowed matrices is the matrix obtained by applying the row operation to the

[†] Strangely, there is no commonly accepted term to describe these kinds of matrices. For lack of a better alternative, we propose to use the adjective "regular" in the sequel.

$m \times m$ identity matrix I_m.

For example, applying the elementary row operation that adds -2 times the first row to the second row of the 3×3 identity matrix $\mathrm{I} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ results in the corresponding elementary matrix $E_1 = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. We claim that, if A is *any* 3–rowed matrix, then multiplying $E_1 A$ has the same effect as the given elementary row operation. For example,

$$\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & -1 \\ 1 & 1 & 4 \end{pmatrix},$$

which you may recognize as the first elementary row operation we used to solve our illustrative example. If we set

$$E_1 = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad E_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}, \qquad E_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \end{pmatrix}, \tag{4.10}$$

then multiplication by E_1 will subtract twice the first row from the second row, multiplication by E_2 will subtract the first row from the third row, and multiplication by E_3 will add $\frac{1}{2}$ the second row to the third row — precisely the row operations used to place our original system in triangular form. Therefore, performing them in the correct order (and using the associativity of matrix multiplication), we conclude that when

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 4 \end{pmatrix}, \qquad \text{then} \qquad E_3 E_2 E_1 A = U = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & -1 \\ 0 & 0 & \frac{5}{2} \end{pmatrix}. \tag{4.11}$$

The reader is urged to check this by directly multiplying the indicated matrices.

In general, then, an *$m \times m$ elementary matrix E of the first type* will have all 1's on the diagonal, one nonzero entry c in some off-diagonal position (i,j), with $i \neq j$, and all other entries equal to zero. If A is any $m \times n$ matrix, then the matrix product $E A$ is equal to the matrix obtained from A by the elementary row operation adding c times row j to row i. (Note that the order of i and j is reversed.)

To undo the operation of adding c times row j to row i, we must perform the inverse row operation that subtracts c (or, equivalently, adds $-c$) times row j from row i. The corresponding *inverse elementary matrix* again has 1's along the diagonal and $-c$ in the (i,j) slot. Let us denote the inverses of the particular elementary matrices (4.10) by L_i, so that, according to our general rule,

$$L_1 = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \qquad L_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{pmatrix}. \tag{4.12}$$

Note that the products

$$L_1 E_1 = L_2 E_2 = L_3 E_3 = \mathrm{I} \tag{4.13}$$

yield the 3×3 identity matrix, reflecting the fact that the matrices represent mutually inverse row operations.

The product of the latter three elementary matrices (4.12) is equal to

$$L = L_1 L_2 L_3 = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -\frac{1}{2} & 1 \end{pmatrix}. \tag{4.14}$$

The matrix L is called a *special lower triangular* matrix, where "lower triangular" means that all the entries above the main diagonal are 0, while "special" indicates that all the entries on the diagonal are equal to 1. Observe that the entries of L below the diagonal are the same as the corresponding nonzero entries in the L_i. This is a general fact that holds when the lower triangular elementary matrices are multiplied in the correct order. More generally, the following elementary consequence of the laws of matrix multiplication will be used extensively.

Lemma 4.2. *If L and $\widehat{L}$ are lower triangular matrices of the same size, so is their product $L\widehat{L}$. If they are both special lower triangular, so is their product. Similarly, if $U, \widehat{U}$ are (special) upper triangular matrices, so is their product $U\widehat{U}$.*

The LU Factorization

We have almost arrived at our first important result. Let us compute the product of the matrices L and U in (4.11), (4.14). Using associativity of matrix multiplication, equations (4.13), and the basic property of the identity matrix I, we conclude that

$$\begin{aligned} LU &= (L_1 L_2 L_3)(E_3 E_2 E_1 A) = L_1 L_2 (L_3 E_3) E_2 E_1 A = L_1 L_2 \,\mathrm{I}\, E_2 E_1 A \\ &= L_1 (L_2 E_2) E_1 A = L_1 \,\mathrm{I}\, E_1 A = (L_1 E_1) A = \mathrm{I}\, A = A. \end{aligned}$$

In other words, we have *factored* the coefficient matrix $A = LU$ into a product of a special lower triangular matrix L and an upper triangular matrix U with the nonzero pivots on its main diagonal. By similar reasoning, the same holds true for almost all square matrices.

Theorem 4.3. *A matrix A is regular if and only if it can be factored*

$$A = LU, \tag{4.15}$$

where L is a special lower triangular matrix, having all 1's on the diagonal, and U is upper triangular with nonzero diagonal entries, which are the pivots of A. The nonzero off-diagonal entries l_{ij} for $i > j$ appearing in L prescribe the elementary row operations that bring A into upper triangular form; namely, one subtracts l_{ij} times row j from row i at the appropriate step of the Gaussian Elimination process.

In practice, to find the LU factorization of a square matrix A, one applies the regular Gaussian Elimination algorithm to reduce A to its upper triangular form U. The entries of L can be filled in during the course of the calculation with the negatives of the multiples used in the elementary row operations. If the algorithm fails to be completed, which happens whenever zero appears in any diagonal pivot position, then the original matrix is *not* regular, and does *not* have an LU factorization.

Example 4.4. Let us compute the LU factorization of the matrix $A = \begin{pmatrix} 2 & 1 & 1 \\ 4 & 5 & 2 \\ 2 & -2 & 0 \end{pmatrix}$. Applying the Gaussian Elimination algorithm, we begin by adding -2 times the first row to the second row, and then adding -1 times the first row to the third. The result is the matrix $\begin{pmatrix} 2 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & -3 & -1 \end{pmatrix}$. The next step adds the second row to the third row, leading to the upper triangular matrix $U = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix}$, whose diagonal entries are the pivots. The corresponding lower triangular matrix is $L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix}$; its entries lying below the main diagonal are the *negatives* of the multiples we used during the elimination procedure. For instance, the $(2,1)$ entry indicates that we added -2 times the first row to the second row, and so on. The reader might wish to verify the resulting factorization

$$\begin{pmatrix} 2 & 1 & 1 \\ 4 & 5 & 2 \\ 2 & -2 & 0 \end{pmatrix} = A = LU = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix}.$$

Forward and Back Substitution

Once we know the LU factorization of a regular matrix A, we are able to solve any associated linear system $A\mathbf{x} = \mathbf{b}$ in two easy stages:

(1) First, solve the lower triangular system

$$L\mathbf{c} = \mathbf{b} \tag{4.16}$$

for the vector $\mathbf{c}$ by *Forward Substitution.* This is the same as Back Substitution, except one solves the equations for the variables in the direct order — from first to last. Explicitly,

$$c_1 = b_1, \qquad c_i = b_i - \sum_{j=1}^{i-1} l_{ij} c_j, \qquad \text{for} \qquad i = 2, 3, \ldots, n, \tag{4.17}$$

noting that the previously computed values of $c_1, \ldots, c_{i-1}$ are used to determine c_i.

(2) Second, solve the resulting upper triangular system

$$U\mathbf{x} = \mathbf{c} \tag{4.18}$$

by *Back Substitution*. The values of the unknowns

$$x_n = \frac{c_n}{u_{nn}}, \qquad x_i = \frac{1}{u_{ii}} \left(c_i - \sum_{j=i+1}^{n} u_{ij}\, x_j \right), \qquad \text{for} \qquad i = n-1, \ldots, 2, 1, \tag{4.19}$$

are successively computed, but now in reverse order. It is worth pointing out that the requirement that each pivot $u_{ii} \neq 0$ is essential here, as otherwise we would not be able to solve for the corresponding variable x_i.

Note that the combined algorithm does indeed solve the original system, since if

$$U\mathbf{x} = \mathbf{c} \qquad \text{and} \qquad L\mathbf{c} = \mathbf{b}, \qquad \text{then} \qquad A\mathbf{x} = L\,U\mathbf{x} = L\mathbf{c} = \mathbf{b}.$$

Example 4.5. With the $L\,U$ decomposition

$$\begin{pmatrix} 2 & 1 & 1 \\ 4 & 5 & 2 \\ 2 & -2 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

found in Example 4.4, we can readily solve any linear system with the given coefficient matrix by Forward and Back Substitution. For instance, to find the solution to

$$\begin{pmatrix} 2 & 1 & 1 \\ 4 & 5 & 2 \\ 2 & -2 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix},$$

we first solve the lower triangular system

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}, \qquad \text{or, explicitly,} \qquad \begin{aligned} a \qquad\quad &= 1, \\ 2a + b \qquad &= 2, \\ a - b + c &= 2. \end{aligned}$$

The first equation says $a = 1$; substituting into the second, we find $b = 0$; the final equation yields $c = 1$. We then use Back Substitution to solve the upper triangular system

$$\begin{pmatrix} 2 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \qquad \text{which is} \qquad \begin{aligned} 2x + y + z &= 1, \\ 3y \qquad &= 0, \\ -z &= 1. \end{aligned}$$

We find $z = -1$, then $y = 0$, and then $x = 1$, which is indeed the solution.

Thus, once we have found the $L\,U$ factorization of the coefficient matrix A, the Forward and Back Substitution processes quickly produce the solution to any system $A\mathbf{x} = \mathbf{b}$. Moreover, they can be straightforwardly programmed on a computer. In practice, to solve a system from scratch, it is a matter of taste whether you work directly with the augmented matrix, or first determine the $L\,U$ factorization of the coefficient matrix, and then apply Forward and Back Substitution to compute the solution.

4.3. Pivoting and Permutations.

The method of Gaussian Elimination presented so far applies only to regular matrices. But not every square matrix is regular; a simple class of examples is matrices whose upper left, i.e., $(1,1)$, entry is zero, and so cannot serve as the first pivot. More generally, the algorithm cannot proceed whenever a zero entry appears in the current pivot position on the diagonal. What then to do? The answer requires revisiting the source of the method.

Consider, as a specific example, the linear system

$$\begin{aligned} 2y + z &= 2, \\ 2x + 6y + z &= 7, \\ x + y + 4z &= 3. \end{aligned} \tag{4.20}$$

The augmented coefficient matrix is

$$\left(\begin{array}{ccc|c} 0 & 2 & 1 & 2 \\ 2 & 6 & 1 & 7 \\ 1 & 1 & 4 & 3 \end{array}\right).$$

In this case, the $(1,1)$ entry is 0, and so is not a legitimate pivot. The problem, of course, is that the first variable x does not appear in the first equation, and so we cannot use it to eliminate x in the other two equations. But this "problem" is actually a bonus — we already have an equation with only two variables in it, and so we only need to eliminate x from one of the other two equations. To be systematic, we rewrite the system in a different order,

$$\begin{aligned} 2x + 6y + z &= 7, \\ 2y + z &= 2, \\ x + y + 4z &= 3, \end{aligned}$$

by interchanging the first two equations. In other words, we employ

Linear System Operation #2: Interchange two equations.

Clearly, this operation does not change the solution and so produces an equivalent linear system. In our case, the augmented coefficient matrix

$$\left(\begin{array}{ccc|c} 2 & 6 & 1 & 7 \\ 0 & 2 & 1 & 2 \\ 1 & 1 & 4 & 3 \end{array}\right),$$

can be obtained from the original by performing the second type of row operation:

Elementary Row Operation #2: Interchange two rows of the matrix.

The new nonzero upper left entry, 2, can now serve as the first pivot, and we may continue to apply elementary row operations of Type #1 to reduce our matrix to upper triangular form. For this particular example, we eliminate the remaining nonzero entry in the first column by subtracting $\frac{1}{2}$ the first row from the last:

$$\left(\begin{array}{rrr|r} 2 & 6 & 1 & 7 \\ 0 & 2 & 1 & 2 \\ 0 & -2 & \frac{7}{2} & -\frac{1}{2} \end{array}\right).$$

The $(2,2)$ entry serves as the next pivot. To eliminate the nonzero entry below it, we add the second to the third row:

$$\left(\begin{array}{ccc|c} 2 & 6 & 1 & 7 \\ 0 & 2 & 1 & 2 \\ 0 & 0 & \frac{9}{2} & \frac{3}{2} \end{array}\right).$$

We have now placed the system in upper triangular form, with the three pivots $2, 2$, and $\frac{9}{2}$ along the diagonal. Back Substitution produces the solution $x = \frac{5}{6}$, $y = \frac{5}{6}$, $z = \frac{1}{3}$.

The row interchange that is required when a zero shows up in the diagonal pivot position is known as *pivoting*. Later, in Section 4.5, we will discuss practical reasons for pivoting even when a diagonal entry is nonzero. Let us distinguish the class of matrices that can be reduced to upper triangular form by Gaussian Elimination with pivoting. These matrices will prove to be of fundamental importance throughout linear algebra.

Definition 4.6. A square matrix is called *nonsingular* if it can be reduced to upper triangular form with all non-zero elements on the diagonal — the pivots — by elementary row operations of Types 1 and 2.

In contrast, a *singular* square matrix cannot be reduced to such upper triangular form by such row operations, because at some stage in the elimination procedure the diagonal entry and all the entries below it are zero. Every regular matrix is nonsingular, but, as we just saw, not every nonsingular matrix is regular. Uniqueness of solutions is the key defining characteristic of nonsingularity.

Theorem 4.7. *A linear system $A\mathbf{x} = \mathbf{b}$ has a unique solution for* every *choice of right hand side $\mathbf{b}$ if and only if its coefficient matrix A is square and nonsingular.*

We are able to prove the "if" part of this theorem, since nonsingularity implies reduction to an equivalent upper triangular form that has the same solutions as the original system. The unique solution to the system is then found by Back Substitution. The "only if" part will be proved later.

The revised version of the Gaussian Elimination algorithm, valid for all nonsingular coefficient matrices, is implemented by the accompanying pseudocode program. The starting point is the augmented matrix $M = \left(A \mid \mathbf{b} \right)$ representing the linear system $A\mathbf{x} = \mathbf{b}$. After successful termination of the program, the result is an augmented matrix in upper triangular form $M = \left(U \mid \mathbf{c} \right)$ representing the equivalent linear system $U\mathbf{x} = \mathbf{c}$. One then uses Back Substitution to determine the solution $\mathbf{x}$ to the linear system.

Gaussian Elimination — Nonsingular Case

```
start
    for $j = 1$ to $n$
        if $m_{kj} = 0$ for all $k \geq j$, stop; print "$A$ is singular"
        if $m_{jj} = 0$ but $m_{kj} \neq 0$ for some $k > j$, switch rows $k$ and $j$
        for $i = j + 1$ to $n$
            set $l_{ij} = m_{ij}/m_{jj}$
            add $-l_{ij}$ times row $j$ to row $i$ of $M$
        next $i$
    next $j$
end
```

Permutation Matrices

As with the first type of elementary row operation, row interchanges can be accomplished by multiplication by a second type of elementary matrix, which is found by applying the row operation to the identity matrix of the appropriate size. For instance, interchanging rows 1 and 2 of the 3×3 identity matrix produces the elementary interchange matrix $P = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. The result PA of multiplying any 3–rowed matrix A on the left by P is the same as interchanging the first two rows of A. For instance,

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \\ 7 & 8 & 9 \end{pmatrix}.$$

Multiple row interchanges are accomplished by combining such elementary interchange matrices. Each such combination of row interchanges corresponds to a unique permutation matrix.

Definition 4.8. A *permutation matrix* is a matrix obtained from the identity matrix by any combination of row interchanges.

In particular, applying a row interchange to a permutation matrix produces another permutation matrix. The following result is easily established.

Lemma 4.9. *A matrix P is a permutation matrix if and only if each row of P contains all* 0 *entries except for a single* 1, *and, in addition, each column of P also contains all* 0 *entries except for a single* 1.

In general, if a permutation matrix P has a 1 in position (i,j), then the effect of multiplication by P is to move the j^{th} row of A into the i^{th} row of the product PA.

Example 4.10. There are six different 3×3 permutation matrices, namely

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}. \tag{4.21}$$

These have the following effects: if A is a matrix with row vectors $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$, then multiplication on the left by each of the six permutation matrices produces, respectively,

$$\begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{pmatrix}, \qquad \begin{pmatrix} \mathbf{r}_2 \\ \mathbf{r}_3 \\ \mathbf{r}_1 \end{pmatrix}, \qquad \begin{pmatrix} \mathbf{r}_3 \\ \mathbf{r}_1 \\ \mathbf{r}_2 \end{pmatrix}, \qquad \begin{pmatrix} \mathbf{r}_2 \\ \mathbf{r}_1 \\ \mathbf{r}_3 \end{pmatrix}, \qquad \begin{pmatrix} \mathbf{r}_3 \\ \mathbf{r}_2 \\ \mathbf{r}_1 \end{pmatrix}, \qquad \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_3 \\ \mathbf{r}_2 \end{pmatrix}.$$

Thus, the first permutation matrix, which is the identity, does nothing. The fourth, fifth and sixth represent row interchanges. The second and third are non-elementary permutations, and can be realized by a pair of successive row interchanges.

An elementary combinatorial argument proves that there are a total of

$$n! = n\,(n-1)\,(n-2)\ \cdots\ 3 \cdot 2 \cdot 1 \tag{4.22}$$

different permutation matrices of size $n \times n$. Moreover, the product $P = P_1 P_2$ of any two permutation matrices is also a permutation matrix. An important point is that multiplication of permutation matrices is *noncommutative* — the order in which one permutes makes a difference. Switching the first and second rows, and then switching the second and third rows *does not* have the same effect as first switching the second and third rows and then switching the first and second rows!

The Permuted LU Factorization

As we now know, any nonsingular matrix A can be reduced to upper triangular form by elementary row operations of types #1 and #2. The row interchanges merely reorder the equations. If one performs all of the required row interchanges in advance, then the elimination algorithm can proceed without requiring any further pivoting. Thus, the matrix obtained by permuting the rows of A in the prescribed manner is regular. In other words, if A is a nonsingular matrix, then there is a permutation matrix P such that the product PA is regular, and hence admits an LU factorization. As a result, we deduce the general *permuted LU factorization*

$$PA = LU, \tag{4.23}$$

where P is a permutation matrix, L is special lower triangular, and U is upper triangular with the pivots on the diagonal. For instance, in the preceding example, we permuted the first and second rows, and hence equation (4.23) has the explicit form

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2} & -1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 6 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & \frac{9}{2} \end{pmatrix}.$$

We have now established the following generalization of Theorem 4.3.

Theorem 4.11. *Let A be an $n \times n$ matrix. Then the following conditions are equivalent*:

(i) *A is nonsingular.*

(ii) *A has n nonzero pivots.*

(iii) *A admits a permuted LU factorization*: $PA = LU$.

A practical method to construct a permuted LU factorization of a given matrix A would proceed as follows. First set up $P = L = \mathrm{I}$ as $n \times n$ identity matrices. The matrix P will keep track of the permutations performed during the Gaussian Elimination process, while the entries of L below the diagonal are gradually replaced by the negatives of the multiples used in the corresponding row operations of type #1. Each time two rows of A are interchanged, the same two rows of P will be interchanged. Moreover, any pair of entries that both lie *below* the diagonal in these same two rows of L must also be interchanged, while entries lying on and above its diagonal need to stay in their place. At a successful conclusion to the procedure, A will have been converted into the upper triangular matrix U, while L and P will assume their final form. Here is an illustrative example.

Example 4.12. Our goal is to produce a permuted LU factorization of the matrix

$$A = \begin{pmatrix} 1 & 2 & -1 & 0 \\ 2 & 4 & -2 & -1 \\ -3 & -5 & 6 & 1 \\ -1 & 2 & 8 & -2 \end{pmatrix}.$$

To begin the procedure, we apply row operations of type #1 to eliminate the entries below the first pivot. The updated matrices[†] are

$$A = \begin{pmatrix} 1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 1 & 3 & 1 \\ 0 & 4 & 7 & -2 \end{pmatrix}, \qquad L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ -3 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}, \qquad P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

where L keeps track of the row operations, and we initialize P to be the identity matrix. The $(2,2)$ entry of the new A is zero, and so we interchange its second and third rows, leading to

$$A = \begin{pmatrix} 1 & 2 & -1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 0 & -1 \\ 0 & 4 & 7 & -2 \end{pmatrix}, \qquad L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}, \qquad P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

We interchanged the same two rows of P, while in L we only interchanged the already computed entries in its second and third rows that lie in its first column below the diagonal.

[†] Here, we are adopting computer programming conventions, where updates of a matrix are all given the same name.

We then eliminate the nonzero entry lying below the $(2,2)$ pivot, leading to

$$A = \begin{pmatrix} 1 & 2 & -1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -5 & -6 \end{pmatrix}, \qquad L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ -1 & 4 & 0 & 1 \end{pmatrix}, \qquad P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

A final row interchange places the matrix in upper triangular form:

$$U = A = \begin{pmatrix} 1 & 2 & -1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & -5 & -6 \\ 0 & 0 & 0 & -1 \end{pmatrix}, \qquad L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ -1 & 4 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{pmatrix}, \qquad P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Again, we performed the same row interchange on P, while only interchanging the third and fourth row entries of L that lie below the diagonal. You can verify that

$$PA = \begin{pmatrix} 1 & 2 & -1 & 0 \\ -3 & -5 & 6 & 1 \\ -1 & 2 & 8 & -2 \\ 2 & 4 & -2 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ -1 & 4 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & -1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & -5 & -6 \\ 0 & 0 & 0 & -1 \end{pmatrix} = LU, \qquad (4.24)$$

as promised. Thus, by rearranging the equations in the order first, third, fourth, second, as prescribed by P, we obtain an equivalent linear system with regular coefficient matrix PA.

Once the permuted LU factorization is established, the solution to the original system $A\mathbf{x} = \mathbf{b}$ is obtained by applying the same Forward and Back Substitution algorithm presented above. Explicitly, we first multiply the system $A\mathbf{x} = \mathbf{b}$ by the permutation matrix, leading to

$$PA\mathbf{x} = P\mathbf{b} = \widehat{\mathbf{b}}, \qquad (4.25)$$

whose right hand side $\widehat{\mathbf{b}}$ has been obtained by permuting the entries of $\mathbf{b}$ in the same fashion as the rows of A. We then solve the two triangular systems

$$L\mathbf{c} = \widehat{\mathbf{b}} \qquad \text{and} \qquad U\mathbf{x} = \mathbf{c} \qquad (4.26)$$

by, respectively, Forward and Back Substitution.

Example 4.12. (*continued*) Suppose we wish to solve the linear system

$$\begin{pmatrix} 1 & 2 & -1 & 0 \\ 2 & 4 & -2 & -1 \\ -3 & -5 & 6 & 1 \\ -1 & 2 & 8 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 3 \\ 0 \end{pmatrix}.$$

In view of the $PA = LU$ factorization established in (4.24), we need only solve the two auxiliary lower and upper triangular systems (4.26). The lower triangular system is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ -1 & 4 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 0 \\ -1 \end{pmatrix};$$

whose right hand side was obtained by applying the permutation matrix P to the right hand side of the original system. Its solution, namely $a = 1$, $b = 6$, $c = -23$, $d = -3$, is obtained through Forward Substitution. The resulting upper triangular system is

$$\begin{pmatrix} 1 & 2 & -1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & -5 & -6 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} 1 \\ 6 \\ -23 \\ -3 \end{pmatrix}.$$

Its solution, $w = 3$, $z = 1$, $y = 0$, $x = 2$, which is also the solution to the original system, is easily obtained by Back Substitution.

4.4. Gauss–Jordan Elimination.

The principal algorithm used to compute the inverse of a nonsingular matrix is known as *Gauss–Jordan Elimination*, in honor of Gauss and Wilhelm Jordan, a nineteenth century German engineer. A key fact is that we only need to solve the right inverse equation

$$A X = \mathrm{I} \tag{4.27}$$

in order to compute $X = A^{-1}$. The left inverse equation in (3.7), namely $X A = \mathrm{I}$, will then follow as an automatic consequence. In other words, for square matrices, a right inverse is automatically a left inverse, and conversely! A proof will appear below.

The reader may well ask, then, why use both left and right inverse conditions in the original definition? There are several good reasons. First of all, a non-square matrix may satisfy one of the two conditions — having either a left inverse or a right inverse — but can never satisfy both. Moreover, even when we restrict our attention to square matrices, starting with only one of the conditions makes the logical development of the subject considerably more difficult, and not really worth the extra effort. Once we have established the basic properties of the inverse of a square matrix, we can then safely discard the superfluous left inverse condition. Finally, when we generalize the notion of an inverse to linear operators, then, unlike square matrices, we *cannot* dispense with either of the conditions.

Let us write out the individual columns of the right inverse equation (4.27). The j^{th} column of the $n \times n$ identity matrix I is the vector $\mathbf{e}_j$ that has a single 1 in the j^{th} slot and 0's elsewhere, so

$$\mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \qquad \mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \qquad \ldots \qquad \mathbf{e}_n = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}. \tag{4.28}$$

According to (3.6), the j^{th} column of the matrix product $A X$ is equal to $A \mathbf{x}_j$, where $\mathbf{x}_j$ denotes the j^{th} column of the inverse matrix X. Therefore, the single matrix equation (4.27) is equivalent to n linear systems

$$A \mathbf{x}_1 = \mathbf{e}_1, \qquad A \mathbf{x}_2 = \mathbf{e}_2, \qquad \ldots \qquad A \mathbf{x}_n = \mathbf{e}_n, \tag{4.29}$$

all having the same coefficient matrix. As such, to solve them we should form the n augmented matrices $M_1 = \bigl(A \mid \mathbf{e}_1 \bigr), \ldots, M_n = \bigl(A \mid \mathbf{e}_n \bigr)$, and then apply our Gaussian Elimination algorithm to each. But this would be a waste of effort. Since the coefficient matrix is the same, we will end up performing *identical* row operations on each augmented matrix. Clearly, it will be more efficient to combine them into one large augmented matrix $M = \bigl(A \mid \mathbf{e}_1 \ \ldots \ \mathbf{e}_n \bigr) = \bigl(A \mid \mathrm{I} \bigr)$, of size $n \times (2n)$, in which the right hand sides $\mathbf{e}_1, \ldots, \mathbf{e}_n$ of our systems are placed into n different columns, which we then recognize as reassembling the columns of an $n \times n$ identity matrix. We may then simultaneously apply our elementary row operations to reduce, if possible, the large augmented matrix so that its first n columns are in upper triangular form.

Example 4.13. For example, to find the inverse of the matrix $A = \begin{pmatrix} 0 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 4 \end{pmatrix}$, we form the large augmented matrix

$$\left(\begin{array}{ccc|ccc} 0 & 2 & 1 & 1 & 0 & 0 \\ 2 & 6 & 1 & 0 & 1 & 0 \\ 1 & 1 & 4 & 0 & 0 & 1 \end{array}\right).$$

Applying the same sequence of elementary row operations as in Section 4.3, we first interchange the rows

$$\left(\begin{array}{ccc|ccc} 2 & 6 & 1 & 0 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 1 & 1 & 4 & 0 & 0 & 1 \end{array}\right),$$

and then eliminate the nonzero entries below the first pivot,

$$\left(\begin{array}{ccc|ccc} 2 & 6 & 1 & 0 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 0 & -2 & \frac{7}{2} & 0 & -\frac{1}{2} & 1 \end{array}\right).$$

Next we eliminate the entry below the second pivot:

$$\left(\begin{array}{ccc|ccc} 2 & 6 & 1 & 0 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & \frac{9}{2} & 1 & -\frac{1}{2} & 1 \end{array}\right).$$

At this stage, we have reduced our augmented matrix to the form $\bigl(U \mid C \bigr)$ where U is upper triangular. This is equivalent to reducing the original n linear systems $A\,\mathbf{x}_i = \mathbf{e}_i$ to n upper triangular systems $U\,\mathbf{x}_i = \mathbf{c}_i$. We can therefore perform n back substitutions to produce the solutions $\mathbf{x}_i$, which would form the individual columns of the inverse matrix $X = (\mathbf{x}_1 \ \ldots \ \mathbf{x}_n)$. In the more common version of the Gauss–Jordan scheme, one instead continues to employ elementary row operations to fully reduce the augmented matrix. The goal is to produce an augmented matrix $\bigl(\mathrm{I} \mid X \bigr)$ in which the left hand $n \times n$ matrix has become the identity, while the right hand matrix is the desired solution $X = A^{-1}$. Indeed, $\bigl(\mathrm{I} \mid X \bigr)$ represents the n trivial linear systems $\mathrm{I}\,\mathbf{x} = \mathbf{x}_i$ whose solutions $\mathbf{x} = \mathbf{x}_i$ are the columns of the inverse matrix X.

Now, the identity matrix has 0's below the diagonal, just like U. It also has 1's along the diagonal, whereas U has the pivots (which are all nonzero) along the diagonal. Thus, the next phase in the reduction process is to make all the diagonal entries of U equal to 1. To proceed, we need to introduce the last, and least, of our linear systems operations.

Linear System Operation #3: Multiply an equation by a nonzero constant.

This operation clearly does not affect the solution, and so yields an equivalent linear system. The corresponding elementary row operation is:

Elementary Row Operation #3: Multiply a row of the matrix by a nonzero scalar.

Dividing the rows of the upper triangular augmented matrix $\left(U \mid C \right)$ by the diagonal pivots of U will produce a matrix of the form $\left(V \mid B \right)$ where V is *special upper triangular*, meaning it has all 1's along the diagonal. In our particular example, the result of these three elementary row operations of Type #3 is

$$\left(\begin{array}{ccc|ccc} 1 & 3 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 1 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & \frac{2}{9} & -\frac{1}{9} & \frac{2}{9} \end{array}\right),$$

where we multiplied the first and second rows by $\frac{1}{2}$ and the third row by $\frac{2}{9}$.

We are now over halfway towards our goal. We need only make the entries above the diagonal of the left hand matrix equal to zero. This can be done by elementary row operations of Type #1, but now we work backwards. First, we eliminate the nonzero entries in the third column lying above the $(3,3)$ entry by subtracting one half the third row from the second and also from the first:

$$\left(\begin{array}{ccc|ccc} 1 & 3 & 0 & -\frac{1}{9} & \frac{5}{9} & -\frac{1}{9} \\ 0 & 1 & 0 & \frac{7}{18} & \frac{1}{18} & -\frac{1}{9} \\ 0 & 0 & 1 & \frac{2}{9} & -\frac{1}{9} & \frac{2}{9} \end{array}\right).$$

Finally, we subtract 3 times the second row from the first to eliminate the remaining nonzero off-diagonal entry, thereby completing the Gauss–Jordan procedure:

$$\left(\begin{array}{ccc|ccc} 1 & 0 & 0 & -\frac{23}{18} & \frac{7}{18} & \frac{2}{9} \\ 0 & 1 & 0 & \frac{7}{18} & \frac{1}{18} & -\frac{1}{9} \\ 0 & 0 & 1 & \frac{2}{9} & -\frac{1}{9} & \frac{2}{9} \end{array}\right).$$

The left hand matrix is the identity, and therefore the final right hand matrix is our desired inverse:

$$A^{-1} = \begin{pmatrix} -\frac{23}{18} & \frac{7}{18} & \frac{2}{9} \\ \frac{7}{18} & \frac{1}{18} & -\frac{1}{9} \\ \frac{2}{9} & -\frac{1}{9} & \frac{2}{9} \end{pmatrix}. \tag{4.30}$$

The reader may wish to verify that the final result does satisfy both inverse conditions $A\,A^{-1} = \mathrm{I} = A^{-1}A$.

We are now able to complete the proofs of the basic results on inverse matrices. First, we need to determine the elementary matrix corresponding to an elementary row operation of type #3. Again, this is obtained by performing the row operation in question on the identity matrix. Thus, the elementary matrix that multiplies row i by the nonzero scalar c is the diagonal matrix having c in the i^{th} diagonal position, and 1's elsewhere along the diagonal. The inverse elementary matrix is the diagonal matrix with $1/c$ in the i^{th} diagonal position and 1's elsewhere on the main diagonal; it corresponds to the inverse operation that divides row i by c. For example, the elementary matrix that multiplies the second row of a 3–rowed matrix by 5 is $E = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, and has inverse $E^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{5} & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

In summary:

Lemma 4.14. *Every elementary matrix is nonsingular, and its inverse is also an elementary matrix of the same type.*

The Gauss–Jordan method tells us how to reduce any nonsingular square matrix A to the identity matrix by a sequence of elementary row operations. Let $E_1, E_2, \ldots, E_N$ be the corresponding elementary matrices. The elimination procedure that reduces A to I amounts to multiplying A by a succession of elementary matrices:

$$E_N E_{N-1} \,\cdots\, E_2 E_1 A = \mathrm{I}. \tag{4.31}$$

We claim that the product matrix

$$X = E_N E_{N-1} \,\cdots\, E_2 E_1 \tag{4.32}$$

is the inverse of A. Indeed, formula (4.31) says that $X A = \mathrm{I}$, and so X is a left inverse. Furthermore, each elementary matrix has an inverse, and so by (3.11), X itself is invertible, with

$$X^{-1} = E_1^{-1} E_2^{-1} \,\cdots\, E_{N-1}^{-1} E_N^{-1}. \tag{4.33}$$

Therefore, multiplying formula (4.31), namely $X A = \mathrm{I}$, on the left by X^{-1} leads to $A = X^{-1}$. Lemma 3.5 implies $X = A^{-1}$. We have thus proved

Theorem 4.15. *A square matrix A has an inverse if and only if it is nonsingular.*

Consequently, an $n \times n$ matrix will have an inverse if and only if it can be reduced to upper triangular form, with n nonzero pivots on the diagonal, by a combination of elementary row operations. Indeed, "invertible" is often used as a synonym for "nonsingular". All other matrices are singular and do not have an inverse as defined above. Before attempting to prove Theorem 4.15, we need to first become familiar with some elementary properties of matrix inverses.

Finally, equating $A = X^{-1}$ to the product (4.33), and invoking Lemma 4.14, we have established the following result.

Proposition 4.16. *Every nonsingular matrix A can be written as the product of elementary matrices.*

Example 4.17. The 2×2 matrix $A = \begin{pmatrix} 0 & -1 \\ 1 & 3 \end{pmatrix}$ is converted into the identity matrix by first interchanging its rows, $\begin{pmatrix} 1 & 3 \\ 0 & -1 \end{pmatrix}$, then scaling the second row by -1, $\begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix}$, and, finally, subtracting 3 times the second row from the first to obtain $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \mathrm{I}$. The corresponding elementary matrices are

$$E_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad E_2 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \qquad E_3 = \begin{pmatrix} 1 & -3 \\ 0 & 1 \end{pmatrix}.$$

Therefore, by (4.32),

$$A^{-1} = E_3\,E_2\,E_1 = \begin{pmatrix} 1 & -3 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 3 & 1 \\ -1 & 0 \end{pmatrix},$$

while

$$A = E_1^{-1}\,E_2^{-1}\,E_3^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 3 \end{pmatrix}.$$

As an application, let us prove that the inverse of a nonsingular triangular matrix is also triangular. Specifically:

Proposition 4.18. *If L is a lower triangular matrix with all nonzero entries on the main diagonal, then L is nonsingular and its inverse L^{-1} is also lower triangular. In particular, if L is special lower triangular, so is L^{-1}. A similar result holds for upper triangular matrices.*

Proof: It suffices to note that if L has all nonzero diagonal entries, one can reduce L to the identity by elementary row operations of Types #1 and #3, whose associated elementary matrices are all lower triangular. Lemma 4.2 implies that the product (4.32) is then also lower triangular. If L is special, then all the pivots are equal to 1. Thus, no elementary row operations of Type #3 are required, and so L can be reduced to the identity matrix by elementary row operations of Type #1 alone. Therefore, its inverse is a product of special lower triangular matrices, and hence is itself special lower triangular. A similar argument applies in the upper triangular cases. *Q.E.D.*

Solving Linear Systems with the Inverse

The primary motivation for introducing the matrix inverse is that it provides a compact formula for the solution to any linear system with an invertible coefficient matrix.

Theorem 4.19. *If A is nonsingular, then $\mathbf{x} = A^{-1}\,\mathbf{b}$ is the unique solution to the linear system $A\,\mathbf{x} = \mathbf{b}$.*

Proof: We merely multiply the system by A^{-1}, which yields $\mathbf{x} = A^{-1}A\,\mathbf{x} = A^{-1}\mathbf{b}$. Moreover, $A\,\mathbf{x} = A\,A^{-1}\mathbf{b} = \mathbf{b}$, proving that $\mathbf{x} = A^{-1}\mathbf{b}$ is indeed the solution. *Q.E.D.*

For example, let us return to the linear system (4.20). Since we computed the inverse of its coefficient matrix in (4.30), a "direct" way to solve the system is to multiply the right hand side by the inverse matrix:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -\frac{23}{18} & \frac{7}{18} & \frac{2}{9} \\ \frac{7}{18} & \frac{1}{18} & -\frac{1}{9} \\ \frac{2}{9} & -\frac{1}{9} & \frac{2}{9} \end{pmatrix} \begin{pmatrix} 2 \\ 7 \\ 3 \end{pmatrix} = \begin{pmatrix} \frac{5}{6} \\ \frac{5}{6} \\ \frac{1}{3} \end{pmatrix},$$

reproducing our earlier solution.

However, while æsthetically appealing, the solution method based on the inverse matrix is hopelessly inefficient as compared to direct Gaussian Elimination, and, despite what you may have learned, *should not be used in practical computations*. (A complete justification of this dictum will be provided in Section 4.5.) On the other hand, the inverse does play a useful role in theoretical developments, as well as providing insight into the design of practical algorithms. But the principal message of applied linear algebra is that LU decomposition and Gaussian Elimination are fundamental; matrix inverses are to be avoided in all but the most elementary computations.

Remark: The reader may have learned a version of the Gauss–Jordan algorithm for solving a single linear system that replaces the Back Substitution step by a complete reduction of the coefficient matrix to the identity. In other words, to solve $A\mathbf{x} = \mathbf{b}$, we start with the augmented matrix $M = \left(A \mid \mathbf{b} \right)$ and use all three types of elementary row operations to produce (assuming nonsingularity) the fully reduced form $\left(\mathrm{I} \mid \mathbf{d} \right)$, representing the trivially soluble, equivalent system $\mathbf{x} = \mathbf{d}$, which is the solution to the original system. However, Back Substitution is more efficient, and remains the method of choice in practical computations.

The LDV Factorization

The second phase of the Gauss–Jordan process leads to a slightly more detailed version of the LU factorization. Let D denote the diagonal matrix having the same diagonal entries as U; in other words, D contains the pivots on its diagonal and zeros everywhere else. Let V be the special upper triangular matrix obtained from U by dividing each row by its pivot, so that V has all 1's on the diagonal. We already encountered V during the course of the Gauss–Jordan procedure. It is easily seen that $U = DV$, which implies the following result.

Theorem 4.20. *A matrix A is regular if and only if it admits a factorization*

$$A = LDV, \tag{4.34}$$

where L is a special lower triangular matrix, D is a diagonal matrix having the nonzero pivots on the diagonal, and V is a special upper triangular matrix.

For the matrix appearing in Example 4.4, we have $U = DV$, where

$$U = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \qquad D = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \qquad V = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

This leads to the factorization

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 4 & 5 & 2 \\ 2 & -2 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = L\,D\,V.$$

Proposition 4.21. *If $A = L\,U$ is regular, then the factors L and U are uniquely determined. The same holds for the $A = L\,D\,V$ factorization.*

Proof: Suppose $L\,U = \widetilde{L}\,\widetilde{U}$. Since the diagonal entries of all four matrices are nonzero, Proposition 4.18 implies that they are invertible. Therefore,

$$\widetilde{L}^{-1}L = \widetilde{L}^{-1}L\,U\,U^{-1} = \widetilde{L}^{-1}\widetilde{L}\,\widetilde{U}\,U^{-1} = \widetilde{U}\,U^{-1}. \tag{4.35}$$

The left hand side of the matrix equation (4.35) is the product of two special lower triangular matrices, and so, by Lemma 4.2, is itself special lower triangular. The right hand side is the product of two upper triangular matrices, and hence is upper triangular. But the only way a special lower triangular matrix could equal an upper triangular matrix is if they both equal the diagonal identity matrix. Therefore, $\widetilde{L}^{-1}L = \mathrm{I} = \widetilde{U}\,U^{-1}$, and so $\widetilde{L} = L$ and $\widetilde{U} = U$, proving the first result. The $L\,D\,V$ version is an immediate consequence. *Q.E.D.*

As you may have guessed, the more general cases requiring one or more row interchanges lead to a permuted $L\,D\,V$ factorization in the following form.

Theorem 4.22. *A matrix A is nonsingular if and only if there is a permutation matrix P such that*

$$P\,A = L\,D\,V, \tag{4.36}$$

where L, D, V are, respectively, special lower triangular, diagonal, and special upper triangular matrices.

Uniqueness does not hold for the more general permuted factorizations (4.23), (4.36), since there may be several permutation matrices that place a matrix in regular form. Moreover, unlike regular elimination, the pivots, i.e., the diagonal entries of U, are no longer uniquely defined, but depend on the particular combination of row interchanges employed during the course of the computation.

The $L\,D\,V$ factorization of a nonsingular matrix takes a particularly simple form if the matrix also happens to be symmetric. This result will form the foundation of some significant later developments.

Theorem 4.23. *A symmetric matrix A is regular if and only if it can be factored as*

$$A = L\,D\,L^T, \tag{4.37}$$

where L is a special lower triangular matrix and D is a diagonal matrix with nonzero diagonal entries.

Proof: We already know, according to Theorem 4.20, that we can factor

$$A = L\,D\,V. \tag{4.38}$$

We take the transpose of both sides of this equation:

$$A^T = (L\,D\,V)^T = V^T D^T L^T = V^T D\,L^T, \tag{4.39}$$

since diagonal matrices are automatically symmetric: $D^T = D$. Note that V^T is special lower triangular, and L^T is special upper triangular. Therefore (4.39) is the $L\,D\,V$ factorization of A^T.

In particular, if A is symmetric, then

$$L\,D\,V = A = A^T = V^T D\,L^T.$$

Uniqueness of the $L\,D\,V$ factorization implies that

$$L = V^T \qquad \text{and} \qquad V = L^T$$

(which are two versions of the same equation). Replacing V by L^T in (4.38) establishes the factorization (4.37). *Q.E.D.*

Remark: If $A = L\,D\,L^T$, then A is necessarily symmetric. Indeed,

$$A^T = (L\,D\,L^T)^T = (L^T)^T D^T L^T = L\,D\,L^T = A.$$

However, not every symmetric matrix has an $L\,D\,L^T$ factorization. A simple example is the irregular but nonsingular 2×2 matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

Example 4.24. The problem is to find the $L\,D\,L^T$ factorization of the particular symmetric matrix $A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 4 \end{pmatrix}$. This requires performing the usual Gaussian Elimination algorithm. Subtracting twice the first row from the second and also the first row from the third produces the matrix $\begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & -1 \\ 0 & -1 & 3 \end{pmatrix}$. We then add one half of the second row of the latter matrix to its third row, resulting in the upper triangular form

$$U = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & -1 \\ 0 & 0 & \frac{5}{2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & \frac{5}{2} \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & -\frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix} = D\,V,$$

which we further factor by dividing each row of U by its pivot. On the other hand, the special lower triangular matrix associated with the preceding row operations is $L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -\frac{1}{2} & 1 \end{pmatrix}$, which, as guaranteed by Theorem 4.23, is the transpose of $V = L^T$.

Therefore, the desired $A = LU = LDL^T$ factorizations of this particular symmetric matrix are

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -\frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & -1 \\ 0 & 0 & \frac{5}{2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -\frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & \frac{5}{2} \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & -\frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix}.$$

Example 4.25. Let us look at a general 2×2 symmetric matrix $A = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$. Regularity requires that the first pivot be $a \neq 0$. A single row operation will place A in upper triangular form $U = \begin{pmatrix} a & c \\ 0 & \dfrac{ac - b^2}{a} \end{pmatrix}$. The associated lower triangular matrix is $L = \begin{pmatrix} 1 & 0 \\ \dfrac{b}{a} & 1 \end{pmatrix}$. Thus, $A = LU$. Finally, $D = \begin{pmatrix} a & 0 \\ 0 & \dfrac{ac - b^2}{a} \end{pmatrix}$ is just the diagonal part of U, and we find $U = DL^T$, so that the LDL^T factorization is explicitly given by

$$\begin{pmatrix} a & b \\ b & c \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \dfrac{b}{a} & 1 \end{pmatrix} \begin{pmatrix} a & 0 \\ 0 & \dfrac{ac - b^2}{a} \end{pmatrix} \begin{pmatrix} 1 & \dfrac{b}{a} \\ 0 & 1 \end{pmatrix}. \tag{4.40}$$

4.5. Practical Linear Algebra.

For pedagogical and practical reasons, the examples and exercises we have chosen to illustrate the algorithms are all based on relatively small matrices. When dealing with matrices of moderate size, the differences between the various approaches to solving linear systems (Gauss, Gauss–Jordan, matrix inverse, etc.) are relatively unimportant, particularly if one has a decent computer or even hand calculator to do the tedious parts. However, real-world applied mathematics deals with much larger linear systems, and the design of efficient algorithms is a must. For example, numerical solution schemes for ordinary differential equations will typically lead to matrices with thousands of entries, while numerical schemes for partial differential equations arising in fluid and solid mechanics, weather prediction, image and video processing, quantum mechanics, molecular dynamics, chemical processes, etc., will often require dealing with matrices with more than a million entries. It is not hard for such systems to tax even the most sophisticated supercomputer. Thus, it is essential that we understand the computational details of competing methods in order to compare their efficiency, and thereby gain some experience with the issues underlying the design of high performance numerical algorithms.

The most basic question is: how many arithmetic operations[†] are required to complete an algorithm? The number will directly influence the time spent running the algorithm on a computer. We shall keep track of additions and multiplications separately, since the

[†] For simplicity, we will only count basic arithmetic operations. But it is worth noting that other issues, such as the number of I/O operations, may also play a role in estimating the computational complexity of a numerical algorithm.

latter typically take longer to process. But we shall not distinguish between addition and subtraction, nor between multiplication and division, as these typically rely on the same floating point algorithm. We shall also assume that the matrices and vectors we deal with are *generic*, with few, if any, zero entries. Modifications of the basic algorithms for *sparse matrices*, meaning those that have lots of zero entries, are an important topic of research, since these include many of the large matrices that appear in applications to differential equations. We refer the interested reader to more advanced treatments of numerical linear algebra, such as [**13**, **24**, **43**, **48**], for further developments.

First, when multiplying an $n \times n$ matrix A and an $n \times 1$ column vector $\mathbf{b}$, each entry of the product $A\mathbf{b}$ requires n multiplications of the form $a_{ij}\, b_j$ and $n-1$ additions to sum the resulting products. Since there are n entries, this means a total of n^2 multiplications and $n(n-1) = n^2 - n$ additions. Thus, for a matrix of size $n = 100$, one needs about $10,000$ distinct multiplications and a similar number of additions. If $n = 1,000,000 = 10^6$, then $n^2 = 10^{12}$, which is phenomenally large, and the total time required to perform the computation becomes a significant issue.

Let us next look at the (regular) Gaussian Elimination algorithm, referring back to our pseudocode program for the notational details. First, we count how many arithmetic operations are based on the j^{th} pivot m_{jj}. For each of the $n-j$ rows lying below it, we must perform one division to compute the factor $l_{ij} = m_{ij}/m_{jj}$ used in the elementary row operation. The entries in the column below the pivot will be set to zero automatically, and so we need only compute the updated entries lying strictly below and to the right of the pivot. There are $(n-j)^2$ such entries in the coefficient matrix and an additional $n-j$ entries in the last column of the augmented matrix. Let us concentrate on the former for the moment. For each of these, we replace m_{ik} by $m_{ik} - l_{ij}\, m_{jk}$, and so must perform one multiplication and one addition. For the j^{th} pivot, there are a total of $(n-j)(n-j+1)$ multiplications — including the initial $n-j$ divisions needed to produce the l_{ij} — and $(n-j)^2$ additions needed to update the coefficient matrix. Therefore, to reduce a regular $n \times n$ matrix to upper triangular form requires a total of

$$\begin{aligned} \sum_{j=1}^{n} (n-j)(n-j+1) &= \frac{n^3-n}{3} && \text{multiplications and} \\ \sum_{j=1}^{n} (n-j)^2 &= \frac{2n^3-3n^2+n}{6} && \text{additions.} \end{aligned} \tag{4.41}$$

Thus, when n is large, both involve approximately $\frac{1}{3}n^3$ operations.

We should also be keeping track of the number of operations on the right hand side of the system. No pivots appear there, and so there are

$$\sum_{j=1}^{n} (n-j) = \frac{n^2-n}{2} \tag{4.42}$$

multiplications and the same number of additions required to produce the right hand side in the resulting triangular system $U\mathbf{x} = \mathbf{c}$. For large n, this count is considerably smaller than the coefficient matrix totals (4.41). We note that the Forward Substitution equations

(4.17) require precisely the same number of arithmetic operations to solve $L\mathbf{c} = \mathbf{b}$ for the right hand side of the upper triangular system. Indeed, the j^{th} equation

$$c_j = b_j \; - \sum_{k=1}^{j-1} l_{jk}\, c_k$$

requires $j-1$ multiplications and the same number of additions, giving a total of

$$\sum_{j=1}^{n} j = \frac{n^2-n}{2}$$

operations of each type. Therefore, to reduce a linear system to upper triangular form, it makes no difference in computational efficiency whether one works directly with the augmented matrix or employs Forward Substitution after the LU factorization of the coefficient matrix has been established.

The Back Substitution phase of the algorithm can be similarly analyzed. To find the value of

$$x_j = \frac{1}{u_{jj}} \left(c_j - \sum_{k=j+1}^{n} u_{jk}\, x_k \right)$$

once we have computed $x_{j+1}, \ldots, x_n$, requires $n-j+1$ multiplications/divisions and $n-j$ additions. Therefore, the Back Substitution phase of the algorithm requires

$$\begin{aligned} &\sum_{j=1}^{n} (n-j+1) = \frac{n^2+n}{2} \qquad && \text{multiplications, along with} \\ &\sum_{j=1}^{n} (n-j) = \frac{n^2-n}{2} && \text{additions.} \end{aligned} \tag{4.43}$$

For n large, both of these are approximately equal to $\frac{1}{2}n^2$. Comparing the counts, we conclude that the bulk of the computational effort goes into the reduction of the coefficient matrix to upper triangular form.

Combining the two counts (4.42–43), we discover that, once we have computed the $A = LU$ decomposition of the coefficient matrix, the Forward and Back Substitution process requires n^2 multiplications and $n^2 - n$ additions to solve a linear system $A\mathbf{x} = \mathbf{b}$. This is exactly the *same* as the number of multiplications and additions needed to compute the product $A^{-1}\mathbf{b}$. Thus, even if we happen to know the inverse of A, it is still *just as efficient* to use Forward and Back Substitution to compute the solution!

On the other hand, the computation of A^{-1} is decidedly more inefficient. There are two possible strategies. First, we can solve the n linear systems

$$A\mathbf{x} = \mathbf{e}_i, \qquad i = 1, \ldots, n, \tag{4.44}$$

for the individual columns of A^{-1}. This requires first computing the LU decomposition, which uses about $\frac{1}{3}n^3$ multiplications and a similar number of additions, followed by applying Forward and Back Substitution to each of the systems, using $n \cdot n^2 = n^3$ multiplications and $n(n^2-n) \approx n^3$ additions, for a grand total of about $\frac{4}{3}n^3$ operations of each type in order to compute A^{-1}. Gauss–Jordan Elimination fares no better (in fact, slightly worse), also requiring about the same number, $\frac{4}{3}n^3$, of each type of arithmetic operation. Both algorithms can be made more efficient by exploiting the fact that there are lots of zeros on the right hand sides of the systems (4.44). Designing the algorithm to avoid adding or subtracting a preordained 0, or multiplying or dividing by a preordained ± 1, reduces the total number of operations required to compute A^{-1} to exactly n^3 multiplications and $n(n-1)^2 \approx n^3$ additions. And don't forget we still need to multiply $A^{-1}\mathbf{b}$ to solve the original system. As a result, solving a linear system with the inverse matrix requires approximately *three* times as many arithmetic operations, and so would take three times as long to complete, as the more elementary Gaussian Elimination and Back Substitution algorithm. This justifies our earlier contention that matrix inversion is inefficient, and, except in very special situations, should never be used for solving linear systems in practice.

Tridiagonal Matrices

Of course, in special cases, the actual arithmetic operation count might be considerably reduced, particularly if A is a sparse matrix with many zero entries. A number of specialized techniques have been designed to handle sparse linear systems. A particularly important class are the *tridiagonal matrices*

$$A = \begin{pmatrix} q_1 & r_1 & & & & \\ p_1 & q_2 & r_2 & & & \\ & p_2 & q_3 & r_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & p_{n-2} & q_{n-1} & r_{n-1} \\ & & & & p_{n-1} & q_n \end{pmatrix} \tag{4.45}$$

with all entries zero except for those on the main diagonal, namely $a_{ii} = q_i$, the *subdiagonal*, meaning the $n-1$ entries $a_{i+1,i} = p_i$ immediately below the main diagonal, and the *superdiagonal*, meaning the entries $a_{i,i+1} = r_i$ immediately above the main diagonal. (Blank entries indicate a 0.) Such matrices arise in the numerical solution of ordinary differential equations and the spline fitting of curves for interpolation and computer graphics. If $A = LU$ is regular, it turns out that the factors are lower and upper *bidiagonal matrices*, of the form

$$L = \begin{pmatrix} 1 & & & & & \\ l_1 & 1 & & & & \\ & l_2 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & l_{n-2} & 1 & \\ & & & & l_{n-1} & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} d_1 & u_1 & & & & \\ & d_2 & u_2 & & & \\ & & d_3 & u_3 & & \\ & & & \ddots & \ddots & \\ & & & & d_{n-1} & u_{n-1} \\ & & & & & d_n \end{pmatrix}. \tag{4.46}$$

Multiplying out LU and equating the result to A leads to the equations

$$\begin{aligned}
&d_1 = q_1, && u_1 = r_1, && l_1 d_1 = p_1, \\
&l_1 u_1 + d_2 = q_2, && u_2 = r_2, && l_2 d_2 = p_2, \\
&\quad\vdots && \quad\vdots && \quad\vdots \\
&l_{j-1} u_{j-1} + d_j = q_j, && u_j = r_j, && l_j d_j = p_j, \\
&\quad\vdots && \quad\vdots && \quad\vdots \\
&l_{n-2} u_{n-2} + d_{n-1} = q_{n-1}, && u_{n-1} = r_{n-1}, && l_{n-1} d_{n-1} = p_{n-1}, \\
&l_{n-1} u_{n-1} + d_n = q_n.
\end{aligned} \tag{4.47}$$

These elementary algebraic equations can be successively solved for the entries of L and U in the following order: $d_1, u_1, l_1, d_2, u_2, l_2, d_3, u_3 \ldots$. The original matrix A is regular provided none of the diagonal entries $d_1, d_2, \ldots$ are zero, which allows the recursive procedure to successfully proceed to termination.

Once the LU factors are in place, we can apply Forward and Back Substitution to solve the tridiagonal linear system $A\mathbf{x} = \mathbf{b}$. We first solve the lower triangular system $L\mathbf{c} = \mathbf{b}$ by Forward Substitution, which leads to the recursive equations

$$c_1 = b_1, \qquad c_2 = b_2 - l_1 c_1, \qquad \ldots \qquad c_n = b_n - l_{n-1} c_{n-1}. \tag{4.48}$$

We then solve the upper triangular system $U\mathbf{x} = \mathbf{c}$ by Back Substitution, again recursively:

$$x_n = \frac{c_n}{d_n}, \qquad x_{n-1} = \frac{c_{n-1} - u_{n-1} x_n}{d_{n-1}}, \qquad \ldots \qquad x_1 = \frac{c_1 - u_1 x_2}{d_1}. \tag{4.49}$$

As you can check, there are a total of $5n - 4$ multiplications/divisions and $3n - 3$ additions/subtractions required to solve a general tridiagonal system of n linear equations — a striking improvement over the general case.

Example 4.26. Consider the $n \times n$ tridiagonal matrix

$$A = \begin{pmatrix} 4 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 4 \end{pmatrix}$$

in which the diagonal entries are all $q_i = 4$, while the entries immediately above and below the main diagonal are all $p_i = r_i = 1$. According to (4.47), the tridiagonal factorization (4.46) has $u_1 = u_2 = \ldots = u_{n-1} = 1$, while

$$d_1 = 4, \qquad l_j = 1/d_j, \qquad d_{j+1} = 4 - l_j, \qquad j = 1, 2, \ldots, n-1.$$

The computed values are

j	1	2	3	4	5	6	7
d_j	4.0	3.75	3.733333	3.732143	3.732057	3.732051	3.732051
l_j	.25	.266666	.267857	.267942	.267948	.267949	.267949

These converge rapidly to

$$d_j \quad \longrightarrow \quad 2+\sqrt{3} = 3.732051\ldots\,, \qquad\qquad l_j \quad \longrightarrow \quad 2-\sqrt{3} = .267949\ldots\,,$$

which makes the factorization for large n almost trivial. The numbers $2 \pm \sqrt{3}$ are the roots of the quadratic equation $x^2 - 4x + 1 = 0$, and are characterized as the fixed points of the nonlinear iterative system $d_{j+1} = 4 - 1/d_j$.

Pivoting Strategies

Let us now investigate the practical side of pivoting. As we know, in the irregular situations when a zero shows up in a diagonal pivot position, a row interchange is required to proceed with the elimination algorithm. But even when a nonzero pivot element is in place, there may be good numerical reasons for exchanging rows in order to install a more desirable element in the pivot position. Here is a simple example:

$$.01\,x + 1.6\,y = 32.1, \qquad\qquad x + .6\,y = 22. \tag{4.50}$$

The exact solution to the system is easily found:

$$x = 10, \qquad\qquad y = 20.$$

Suppose we are working with a very primitive calculator that only retains 3 digits of accuracy. (Of course, this is not a very realistic situation, but the example could be suitably modified to produce similar difficulties no matter how many digits of accuracy our computer retains.) The augmented matrix is

$$\left(\begin{array}{cc|c} .01 & 1.6 & 32.1 \\ 1 & .6 & 22 \end{array}\right).$$

Choosing the $(1,1)$ entry as our pivot, and subtracting 100 times the first row from the second produces the upper triangular form

$$\left(\begin{array}{cc|c} .01 & 1.6 & 32.1 \\ 0 & -159.4 & -3188 \end{array}\right).$$

Since our calculator has only three–place accuracy, it will round the entries in the second row, producing the augmented coefficient matrix

$$\left(\begin{array}{cc|c} .01 & 1.6 & 32.1 \\ 0 & -159.0 & -3190 \end{array}\right).$$

The solution by Back Substitution gives

$$\begin{aligned} &y = 3190/159 = 20.0628\ldots \simeq\ 20.1, \qquad\qquad \text{and then} \\ &x = 100\,(32.1 - 1.6\,y) = 100\,(32.1 - 32.16) \simeq 100\,(32.1 - 32.2) = -10. \end{aligned}$$

Gaussian Elimination With Partial Pivoting

```
start
    for $i = 1$ to $n$
        set $\rho(i) = i$
    next $i$
    for $j = 1$ to $n$
        if $m_{\rho(i),j} = 0$ for all $i \geq j$, stop; print "$A$ is singular"
        choose $i > j$ such that $m_{\rho(i),j}$ is maximal
        interchange $\rho(i) \longleftrightarrow \rho(j)$
        for $i = j+1$ to $n$
            set $l_{\rho(i)j} = m_{\rho(i)j} / m_{\rho(j)j}$
            for $k = j+1$ to $n+1$
                set $m_{\rho(i)k} = m_{\rho(i)k} - l_{\rho(i)j} m_{\rho(j)k}$
            next $k$
        next $i$
    next $j$
end
```

The relatively small error in y has produced a very large error in x — not even its sign is correct!

The problem is that the first pivot, .01, is much smaller than the other element, 1, that appears in the column below it. Interchanging the two rows before performing the row operation would resolve the difficulty — even with such an inaccurate calculator! After the interchange, we have

$$\left(\begin{array}{cc|c} 1 & .6 & 22 \\ .01 & 1.6 & 32.1 \end{array}\right),$$

which results in the rounded-off upper triangular form

$$\left(\begin{array}{cc|c} 1 & .6 & 22 \\ 0 & 1.594 & 31.88 \end{array}\right) \simeq \left(\begin{array}{cc|c} 1 & .6 & 22 \\ 0 & 1.59 & 31.9 \end{array}\right).$$

The solution by Back Substitution now gives a respectable answer:

$$y = 31.9/1.59 = 20.0628\ldots \simeq 20.1, \qquad x = 22 - .6\,y = 22 - 12.06 \simeq 22 - 12.1 = 9.9.$$

The general strategy, known as *Partial Pivoting*, says that at each stage, we should use the largest (in absolute value) legitimate (i.e., in the pivot column on or below the diagonal) element as the pivot, even if the diagonal element is nonzero. Partial pivoting can help suppress the undesirable effects of round-off errors during the computation.

In a computer implementation of pivoting, there is no need to waste processor time physically exchanging the row entries in memory. Rather, one introduces a separate array of pointers that serve to indicate which original row is currently in which permuted position. More concretely, one initializes n row pointers $\rho(1) = 1, \ldots, \rho(n) = n$. Interchanging row i and row j of the coefficient or augmented matrix is then accomplished by merely interchanging $\rho(i)$ and $\rho(j)$. Thus, to access a matrix element that is currently in row i of the augmented matrix, one merely retrieves the element that is in row $\rho(i)$ in the computer's memory. An explicit implementation of this strategy is provided in the accompanying pseudocode program.

Partial pivoting will solve most problems, although there can still be difficulties. For instance, it does not accurately solve the system

$$10\,x + 1600\,y = 3210, \qquad x + .6\,y = 22,$$

obtained by multiplying the first equation in (4.50) by 1000. The tip-off is that, while the entries in the column containing the pivot are smaller, those in its row are much larger. The solution to this difficulty is *Full Pivoting*, in which one also performs column interchanges — preferably with a column pointer — to move the largest legitimate element into the pivot position. In practice, a column interchange amounts to reordering the variables in the system, which, as long as one keeps proper track of the order, also doesn't change the solutions. Thus, switching the order of x, y leads to the augmented matrix $\left(\begin{array}{cc|c} 1600 & 10 & 3210 \\ .6 & 1 & 22 \end{array}\right)$ in which the first column now refers to y and the second to x. Now Gaussian Elimination will produce a reasonably accurate solution to the system.

Finally, there are some matrices that are hard to handle even with sophisticated pivoting strategies. Such *ill-conditioned* matrices are typically characterized by being "almost" singular. A famous example of an ill-conditioned matrix is the $n \times n$ *Hilbert matrix*

$$H_n = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \cdots & \frac{1}{n+2} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \cdots & \frac{1}{n+3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \frac{1}{n+3} & \cdots & \frac{1}{2n-1} \end{pmatrix}. \tag{4.51}$$

It can be shown that H_n is nonsingular for all n. However, the solution of a linear system whose coefficient matrix is a Hilbert matrix H_n, even for moderately large n, is a very challenging problem, even using high precision computer arithmetic. This is because the larger n is, the closer H_n is, in a sense, to being singular.

The reader is urged to try the following computer experiment. Fix a moderately large value of n, say 20. Choose a column vector $\mathbf{x}$ with n entries chosen at random. Compute

$\mathbf{b} = H_n \mathbf{x}$ directly. Then try to solve the system $H_n \mathbf{x} = \mathbf{b}$ by Gaussian Elimination, and compare the result with the original vector $\mathbf{x}$. If you obtain an accurate solution with $n = 20$, try $n = 50$ or 100. This will give you a good indicator of the degree of arithmetic precision used by your computer hardware, and the accuracy of the numerical solution algorithm(s) in your software.

AIMS Lecture Notes 2006

Peter J. Olver

5. Inner Products and Norms

The norm of a vector is a measure of its size. Besides the familiar Euclidean norm based on the dot product, there are a number of other important norms that are used in numerical analysis. In this section, we review the basic properties of inner products and norms.

5.1. Inner Products.

Some, but not all, norms are based on inner products. The most basic example is the familiar *dot product*

$$\langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v} \cdot \mathbf{w} = v_1 w_1 + v_2 w_2 + \cdots + v_n w_n = \sum_{i=1}^{n} v_i w_i, \tag{5.1}$$

between (column) vectors $\mathbf{v} = (v_1, v_2, \ldots, v_n)^T$, $\mathbf{w} = (w_1, w_2, \ldots, w_n)^T$, lying in the Euclidean space $\mathbb{R}^n$. A key observation is that the dot product (5.1) is equal to the matrix product

$$\mathbf{v} \cdot \mathbf{w} = \mathbf{v}^T \mathbf{w} = \begin{pmatrix} v_1 & v_2 & \ldots & v_n \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} \tag{5.2}$$

between the row vector $\mathbf{v}^T$ and the column vector $\mathbf{w}$. The key fact is that the dot product of a vector with itself,

$$\mathbf{v} \cdot \mathbf{v} = v_1^2 + v_2^2 + \cdots + v_n^2,$$

is the sum of the squares of its entries, and hence, by the classical Pythagorean Theorem, equals the square of its length; see Figure 5.1. Consequently, the *Euclidean norm* or *length* of a vector is found by taking the square root:

$$\| \mathbf{v} \| = \sqrt{\mathbf{v} \cdot \mathbf{v}} = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}\,. \tag{5.3}$$

Note that every nonzero vector $\mathbf{v} \neq \mathbf{0}$ has positive Euclidean norm, $\| \mathbf{v} \| > 0$, while only the zero vector has zero norm: $\| \mathbf{v} \| = 0$ if and only if $\mathbf{v} = \mathbf{0}$. The elementary properties of dot product and Euclidean norm serve to inspire the abstract definition of more general inner products.

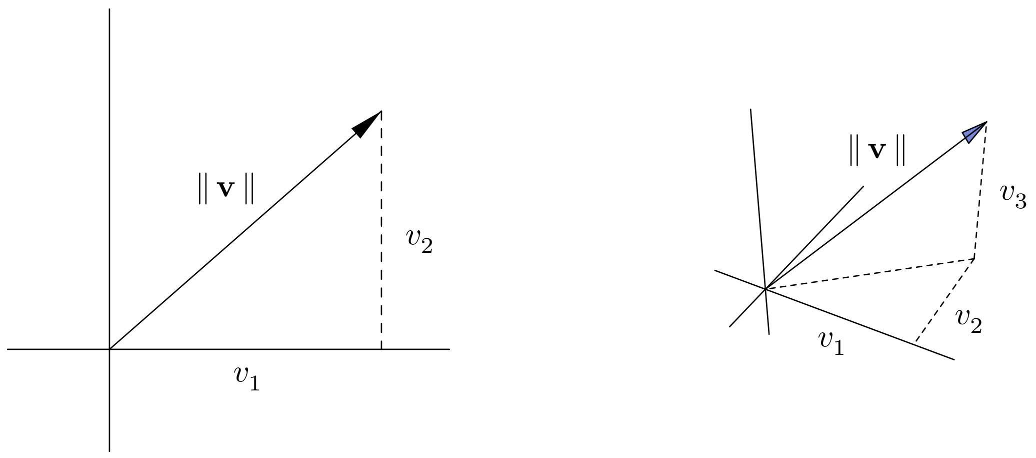


Figure 5.1. The Euclidean Norm in $\mathbb{R}^2$ and $\mathbb{R}^3$.

Definition 5.1. An *inner product* on the vector space $\mathbb{R}^n$ is a pairing that takes two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ and produces a real number $\langle \mathbf{v}, \mathbf{w} \rangle \in \mathbb{R}$. The inner product is required to satisfy the following three axioms for all $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$, and scalars $c, d \in \mathbb{R}$.

(i) *Bilinearity*:

$$\begin{aligned} \langle c\,\mathbf{u} + d\,\mathbf{v}, \mathbf{w} \rangle &= c\,\langle \mathbf{u}, \mathbf{w} \rangle + d\,\langle \mathbf{v}, \mathbf{w} \rangle, \\ \langle \mathbf{u}, c\,\mathbf{v} + d\,\mathbf{w} \rangle &= c\,\langle \mathbf{u}, \mathbf{v} \rangle + d\,\langle \mathbf{u}, \mathbf{w} \rangle. \end{aligned} \tag{5.4}$$

(ii) *Symmetry*:

$$\langle \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{w}, \mathbf{v} \rangle. \tag{5.5}$$

(iii) *Positivity*:

$$\langle \mathbf{v}, \mathbf{v} \rangle > 0 \qquad \text{whenever} \qquad \mathbf{v} \neq \mathbf{0}, \qquad \text{while} \qquad \langle \mathbf{0}, \mathbf{0} \rangle = 0. \tag{5.6}$$

Given an inner product, the associated *norm* of a vector $\mathbf{v} \in V$ is defined as the positive square root of the inner product of the vector with itself:

$$\| \mathbf{v} \| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}\,. \tag{5.7}$$

The positivity axiom implies that $\| \mathbf{v} \| \geq 0$ is real and non-negative, and equals 0 if and only if $\mathbf{v} = \mathbf{0}$ is the zero vector.

Example 5.2. While certainly the most common inner product on $\mathbb{R}^2$, the dot product

$$\mathbf{v} \cdot \mathbf{w} = v_1\, w_1 + v_2\, w_2$$

is by no means the only possibility. A simple example is provided by the *weighted inner product*

$$\langle \mathbf{v}, \mathbf{w} \rangle = 2\,v_1\, w_1 + 5\,v_2\, w_2, \qquad \mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, \qquad \mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}. \tag{5.8}$$

Let us verify that this formula does indeed define an inner product. The symmetry axiom (5.5) is immediate. Moreover,

$$\begin{aligned} \langle c\,\mathbf{u} + d\,\mathbf{v}, \mathbf{w} \rangle &= 2\,(c u_1 + d v_1)\, w_1 + 5\,(c u_2 + d v_2)\, w_2 \\ &= c(2\,u_1\, w_1 + 5\,u_2\, w_2) + d(2\,v_1\, w_1 + 5\,v_2\, w_2) = c\,\langle \mathbf{u}, \mathbf{w} \rangle + d\,\langle \mathbf{v}, \mathbf{w} \rangle, \end{aligned}$$

which verifies the first bilinearity condition; the second follows by a very similar computation. Moreover, $\langle \mathbf{0}, \mathbf{0} \rangle = 0$, while

$$\langle \mathbf{v}, \mathbf{v} \rangle = 2v_1^2 + 5v_2^2 > 0 \qquad \text{whenever} \qquad \mathbf{v} \neq \mathbf{0},$$

since at least one of the summands is strictly positive. This establishes (5.8) as a legitimate inner product on $\mathbb{R}^2$. The associated *weighted norm* $\| \mathbf{v} \| = \sqrt{2v_1^2 + 5v_2^2}$ defines an alternative, "non-Pythagorean" notion of length of vectors and distance between points in the plane.

A less evident example of an inner product on $\mathbb{R}^2$ is provided by the expression

$$\langle \mathbf{v}, \mathbf{w} \rangle = v_1 w_1 - v_1 w_2 - v_2 w_1 + 4 v_2 w_2. \tag{5.9}$$

Bilinearity is verified in the same manner as before, and symmetry is immediate. Positivity is ensured by noticing that

$$\langle \mathbf{v}, \mathbf{v} \rangle = v_1^2 - 2v_1 v_2 + 4v_2^2 = (v_1 - v_2)^2 + 3v_2^2 \geq 0$$

is always non-negative, and, moreover, is equal to zero if and only if $v_1 - v_2 = 0, v_2 = 0$, i.e., only when $\mathbf{v} = \mathbf{0}$. We conclude that (5.9) defines yet another inner product on $\mathbb{R}^2$, with associated norm

$$\| \mathbf{v} \| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle} = \sqrt{v_1^2 - 2v_1 v_2 + 4v_2^2}\,.$$

The second example (5.8) is a particular case of a general class of inner products.

Example 5.3. Let $c_1, \ldots, c_n > 0$ be a set of *positive* numbers. The corresponding *weighted inner product* and *weighted norm* on $\mathbb{R}^n$ are defined by

$$\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{i=1}^{n} c_i v_i w_i, \qquad \| \mathbf{v} \| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle} = \sqrt{\sum_{i=1}^{n} c_i v_i^2}\,. \tag{5.10}$$

The numbers c_i are the *weights*. Observe that the larger the weight c_i, the more the i^{th} coordinate of $\mathbf{v}$ contributes to the norm. We can rewrite the weighted inner product in the useful vector form

$$\langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v}^T C \mathbf{w}, \qquad \text{where} \qquad C = \begin{pmatrix} c_1 & 0 & 0 & \ldots & 0 \\ 0 & c_2 & 0 & \ldots & 0 \\ 0 & 0 & c_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & c_n \end{pmatrix} \tag{5.11}$$

is the diagonal *weight matrix*. Weighted norms are particularly relevant in statistics and data fitting, [**12**], where one wants to emphasize certain quantities and de-emphasize others; this is done by assigning appropriate weights to the different components of the data vector $\mathbf{v}$.

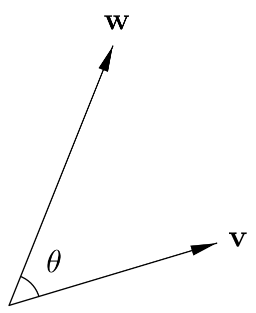


Figure 5.2. Angle Between Two Vectors.

5.2. Inequalities.

There are two absolutely fundamental inequalities that are valid for *any* inner product on any vector space. The first is inspired by the geometric interpretation of the dot product on Euclidean space in terms of the angle between vectors. It is named after two of the founders of modern analysis, Augustin Cauchy and Herman Schwarz, who established it in the case of the L^2 inner product on function space[†]. The more familiar triangle inequality, that the length of any side of a triangle is bounded by the sum of the lengths of the other two sides is, in fact, an immediate consequence of the Cauchy–Schwarz inequality, and hence also valid for any norm based on an inner product.

The Cauchy–Schwarz Inequality

In Euclidean geometry, the dot product between two vectors can be geometrically characterized by the equation

$$\mathbf{v} \cdot \mathbf{w} = \| \mathbf{v} \| \, \| \mathbf{w} \| \, \cos\theta, \tag{5.12}$$

where θ measures the angle between the vectors $\mathbf{v}$ and $\mathbf{w}$, as drawn in Figure 5.2. Since

$$| \cos\theta | \le 1,$$

the absolute value of the dot product is bounded by the product of the lengths of the vectors:

$$| \mathbf{v} \cdot \mathbf{w} | \ \le \ \| \mathbf{v} \| \, \| \mathbf{w} \|.$$

This is the simplest form of the general *Cauchy–Schwarz inequality*. We present a simple, algebraic proof that does not rely on the geometrical notions of length and angle and thus demonstrates its universal validity for *any* inner product.

Theorem 5.4. *Every inner product satisfies the Cauchy–Schwarz inequality*

$$| \langle \mathbf{v} , \mathbf{w} \rangle | \ \le \ \| \mathbf{v} \| \, \| \mathbf{w} \|, \qquad \textit{for all} \qquad \mathbf{v}, \mathbf{w} \in V. \tag{5.13}$$

Here, $\| \mathbf{v} \|$ is the associated norm, while $| \cdot |$ denotes absolute value of real numbers. Equality holds if and only if $\mathbf{v}$ and $\mathbf{w}$ are parallel vectors.

[†] Russians also give credit for its discovery to their compatriot Viktor Bunyakovskii, and, indeed, some authors append his name to the inequality.

Proof: The case when $\mathbf{w} = \mathbf{0}$ is trivial, since both sides of (5.13) are equal to 0. Thus, we may suppose $\mathbf{w} \neq \mathbf{0}$. Let $t \in \mathbb{R}$ be an arbitrary scalar. Using the three inner product axioms, we have

$$0 \leq \| \mathbf{v} + t\,\mathbf{w} \|^2 = \langle \mathbf{v} + t\,\mathbf{w}\,, \mathbf{v} + t\,\mathbf{w} \rangle = \| \mathbf{v} \|^2 + 2t\,\langle \mathbf{v}\,, \mathbf{w} \rangle + t^2\, \| \mathbf{w} \|^2, \tag{5.14}$$

with equality holding if and only if $\mathbf{v} = -\,t\,\mathbf{w}$ — which requires $\mathbf{v}$ and $\mathbf{w}$ to be parallel vectors. We fix $\mathbf{v}$ and $\mathbf{w}$, and consider the right hand side of (5.14) as a quadratic function,

$$0 \leq p(t) = a t^2 + 2\,b\,t + c, \qquad \text{where} \qquad a = \| \mathbf{w} \|^2, \qquad b = \langle \mathbf{v}\,, \mathbf{w} \rangle, \qquad c = \| \mathbf{v} \|^2,$$

of the scalar variable t. To get the maximum mileage out of the fact that $p(t) \geq 0$, let us look at where it assumes its minimum, which occurs when its derivative is zero:

$$p'(t) = 2\,a\,t + 2\,b = 0, \qquad \text{and so} \qquad t \;=\; -\,\frac{b}{a} \;=\; -\,\frac{\langle \mathbf{v}\,, \mathbf{w} \rangle}{\| \mathbf{w} \|^2}\,.$$

Substituting this particular value of t into (5.14), we find

$$0 \;\leq\; \| \mathbf{v} \|^2 - 2\,\frac{\langle \mathbf{v}\,, \mathbf{w} \rangle^2}{\| \mathbf{w} \|^2} + \frac{\langle \mathbf{v}\,, \mathbf{w} \rangle^2}{\| \mathbf{w} \|^2} \;=\; \| \mathbf{v} \|^2 - \frac{\langle \mathbf{v}\,, \mathbf{w} \rangle^2}{\| \mathbf{w} \|^2}\,.$$

Rearranging this last inequality, we conclude that

$$\frac{\langle \mathbf{v}\,, \mathbf{w} \rangle^2}{\| \mathbf{w} \|^2} \;\leq\; \| \mathbf{v} \|^2, \qquad \text{or} \qquad \langle \mathbf{v}\,, \mathbf{w} \rangle^2 \;\leq\; \| \mathbf{v} \|^2\, \| \mathbf{w} \|^2.$$

Also, as noted above, equality holds if and only if $\mathbf{v}$ and $\mathbf{w}$ are parallel. Taking the (positive) square root of both sides of the final inequality completes the proof of the Cauchy–Schwarz inequality (5.13). *Q.E.D.*

Given any inner product, we can use the quotient

$$\cos\theta = \frac{\langle \mathbf{v}\,, \mathbf{w} \rangle}{\| \mathbf{v} \|\; \| \mathbf{w} \|} \tag{5.15}$$

to define the "angle" between the vector space elements $\mathbf{v}, \mathbf{w} \in V$. The Cauchy–Schwarz inequality tells us that the ratio lies between -1 and $+1$, and hence the angle θ is well defined, and, in fact, unique if we restrict it to lie in the range $0 \leq \theta \leq \pi$.

For example, the vectors $\mathbf{v} = (\,1, 0, 1\,)^T$, $\mathbf{w} = (\,0, 1, 1\,)^T$ have dot product $\mathbf{v} \cdot \mathbf{w} = 1$ and norms $\| \mathbf{v} \| = \| \mathbf{w} \| = \sqrt{2}$. Hence the Euclidean angle between them is given by

$$\cos\theta = \frac{1}{\sqrt{2} \cdot \sqrt{2}} = \frac{1}{2}\,, \qquad \text{and so} \qquad \theta = \tfrac{1}{3}\,\pi = 1.0472\ldots\,.$$

On the other hand, if we adopt the weighted inner product $\langle \mathbf{v}\,, \mathbf{w} \rangle = v_1 w_1 + 2 v_2 w_2 + 3 v_3 w_3$, then $\mathbf{v} \cdot \mathbf{w} = 3$, $\| \mathbf{v} \| = 2$, $\| \mathbf{w} \| = \sqrt{5}$, and hence their "weighted" angle becomes

$$\cos\theta = \frac{3}{2\,\sqrt{5}} = .67082\ldots\,, \qquad \text{with} \qquad \theta = .835482\ldots\,.$$

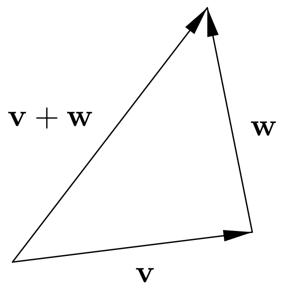


Figure 5.3. Triangle Inequality.

Thus, the measurement of angle (and length) is dependent upon the choice of an underlying inner product.

In Euclidean geometry, perpendicular vectors meet at a right angle, $\theta = \frac{1}{2}\pi$ or $\frac{3}{2}\pi$, with $\cos\theta = 0$. The angle formula (5.12) implies that the vectors $\mathbf{v}, \mathbf{w}$ are perpendicular if and only if their dot product vanishes: $\mathbf{v}\cdot\mathbf{w} = 0$. Perpendicularity is of interest in general inner product spaces, but, for historical reasons, has been given a more suggestive name.

Definition 5.5. Two elements $\mathbf{v}, \mathbf{w} \in V$ of an inner product space V are called *orthogonal* if their inner product vanishes: $\langle\, \mathbf{v}\,, \mathbf{w}\,\rangle = 0$.

In particular, the zero element is orthogonal to everything: $\langle\, \mathbf{0}\,, \mathbf{v}\,\rangle = 0$ for all $\mathbf{v} \in V$. Orthogonality is a remarkably powerful tool in all applications of linear algebra, and often serves to dramatically simplify many computations.

The Triangle Inequality

The familiar triangle inequality states that the length of one side of a triangle is at most equal to the sum of the lengths of the other two sides. Referring to Figure 5.3, if the first two sides are represented by vectors $\mathbf{v}$ and $\mathbf{w}$, then the third corresponds to their sum $\mathbf{v}+\mathbf{w}$. The triangle inequality turns out to be an elementary consequence of the Cauchy–Schwarz inequality, and hence is valid in *any* inner product space.

Theorem 5.6. *The norm associated with an inner product satisfies the* triangle inequality

$$\|\,\mathbf{v}+\mathbf{w}\,\| \;\le\; \|\,\mathbf{v}\,\| + \|\,\mathbf{w}\,\| \qquad \textit{for all} \qquad \mathbf{v}, \mathbf{w} \in V. \tag{5.16}$$

Equality holds if and only if $\mathbf{v}$ *and* $\mathbf{w}$ *are parallel vectors.*

Proof: We compute

$$\begin{aligned}\|\,\mathbf{v}+\mathbf{w}\,\|^2 &= \langle\, \mathbf{v}+\mathbf{w}\,, \mathbf{v}+\mathbf{w}\,\rangle = \|\,\mathbf{v}\,\|^2 + 2\,\langle\, \mathbf{v}\,, \mathbf{w}\,\rangle + \|\,\mathbf{w}\,\|^2 \\ &\le \|\,\mathbf{v}\,\|^2 + 2\,|\,\langle\, \mathbf{v}\,, \mathbf{w}\,\rangle\,| + \|\,\mathbf{w}\,\|^2 \le \|\,\mathbf{v}\,\|^2 + 2\,\|\,\mathbf{v}\,\|\,\|\,\mathbf{w}\,\| + \|\,\mathbf{w}\,\|^2 \\ &= \big(\,\|\,\mathbf{v}\,\| + \|\,\mathbf{w}\,\|\,\big)^2,\end{aligned}$$

where the middle inequality follows from Cauchy–Schwarz. Taking square roots of both sides and using positivity completes the proof. *Q.E.D.*

Example 5.7. The vectors $\mathbf{v} = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}$ and $\mathbf{w} = \begin{pmatrix} 2 \\ 0 \\ 3 \end{pmatrix}$ sum to $\mathbf{v}+\mathbf{w} = \begin{pmatrix} 3 \\ 2 \\ 2 \end{pmatrix}$. Their Euclidean norms are $\|\mathbf{v}\| = \sqrt{6}$ and $\|\mathbf{w}\| = \sqrt{13}$, while $\|\mathbf{v}+\mathbf{w}\| = \sqrt{17}$. The triangle inequality (5.16) in this case says $\sqrt{17} \le \sqrt{6} + \sqrt{13}$, which is valid.

5.3. Norms.

Every inner product gives rise to a norm that can be used to measure the magnitude or length of the elements of the underlying vector space. However, not every norm that is used in analysis and applications arises from an inner product. To define a general norm, we will extract those properties that do not directly rely on the inner product structure.

Definition 5.8. A *norm* on the vector space $\mathbb{R}^n$ assigns a real number $\|\mathbf{v}\|$ to each vector $\mathbf{v} \in V$, subject to the following axioms for every $\mathbf{v}, \mathbf{w} \in V$, and $c \in \mathbb{R}$.

(*i*) *Positivity*: $\|\mathbf{v}\| \ge 0$, with $\|\mathbf{v}\| = 0$ if and only if $\mathbf{v} = \mathbf{0}$.

(*ii*) *Homogeneity*: $\|c\,\mathbf{v}\| = |c|\,\|\mathbf{v}\|$.

(*iii*) *Triangle inequality*: $\|\mathbf{v}+\mathbf{w}\| \le \|\mathbf{v}\| + \|\mathbf{w}\|$.

As we now know, every inner product gives rise to a norm. Indeed, positivity of the norm is one of the inner product axioms. The homogeneity property follows since

$$\|c\,\mathbf{v}\| = \sqrt{\langle c\,\mathbf{v}\,, c\,\mathbf{v}\rangle} = \sqrt{c^2\,\langle \mathbf{v}\,,\mathbf{v}\rangle} = |c|\,\sqrt{\langle \mathbf{v}\,,\mathbf{v}\rangle} = |c|\,\|\mathbf{v}\|.$$

Finally, the triangle inequality for an inner product norm was established in Theorem 5.6. Let us introduce some of the principal examples of norms that do not come from inner products.

The 1*–norm* of a vector $\mathbf{v} = (v_1, v_2, \ldots, v_n)^T \in \mathbb{R}^n$ is defined as the sum of the absolute values of its entries:

$$\|\mathbf{v}\|_1 = |v_1| + |v_2| + \cdots + |v_n|. \tag{5.17}$$

The *max* or ∞*–norm* is equal to its maximal entry (in absolute value):

$$\|\mathbf{v}\|_\infty = \max\{\,|v_1|, |v_2|, \ldots, |v_n|\,\}. \tag{5.18}$$

Verification of the positivity and homogeneity properties for these two norms is straightforward; the triangle inequality is a direct consequence of the elementary inequality

$$|a+b| \le |a| + |b|, \qquad a, b \in \mathbb{R},$$

for absolute values.

The Euclidean norm, 1–norm, and ∞–norm on $\mathbb{R}^n$ are just three representatives of the general *p–norm*

$$\|\mathbf{v}\|_p = \sqrt[p]{\sum_{i=1}^{n} |v_i|^p}\,. \tag{5.19}$$

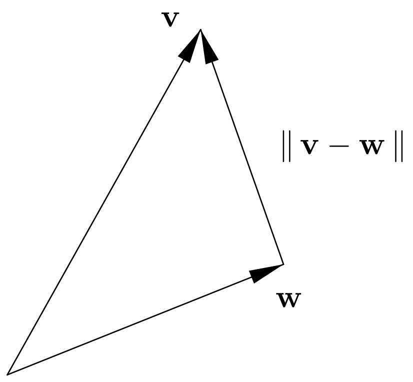


Figure 5.4. Distance Between Vectors.

This quantity defines a norm for any $1 \le p < \infty$. The ∞–norm is a limiting case of (5.19) as $p \to \infty$. Note that the Euclidean norm (5.3) is the 2–norm, and is often designated as such; it is the only p–norm which comes from an inner product. The positivity and homogeneity properties of the p–norm are not hard to establish. The triangle inequality, however, is not trivial; in detail, it reads

$$\sqrt[p]{\sum_{i=1}^{n} |\,v_i + w_i\,|^p} \;\le\; \sqrt[p]{\sum_{i=1}^{n} |\,v_i\,|^p} + \sqrt[p]{\sum_{i=1}^{n} |\,w_i\,|^p}\,, \tag{5.20}$$

and is known as *Minkowski's inequality*. A complete proof can be found in [**30**].

Every norm defines a *distance* between vector space elements, namely

$$d(\mathbf{v},\mathbf{w}) = \|\,\mathbf{v}-\mathbf{w}\,\|. \tag{5.21}$$

For the standard dot product norm, we recover the usual notion of distance between points in Euclidean space. Other types of norms produce alternative (and sometimes quite useful) notions of distance that are, nevertheless, subject to all the familiar properties:

(a) *Symmetry*: $d(\mathbf{v},\mathbf{w}) = d(\mathbf{w},\mathbf{v})$;
(b) *Positivity*: $d(\mathbf{v},\mathbf{w}) = 0$ if and only if $\mathbf{v} = \mathbf{w}$;
(c) *Triangle Inequality*: $d(\mathbf{v},\mathbf{w}) \le d(\mathbf{v},\mathbf{z}) + d(\mathbf{z},\mathbf{w})$.

Equivalence of Norms

While there are many different types of norms on $\mathbb{R}^n$, in a certain sense, they are all more or less equivalent[†]. "Equivalence" does not mean that they assume the same value, but rather that they are always close to one another, and so, for many analytical purposes, may be used interchangeably. As a consequence, we may be able to simplify the analysis of a problem by choosing a suitably adapted norm.

[†] This statement remains valid in any finite-dimensional vector space, but is *not* correct in infinite-dimensional function spaces.

Theorem 5.9. *Let* $\| \cdot \|_1$ *and* $\| \cdot \|_2$ *be any two norms on* $\mathbb{R}^n$. *Then there exist positive constants* $c^\star, C^\star > 0$ *such that*

$$c^\star \, \| \mathbf{v} \|_1 \leq \| \mathbf{v} \|_2 \leq C^\star \, \| \mathbf{v} \|_1 \qquad \textit{for every} \qquad \mathbf{v} \in \mathbb{R}^n. \tag{5.22}$$

Proof: We just sketch the basic idea, leaving the details to a more rigorous real analysis course, cf. [**11**; §7.6]. We begin by noting that a norm defines a continuous real-valued function $f(\mathbf{v}) = \| \mathbf{v} \|$ on $\mathbb{R}^n$. (Continuity is, in fact, a consequence of the triangle inequality.) Let $S_1 = \left\{ \| \mathbf{u} \|_1 = 1 \right\}$ denote the unit sphere of the first norm. Any continuous function defined on a compact set achieves both a maximum and a minimum value. Thus, restricting the second norm function to the unit sphere S_1 of the first norm, we can set

$$c^\star = \min \{ \, \| \mathbf{u} \|_2 \; | \; \mathbf{u} \in S_1 \, \}, \qquad C^\star = \max \{ \, \| \mathbf{u} \|_2 \; | \; \mathbf{u} \in S_1 \, \}. \tag{5.23}$$

Moreover, $0 < c^\star \leq C^\star < \infty$, with equality holding if and only if the norms are the same. The minimum and maximum (5.23) will serve as the constants in the desired inequalities (5.22). Indeed, by definition,

$$c^\star \leq \| \mathbf{u} \|_2 \leq C^\star \qquad \text{when} \qquad \| \mathbf{u} \|_1 = 1, \tag{5.24}$$

which proves that (5.22) is valid for all unit vectors $\mathbf{v} = \mathbf{u} \in S_1$. To prove the inequalities in general, assume $\mathbf{v} \neq \mathbf{0}$. (The case $\mathbf{v} = \mathbf{0}$ is trivial.) The homogeneity property of the norm implies that $\mathbf{u} = \mathbf{v} / \| \mathbf{v} \|_1 \in S_1$ is a unit vector in the first norm: $\| \mathbf{u} \|_1 = \| \mathbf{v} \| / \| \mathbf{v} \|_1 = 1$. Moreover, $\| \mathbf{u} \|_2 = \| \mathbf{v} \|_2 / \| \mathbf{v} \|_1$. Substituting into (5.24) and clearing denominators completes the proof of (5.22). *Q.E.D.*

Example 5.10. For example, consider the Euclidean norm $\| \cdot \|_2$ and the max norm $\| \cdot \|_\infty$ on $\mathbb{R}^n$. The bounding constants are found by minimizing and maximizing $\| \mathbf{u} \|_\infty = \max\{ \, | u_1 |, \ldots, | u_n | \, \}$ over all unit vectors $\| \mathbf{u} \|_2 = 1$ on the (round) unit sphere. The maximal value is achieved at the poles $\pm \mathbf{e}_k$, with $\| \pm \mathbf{e}_k \|_\infty = C^\star = 1$ The minimal value is attained at the points $\left(\pm \frac{1}{\sqrt{n}}, \ldots, \pm \frac{1}{\sqrt{n}} \right)$, whereby $c^\star = \frac{1}{\sqrt{n}}$. Therefore,

$$\frac{1}{\sqrt{n}} \, \| \mathbf{v} \|_2 \leq \| \mathbf{v} \|_\infty \leq \| \mathbf{v} \|_2. \tag{5.25}$$

We can interpret these inequalities as follows. Suppose $\mathbf{v}$ is a vector lying on the unit sphere in the Euclidean norm, so $\| \mathbf{v} \|_2 = 1$. Then (5.25) tells us that its ∞ norm is bounded from above and below by $\frac{1}{\sqrt{n}} \leq \| \mathbf{v} \|_\infty \leq 1$. Therefore, the Euclidean unit sphere sits inside the ∞ norm unit sphere and outside the ∞ norm sphere of radius $\frac{1}{\sqrt{n}}$.

AIMS Lecture Notes 2006

Peter J. Olver

6. Eigenvalues and Singular Values

In this section, we collect together the basic facts about eigenvalues and eigenvectors. From a geometrical viewpoint, the eigenvectors indicate the directions of pure stretch and the eigenvalues the extent of stretching. Most matrices are complete, meaning that their (complex) eigenvectors form a basis of the underlying vector space. A particularly important class are the symmetric matrices, whose eigenvectors form an orthogonal basis of $\mathbb{R}^n$. A non-square matrix A does not have eigenvalues. In their place, one uses the square roots of the eigenvalues of the associated square Gram matrix $K = A^T A$, which are called singular values of the original matrix. The numerical computation of eigenvalues and eigenvectors is a challenging issue, and must be be deferred until later.

6.1. Eigenvalues and Eigenvectors.

We inaugurate our discussion of eigenvalues and eigenvectors with the basic definition.

Definition 6.1. Let A be an $n \times n$ matrix. A scalar λ is called an *eigenvalue* of A if there is a *non-zero* vector $\mathbf{v} \neq \mathbf{0}$, called an *eigenvector*, such that

$$A\mathbf{v} = \lambda \mathbf{v}. \tag{6.1}$$

In other words, the matrix A stretches the eigenvector $\mathbf{v}$ by an amount specified by the eigenvalue λ.

Remark: The odd-looking terms "eigenvalue" and "eigenvector" are hybrid German–English words. In the original German, they are *Eigenwert* and *Eigenvektor*, which can be fully translated as "proper value" and "proper vector". For some reason, the half-translated terms have acquired a certain charm, and are now standard. The alternative English terms *characteristic value* and *characteristic vector* can be found in some (mostly older) texts. Oddly, the term *characteristic equation*, to be defined below, is still used.

The requirement that the eigenvector $\mathbf{v}$ be nonzero is important, since $\mathbf{v} = \mathbf{0}$ is a trivial solution to the eigenvalue equation (6.1) for *any* scalar λ. Moreover, as far as solving linear ordinary differential equations goes, the zero vector $\mathbf{v} = \mathbf{0}$ gives $\mathbf{u}(t) \equiv \mathbf{0}$, which is certainly a solution, but one that we already knew.

The eigenvalue equation (6.1) is a system of linear equations for the entries of the eigenvector $\mathbf{v}$ — provided that the eigenvalue λ is specified in advance — but is "mildly" nonlinear as a combined system for λ and $\mathbf{v}$. Gaussian Elimination per se will not solve the problem, and we are in need of a new idea. Let us begin by rewriting the equation in the form

$$(A - \lambda\,\mathrm{I}\,)\mathbf{v} = \mathbf{0}, \tag{6.2}$$

where I is the identity matrix of the correct size[†]. Now, for given λ, equation (6.2) is a homogeneous linear system for $\mathbf{v}$, and always has the trivial zero solution $\mathbf{v} = \mathbf{0}$. But we are specifically seeking a nonzero solution! A homogeneous linear system has a nonzero solution $\mathbf{v} \neq \mathbf{0}$ if and only if its coefficient matrix, which in this case is $A - \lambda\,\mathrm{I}$, is singular. This observation is the key to resolving the eigenvector equation.

Theorem 6.2. *A scalar λ is an eigenvalue of the $n \times n$ matrix A if and only if the matrix $A - \lambda\,\mathrm{I}$ is singular, i.e., of rank $< n$. The corresponding eigenvectors are the nonzero solutions to the eigenvalue equation $(A - \lambda\,\mathrm{I}\,)\mathbf{v} = \mathbf{0}$.*

Proposition 6.3. *A scalar λ is an eigenvalue of the matrix A if and only if λ is a solution to the* characteristic equation

$$\det(A - \lambda\,\mathrm{I}\,) = 0. \tag{6.3}$$

In practice, when finding eigenvalues and eigenvectors by hand, one first solves the characteristic equation (6.3). Then, for each eigenvalue λ one uses standard linear algebra methods, i.e., Gaussian Elimination, to solve the corresponding linear system (6.2) for the eigenvector $\mathbf{v}$.

Example 6.4. Consider the 2×2 matrix

$$A = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}.$$

We compute the determinant in the characteristic equation using formula (3.8):

$$\det(A - \lambda\,\mathrm{I}\,) = \det \begin{pmatrix} 3-\lambda & 1 \\ 1 & 3-\lambda \end{pmatrix} = (3-\lambda)^2 - 1 = \lambda^2 - 6\,\lambda + 8.$$

Thus, the characteristic equation is a quadratic polynomial equation, and can be solved by factorization:

$$\lambda^2 - 6\,\lambda + 8 = (\lambda - 4)\,(\lambda - 2) = 0.$$

We conclude that A has two eigenvalues: $\lambda_1 = 4$ and $\lambda_2 = 2$.

For each eigenvalue, the corresponding eigenvectors are found by solving the associated homogeneous linear system (6.2). For the first eigenvalue, the eigenvector equation is

$$(A - 4\,\mathrm{I}\,)\,\mathbf{v} = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \text{or} \qquad \begin{aligned} -x + y &= 0, \\ x - y &= 0. \end{aligned}$$

[†] Note that it is not legal to write (6.2) in the form $(A - \lambda)\mathbf{v} = \mathbf{0}$ since we do not know how to subtract a scalar λ from a matrix A. Worse, if you type $A - \lambda$ in Matlab or Mathematica, the result will be to subtract λ from *all* the entries of A, which is *not* what we are after!

The general solution is

$$x = y = a, \qquad \text{so} \qquad \mathbf{v} = \begin{pmatrix} a \\ a \end{pmatrix} = a \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

where a is an arbitrary scalar. Only the nonzero solutions[†] count as eigenvectors, and so the eigenvectors for the eigenvalue $\lambda_1 = 4$ must have $a \neq 0$, i.e., they are all nonzero scalar multiples of the basic eigenvector $\mathbf{v}_1 = (\,1, 1\,)^T$.

Remark: In general, if $\mathbf{v}$ is an eigenvector of A for the eigenvalue λ, then so is any nonzero scalar multiple of $\mathbf{v}$. In practice, we only distinguish linearly independent eigenvectors. Thus, in this example, we shall say "$\mathbf{v}_1 = (\,1, 1\,)^T$ is *the* eigenvector corresponding to the eigenvalue $\lambda_1 = 4$", when we really mean that the eigenvectors for $\lambda_1 = 4$ consist of all nonzero scalar multiples of $\mathbf{v}_1$.

Similarly, for the second eigenvalue $\lambda_2 = 2$, the eigenvector equation is

$$(A - 2\,\mathrm{I}\,)\,\mathbf{v} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

The solution $(\,-a, a\,)^T = a\,(\,-1, 1\,)^T$ is the set of scalar multiples of the eigenvector $\mathbf{v}_2 = (\,-1, 1\,)^T$. Therefore, the complete list of eigenvalues and eigenvectors (up to scalar multiple) for this particular matrix is

$$\lambda_1 = 4, \qquad \mathbf{v}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad\qquad \lambda_2 = 2, \qquad \mathbf{v}_2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

Example 6.5. Consider the 3×3 matrix

$$A = \begin{pmatrix} 0 & -1 & -1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}.$$

Using the formula for a 3×3 determinant, we compute the characteristic equation

$$\begin{aligned} 0 = \det(A - \lambda\,\mathrm{I}\,) &= \det \begin{pmatrix} -\lambda & -1 & -1 \\ 1 & 2-\lambda & 1 \\ 1 & 1 & 2-\lambda \end{pmatrix} \\ &= (-\lambda)(2-\lambda)^2 + (-1)\cdot 1 \cdot 1 + (-1)\cdot 1 \cdot 1 - \\ &\qquad - 1\cdot(2-\lambda)(-1) - 1\cdot 1\cdot(-\lambda) - (2-\lambda)\cdot 1\cdot(-1) \\ &= -\lambda^3 + 4\lambda^2 - 5\lambda + 2. \end{aligned}$$

The resulting cubic polynomial can be factored:

$$-\lambda^3 + 4\lambda^2 - 5\lambda + 2 = -(\lambda - 1)^2\,(\lambda - 2) = 0.$$

[†] If, at this stage, you end up with a linear system with only the trivial zero solution, you've done something wrong! Either you don't have a correct eigenvalue — maybe you made a mistake setting up and/or solving the characteristic equation — or you've made an error solving the homogeneous eigenvector system.

Most 3×3 matrices have three different eigenvalues, but this particular one has only two: $\lambda_1 = 1$, which is called a *double eigenvalue* since it is a double root of the characteristic equation, along with a *simple eigenvalue* $\lambda_2 = 2$.

The eigenvector equation (6.2) for the double eigenvalue $\lambda_1 = 1$ is

$$(A - \mathrm{I})\mathbf{v} = \begin{pmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

The general solution to this homogeneous linear system

$$\mathbf{v} = \begin{pmatrix} -a-b \\ a \\ b \end{pmatrix} = a \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$$

depends upon two free variables: $y = a$ and $z = b$. Any nonzero solution forms a valid eigenvector for the eigenvalue $\lambda_1 = 1$, and so the general eigenvector is any non-zero linear combination of the two "basis eigenvectors" $\mathbf{v}_1 = (-1, 1, 0)^T$, $\widehat{\mathbf{v}}_1 = (-1, 0, 1)^T$.

On the other hand, the eigenvector equation for the simple eigenvalue $\lambda_2 = 2$ is

$$(A - 2\,\mathrm{I})\mathbf{v} = \begin{pmatrix} -2 & -1 & -1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

The general solution

$$\mathbf{v} = \begin{pmatrix} -a \\ a \\ a \end{pmatrix} = a \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}$$

consists of all scalar multiples of the eigenvector $\mathbf{v}_2 = (-1, 1, 1)^T$.

In summary, the eigenvalues and (basis) eigenvectors for this matrix are

$$\begin{aligned} \lambda_1 &= 1, & \mathbf{v}_1 &= \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}, & \widehat{\mathbf{v}}_1 &= \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \\ \lambda_2 &= 2, & \mathbf{v}_2 &= \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}. \end{aligned} \tag{6.4}$$

In general, given a real eigenvalue λ, the corresponding *eigenspace* $V_\lambda \subset \mathbb{R}^n$ is the subspace spanned by all its eigenvectors. Equivalently, the eigenspace is the kernel

$$V_\lambda = \ker(A - \lambda\,\mathrm{I}). \tag{6.5}$$

In particular, $\lambda \in \mathbb{R}$ is an eigenvalue if and only if $V_\lambda \neq \{\mathbf{0}\}$ is a nontrivial subspace, and then every nonzero element of V_λ is a corresponding eigenvector. The most economical way to indicate each eigenspace is by writing out a basis, as in (6.4) with $\mathbf{v}_1, \widehat{\mathbf{v}}_1$ giving a basis for V_1, while $\mathbf{v}_2$ is a basis for V_2.

Example 6.6. The characteristic equation of the matrix $A = \begin{pmatrix} 1 & 2 & 1 \\ 1 & -1 & 1 \\ 2 & 0 & 1 \end{pmatrix}$ is

$$0 = \det(A - \lambda\,\mathrm{I}\,) = -\lambda^3 + \lambda^2 + 5\,\lambda + 3 = -\,(\lambda+1)^2\,(\lambda - 3).$$

Again, there is a double eigenvalue $\lambda_1 = -1$ and a simple eigenvalue $\lambda_2 = 3$. However, in this case the matrix

$$A - \lambda_1\,\mathrm{I} = A + \mathrm{I} = \begin{pmatrix} 2 & 2 & 1 \\ 1 & 0 & 1 \\ 2 & 0 & 2 \end{pmatrix}$$

has only a one-dimensional kernel, spanned by $\mathbf{v}_1 = (\,2, -1, -2\,)^T$. Thus, even though λ_1 is a double eigenvalue, it only admits a one-dimensional eigenspace. The list of eigenvalues and eigenvectors is, in a sense, incomplete:

$$\lambda_1 = -1, \qquad \mathbf{v}_1 = \begin{pmatrix} 2 \\ -1 \\ -2 \end{pmatrix}, \qquad\qquad \lambda_2 = 3, \qquad \mathbf{v}_2 = \begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix}.$$

Example 6.7. Finally, consider the matrix $A = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & -2 \\ 2 & 2 & -1 \end{pmatrix}$. The characteristic equation is

$$0 = \det(A - \lambda\,\mathrm{I}\,) = -\lambda^3 + \lambda^2 - 3\,\lambda - 5 = -\,(\lambda+1)\,(\lambda^2 - 2\,\lambda + 5).$$

The linear factor yields the eigenvalue -1. The quadratic factor leads to two complex roots, $1 + 2\,\mathrm{i}$ and $1 - 2\,\mathrm{i}$, which can be obtained via the quadratic formula. Hence A has one real and two complex eigenvalues:

$$\lambda_1 = -1, \qquad\qquad \lambda_2 = 1 + 2\,\mathrm{i}\,, \qquad\qquad \lambda_3 = 1 - 2\,\mathrm{i}\,.$$

Solving the associated linear system, the real eigenvalue is found to have corresponding eigenvector $\mathbf{v}_1 = (\,-1, 1, 1\,)^T$.

Complex eigenvalues are as important as real eigenvalues, and we need to be able to handle them too. To find the corresponding eigenvectors, which will also be complex, we need to solve the usual eigenvalue equation (6.2), which is now a complex homogeneous linear system. For example, the eigenvector(s) for $\lambda_2 = 1 + 2\,\mathrm{i}$ are found by solving

$$\bigl[\,A - (1 + 2\,\mathrm{i}\,)\,\mathrm{I}\,\bigr]\mathbf{v} = \begin{pmatrix} -2\,\mathrm{i} & 2 & 0 \\ 0 & -2\,\mathrm{i} & -2 \\ 2 & 2 & -2 - 2\,\mathrm{i} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

This linear system can be solved by Gaussian Elimination (with complex pivots). A simpler strategy is to work directly: the first equation $-\,2\,\mathrm{i}\,x + 2\,y = 0$ tells us that $y = \mathrm{i}\,x$, while the second equation $-\,2\,\mathrm{i}\,y - 2\,z = 0$ says $z = -\,\mathrm{i}\,y = x$. If we trust our calculations so far, we do not need to solve the final equation $2\,x + 2\,y + (-2 - 2\,\mathrm{i}\,)z = 0$, since we know that the coefficient matrix is singular and hence this equation must be a consequence of the first two. (However, it does serve as a useful check on our work.) So, the general solution $\mathbf{v} = (x, \mathrm{i}\,x, x)^T$ is an arbitrary constant multiple of the complex eigenvector $\mathbf{v}_2 = (1, \mathrm{i}, 1)^T$. The eigenvector equation for $\lambda_3 = 1 - 2\,\mathrm{i}$ is similarly solved for the third eigenvector $\mathbf{v}_3 = (1, -\mathrm{i}, 1)^T$.

Summarizing, the matrix under consideration has three complex eigenvalues and three corresponding eigenvectors, each unique up to (complex) scalar multiple:

$$\begin{array}{lll} \lambda_1 = -1, & \lambda_2 = 1 + 2\,\mathrm{i}, & \lambda_3 = 1 - 2\,\mathrm{i}, \\ \mathbf{v}_1 = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}, & \mathbf{v}_2 = \begin{pmatrix} 1 \\ \mathrm{i} \\ 1 \end{pmatrix}, & \mathbf{v}_3 = \begin{pmatrix} 1 \\ -\mathrm{i} \\ 1 \end{pmatrix}. \end{array}$$

Note that the third complex eigenvalue is the complex conjugate of the second, and the eigenvectors are similarly related. This is indicative of a general fact for real matrices:

Proposition 6.8. *If A is a real matrix with a complex eigenvalue $\lambda = \mu + \mathrm{i}\,\nu$ and corresponding complex eigenvector $\mathbf{v} = \mathbf{x} + \mathrm{i}\,\mathbf{y}$, then the complex conjugate $\overline{\lambda} = \mu - \mathrm{i}\,\nu$ is also an eigenvalue with complex conjugate eigenvector $\overline{\mathbf{v}} = \mathbf{x} - \mathrm{i}\,\mathbf{y}$.*

Proof: First take complex conjugates of the eigenvalue equation (6.1):

$$\overline{A}\,\overline{\mathbf{v}} = \overline{A\,\mathbf{v}} = \overline{\lambda\,\mathbf{v}} = \overline{\lambda}\,\overline{\mathbf{v}}.$$

Using the fact that a real matrix is unaffected by conjugation, so $\overline{A} = A$, we conclude $A\,\overline{\mathbf{v}} = \overline{\lambda}\,\overline{\mathbf{v}}$, which is the equation for the eigenvalue $\overline{\lambda}$ and eigenvector $\overline{\mathbf{v}}$. *Q.E.D.*

As a consequence, when dealing with real matrices, we only need to compute the eigenvectors for *one* of each complex conjugate pair of eigenvalues. This observation effectively halves the amount of work in the unfortunate event that we are confronted with complex eigenvalues.

The *eigenspace* associated with a complex eigenvalue λ is the subspace $V_\lambda \subset \mathbb{C}^n$ spanned by the associated eigenvectors. One might also consider complex eigenvectors associated with a real eigenvalue, but this doesn't add anything to the picture — they are merely complex linear combinations of the real eigenvalues. Thus, we only introduce complex eigenvectors when dealing with genuinely complex eigenvalues.

Remark: The reader may recall that we said one should never use determinants in practical computations. So why have we reverted to using determinants to find eigenvalues? The truthful answer is that the practical computation of eigenvalues and eigenvectors *never* resorts to the characteristic equation! The method is fraught with numerical traps and inefficiencies when (*a*) computing the determinant leading to the characteristic equation, then (*b*) solving the resulting polynomial equation, which is itself a nontrivial numerical problem[†], [**7**, **43**], and, finally, (*c*) solving each of the resulting linear eigenvector systems.

[†] In fact, one effective numerical strategy for finding the roots of a polynomial is to turn the procedure on its head, and calculate the eigenvalues of a matrix whose characteristic equation is the polynomial in question! See [**43**] for details.

Worse, if we only know an approximation $\widetilde{\lambda}$ to the true eigenvalue λ, the approximate eigenvector system $(A - \widetilde{\lambda}\,)\mathbf{v} = \mathbf{0}$ will almost certainly have a nonsingular coefficient matrix, and hence only admits the trivial solution $\mathbf{v} = \mathbf{0}$ — which does not even qualify as an eigenvector!

Nevertheless, the characteristic equation does give us important theoretical insight into the structure of the eigenvalues of a matrix, and can be used when dealing with small matrices, e.g., 2×2 and 3×3, presuming exact arithmetic is employed. Numerical algorithms for computing eigenvalues and eigenvectors are based on completely different ideas.

Proposition 6.9. *A matrix A is singular if and only if 0 is an eigenvalue.*

Proof: By definition, 0 is an eigenvalue of A if and only if there is a nonzero solution to the eigenvector equation $A\mathbf{v} = 0\mathbf{v} = \mathbf{0}$. Thus, 0 is an eigenvector of A if and only if it has a non-zero vector in its kernel, $\ker A \neq \{\mathbf{0}\}$, and hence A is necessarily singular. *Q.E.D.*

Basic Properties of Eigenvalues

If A is an $n \times n$ matrix, then its *characteristic polynomial* is

$$p_A(\lambda) = \det(A - \lambda\,\mathrm{I}\,) = c_n\,\lambda^n + c_{n-1}\,\lambda^{n-1} + \;\cdots\; + c_1\,\lambda + c_0. \tag{6.6}$$

The fact that $p_A(\lambda)$ is a polynomial of degree n is a consequence of the general determinantal formula. Indeed, every term is prescribed by a permutation π of the rows of the matrix, and equals plus or minus a product of n distinct matrix entries including one from each row and one from each column. The term corresponding to the identity permutation is obtained by multiplying the diagonal entries together, which, in this case, is

$$(a_{11}-\lambda)\,(a_{22}-\lambda)\;\cdots\;(a_{nn}-\lambda) = (-1)^n\lambda^n + (-1)^{n-1}\big(\,a_{11} + a_{22} + \;\cdots\; + a_{nn}\,\big)\,\lambda^{n-1} + \;\cdots\;. \tag{6.7}$$

All of the other terms have at most $n-2$ diagonal factors $a_{ii} - \lambda$, and so are polynomials of degree $\leq n-2$ in λ. Thus, (6.7) is the only summand containing the monomials λ^n and λ^{n-1}, and so their respective coefficients are

$$c_n = (-1)^n, \qquad\qquad c_{n-1} = (-1)^{n-1}(a_{11} + a_{22} + \;\cdots\; + a_{nn}) = (-1)^{n-1}\,\mathrm{tr}\,A, \tag{6.8}$$

where $\mathrm{tr}\,A$, the sum of its diagonal entries, is called the *trace* of the matrix A. The other coefficients $c_{n-2}, \ldots, c_1, c_0$ in (6.6) are more complicated combinations of the entries of A. However, setting $\lambda = 0$ implies

$$p_A(0) = \det A = c_0,$$

and hence the constant term in the characteristic polynomial equals the determinant of the matrix. In particular, if $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is a 2×2 matrix, its characteristic polynomial has the explicit form

$$\begin{aligned} p_A(\lambda) = \det(A - \lambda\,\mathrm{I}\,) &= \det\begin{pmatrix} a-\lambda & b \\ c & d-\lambda \end{pmatrix} \\ &= \lambda^2 - (a+d)\lambda + (a\,d - b\,c) = \lambda^2 - (\mathrm{tr}\,A)\lambda + (\det A). \end{aligned} \tag{6.9}$$

As a result of these considerations, the characteristic equation of an $n \times n$ matrix A is a polynomial equation of degree n. According to the Fundamental Theorem of Algebra, [**17**], every (complex) polynomial of degree $n \geq 1$ can be completely factored, and so we can write the characteristic polynomial in factored form:

$$p_A(\lambda) = (-1)^n (\lambda - \lambda_1)(\lambda - \lambda_2) \ \cdots \ (\lambda - \lambda_n). \tag{6.10}$$

The complex numbers $\lambda_1, \ldots, \lambda_n$, some of which may be repeated, are the *roots* of the characteristic equation $p_A(\lambda) = 0$, and hence the eigenvalues of the matrix A. Therefore, we immediately conclude:

Theorem 6.10. *An $n \times n$ matrix A has at least one and at most n distinct complex eigenvalues.*

Most $n \times n$ matrices — meaning those for which the characteristic polynomial factors into n *distinct* factors — have *exactly* n complex eigenvalues. More generally, an eigenvalue λ_j is said to have *multiplicity* m if the factor $(\lambda - \lambda_j)$ appears exactly m times in the factorization (6.10) of the characteristic polynomial. An eigenvalue is *simple* if it has multiplicity 1. In particular, A has n distinct eigenvalues if and only if all its eigenvalues are simple. In all cases, when the repeated eigenvalues are counted in accordance with their multiplicity, every $n \times n$ matrix has a total of n, possibly repeated, eigenvalues.

An example of a matrix with just one eigenvalue, of multiplicity n, is the $n \times n$ identity matrix I, whose only eigenvalue is $\lambda = 1$. In this case, *every* nonzero vector in $\mathbb{R}^n$ is an eigenvector of the identity matrix, and so the eigenspace is all of $\mathbb{R}^n$. At the other extreme, the "bidiagonal" *Jordan block matrix*[†]

$$J_a = \begin{pmatrix} a & 1 & & & & \\ & a & 1 & & & \\ & & a & 1 & & \\ & & & \ddots & \ddots & \\ & & & & a & 1 \\ & & & & & a \end{pmatrix} \tag{6.11}$$

also has only one eigenvalue, $\lambda = a$, again of multiplicity n. But in this case, J_a has only one eigenvector (up to scalar multiple), which is the first standard basis vector $\mathbf{e}_1$, and so its eigenspace is one-dimensional.

Remark: If λ is a complex eigenvalue of multiplicity k for the real matrix A, then its complex conjugate $\overline{\lambda}$ also has multiplicity k. This is because complex conjugate roots of a real polynomial necessarily appear with identical multiplicities.

If we explicitly multiply out the factored product (6.10) and equate the result to the characteristic polynomial (6.6), we find that its coefficients $c_0, c_1, \ldots c_{n-1}$ can be written

[†] All non-displayed entries are zero.

as certain polynomials of the roots, known as the *elementary symmetric polynomials*. The first and last are of particular importance:

$$c_0 = \lambda_1 \lambda_2 \cdots \lambda_n, \qquad c_{n-1} = (-1)^{n-1}(\lambda_1 + \lambda_2 + \cdots + \lambda_n). \tag{6.12}$$

Comparison with our previous formulae for the coefficients c_0 and c_{n-1} leads to the following useful result.

Proposition 6.11. *The sum of the eigenvalues of a matrix equals its trace*:

$$\lambda_1 + \lambda_2 + \cdots + \lambda_n = \operatorname{tr} A = a_{11} + a_{22} + \cdots + a_{nn}. \tag{6.13}$$

The product of the eigenvalues equals its determinant:

$$\lambda_1 \lambda_2 \cdots \lambda_n = \det A. \tag{6.14}$$

Remark: For repeated eigenvalues, one must add or multiply them in the formulae (6.13–14) according to their multiplicity.

Example 6.12. The matrix $A = \begin{pmatrix} 1 & 2 & 1 \\ 1 & -1 & 1 \\ 2 & 0 & 1 \end{pmatrix}$ considered in Example 6.6 has trace and determinant

$$\operatorname{tr} A = 1, \qquad \det A = 3,$$

which fix, respectively, the coefficient of λ^2 and the constant term in its characteristic equation. This matrix has two distinct eigenvalues: -1, which is a double eigenvalue, and 3, which is simple. For this particular matrix, formulae (6.13–14) become

$$1 = \operatorname{tr} A = (-1) + (-1) + 3, \qquad 3 = \det A = (-1)(-1)\,3.$$

Note that the double eigenvalue contributes twice to the sum and to the product.

6.2. Completeness.

Most of the vector space bases that play a distinguished role in applications are assembled from the eigenvectors of a particular matrix. In this section, we show that the eigenvectors of any "complete" matrix automatically form a basis for $\mathbb{R}^n$ or, in the complex case, $\mathbb{C}^n$. In the following subsection, we use the eigenvector basis to rewrite the linear transformation determined by the matrix in a simple diagonal form. The most important cases — symmetric and positive definite matrices — will be treated in the following section.

The first task is to show that eigenvectors corresponding to distinct eigenvalues are automatically linearly independent.

Lemma 6.13. *If $\lambda_1, \ldots, \lambda_k$ are* distinct *eigenvalues of the same matrix A, then the corresponding eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are linearly independent.*

Proof: The result is proved by induction on the number of eigenvalues. The case $k = 1$ is immediate since an eigenvector cannot be zero. Assume that we know the result is valid for $k - 1$ eigenvalues. Suppose we have a vanishing linear combination:

$$c_1 \mathbf{v}_1 + \cdots + c_{k-1} \mathbf{v}_{k-1} + c_k \mathbf{v}_k = \mathbf{0}. \tag{6.15}$$

Let us multiply this equation by the matrix A:

$$\begin{aligned} A\bigl(c_1 \mathbf{v}_1 + \cdots + c_{k-1} \mathbf{v}_{k-1} + c_k \mathbf{v}_k \bigr) &= c_1 A\mathbf{v}_1 + \cdots + c_{k-1} A\mathbf{v}_{k-1} + c_k A\mathbf{v}_k \\ &= c_1 \lambda_1 \mathbf{v}_1 + \cdots + c_{k-1} \lambda_{k-1} \mathbf{v}_{k-1} + c_k \lambda_k \mathbf{v}_k = \mathbf{0}. \end{aligned}$$

On the other hand, if we multiply the original equation (6.15) by λ_k, we also have

$$c_1 \lambda_k \mathbf{v}_1 + \cdots + c_{k-1} \lambda_k \mathbf{v}_{k-1} + c_k \lambda_k \mathbf{v}_k = \mathbf{0}.$$

Subtracting this from the previous equation, the final terms cancel and we are left with the equation

$$c_1(\lambda_1 - \lambda_k)\mathbf{v}_1 + \cdots + c_{k-1}(\lambda_{k-1} - \lambda_k)\mathbf{v}_{k-1} = \mathbf{0}.$$

This is a vanishing linear combination of the first $k - 1$ eigenvectors, and so, by our induction hypothesis, can only happen if all the coefficients are zero:

$$c_1(\lambda_1 - \lambda_k) = 0, \qquad \ldots \qquad c_{k-1}(\lambda_{k-1} - \lambda_k) = 0.$$

The eigenvalues were assumed to be distinct, so $\lambda_j \neq \lambda_k$ when $j \neq k$. Consequently, $c_1 = \cdots = c_{k-1} = 0$. Substituting these values back into (6.15), we find $c_k \mathbf{v}_k = \mathbf{0}$, and so $c_k = 0$ also, since the eigenvector $\mathbf{v}_k \neq \mathbf{0}$. Thus we have proved that (6.15) holds if and only if $c_1 = \cdots = c_k = 0$, which implies the linear independence of the eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$. This completes the induction step. *Q.E.D.*

The most important consequence of this result is when a matrix has the maximum allotment of eigenvalues.

Theorem 6.14. *If the $n \times n$ real matrix A has n distinct real eigenvalues $\lambda_1, \ldots, \lambda_n$, then the corresponding real eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ form a basis of $\mathbb{R}^n$. If A (which may now be either a real or a complex matrix) has n distinct complex eigenvalues, then the corresponding eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ form a basis of $\mathbb{C}^n$.*

For instance, the 2×2 matrix in Example 6.4 has two distinct real eigenvalues, and its two independent eigenvectors form a basis of $\mathbb{R}^2$. The 3×3 matrix in Example 6.7 has three distinct complex eigenvalues, and its eigenvectors form a basis for $\mathbb{C}^3$. If a matrix has multiple eigenvalues, then there may or may not be an eigenvector basis of $\mathbb{R}^n$ (or $\mathbb{C}^n$). The matrix in Example 6.5 admits an eigenvector basis, whereas the matrix in Example 6.6 does not. In general, it can be proved that the dimension of the eigenspace is less than or equal to the eigenvalue's multiplicity. In particular, every simple eigenvalue has a one-dimensional eigenspace, and hence, up to scalar multiple, only one associated eigenvector.

Definition 6.15. An eigenvalue λ of a matrix A is called *complete* if the corresponding eigenspace $V_\lambda = \ker(A - \lambda\,\mathrm{I})$ has the same dimension as its multiplicity. The matrix A is *complete* if all its eigenvalues are.

Note that a simple eigenvalue is automatically complete, since its eigenspace is the one-dimensional subspace spanned by the corresponding eigenvector. Thus, only multiple eigenvalues can cause a matrix to be incomplete.

Remark: The multiplicity of an eigenvalue λ_i is sometimes referred to as its *algebraic multiplicity*. The dimension of the eigenspace V_λ is its *geometric multiplicity*, and so completeness requires that the two multiplicities are equal. The word "complete" is not completely standard; other common terms for such matrices are *perfect*, *semi-simple* and, as discussed shortly, *diagonalizable*.

Theorem 6.16. *An $n \times n$ real or complex matrix A is complete if and only if its eigenvectors span $\mathbb{C}^n$. In particular, any $n \times n$ matrix that has n distinct eigenvalues is complete.*

Or, stated another way, a matrix is complete if and only if its eigenvectors can be used to form a basis of $\mathbb{C}^n$. Most matrices are complete. Incomplete $n \times n$ matrices, which have fewer than n linearly independent complex eigenvectors, are considerably less pleasant to deal with.

6.3. Eigenvalues of Symmetric Matrices.

Fortunately, the matrices that arise in most applications are complete and, in fact, possess some additional structure that ameliorates the calculation of their eigenvalues and eigenvectors. The most important class are the symmetric, including positive definite, matrices. In fact, not only are the eigenvalues of a symmetric matrix necessarily real, the eigenvectors always form an *orthogonal basis* of the underlying Euclidean space. In fact, this is by far the most common way for orthogonal bases to appear — as the eigenvector bases of symmetric matrices. Let us state this important result, but defer its proof until the end of the section.

Theorem 6.17. *Let $A = A^T$ be a real symmetric $n \times n$ matrix. Then*

(*a*) *All the eigenvalues of A are real.*

(*b*) *Eigenvectors corresponding to distinct eigenvalues are orthogonal.*

(*c*) *There is an orthonormal basis of $\mathbb{R}^n$ consisting of n eigenvectors of A.*

In particular, all symmetric matrices are complete.

Example 6.18. The 2×2 matrix $A = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$ considered in Example 6.4 is symmetric, and so has real eigenvalues $\lambda_1 = 4$ and $\lambda_2 = 2$. You can easily check that the corresponding eigenvectors $\mathbf{v}_1 = (1,1)^T$ and $\mathbf{v}_2 = (-1,1)^T$ are orthogonal: $\mathbf{v}_1 \cdot \mathbf{v}_2 = 0$, and hence form an orthogonal basis of $\mathbb{R}^2$. The orthonormal eigenvector basis promised by Theorem 6.17 is obtained by dividing each eigenvector by its Euclidean norm:

$$\mathbf{u}_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}, \qquad \mathbf{u}_2 = \begin{pmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}.$$

Example 6.19. Consider the symmetric matrix $A = \begin{pmatrix} 5 & -4 & 2 \\ -4 & 5 & 2 \\ 2 & 2 & -1 \end{pmatrix}$. A straightforward computation produces its eigenvalues and eigenvectors:

$$\lambda_1 = 9, \qquad \lambda_2 = 3, \qquad \lambda_3 = -3,$$
$$\mathbf{v}_1 = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \qquad \mathbf{v}_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \qquad \mathbf{v}_3 = \begin{pmatrix} 1 \\ 1 \\ -2 \end{pmatrix}.$$

As the reader can check, the eigenvectors form an orthogonal basis of $\mathbb{R}^3$. An orthonormal basis is provided by the unit eigenvectors

$$\mathbf{u}_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}, \qquad \mathbf{u}_2 = \begin{pmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{pmatrix}, \qquad \mathbf{u}_3 = \begin{pmatrix} \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \\ -\frac{2}{\sqrt{6}} \end{pmatrix}.$$

In particular, the eigenvalues of a symmetric matrix can be used to test its positive definiteness.

Theorem 6.20. *A symmetric matrix $K = K^T$ is positive definite if and only if all of its eigenvalues are strictly positive.*

Example 6.21. Consider the symmetric matrix $K = \begin{pmatrix} 8 & 0 & 1 \\ 0 & 8 & 1 \\ 1 & 1 & 7 \end{pmatrix}$. Its characteristic equation is

$$\det(K - \lambda I) = -\lambda^3 + 23\lambda^2 - 174\lambda + 432 = -(\lambda - 9)(\lambda - 8)(\lambda - 6),$$

and so its eigenvalues are $9, 8$, and 6. Since they are all positive, K is a positive definite matrix. The associated eigenvectors are

$$\lambda_1 = 9, \quad \mathbf{v}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \qquad \lambda_2 = 8, \quad \mathbf{v}_2 = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}, \qquad \lambda_3 = 6, \quad \mathbf{v}_3 = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix}.$$

Note that the eigenvectors form an orthogonal basis of $\mathbb{R}^3$, as guaranteed by Theorem 6.17. As usual, we can construct an corresponding orthonormal eigenvector basis

$$\mathbf{u}_1 = \begin{pmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{pmatrix}, \qquad \mathbf{u}_2 = \begin{pmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}, \qquad \mathbf{u}_3 = \begin{pmatrix} -\frac{1}{\sqrt{6}} \\ -\frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{6}} \end{pmatrix},$$

by dividing each eigenvector by its norm.

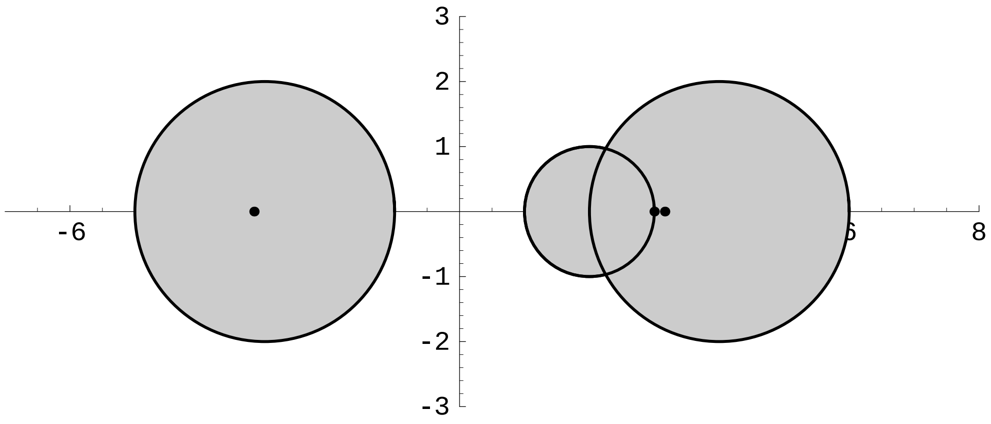


Figure 6.1. Gerschgorin Disks and Eigenvalues.

6.4. The Gerschgorin Circle Theorem.

In general, precisely computing the eigenvalues is not easy, and, in most cases, must be done through a numerical eigenvalue routine. In applications, though, we may not require their exact numerical values, but only approximate locations. The *Gerschgorin Circle Theorem*, due to the early twentieth century Russian mathematician Semen Gerschgorin, serves to restrict the eigenvalues to a certain well-defined region in the complex plane.

Definition 6.22. Let A be an $n \times n$ matrix, either real or complex. For each $1 \le i \le n$, define the *Gerschgorin disk*

$$D_i = \{\, |\, z - a_{ii} \,| \le r_i \;|\; z \in \mathbb{C} \,\}, \qquad \text{where} \qquad r_i = \sum_{\substack{j=1 \\ j \ne i}}^{n} |\, a_{ij} \,|. \tag{6.16}$$

The *Gerschgorin domain* $D_A = \bigcup_{i=1}^{n} D_i \subset \mathbb{C}$ is the union of the Gerschgorin disks.

Thus, the i^{th} Gerschgorin disk D_i is centered at the i^{th} diagonal entry a_{ii}, and has radius r_i equal to the sum of the absolute values of the off-diagonal entries that are in the i^{th} row of A. We can now state the Gerschgorin Circle Theorem.

Theorem 6.23. *All real and complex eigenvalues of the matrix A lie in its Gerschgorin domain D_A.*

Example 6.24. The matrix $A = \begin{pmatrix} 2 & -1 & 0 \\ 1 & 4 & -1 \\ -1 & -1 & -3 \end{pmatrix}$ has Gerschgorin disks

$$D_1 = \{\, |\, z - 2 \,| \le 1 \,\}, \qquad D_2 = \{\, |\, z - 4 \,| \le 2 \,\}, \qquad D_3 = \{\, |\, z + 3 \,| \le 2 \,\},$$

which are plotted in Figure 6.1. The eigenvalues of A are

$$\lambda_1 = 3, \qquad \lambda_2 = \sqrt{10} = 3.1623\ldots, \qquad \lambda_3 = -\sqrt{10} = -3.1623\ldots.$$

Observe that λ_1 belongs to both D_1 and D_2, while λ_2 lies in D_2, and λ_3 is in D_3. We thus confirm that all three eigenvalues are in the Gerschgorin domain $D_A = D_1 \cup D_2 \cup D_3$.

Proof of Theorem 6.23: Let $\mathbf{v}$ be an eigenvector of A with eigenvalue λ. Let $\mathbf{u} = \mathbf{v}/\|\mathbf{v}\|_\infty$ be the corresponding unit eigenvector with respect to the ∞ norm, so

$$\|\mathbf{u}\|_\infty = \max\{\,|u_1|,\ \ldots\ ,|u_n|\,\} = 1.$$

Let u_i be an entry of $\mathbf{u}$ that achieves the maximum: $|u_i| = 1$. Writing out the i^{th} component of the eigenvalue equation $A\mathbf{u} = \lambda\mathbf{u}$, we find

$$\sum_{j=1}^{n} a_{ij}\,u_j = \lambda\,u_i, \qquad \text{which we rewrite as} \qquad \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}\,u_j = (\lambda - a_{ii})\,u_i.$$

Therefore, since all $|u_j| \leq 1$ while $|u_i| = 1$,

$$|\lambda - a_{ii}| = |\lambda - a_{ii}|\,|u_i| = \left|\,\sum_{j\neq i} a_{ij}\,u_j\,\right| \leq \sum_{j\neq i} |a_{ij}|\,|u_j| \leq \sum_{j\neq i} |a_{ij}| = r_i.$$

This immediately implies that $\lambda \in D_i \subset D_A$ belongs to the i^{th} Gerschgorin disk. *Q.E.D.*

One application is a simple direct test that guarantees invertibility of a matrix without requiring Gaussian Elimination or computing determinants. According to Proposition 6.9, a matrix A is nonsingular if and only if it does not admit zero as an eigenvalue. Thus, if its Gerschgorin domain does not contain 0, it cannot be an eigenvalue, and hence A is necessarily invertible. The condition $0 \notin D_A$ requires that the matrix have large diagonal entries, as quantified by the following definition.

Definition 6.25. A square matrix A is called *strictly diagonally dominant* if

$$|a_{ii}| > \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|, \qquad \text{for all} \qquad i = 1,\ldots,n. \tag{6.17}$$

In other words, strict diagonal dominance requires each diagonal entry to be larger, in absolute value, than the sum of the absolute values of *all* the other entries in its row. For example, the matrix $\begin{pmatrix} 3 & -1 & 1 \\ 1 & -4 & 2 \\ -2 & -1 & 5 \end{pmatrix}$ is strictly diagonally dominant since

$$|3| > |-1| + |1|, \qquad |-4| > |1| + |2|, \qquad |5| > |-2| + |-1|.$$

Diagonally dominant matrices appear frequently in numerical solution methods for both ordinary and partial differential equations. As we shall see, they are the most common class of matrices to which iterative solution methods can be successfully applied.

Theorem 6.26. *A strictly diagonally dominant matrix is nonsingular.*

Proof: The diagonal dominance inequalities (6.17) imply that the radius of the i^{th} Gerschgorin disk is strictly less than the modulus of its center: $r_i < |a_{ii}|$. Thus, the disk cannot contain 0; indeed, if $z \in D_i$, then, by the triangle inequality,

$$r_i > |z - a_{ii}| \geq |a_{ii}| - |z| > r_i - |z|, \qquad \text{and hence} \qquad |z| > 0.$$

Thus, $0 \notin D_A$ does not lie in the Gerschgorin domain and so cannot be an eigenvalue. *Q.E.D.*

Warning: The converse to this result is obviously not true; there are plenty of non-singular matrices that are not diagonally dominant.

6.5. Singular Values.

We have already indicated the central role played by the eigenvalues and eigenvectors of a square matrix in both theory and applications. Much more evidence to this effect will appear in the ensuing chapters. Alas, rectangular matrices do not have eigenvalues (why?), and so, at first glance, do not appear to possess any quantities of comparable significance. But you no doubt recall that our earlier treatment of least squares minimization problems, as well as the equilibrium equations for structures and circuits, made essential use of the symmetric, positive semi-definite *square* Gram matrix $K = A^T A$ — which can be naturally formed even when A is not square. Perhaps the eigenvalues of K might play a comparably important role for general matrices. Since they are not easily related to the eigenvalues of A — which, in the non-square case, don't even exist — we shall endow them with a new name.

Definition 6.27. The *singular values* $\sigma_1, \ldots, \sigma_r$ of an $m \times n$ matrix A are the positive square roots, $\sigma_i = \sqrt{\lambda_i} > 0$, of the nonzero eigenvalues of the associated Gram matrix $K = A^T A$. The corresponding eigenvectors of K are known as the *singular vectors* of A.

Since K is necessarily positive semi-definite, its eigenvalues are always non-negative, $\lambda_i \geq 0$, which justifies the positivity of the singular values of A — independently of whether A itself has positive, negative, or even complex eigenvalues — or is rectangular and has no eigenvalues at all. The standard convention is to label the singular values in decreasing order, so that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. Thus, σ_1 will always denote the largest or *dominant* singular value. If $K = A^T A$ has repeated eigenvalues, the singular values of A are repeated with the same multiplicities. As we will see, the number r of singular values is always equal to the rank of the matrix.

Warning: Many texts include the zero eigenvalues of K as singular values of A. We find this to be somewhat less convenient, but you should be aware of the differences in the two conventions.

Example 6.28. Let $A = \begin{pmatrix} 3 & 5 \\ 4 & 0 \end{pmatrix}$. The associated Gram matrix $K = A^T A = \begin{pmatrix} 25 & 15 \\ 15 & 25 \end{pmatrix}$ has eigenvalues $\lambda_1 = 40$, $\lambda_2 = 10$, and corresponding eigenvectors $\mathbf{v}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $\mathbf{v}_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$. Thus, the singular values of A are $\sigma_1 = \sqrt{40} \approx 6.3246$ and $\sigma_2 = \sqrt{10} \approx 3.1623$, with $\mathbf{v}_1, \mathbf{v}_2$ being the singular vectors. Note that the singular values are *not* the same as its eigenvalues, which are $\lambda_1 = \frac{1}{2}(3 + \sqrt{89}) \approx 6.2170$ and $\lambda_2 = \frac{1}{2}(3 - \sqrt{89}) \approx -3.2170$ — nor are the singular vectors eigenvectors of A.

Only in the special case of symmetric matrices is there a direct connection between the singular values and the eigenvalues.

Proposition 6.29. *If $A = A^T$ is a symmetric matrix, its singular values are the absolute values of its nonzero eigenvalues:* $\sigma_i = |\,\lambda_i\,| > 0$; *its singular vectors coincide with the associated non-null eigenvectors.*

Proof: When A is symmetric, $K = A^T A = A^2$. So, if $A\mathbf{v} = \lambda\mathbf{v}$, then $K\mathbf{v} = A^2\mathbf{v} = \lambda^2\mathbf{v}$. Thus, every eigenvector $\mathbf{v}$ of A is also an eigenvector of K with eigenvalue λ^2. Therefore, the eigenvector basis of A is also an eigenvector basis for K, and hence also forms a complete system of singular vectors for A. *Q.E.D.*

Condition Number, Rank, and Principal Component Analysis

The singular values not only provide a pretty geometric interpretation of the action of the matrix, they also play a key role in modern computational algorithms. The relative magnitudes of the singular values can be used to distinguish well-behaved linear systems from ill-conditioned systems which are much trickier to solve accurately. Since the number of singular values equals the matrix's rank, an $n\times n$ matrix with fewer than n singular values is singular. For the same reason, a square matrix with one or more very small singular values should be considered to be close to singular. The potential difficulty of accurately solving a linear algebraic system with coefficient matrix A is traditionally quantified as follows.

Definition 6.30. The *condition number* of a nonsingular $n \times n$ matrix is the ratio between its largest and smallest singular value: $\kappa(A) = \sigma_1/\sigma_n$.

If A is singular, it is said to have condition number ∞. A matrix with a very large condition number is said to be *ill-conditioned*; in practice, this occurs when the condition number is larger than the reciprocal of the machine's precision, e.g., 10^7 for typical single precision arithmetic. As the name implies, it is much harder to solve a linear system $A\mathbf{x} = \mathbf{b}$ when its coefficient matrix is ill-conditioned.

Determining the rank of a large matrix can be a numerical challenge. Small numerical errors in the entries can have an unpredictable effect. For example, the matrix $A = \begin{pmatrix} 1 & 1 & -1 \\ 2 & 2 & -2 \\ 3 & 3 & -3 \end{pmatrix}$ has rank $r = 1$, but a tiny change, e.g., $\widetilde{A} = \begin{pmatrix} 1.00001 & 1. & -1. \\ 2. & 2.00001 & -2. \\ 3. & 3. & -3.00001 \end{pmatrix}$, will produce a nonsingular matrix with rank $r = 3$. The latter matrix, however, is very close to singular, and this is highlighted by its singular values, which are $\sigma_1 \approx 6.48075$ while $\sigma_2 \approx \sigma_3 \approx .000001$. The fact that the second and third singular values are very small indicates that $\widetilde{A}$ is very close to a matrix of rank 1 and should be viewed as a numerical (or experimental) perturbation of such a matrix. Thus, an effective practical method for computing the rank of a matrix is to first assign a threshold, e.g., 10^{-5}, for singular values, and then treat any small singular value lying below the threshold as if it were zero.

This idea underlies the method of *Principal Component Analysis* that is assuming an increasingly visible role in modern statistics, data mining, imaging, speech recognition, semantics, and a variety of other fields, [**29**]. The singular vectors associated with the larger singular values indicate the *principal components* of the matrix, while small singular values indicate relatively unimportant directions. In applications, the columns of the matrix A represent the data vectors, which are normalized to have mean $\mathbf{0}$. The corresponding Gram matrix $K = A^T A$ can be identified as the associated *covariance matrix*, [**12**]. Its eigenvectors are the principal components that serve to indicate directions of correlation and clustering in the data.

AIMS Lecture Notes 2006

Peter J. Olver

7. Iterative Methods for Linear Systems

Linear iteration coincides with multiplication by successive powers of a matrix; convergence of the iterates depends on the magnitude of its eigenvalues. We discuss in some detail a variety of convergence criteria based on the spectral radius, on matrix norms, and on eigenvalue estimates provided by the Gerschgorin Circle Theorem.

We will then turn our attention to the three most important iterative schemes used to accurately approximate the solutions to linear algebraic systems. The classical Jacobi method is the simplest, while an evident serialization leads to the popular Gauss–Seidel method. Completely general convergence criteria are hard to formulate, although convergence is assured for the important class of diagonally dominant matrices that arise in many applications. A simple modification of the Gauss–Seidel scheme, known as Successive Over-Relaxation (SOR), can dramatically speed up the convergence rate, and is the method of choice in many modern applications. Finally, we introduce the method of conjugate gradients, a powerful "semi-direct" iterative scheme that, in contrast to the classical iterative schemes, is guaranteed to eventually produce the exact solution.

7.1. Linear Iterative Systems.

We begin with the basic definition of an iterative system of linear equations.

Definition 7.1. A *linear iterative system* takes the form

$$\mathbf{u}^{(k+1)} = T\,\mathbf{u}^{(k)}, \qquad \mathbf{u}^{(0)} = \mathbf{a}. \tag{7.1}$$

The *coefficient matrix* T has size $n \times n$. We will consider both real and complex systems, and so the *iterates*[†] $\mathbf{u}^{(k)}$ are vectors either in $\mathbb{R}^n$ (which assumes that the coefficient matrix T is also real) or in $\mathbb{C}^n$. For $k = 1, 2, 3, \ldots$, the solution $\mathbf{u}^{(k)}$ is uniquely determined by the *initial conditions* $\mathbf{u}^{(0)} = \mathbf{a}$.

Powers of Matrices

The solution to the general linear iterative system (7.1) is, at least at first glance, immediate. Clearly,

$$\mathbf{u}^{(1)} = T\,\mathbf{u}^{(0)} = T\,\mathbf{a}, \qquad \mathbf{u}^{(2)} = T\,\mathbf{u}^{(1)} = T^2\mathbf{a}, \qquad \mathbf{u}^{(3)} = T\,\mathbf{u}^{(2)} = T^3\mathbf{a},$$

[†] *Warning*: The superscripts on $\mathbf{u}^{(k)}$ refer to the iterate number, and should not be mistaken for derivatives.

and, in general,

$$\mathbf{u}^{(k)} = T^k \mathbf{a}. \tag{7.2}$$

Thus, the iterates are simply determined by multiplying the initial vector $\mathbf{a}$ by the successive powers of the coefficient matrix T. And so, unlike differential equations, proving the existence and uniqueness of solutions to an iterative system is completely trivial.

However, unlike real or complex scalars, the general formulae and qualitative behavior of the powers of a square matrix are not nearly so immediately apparent. (Before continuing, the reader is urged to experiment with simple 2×2 matrices, trying to detect patterns.) To make progress, recall how we managed to solve linear systems of differential equations by suitably adapting the known exponential solution from the scalar version. In the iterative case, the scalar solution formula (2.8) is written in terms of powers, not exponentials. This motivates us to try the power ansatz

$$\mathbf{u}^{(k)} = \lambda^k \mathbf{v}, \tag{7.3}$$

in which λ is a scalar and $\mathbf{v}$ is a fixed vector, as a possible solution to the system. We find

$$\mathbf{u}^{(k+1)} = \lambda^{k+1} \mathbf{v}, \qquad \text{while} \qquad T\mathbf{u}^{(k)} = T(\lambda^k \mathbf{v}) = \lambda^k T \mathbf{v}.$$

These two expressions will be equal if and only if

$$T\mathbf{v} = \lambda \mathbf{v}.$$

Therefore, (7.3) is a nontrivial solution to (7.1) if and only if λ is an *eigenvalue* of the coefficient matrix T and $\mathbf{v} \neq \mathbf{0}$ an associated *eigenvector*.

Thus, to each eigenvector and eigenvalue of the coefficient matrix, we can construct a solution to the iterative system. We can then appeal to linear superposition to combine the basic power solutions to form more general solutions. In particular, if the coefficient matrix is complete, then this method will, as in the case of linear ordinary differential equations, produce the general solution.

Theorem 7.2. *If the coefficient matrix T is complete, then the general solution to the linear iterative system $\mathbf{u}^{(k+1)} = T\mathbf{u}^{(k)}$ is given by*

$$\mathbf{u}^{(k)} = c_1 \lambda_1^k \mathbf{v}_1 + c_2 \lambda_2^k \mathbf{v}_2 + \cdots + c_n \lambda_n^k \mathbf{v}_n, \tag{7.4}$$

where $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are the linearly independent eigenvectors and $\lambda_1, \ldots, \lambda_n$ the corresponding eigenvalues of T. The coefficients $c_1, \ldots, c_n$ are arbitrary scalars and are uniquely prescribed by the initial conditions $\mathbf{u}^{(0)} = \mathbf{a}$.

Proof: Since we already know that (7.4) is a solution to the system for arbitrary $c_1, \ldots, c_n$, it suffices to show that we can match any prescribed initial conditions. To this end, we need to solve the linear system

$$\mathbf{u}^{(0)} = c_1 \mathbf{v}_1 + \cdots + c_n \mathbf{v}_n = \mathbf{a}. \tag{7.5}$$

Completeness of T implies that its eigenvectors form a basis of $\mathbb{C}^n$, and hence (7.5) always admits a solution. In matrix form, we can rewrite (7.5) as

$$S\mathbf{c} = \mathbf{a}, \qquad \text{so that} \qquad \mathbf{c} = S^{-1}\mathbf{a},$$

where $S = (\mathbf{v}_1\ \mathbf{v}_2\ \ldots\ \mathbf{v}_n)$ is the (nonsingular) matrix whose columns are the eigenvectors. *Q.E.D.*

Remark: Solutions in the incomplete cases rely on the Jordan canonical form. As with systems of differential equations, the formulas are more complicated, and will not be written out.

Example 7.3. Consider the iterative system

$$x^{(k+1)} = \tfrac{3}{5}\, x^{(k)} + \tfrac{1}{5}\, y^{(k)}, \qquad y^{(k+1)} = \tfrac{1}{5}\, x^{(k)} + \tfrac{3}{5}\, y^{(k)}, \tag{7.6}$$

with initial conditions

$$x^{(0)} = a, \qquad y^{(0)} = b. \tag{7.7}$$

The system can be rewritten in our matrix form (7.1), with

$$T = \begin{pmatrix} .6 & .2 \\ .2 & .6 \end{pmatrix}, \qquad \mathbf{u}^{(k)} = \begin{pmatrix} x^{(k)} \\ y^{(k)} \end{pmatrix}, \qquad \mathbf{a} = \begin{pmatrix} a \\ b \end{pmatrix}.$$

Solving the characteristic equation

$$\det(T - \lambda\,\mathrm{I}\,) = \lambda^2 - 1.2\,\lambda - .32 = 0$$

produces the eigenvalues $\lambda_1 = .8, \lambda_2 = .4$. We then solve the associated linear systems $(T - \lambda_j\,\mathrm{I}\,)\mathbf{v}_j = \mathbf{0}$ for the corresponding eigenvectors:

$$\lambda_1 = .8, \qquad \mathbf{v}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad\qquad \lambda_2 = .4, \qquad \mathbf{v}_2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

Therefore, the basic power solutions are

$$\mathbf{u}_1^{(k)} = (\,.8)^k \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad \mathbf{u}_2^{(k)} = (\,.4)^k \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

Theorem 7.2 tells us that the general solution is given as a linear combination,

$$\mathbf{u}^{(k)} = c_1\,\mathbf{u}_1^{(k)} + c_2\,\mathbf{u}_2^{(k)} = c_1\,(\,.8)^k \begin{pmatrix} 1 \\ 1 \end{pmatrix} + c_2\,(\,.4)^k \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} c_1\,(\,.8)^k - c_2\,(\,.4)^k \\ c_1\,(\,.8)^k + c_2\,(\,.4)^k \end{pmatrix},$$

where c_1, c_2 are determined by the initial conditions:

$$\mathbf{u}^{(0)} = \begin{pmatrix} c_1 - c_2 \\ c_1 + c_2 \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}, \qquad \text{and hence} \qquad c_1 = \frac{a+b}{2}\,, \qquad c_2 = \frac{b-a}{2}\,.$$

Therefore, the explicit formula for the solution to the initial value problem (7.6–7) is

$$x^{(k)} = (\,.8)^k\,\frac{a+b}{2} + (\,.4)^k\,\frac{a-b}{2}, \qquad y^{(k)} = (\,.8)^k\,\frac{a+b}{2} + (\,.4)^k\,\frac{b-a}{2}.$$

In particular, as $k \to \infty$, the iterates $\mathbf{u}^{(k)} \to \mathbf{0}$ converge to zero at a rate governed by the dominant eigenvalue $\lambda_1 = .8$. Thus, (7.6) defines a stable iterative system. Figure 7.1 illustrates the cumulative effect of the iteration. The initial conditions consist of a large number of points on the unit circle $x^2 + y^2 = 1$, which are successively mapped to points on progressively smaller and flatter ellipses, all converging towards the origin.

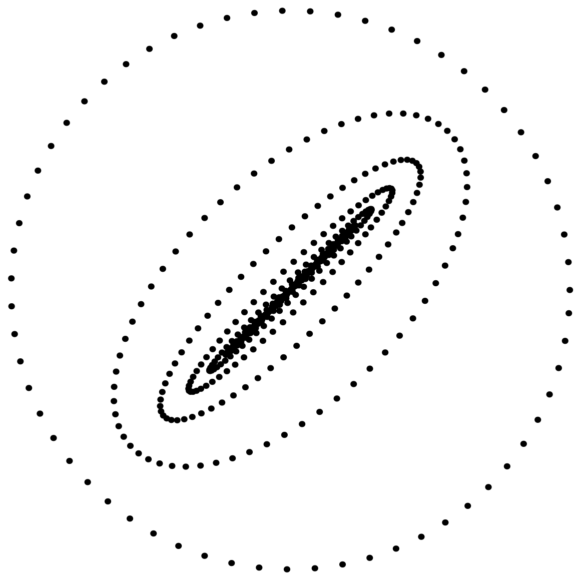


Figure 7.1. Stable Iterative System.

Example 7.4. The *Fibonacci numbers* are defined by the second order[†] iterative scheme

$$u^{(k+2)} = u^{(k+1)} + u^{(k)}, \tag{7.8}$$

with initial conditions

$$u^{(0)} = a, \qquad u^{(1)} = b. \tag{7.9}$$

In short, to obtain the next Fibonacci number, add the previous two. The classical *Fibonacci integers* start with $a = 0$, $b = 1$; the next few are

$$u^{(0)} = 0, \ \ u^{(1)} = 1, \ \ u^{(2)} = 1, \ \ u^{(3)} = 2, \ \ u^{(4)} = 3, \ \ u^{(5)} = 5, \ \ u^{(6)} = 8, \ \ u^{(7)} = 13, \ \ \ldots .$$

The Fibonacci integers occur in a surprising variety of natural objects, including leaves, flowers, and fruit, [**46**]. They were originally introduced by the Italian Renaissance mathematician Fibonacci (Leonardo of Pisa) as a crude model of the growth of a population of rabbits. In Fibonacci's model, the k^{th} Fibonacci number $u^{(k)}$ measures the total number of pairs of rabbits at year k. We start the process with a single juvenile pair[‡] at year 0. Once a year, each pair of rabbits produces a new pair of offspring, but it takes a full year for a rabbit pair to mature enough to produce offspring of their own.

Just as every higher order ordinary differential equation can be replaced by an equivalent first order system, so every higher order iterative equation can be replaced by a first

[†] In general, an iterative system $\mathbf{u}^{(k+j)} = T_1 \mathbf{u}^{(k+j-1)} + \cdots + T_j \mathbf{u}^{(k)}$ in which the new iterate depends upon the preceding j values is said to have *order* j.

[‡] We ignore important details like the sex of the offspring.

order iterative system. In this particular case, we define the vector

$$\mathbf{u}^{(k)} = \begin{pmatrix} u^{(k)} \\ u^{(k+1)} \end{pmatrix} \in \mathbb{R}^2,$$

and note that (7.8) is equivalent to the matrix system

$$\begin{pmatrix} u^{(k+1)} \\ u^{(k+2)} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} u^{(k)} \\ u^{(k+1)} \end{pmatrix}, \qquad \text{or} \qquad \mathbf{u}^{(k+1)} = T\,\mathbf{u}^{(k)}, \qquad \text{where} \qquad T = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}.$$

To find the explicit formula for the Fibonacci numbers, we must determine the eigenvalues and eigenvectors of the coefficient matrix T. A straightforward computation produces

$$\lambda_1 = \frac{1+\sqrt5}{2} = 1.618034\ldots\,, \qquad \lambda_2 = \frac{1-\sqrt5}{2} = -.618034\ldots\,,$$
$$\mathbf{v}_1 = \begin{pmatrix} \frac{-1+\sqrt5}{2} \\ 1 \end{pmatrix}, \qquad \mathbf{v}_2 = \begin{pmatrix} \frac{-1-\sqrt5}{2} \\ 1 \end{pmatrix}.$$

Therefore, according to (7.4), the general solution to the Fibonacci system is

$$\mathbf{u}^{(k)} = \begin{pmatrix} u^{(k)} \\ u^{(k+1)} \end{pmatrix} = c_1 \left(\frac{1+\sqrt5}{2} \right)^k \begin{pmatrix} \frac{-1+\sqrt5}{2} \\ 1 \end{pmatrix} + c_2 \left(\frac{1-\sqrt5}{2} \right)^k \begin{pmatrix} \frac{-1-\sqrt5}{2} \\ 1 \end{pmatrix}. \tag{7.10}$$

The initial data

$$\mathbf{u}^{(0)} = c_1 \begin{pmatrix} \frac{-1+\sqrt5}{2} \\ 1 \end{pmatrix} + c_2 \begin{pmatrix} \frac{-1-\sqrt5}{2} \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}$$

uniquely specifies the coefficients

$$c_1 = \frac{2\,a + (1+\sqrt5)\,b}{2\sqrt5}\,, \qquad c_2 = -\,\frac{2\,a + (1-\sqrt5)\,b}{2\sqrt5}\,.$$

The first entry of the solution vector (7.10) produces the explicit formula

$$u^{(k)} = \frac{(-1+\sqrt5)\,a + 2\,b}{2\sqrt5} \left(\frac{1+\sqrt5}{2} \right)^k + \frac{(1+\sqrt5)\,a - 2\,b}{2\sqrt5} \left(\frac{1-\sqrt5}{2} \right)^k \tag{7.11}$$

for the kth Fibonacci number. For the particular initial conditions $a = 0$, $b = 1$, (7.11) reduces to the classical *Binet formula*

$$u^{(k)} = \frac{1}{\sqrt5} \left[\left(\frac{1+\sqrt5}{2} \right)^k - \left(\frac{1-\sqrt5}{2} \right)^k \right] \tag{7.12}$$

for the kth Fibonacci integer. It is a remarkable fact that, for every value of k, all the $\sqrt5$'s cancel out, and the Binet formula does indeed produce the Fibonacci integers listed above. Another useful observation is that, since

$$0 < |\,\lambda_2\,| = \frac{\sqrt5 - 1}{2} < 1 < \lambda_1 = \frac{1+\sqrt5}{2}\,,$$

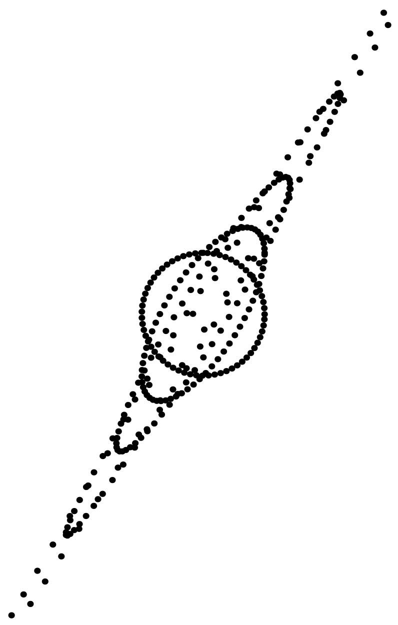


Figure 7.2. Fibonacci Iteration.

the terms involving λ_1^k go to ∞ (and so the zero solution to this iterative system is unstable) while the terms involving λ_2^k go to zero. Therefore, even for k moderately large, the first term in (7.11) is an excellent approximation (and one that gets more and more accurate with increasing k) to the k^{th} Fibonacci number. A plot of the first 4 iterates, starting with the initial data consisting of equally spaced points on the unit circle, can be seen in Figure 7.2. As in the previous example, the circle is mapped to a sequence of progressively more eccentric ellipses; however, their major semi-axes become more and more stretched out, and almost all points end up going off to ∞.

The dominant eigenvalue $\lambda_1 = \frac{1}{2}\left(1+\sqrt{5}\,\right) = 1.618034\ldots$ is known as the *golden ratio* and plays an important role in spiral growth in nature, as well as in art, architecture and design, [**46**]. It describes the overall growth rate of the Fibonacci integers, and, in fact, any sequence of Fibonacci numbers with initial conditions $b \neq \frac{1}{2}\left(1-\sqrt{5}\,\right)a$.

Example 7.5. Let $T = \begin{pmatrix} -3 & 1 & 6 \\ 1 & -1 & -2 \\ -1 & -1 & 0 \end{pmatrix}$ be the coefficient matrix for a three-dimensional iterative system $\mathbf{u}^{(k+1)} = T\,\mathbf{u}^{(k)}$. Its eigenvalues and corresponding eigenvectors are

$$\lambda_1 = -2, \qquad \lambda_2 = -1+\mathrm{i}\,, \qquad \lambda_3 = -1-\mathrm{i}\,,$$
$$\mathbf{v}_1 = \begin{pmatrix} 4 \\ -2 \\ 1 \end{pmatrix}, \qquad \mathbf{v}_2 = \begin{pmatrix} 2-\mathrm{i} \\ -1 \\ 1 \end{pmatrix}, \qquad \mathbf{v}_3 = \begin{pmatrix} 2+\mathrm{i} \\ -1 \\ 1 \end{pmatrix}.$$

Therefore, according to (7.4), the general complex solution to the iterative system is

$$\mathbf{u}^{(k)} = b_1\,(-2)^k \begin{pmatrix} 4 \\ -2 \\ 1 \end{pmatrix} + b_2\,(-1+\mathrm{i}\,)^k \begin{pmatrix} 2-\mathrm{i} \\ -1 \\ 1 \end{pmatrix} + b_3\,(-1-\mathrm{i}\,)^k \begin{pmatrix} 2+\mathrm{i} \\ -1 \\ 1 \end{pmatrix},$$

where b_1, b_2, b_3 are arbitrary complex scalars.

If we are only interested in real solutions, we can, as in the case of systems of differential equations, break up any complex solution into its real and imaginary parts, each of which constitutes a real solution. We begin by writing $\lambda_2 = -1 + \mathrm{i} = \sqrt{2}\,e^{3\pi\,\mathrm{i}/4}$, and hence

$$(-1+\mathrm{i}\,)^k = 2^{k/2}\,e^{3k\pi\,\mathrm{i}/4} = 2^{k/2}\left(\cos\tfrac{3}{4}k\pi + \mathrm{i}\,\sin\tfrac{3}{4}k\pi\right).$$

Therefore, the complex solution

$$(-1+\mathrm{i}\,)^k \begin{pmatrix} 2-\mathrm{i} \\ -1 \\ 1 \end{pmatrix} = 2^{k/2} \begin{pmatrix} 2\cos\frac{3}{4}k\pi + \sin\frac{3}{4}k\pi \\ -\cos\frac{3}{4}k\pi \\ \cos\frac{3}{4}k\pi \end{pmatrix} + \mathrm{i}\;2^{k/2} \begin{pmatrix} 2\sin\frac{3}{4}k\pi - \cos\frac{3}{4}k\pi \\ -\sin\frac{3}{4}k\pi \\ \sin\frac{3}{4}k\pi \end{pmatrix}$$

is a combination of two independent real solutions. The complex conjugate eigenvalue $\lambda_3 = -1 - \mathrm{i}$ leads, as before, to the complex conjugate solution — and the same two real solutions. The general real solution $\mathbf{u}^{(k)}$ to the system can be written as a linear combination of the three independent real solutions:

$$c_1\,(-2)^k \begin{pmatrix} 4 \\ -2 \\ 1 \end{pmatrix} + c_2\,2^{k/2} \begin{pmatrix} 2\cos\frac{3}{4}k\pi + \sin\frac{3}{4}k\pi \\ -\cos\frac{3}{4}k\pi \\ \cos\frac{3}{4}k\pi \end{pmatrix} + c_3\,2^{k/2} \begin{pmatrix} 2\sin\frac{3}{4}k\pi - \cos\frac{3}{4}k\pi \\ -\sin\frac{3}{4}k\pi \\ \sin\frac{3}{4}k\pi \end{pmatrix}, \tag{7.13}$$

where c_1, c_2, c_3 are arbitrary real scalars, uniquely prescribed by the initial conditions.

7.2. Stability.

With the solution formula (7.4) in hand, we are now in a position to understand the qualitative behavior of solutions to (complete) linear iterative systems. The most important case for applications is when all the iterates converge to $\mathbf{0}$.

Definition 7.6. The equilibrium solution $\mathbf{u}^\star = \mathbf{0}$ to a linear iterative system (7.1) is called *asymptotically stable* if and only if all solutions $\mathbf{u}^{(k)} \to \mathbf{0}$ as $k \to \infty$.

Asymptotic stability relies on the following property of the coefficient matrix.

Definition 7.7. A matrix T is called *convergent* if its powers converge to the zero matrix, $T^k \to \mathrm{O}$, meaning that the individual entries of T^k all go to 0 as $k \to \infty$.

The equivalence of the convergence condition and stability of the iterative system follows immediately from the solution formula (7.2).

Proposition 7.8. *The linear iterative system* $\mathbf{u}^{(k+1)} = T\,\mathbf{u}^{(k)}$ *has asymptotically stable zero solution if and only if* T *is a convergent matrix.*

Proof: If $T^k \to \mathrm{O}$, and $\mathbf{u}^{(k)} = T^k\,\mathbf{a}$ is any solution, then clearly $\mathbf{u}^{(k)} \to \mathbf{0}$ as $k \to \infty$, proving stability. Conversely, the solution $\mathbf{u}_j^{(k)} = T^k \mathbf{e}_j$ is the same as the j^{th} column of T^k. If the origin is asymptotically stable, then $\mathbf{u}_j^{(k)} \to \mathbf{0}$. Thus, the individual columns of T^k all tend to $\mathbf{0}$, proving that $T^k \to \mathrm{O}$. *Q.E.D.*

To facilitate the analysis of convergence, we shall adopt a norm $\|\cdot\|$ on our underlying vector space, $\mathbb{R}^n$ or $\mathbb{C}^n$. The reader may be inclined to choose the Euclidean (or Hermitian) norm, but, in practice, the ∞ norm

$$\|\,\mathbf{u}\,\|_\infty = \max\{\,|\,u_1\,|,\ \ldots\ ,|\,u_n\,|\,\} \tag{7.14}$$

prescribed by the vector's maximal entry (in modulus) is usually much easier to work with. Convergence of the iterates is equivalent to convergence of their norms:

$$\mathbf{u}^{(k)} \to \mathbf{0} \qquad \text{if and only if} \qquad \|\,\mathbf{u}^{(k)}\,\| \to 0 \qquad \text{as} \qquad k \to \infty.$$

The fundamental stability criterion for linear iterative systems relies on the size of the eigenvalues of the coefficient matrix.

Theorem 7.9. *A linear iterative system* (7.1) *is asymptotically stable if and only if all its* (*complex*) *eigenvalues have modulus strictly less than one*: $|\,\lambda_j\,| < 1$.

Proof: Let us prove this result assuming that the coefficient matrix T is complete. (The proof in the incomplete case relies on the Jordan canonical form, and is outlined in the exercises.) If λ_j is an eigenvalue such that $|\,\lambda_j\,| < 1$, then the corresponding basis solution $\mathbf{u}_j^{(k)} = \lambda_j^k\,\mathbf{v}_j$ tends to zero as $k \to \infty$; indeed,

$$\|\,\mathbf{u}_j^{(k)}\,\| = \|\,\lambda_j^k\,\mathbf{v}_j\,\| = |\,\lambda_j\,|^k\,\|\,\mathbf{v}_j\,\| \ \longrightarrow\ 0 \qquad \text{since} \qquad |\,\lambda_j\,| < 1.$$

Therefore, if all eigenvalues are less than 1 in modulus, all terms in the solution formula (7.4) tend to zero, which proves asymptotic stability: $\mathbf{u}^{(k)} \to \mathbf{0}$. Conversely, if any eigenvalue satisfies $|\,\lambda_j\,| \geq 1$, then the solution $\mathbf{u}^{(k)} = \lambda_j^k\,\mathbf{v}_j$ does not tend to $\mathbf{0}$ as $k \to \infty$, and hence $\mathbf{0}$ is not asymptotically stable. *Q.E.D.*

Consequently, the necessary and sufficient condition for asymptotic stability of a linear iterative system is that all the eigenvalues of the coefficient matrix lie strictly inside the unit circle in the complex plane: $|\,\lambda_j\,| < 1$.

Definition 7.10. The *spectral radius* of a matrix T is defined as the maximal modulus of all of its real and complex eigenvalues: $\rho(T) = \max\,\{\,|\,\lambda_1\,|, \ldots, |\,\lambda_k\,|\,\}$.

We can then restate the Stability Theorem 7.9 as follows:

Theorem 7.11. *The matrix T is convergent if and only if its spectral radius is strictly less than one*: $\rho(T) < 1$.

If T is complete, then we can apply the triangle inequality to (7.4) to estimate

$$\begin{aligned}
\| \mathbf{u}^{(k)} \| &= \| c_1 \lambda_1^k \mathbf{v}_1 + \cdots + c_n \lambda_n^k \mathbf{v}_n \| \\
&\le | \lambda_1 |^k \| c_1 \mathbf{v}_1 \| + \cdots + | \lambda_n |^k \| c_n \mathbf{v}_n \| \\
&\le \rho(T)^k \big(| c_1 | \, \| \mathbf{v}_1 \| + \cdots + | c_n | \, \| \mathbf{v}_n \| \big) = C \, \rho(T)^k,
\end{aligned} \tag{7.15}$$

for some constant $C > 0$ that depends only upon the initial conditions. In particular, if $\rho(T) < 1$, then

$$\| \mathbf{u}^{(k)} \| \;\le\; C \, \rho(T)^k \;\longrightarrow\; 0 \qquad \text{as} \qquad k \to \infty, \tag{7.16}$$

in accordance with Theorem 7.11. Thus, the spectral radius prescribes the rate of convergence of the solutions to equilibrium. The smaller the spectral radius, the faster the solutions go to $\mathbf{0}$.

If T has only one largest (simple) eigenvalue, so $| \lambda_1 | > | \lambda_j |$ for all $j > 1$, then the first term in the solution formula (7.4) will eventually dominate all the others: $\| \lambda_1^k \mathbf{v}_1 \| \gg \| \lambda_j^k \mathbf{v}_j \|$ for $j > 1$ and $k \gg 0$. Therefore, provided that $c_1 \neq 0$, the solution (7.4) has the asymptotic formula

$$\mathbf{u}^{(k)} \;\approx\; c_1 \lambda_1^k \mathbf{v}_1, \tag{7.17}$$

and so most solutions end up parallel to $\mathbf{v}_1$. In particular, if $| \lambda_1 | = \rho(T) < 1$, such a solution approaches $\mathbf{0}$ along the direction of the dominant eigenvector $\mathbf{v}_1$ at a rate governed by the modulus of the dominant eigenvalue. The exceptional solutions, with $c_1 = 0$, tend to $\mathbf{0}$ at a faster rate, along one of the other eigendirections. In practical computations, one rarely observes the exceptional solutions. Indeed, even if the initial condition does not involve the dominant eigenvector, round-off error during the iteration will almost inevitably introduce a small component in the direction of $\mathbf{v}_1$, which will, if you wait long enough, eventually dominate the computation.

Warning: The inequality (7.15) only applies to complete matrices. In the general case, one can prove that the solution satisfies the slightly weaker inequality

$$\| \mathbf{u}^{(k)} \| \le C \, \sigma^k \qquad \text{for all} \qquad k \ge 0, \qquad \text{where} \qquad \sigma > \rho(T) \tag{7.18}$$

is any number larger than the spectral radius, while $C > 0$ is a positive constant (whose value may depend on how close σ is to ρ).

Example 7.12. According to Example 7.5, the matrix

$$T = \begin{pmatrix} -3 & 1 & 6 \\ 1 & -1 & -2 \\ -1 & -1 & 0 \end{pmatrix} \qquad \text{has eigenvalues} \qquad \begin{aligned} \lambda_1 &= -2, \\ \lambda_2 &= -1 + \mathrm{i}\,, \\ \lambda_3 &= -1 - \mathrm{i}\,. \end{aligned}$$

Since $| \lambda_1 | = 2 > | \lambda_2 | = | \lambda_3 | = \sqrt{2}$, the spectral radius is $\rho(T) = | \lambda_1 | = 2$. We conclude that T is not a convergent matrix. As the reader can check, either directly, or from the solution formula (7.13), the vectors $\mathbf{u}^{(k)} = T^k \mathbf{u}^{(0)}$ obtained by repeatedly multiplying any nonzero initial vector $\mathbf{u}^{(0)}$ by T rapidly go off to ∞, at a rate roughly equal to $\rho(T)^k = 2^k$.

On the other hand, the matrix

$$\widetilde{T} = -\tfrac{1}{3}\,T = \begin{pmatrix} 1 & -\frac{1}{3} & -2 \\ -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} \\ \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix} \qquad \text{with eigenvalues} \qquad \begin{aligned} \lambda_1 &= \tfrac{2}{3}, \\ \lambda_2 &= \tfrac{1}{3} - \tfrac{1}{3}\,\mathrm{i}\,, \\ \lambda_3 &= \tfrac{1}{3} + \tfrac{1}{3}\,\mathrm{i}\,, \end{aligned}$$

has spectral radius $\rho(\widetilde{T}) = \frac{2}{3}$, and hence is a convergent matrix. According to (7.17), if we write the initial data $\mathbf{u}^{(0)} = c_1\,\mathbf{v}_1 + c_2\,\mathbf{v}_2 + c_3\,\mathbf{v}_3$ as a linear combination of the eigenvectors, then, provided $c_1 \neq 0$, the iterates have the asymptotic form $\mathbf{u}^{(k)} \approx c_1\,\left(-\frac{2}{3}\right)^k\,\mathbf{v}_1$, where $\mathbf{v}_1 = (\,4, -2, 1\,)^T$ is the eigenvector corresponding to the dominant eigenvalue $\lambda_1 = -\frac{2}{3}$. Thus, for most initial vectors, the iterates end up decreasing in length by a factor of almost exactly $\frac{2}{3}$, eventually becoming parallel to the dominant eigenvector $\mathbf{v}_1$. This is borne out by a sample computation: starting with $\mathbf{u}^{(0)} = (\,1, 1, 1\,)^T$, the first ten iterates are

$$\begin{pmatrix} -.0936 \\ .0462 \\ -.0231 \end{pmatrix}, \quad \begin{pmatrix} -.0627 \\ .0312 \\ -.0158 \end{pmatrix}, \quad \begin{pmatrix} -.0416 \\ .0208 \\ -.0105 \end{pmatrix}, \quad \begin{pmatrix} -.0275 \\ .0138 \\ -.0069 \end{pmatrix}, \quad \begin{pmatrix} -.0182 \\ .0091 \\ -.0046 \end{pmatrix},$$

$$\begin{pmatrix} -.0121 \\ .0061 \\ -.0030 \end{pmatrix}, \quad \begin{pmatrix} -.0081 \\ .0040 \\ -.0020 \end{pmatrix}, \quad \begin{pmatrix} -.0054 \\ .0027 \\ -.0013 \end{pmatrix}, \quad \begin{pmatrix} -.0036 \\ .0018 \\ -.0009 \end{pmatrix}, \quad \begin{pmatrix} -.0024 \\ .0012 \\ -.0006 \end{pmatrix}.$$

7.3. Matrix Norms.

The convergence of a linear iterative system is governed by the spectral radius or largest eigenvalue (in modulus) of the coefficient matrix. Unfortunately, finding accurate approximations to the eigenvalues of most matrices is a nontrivial computational task. Indeed, all practical numerical algorithms rely on some form of iteration. But using iteration to determine the spectral radius defeats the purpose, which is to predict the behavior of the iterative system in advance!

In this section, we present two alternative approaches for directly investigating convergence and stability issues. Matrix norms form a natural class of norms on the vector space of $n \times n$ matrices and can, in many instances, be used to establish convergence with a minimal effort.

Matrix Norms

We work exclusively with real $n \times n$ matrices in this section, although the results straightforwardly extend to complex matrices. We begin by fixing a norm $\|\cdot\|$ on $\mathbb{R}^n$. The norm may or may not come from an inner product — this is irrelevant as far as the construction goes. Each norm on $\mathbb{R}^n$ will naturally induce a norm on the vector space $\mathcal{M}_{n\times n}$ of all $n \times n$ matrices. Roughly speaking, the matrix norm tells us how much a linear transformation stretches vectors relative to the given norm.

Theorem 7.13. *If $\| \cdot \|$ is any norm on $\mathbb{R}^n$, then the quantity*

$$\| A \| = \max \{ \| A \mathbf{u} \| \mid \| \mathbf{u} \| = 1 \} \tag{7.19}$$

defines a norm on $\mathcal{M}_{n \times n}$, known as the natural matrix norm.

Proof: First note that $\| A \| < \infty$, since the maximum is taken on a closed and bounded subset, namely the unit sphere $S_1 = \{ \| \mathbf{u} \| = 1 \}$ for the given norm. To show that (7.19) defines a norm, we need to verify the three basic axioms of Definition 5.8.

Non-negativity, $\| A \| \geq 0$, is immediate. Suppose $\| A \| = 0$. This means that, for every unit vector, $\| A \mathbf{u} \| = 0$, and hence $A \mathbf{u} = \mathbf{0}$ whenever $\| \mathbf{u} \| = 1$. If $\mathbf{0} \neq \mathbf{v} \in \mathbb{R}^n$ is any nonzero vector, then $\mathbf{u} = \mathbf{v}/r$, where $r = \| \mathbf{v} \|$, is a unit vector, so

$$A \mathbf{v} = A(r \, \mathbf{u}) = r \, A \mathbf{u} = \mathbf{0}. \tag{7.20}$$

Therefore, $A \mathbf{v} = \mathbf{0}$ for every $\mathbf{v} \in \mathbb{R}^n$, which implies $A = \mathrm{O}$ is the zero matrix. This serves to prove the positivity property. As for homogeneity, if $c \in \mathbb{R}$ is any scalar,

$$\| c A \| = \max \{ \| c A \mathbf{u} \| \} = \max \{ | c | \, \| A \mathbf{u} \| \} = | c | \max \{ \| A \mathbf{u} \| \} = | c | \, \| A \|.$$

Finally, to prove the triangle inequality, we use the fact that the maximum of the sum of quantities is bounded by the sum of their individual maxima. Therefore, since the norm on $\mathbb{R}^n$ satisfies the triangle inequality,

$$\begin{aligned} \| A + B \| &= \max \{ \| A \mathbf{u} + B \mathbf{u} \| \} \leq \max \{ \| A \mathbf{u} \| + \| B \mathbf{u} \| \} \\ &\leq \max \{ \| A \mathbf{u} \| \} + \max \{ \| B \mathbf{u} \| \} = \| A \| + \| B \|. \end{aligned}$$

Q.E.D.

The property that distinguishes a matrix norm from a generic norm on the space of matrices is the fact that it also obeys a very useful *product inequality*.

Theorem 7.14. *A natural matrix norm satisfies*

$$\| A \mathbf{v} \| \leq \| A \| \, \| \mathbf{v} \|, \qquad \text{for all} \qquad A \in \mathcal{M}_{n \times n}, \quad \mathbf{v} \in \mathbb{R}^n. \tag{7.21}$$

Furthermore,

$$\| A B \| \leq \| A \| \, \| B \|, \qquad \text{for all} \qquad A, B \in \mathcal{M}_{n \times n}. \tag{7.22}$$

Proof: Note first that, by definition $\| A \mathbf{u} \| \leq \| A \|$ for all unit vectors $\| \mathbf{u} \| = 1$. Then, letting $\mathbf{v} = r \, \mathbf{u}$ where $\mathbf{u}$ is a unit vector and $r = \| \mathbf{v} \|$, we have

$$\| A \mathbf{v} \| = \| A (r \, \mathbf{u}) \| = r \, \| A \mathbf{u} \| \leq r \, \| A \| = \| \mathbf{v} \| \, \| A \|,$$

proving the first inequality. To prove the second, we apply the first to compute

$$\begin{aligned} \| A B \| &= \max \{ \| A B \mathbf{u} \| \} = \max \{ \| A (B \mathbf{u}) \| \} \\ &\leq \max \{ \| A \| \, \| B \mathbf{u} \| \} = \| A \| \max \{ \| B \mathbf{u} \| \} = \| A \| \, \| B \|. \end{aligned}$$

Q.E.D.

Remark: In general, a norm on the vector space of $n \times n$ matrices is called a *matrix norm* if it also satisfies the multiplicative inequality (7.22). Most, but not all, matrix norms used in applications come from norms on the underlying vector space.

The multiplicative inequality (7.22) implies, in particular, that $\| A^2 \| \leq \| A \|^2$; equality is not necessarily valid. More generally:

Proposition 7.15. *If A is a square matrix, then $\| A^k \| \leq \| A \|^k$. In particular, if $\| A \| < 1$, then $\| A^k \| \to 0$ as $k \to \infty$, and hence A is a convergent matrix: $A^k \to \mathrm{O}$.*

The converse is not quite true; a convergent matrix does not necessarily have matrix norm less than 1, or even ≤ 1 — see Example 7.20 below. An alternative proof of Proposition 7.15 can be based on the following useful estimate:

Theorem 7.16. *The spectral radius of a matrix is bounded by its matrix norm:*

$$\rho(A) \leq \| A \|. \tag{7.23}$$

Proof: If λ is a real eigenvalue, and $\mathbf{u}$ a corresponding unit eigenvector, so that $A\mathbf{u} = \lambda\,\mathbf{u}$ with $\| \mathbf{u} \| = 1$, then

$$\| A\mathbf{u} \| = \| \lambda\,\mathbf{u} \| = |\,\lambda\,|\ \| \mathbf{u} \| = |\,\lambda\,|. \tag{7.24}$$

Since $\| A \|$ is the maximum of $\| A\mathbf{u} \|$ over all possible unit vectors, this implies that

$$|\,\lambda\,| \leq \| A \|. \tag{7.25}$$

If all the eigenvalues of A are real, then the spectral radius is the maximum of their absolute values, and so it too is bounded by $\| A \|$, proving (7.23).

If A has complex eigenvalues, then we need to work a little harder to establish (7.25). (This is because the matrix norm is defined by the effect of A on *real* vectors, and so we cannot directly use the complex eigenvectors to establish the required bound.) Let $\lambda = r\,e^{\,\mathrm{i}\,\theta}$ be a complex eigenvalue with complex eigenvector $\mathbf{z} = \mathbf{x} + \mathrm{i}\,\mathbf{y}$. Define

$$m = \min \left\{ \| \operatorname{Re}\,(e^{\,\mathrm{i}\,\varphi}\,\mathbf{z}) \| = \| (\cos\varphi)\,\mathbf{x} - (\sin\varphi)\,\mathbf{y} \| \;\middle|\; 0 \leq \varphi \leq 2\pi \right\}. \tag{7.26}$$

Since the indicated subset is a closed curve (in fact, an ellipse) that does not go through the origin, $m > 0$. Let φ_0 denote the value of the angle that produces the minimum, so

$$m = \| (\cos\varphi_0)\,\mathbf{x} - (\sin\varphi_0)\,\mathbf{y} \| = \| \operatorname{Re}\left(e^{\,\mathrm{i}\,\varphi_0}\,\mathbf{z} \right) \|.$$

Define the real unit vector

$$\mathbf{u} = \frac{\operatorname{Re}\left(e^{\,\mathrm{i}\,\varphi_0}\,\mathbf{z} \right)}{m} = \frac{(\cos\varphi_0)\,\mathbf{x} - (\sin\varphi_0)\,\mathbf{y}}{m}, \qquad \text{so that} \qquad \| \mathbf{u} \| = 1.$$

Then

$$A\mathbf{u} = \frac{1}{m} \operatorname{Re}\left(e^{\,\mathrm{i}\,\varphi_0}\,A\,\mathbf{z} \right) = \frac{1}{m} \operatorname{Re}\left(e^{\,\mathrm{i}\,\varphi_0}\,r\,e^{\,\mathrm{i}\,\theta}\,\mathbf{z} \right) = \frac{r}{m} \operatorname{Re}\left(e^{\,\mathrm{i}\,(\varphi_0+\theta)}\,\mathbf{z} \right).$$

Therefore, keeping in mind that m is the minimal value in (7.26),

$$\| A \| \geq \| A\mathbf{u} \| = \frac{r}{m} \| \operatorname{Re}\left(e^{\,\mathrm{i}\,(\varphi_0+\theta)}\,\mathbf{z} \right) \| \geq r = |\,\lambda\,|, \tag{7.27}$$

and so (7.25) also holds for complex eigenvalues. *Q.E.D.*

Explicit Formulae

Let us now determine the explicit formulae for the matrix norms induced by our most important vector norms on $\mathbb{R}^n$. The simplest to handle is the ∞ norm

$$\| \mathbf{v} \|_\infty = \max \{ \, | v_1 |, \; \ldots \; , | v_n | \, \}.$$

Definition 7.17. The i^{th} *absolute row sum* of a matrix A is the sum of the absolute values of the entries in the i^{th} row:

$$s_i = | a_{i1} | + \; \cdots \; + | a_{in} | = \sum_{j=1}^{n} | a_{ij} |. \tag{7.28}$$

Proposition 7.18. *The* ∞ matrix norm *of a matrix A is equal to its maximal absolute row sum*:

$$\| A \|_\infty = \max\{ s_1, \ldots, s_n \} = \max \left\{ \sum_{j=1}^{n} | a_{ij} | \;\middle|\; 1 \leq i \leq n \right\}. \tag{7.29}$$

Proof: Let $s = \max\{ s_1, \ldots, s_n \}$ denote the right hand side of (7.29). Given any $\mathbf{v} \in \mathbb{R}^n$, we compute

$$\begin{aligned} \| A \mathbf{v} \|_\infty &= \max \left\{ \left| \sum_{j=1}^{n} a_{ij} v_j \right| \right\} \leq \max \left\{ \sum_{j=1}^{n} | a_{ij} v_j | \right\} \\ &\leq \max \left\{ \sum_{j=1}^{n} | a_{ij} | \right\} \max \{ \, | v_j | \, \} = s \, \| \mathbf{v} \|_\infty . \end{aligned}$$

In particular, by specializing to $\| \mathbf{v} \|_\infty = 1$, we deduce that $\| A \|_\infty \leq s$.

On the other hand, suppose the maximal absolute row sum occurs at row i, so

$$s_i = \sum_{j=1}^{n} | a_{ij} | = s. \tag{7.30}$$

Let $\mathbf{u} \in \mathbb{R}^n$ be the specific vector that has the following entries: $u_j = +1$ if $a_{ij} > 0$, while $u_j = -1$ if $a_{ij} < 0$. Then $\| \mathbf{u} \|_\infty = 1$. Moreover, since $a_{ij} u_j = | a_{ij} |$, the i^{th} entry of $A \mathbf{u}$ is equal to the i^{th} absolute row sum (7.30). This implies that

$$\| A \|_\infty \geq \| A \mathbf{u} \|_\infty \geq s. \qquad \textit{Q.E.D.}$$

Combining Propositions 7.15 and 7.18, we have established the following convergence criterion.

Corollary 7.19. *If all the absolute row sums of A are strictly less than* 1, *then* $\| A \|_\infty < 1$ *and hence A is a convergent matrix.*

Example 7.20. Consider the symmetric matrix $A = \begin{pmatrix} \frac{1}{2} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{1}{4} \end{pmatrix}$. Its two absolute row sums are $\left| \frac{1}{2} \right| + \left| -\frac{1}{3} \right| = \frac{5}{6}$, $\left| -\frac{1}{3} \right| + \left| \frac{1}{4} \right| = \frac{7}{12}$, so

$$\| A \|_\infty = \max \left\{ \tfrac{5}{6}, \tfrac{7}{12} \right\} = \tfrac{5}{6} \approx .83333\ldots .$$

Since the norm is less than 1, A is a convergent matrix. Indeed, its eigenvalues are

$$\lambda_1 = \frac{9 + \sqrt{73}}{24} \approx .7310\ldots , \qquad \lambda_2 = \frac{9 - \sqrt{73}}{24} \approx .0190\ldots ,$$

and hence the spectral radius is

$$\rho(A) = \frac{9 + \sqrt{73}}{24} \approx .7310\ldots ,$$

which is slightly smaller than its ∞ norm.

The row sum test for convergence is not always conclusive. For example, the matrix

$$A = \begin{pmatrix} \frac{1}{2} & -\frac{3}{5} \\ \frac{3}{5} & \frac{1}{4} \end{pmatrix} \qquad \text{has matrix norm} \qquad \| A \|_\infty = \tfrac{11}{10} > 1. \tag{7.31}$$

On the other hand, its eigenvalues are $(15 \pm \sqrt{601}\,)/40$, and hence its spectral radius is

$$\rho(A) = \frac{15 + \sqrt{601}}{40} \approx .98788\ldots ,$$

which implies that A is (just barely) convergent, even though its maximal row sum is larger than 1.

The matrix norm associated with the Euclidean norm $\| \mathbf{v} \|_2 = \sqrt{v_1^2 + \cdots + v_n^2}$ is given by largest singular value.

Proposition 7.21. *The matrix norm corresponding to the Euclidean norm equals the maximal singular value*:

$$\| A \|_2 = \sigma_1 = \max \{\sigma_1, \ldots, \sigma_r\}, \qquad r = \operatorname{rank} A > 0, \qquad \text{while} \qquad \| \mathrm{O} \|_2 = 0. \tag{7.32}$$

Unfortunately, as we discovered in Example 7.20, matrix norms are not a foolproof test of convergence. There exist convergent matrices such that $\rho(A) < 1$ and yet have matrix norm $\| A \| \geq 1$. In such cases, the matrix norm is not able to predict convergence of the iterative system, although one should expect the convergence to be quite slow. Although such pathology might show up in the chosen matrix norm, it turns out that one can always rig up some matrix norm for which $\| A \| < 1$. This follows from a more general result, whose proof can be found in [**40**].

Theorem 7.22. *Let A have spectral radius $\rho(A)$. If $\varepsilon > 0$ is any positive number, then there exists a matrix norm $\| \cdot \|$ such that*

$$\rho(A) \leq \| A \| < \rho(A) + \varepsilon. \tag{7.33}$$

Corollary 7.23. *If A is a convergent matrix, then there exists a matrix norm such that $\| A \| < 1$.*

Proof: By definition, A is convergent if and only if $\rho(A) < 1$. Choose $\varepsilon > 0$ such that $\rho(A) + \varepsilon < 1$. Any norm that satisfies (7.33) has the desired property. *Q.E.D.*

Remark: Based on the accumulated evidence, one might be tempted to speculate that the spectral radius itself defines a matrix norm. Unfortunately, this is not the case. For example, the nonzero matrix $A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ has zero spectral radius, $\rho(A) = 0$, in violation of a basic norm axiom.

7.4. Iterative Solution of Linear Algebraic Systems.

In this section, we return to the most basic problem in linear algebra: solving the linear algebraic system

$$A\mathbf{u} = \mathbf{b}, \tag{7.34}$$

consisting of n equations in n unknowns. We assume that the coefficient matrix A is nonsingular, and so the solution $\mathbf{u} = A^{-1}\mathbf{b}$ is unique.

We will introduce several popular iterative methods that can be used to approximate the solution for certain classes of coefficient matrices. The resulting algorithms will provide an attractive alternative to Gaussian Elimination, particularly when dealing with the large, sparse systems that arise in the numerical solution to differential equations. One major advantage of an iterative technique is that it (typically) produces progressively more and more accurate approximations to the solution, and hence, by prolonging the iterations, can, at least in principle, compute the solution to any desired order of accuracy. Moreover, even performing just a few iterations may produce a reasonable approximation to the true solution — in stark contrast to Gaussian Elimination, where one must continue the algorithm through to the bitter end before any useful information can be extracted. A partially completed Gaussian Elimination is of scant use! A significant weakness is that iterative schemes are not universally applicable, and their design relies upon the detailed structure of the coefficient matrix.

We shall be attempting to solve the linear system (7.34) by replacing it with an iterative system of the form

$$\mathbf{u}^{(k+1)} = T\mathbf{u}^{(k)} + \mathbf{c}, \qquad \mathbf{u}^{(0)} = \mathbf{u}_0, \tag{7.35}$$

in which T is an $n \times n$ matrix and $\mathbf{c}$ a vector. This represents a slight generalization of our earlier iterative system (7.1), in that the right hand side is now an affine function of $\mathbf{u}^{(k)}$. Suppose that the solutions to the affine iterative system converge: $\mathbf{u}^{(k)} \to \mathbf{u}^\star$ as $k \to \infty$. Then, by taking the limit of both sides of (7.35), we discover that the limit point $\mathbf{u}^\star$ solves the *fixed-point equation*

$$\mathbf{u}^\star = T\mathbf{u}^\star + \mathbf{c}. \tag{7.36}$$

Thus, we need to design our iterative system so that

(*a*) te solution to the fixed-point system $\mathbf{u} = T\mathbf{u} + \mathbf{c}$ coincides with the solution to the original system $A\mathbf{u} = \mathbf{b}$, and

(*b*) the iterates defined by (7.35) are known to converge to the fixed point.

Before exploring these issues in depth, let us look at a simple example.

Example 7.24. Consider the linear system

$$3x + y - z = 3, \qquad x - 4y + 2z = -1, \qquad -2x - y + 5z = 2, \tag{7.37}$$

which has the vectorial form $A\mathbf{u} = \mathbf{b}$, with

$$A = \begin{pmatrix} 3 & 1 & -1 \\ 1 & -4 & 2 \\ -2 & -1 & 5 \end{pmatrix}, \qquad \mathbf{u} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 3 \\ -1 \\ 2 \end{pmatrix}.$$

One easy way to convert a linear system into a fixed-point form is to rewrite it as

$$\mathbf{u} = \mathrm{I}\,\mathbf{u} - A\mathbf{u} + A\mathbf{u} = (\mathrm{I} - A)\mathbf{u} + \mathbf{b} = T\mathbf{u} + \mathbf{c}, \qquad \text{where} \qquad T = \mathrm{I} - A, \qquad \mathbf{c} = \mathbf{b}.$$

In the present case,

$$T = \mathrm{I} - A = \begin{pmatrix} -2 & -1 & 1 \\ -1 & 5 & -2 \\ 2 & 1 & -4 \end{pmatrix}, \qquad \mathbf{c} = \mathbf{b} = \begin{pmatrix} 3 \\ -1 \\ 2 \end{pmatrix}.$$

The resulting iterative system $\mathbf{u}^{(k+1)} = T\mathbf{u}^{(k)} + \mathbf{c}$ has the explicit form

$$\begin{aligned} x^{(k+1)} &= -2x^{(k)} - y^{(k)} + z^{(k)} + 3, \\ y^{(k+1)} &= \;\; -x^{(k)} + 5y^{(k)} - 2z^{(k)} - 1, \\ z^{(k+1)} &= \;\; 2x^{(k)} + y^{(k)} - 4z^{(k)} + 2. \end{aligned} \tag{7.38}$$

Another possibility is to solve the first equation in (7.37) for x, the second for y, and the third for z, so that

$$x = -\tfrac{1}{3}\,y + \tfrac{1}{3}\,z + 1, \qquad y = \tfrac{1}{4}\,x + \tfrac{1}{2}\,z + \tfrac{1}{4}, \qquad z = \tfrac{2}{5}\,x + \tfrac{1}{5}\,y + \tfrac{2}{5}.$$

The resulting equations have the form of a fixed-point system

$$\mathbf{u} = \widehat{T}\mathbf{u} + \widehat{\mathbf{c}}, \qquad \text{in which} \qquad \widehat{T} = \begin{pmatrix} 0 & -\frac{1}{3} & \frac{1}{3} \\ \frac{1}{4} & 0 & \frac{1}{2} \\ \frac{2}{5} & \frac{1}{5} & 0 \end{pmatrix}, \qquad \widehat{\mathbf{c}} = \begin{pmatrix} 1 \\ \frac{1}{4} \\ \frac{2}{5} \end{pmatrix}.$$

The corresponding iteration $\mathbf{u}^{(k+1)} = \widehat{T}\mathbf{u}^{(k)} + \widehat{\mathbf{c}}$ takes the explicit form

$$\begin{aligned} x^{(k+1)} &= -\tfrac{1}{3}\,y^{(k)} + \tfrac{1}{3}\,z^{(k)} + 1, \\ y^{(k+1)} &= \;\; \tfrac{1}{4}\,x^{(k)} + \tfrac{1}{2}\,z^{(k)} + \tfrac{1}{4}, \\ z^{(k+1)} &= \;\; \tfrac{2}{5}\,x^{(k)} + \tfrac{1}{5}\,y^{(k)} + \tfrac{2}{5}. \end{aligned} \tag{7.39}$$

Do the resulting iterative schemes converge to the solution $x = y = z = 1$? The results, starting with initial guess $\mathbf{u}^{(0)} = (0,0,0)$, are tabulated as follows.

k	$\mathbf{u}^{(k+1)} = T\,\mathbf{u}^{(k)} + \mathbf{b}$			$\mathbf{u}^{(k+1)} = \widehat{T}\,\mathbf{u}^{(k)} + \widehat{\mathbf{c}}$		
0	0	0	0	0	0	0
1	3	-1	2	1	.25	.4
2	0	-13	-1	1.05	.7	.85
3	15	-64	-7	1.05	.9375	.96
4	30	-322	-4	1.0075	.9925	1.0075
5	261	-1633	-244	1.005	1.00562	1.0015
6	870	-7939	-133	.9986	1.002	1.0031
7	6069	-40300	-5665	1.0004	1.0012	.9999
8	22500	-196240	-5500	.9995	1.0000	1.0004
9	145743	-992701	-129238	1.0001	1.0001	.9998
10	571980	-4850773	-184261	.9999	.9999	1.0001
11	3522555	-24457324	-2969767	1.0000	1.0000	1.0000

For the first scheme, the answer is clearly no — the iterates become wilder and wilder. Indeed, this occurs no matter how close the initial guess $\mathbf{u}^{(0)}$ is to the actual solution — unless $\mathbf{u}^{(0)} = \mathbf{u}^\star$ happens to be exactly equal. In the second case, the iterates do converge to the solution, and it does not take too long, even starting from a poor initial guess, to obtain a reasonably accurate approximation. Of course, in such a simple example, it would be silly to use iteration, when Gaussian Elimination can be done by hand and produces the solution almost immediately. However, we use the small examples for illustrative purposes, bringing the full power of iterative schemes to bear on the large linear systems arising in applications.

The convergence of solutions to (7.35) to the fixed point $\mathbf{u}^\star$ is based on the behavior of the *error vectors*

$$\mathbf{e}^{(k)} = \mathbf{u}^{(k)} - \mathbf{u}^\star, \tag{7.40}$$

which measure how close the iterates are to the true solution. Let us find out how the successive error vectors are related. We compute

$$\mathbf{e}^{(k+1)} = \mathbf{u}^{(k+1)} - \mathbf{u}^\star = (T\,\mathbf{u}^{(k)} + \mathbf{a}) - (T\,\mathbf{u}^\star + \mathbf{a}) = T(\mathbf{u}^{(k)} - \mathbf{u}^\star) = T\,\mathbf{e}^{(k)},$$

showing that the error vectors satisfy a *linear* iterative system

$$\mathbf{e}^{(k+1)} = T\,\mathbf{e}^{(k)}, \tag{7.41}$$

with the *same* coefficient matrix T. Therefore, they are given by the explicit formula

$$\mathbf{e}^{(k)} = T^k\,\mathbf{e}^{(0)}.$$

Now, the solutions to (7.35) converge to the fixed point, $\mathbf{u}^{(k)} \to \mathbf{u}^\star$, if and only if the error vectors converge to zero: $\mathbf{e}^{(k)} \to \mathbf{0}$ as $k \to \infty$. Our analysis of linear iterative systems, as summarized in Proposition 7.8, establishes the following basic convergence result.

Proposition 7.25. *The affine iterative system* (7.35) *will converge to the solution to the fixed point equation* (7.36) *if and only if* T *is a convergent matrix:* $\rho(T) < 1$.

The spectral radius $\rho(T)$ of the coefficient matrix will govern the speed of convergence. Therefore, our main goal is to construct an iterative scheme whose coefficient matrix has as small a spectral radius as possible. At the very least, the spectral radius must be less than 1. For the two iterative schemes presented in Example 7.24, the spectral radii of the coefficient matrices are found to be

$$\rho(T) \approx 4.9675, \qquad\qquad \rho(\,\widehat{T}\,) = .5.$$

Therefore, T is not a convergent matrix, which explains the wild behavior of its iterates, whereas $\widehat{T}$ is convergent, and one expects the error to roughly decrease by a factor of $\frac{1}{2}$ at each step.

The Jacobi Method

The first general iterative scheme for solving linear systems is based on the same simple idea used in our illustrative Example 7.24. Namely, we solve the i^{th} equation in the system $A\mathbf{u} = \mathbf{b}$, which is

$$\sum_{j=1}^{n} a_{ij}\,u_j = b_i,$$

for the i^{th} variable u_i. To do this, we need to assume that all the diagonal entries of A are nonzero: $a_{ii} \neq 0$. The result is

$$u_i \;=\; -\,\frac{1}{a_{ii}} \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}\,u_j + \frac{b_i}{a_{ii}} \;=\; \sum_{j=1}^{n} t_{ij}\,u_j + c_i, \tag{7.42}$$

where

$$t_{ij} = \begin{cases} -\,\dfrac{a_{ij}}{a_{ii}}\,, & i \neq j, \\ 0, & i = j, \end{cases} \qquad\text{and}\qquad c_i = \frac{b_i}{a_{ii}}\,. \tag{7.43}$$

The result has the form of a fixed-point system $\mathbf{u} = T\,\mathbf{u} + \mathbf{c}$, and forms the basis of the *Jacobi method*

$$\mathbf{u}^{(k+1)} = T\,\mathbf{u}^{(k)} + \mathbf{c}, \qquad\qquad \mathbf{u}^{(0)} = \mathbf{u}_0, \tag{7.44}$$

named after the influential nineteenth century German analyst Carl Jacobi. The explicit form of the Jacobi iterative scheme is

$$u_i^{(k+1)} \;=\; -\,\frac{1}{a_{ii}} \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}\,u_j^{(k)} + \frac{b_i}{a_{ii}}\,. \tag{7.45}$$

It is instructive to rederive the Jacobi method in a direct matrix form. We begin by decomposing the coefficient matrix

$$A = L + D + U \tag{7.46}$$

into the sum of a strictly lower triangular matrix L, a diagonal matrix D, and a strictly upper triangular matrix U, each of which is uniquely specified. For example, when

$$A = \begin{pmatrix} 3 & 1 & -1 \\ 1 & -4 & 2 \\ -2 & -1 & 5 \end{pmatrix}, \tag{7.47}$$

the decomposition (7.46) yields

$$L = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ -2 & -1 & 0 \end{pmatrix}, \qquad D = \begin{pmatrix} 3 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & 5 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & 1 & -1 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix}.$$

Warning: The L, D, U in the elementary additive decomposition (7.46) have nothing to do with the L, D, U appearing in factorizations arising from Gaussian Elimination. The latter play no role in the iterative solution methods considered here.

We then rewrite the system

$$A\mathbf{u} = (L + D + U)\,\mathbf{u} = \mathbf{b} \qquad \text{in the alternative form} \qquad D\,\mathbf{u} = -\,(L + U)\,\mathbf{u} + \mathbf{b}.$$

The Jacobi fixed point equations (7.42) amounts to solving for

$$\mathbf{u} = T\,\mathbf{u} + \mathbf{c}, \qquad \text{where} \qquad T = -\,D^{-1}(L + U), \qquad \mathbf{c} = D^{-1}\mathbf{b}. \tag{7.48}$$

For the example (7.47), we recover the Jacobi iteration matrix as

$$T = -\,D^{-1}(L + U) = \begin{pmatrix} 0 & -\frac{1}{3} & \frac{1}{3} \\ \frac{1}{4} & 0 & \frac{1}{2} \\ \frac{2}{5} & \frac{1}{5} & 0 \end{pmatrix}.$$

Deciding in advance whether or not the Jacobi method will converge is not easy. However, it can be shown that Jacobi iteration *is* guaranteed to converge when the original coefficient matrix has large diagonal entries, in accordance with Definition 6.25.

Theorem 7.26. *If A is strictly diagonally dominant, then the associated Jacobi iteration scheme converges.*

Proof: We shall prove that $\|\,T\,\|_\infty < 1$, and so Corollary 7.19 implies that T is a convergent matrix. The absolute row sums of the Jacobi matrix $T = -\,D^{-1}(L + U)$ are, according to (7.43),

$$s_i = \sum_{j=1}^{n} |\,t_{ij}\,| = \frac{1}{|\,a_{ii}\,|} \sum_{\substack{j=1 \\ j \neq i}}^{n} |\,a_{ij}\,| < 1, \tag{7.49}$$

because A is strictly diagonally dominant. Thus, $\|\,T\,\|_\infty = \max\{s_1, \ldots, s_n\} < 1$, and the result follows. *Q.E.D.*

Example 7.27. Consider the linear system

$$\begin{aligned} 4x + y + w &= 1, \\ x + 4y + z + v &= 2, \\ y + 4z + w &= -1, \\ x + z + 4w + v &= 2, \\ y + w + 4v &= 1. \end{aligned}$$

The Jacobi method solves the respective equations for x, y, z, w, v, leading to the iterative scheme

$$\begin{aligned} x^{(k+1)} &= -\tfrac{1}{4}\,y^{(k)} - \tfrac{1}{4}\,w^{(k)} + \tfrac{1}{4}\,, \\ y^{(k+1)} &= -\tfrac{1}{4}\,x^{(k)} - \tfrac{1}{4}\,z^{(k)} - \tfrac{1}{4}\,v^{(k)} + \tfrac{1}{2}\,, \\ z^{(k+1)} &= -\tfrac{1}{4}\,y^{(k)} - \tfrac{1}{4}\,w^{(k)} - \tfrac{1}{4}\,, \\ w^{(k+1)} &= -\tfrac{1}{4}\,x^{(k)} - \tfrac{1}{4}\,z^{(k)} - \tfrac{1}{4}\,v^{(k)} + \tfrac{1}{2}\,, \\ v^{(k+1)} &= -\tfrac{1}{4}\,y^{(k)} - \tfrac{1}{4}\,w^{(k)} + \tfrac{1}{4}\,. \end{aligned}$$

The coefficient matrix of the original system,

$$A = \begin{pmatrix} 4 & 1 & 0 & 1 & 0 \\ 1 & 4 & 1 & 0 & 1 \\ 0 & 1 & 4 & 1 & 0 \\ 1 & 0 & 1 & 4 & 1 \\ 0 & 1 & 0 & 1 & 4 \end{pmatrix},$$

is diagonally dominant, and so we are guaranteed that the Jacobi iterations will eventually converge to the solution. Indeed, the Jacobi scheme takes the iterative form (7.48), with

$$T = \begin{pmatrix} 0 & -\frac{1}{4} & 0 & -\frac{1}{4} & 0 \\ -\frac{1}{4} & 0 & -\frac{1}{4} & 0 & -\frac{1}{4} \\ 0 & -\frac{1}{4} & 0 & -\frac{1}{4} & 0 \\ -\frac{1}{4} & 0 & -\frac{1}{4} & 0 & -\frac{1}{4} \\ 0 & -\frac{1}{4} & 0 & -\frac{1}{4} & 0 \end{pmatrix}, \qquad \mathbf{c} = \begin{pmatrix} \frac{1}{4} \\ \frac{1}{2} \\ -\frac{1}{4} \\ \frac{1}{2} \\ \frac{1}{4} \end{pmatrix}.$$

Note that $\| T \|_\infty = \frac{3}{4} < 1$, validating convergence of the scheme. Thus, to obtain, say, four decimal place accuracy in the solution, we estimate that it would take less than $\log(.5 \times 10^{-4}) / \log .75 \approx 34$ iterates, assuming a moderate initial error. But the matrix norm always underestimates the true rate of convergence, as prescribed by the spectral radius $\rho(T) = .6124$, which would imply about $\log(.5 \times 10^{-4}) / \log .6124 \approx 20$ iterations to obtain the desired accuracy. Indeed, starting with the initial guess $x^{(0)} = y^{(0)} = z^{(0)} = w^{(0)} = v^{(0)} = 0$, the Jacobi iterates converge to the exact solution

$$x = -.1, \qquad y = .7, \qquad z = -.6, \qquad w = .7, \qquad v = -.1,$$

to within four decimal places in exactly 20 iterations.

The Gauss–Seidel Method

The Gauss–Seidel method relies on a slightly more refined implementation of the Jacobi process. To understand how it works, it will help to write out the Jacobi iteration scheme (7.44) in full detail:

$$\begin{array}{rlllllll}
u_1^{(k+1)} = & & t_{12}\,u_2^{(k)} & + t_{13}\,u_3^{(k)} + & \cdots & + t_{1,n-1}\,u_{n-1}^{(k)} & + t_{1n}\,u_n^{(k)} & + c_1, \\
u_2^{(k+1)} = & t_{21}\,u_1^{(k)} & & + t_{23}\,u_3^{(k)} + & \cdots & + t_{2,n-1}\,u_{n-1}^{(k)} & + t_{2n}\,u_n^{(k)} & + c_2, \\
u_3^{(k+1)} = & t_{31}\,u_1^{(k)} & + t_{32}\,u_2^{(k)} & & \cdots & + t_{3,n-1}\,u_{n-1}^{(k)} & + t_{3n}\,u_n^{(k)} & + c_3, \\
\vdots \quad & \vdots & \vdots & \ddots & & & \ddots & \vdots \\
u_n^{(k+1)} = & t_{n1}\,u_1^{(k)} & + t_{n2}\,u_2^{(k)} & + t_{n3}\,u_3^{(k)} + & \cdots & + t_{n,n-1}\,u_{n-1}^{(k)} & & + c_n,
\end{array} \tag{7.50}$$

where we are explicitly noting the fact that the diagonal entries of T vanish. Observe that we are using the entries of $\mathbf{u}^{(k)}$ to compute *all* of the updated values of $\mathbf{u}^{(k+1)}$. Presumably, if the iterates $\mathbf{u}^{(k)}$ are converging to the solution $\mathbf{u}^\star$, then their individual entries are also converging, and so each $u_j^{(k+1)}$ should be a better approximation to $u_j^\star$ than $u_j^{(k)}$ is. Therefore, if we begin the k^{th} Jacobi iteration by computing $u_1^{(k+1)}$ using the first equation, then we are tempted to use this new and improved value to replace $u_1^{(k)}$ in each of the subsequent equations. In particular, we employ the modified equation

$$u_2^{(k+1)} = t_{21}\,u_1^{(k+1)} + t_{23}\,u_3^{(k)} + \;\cdots\; + t_{1n}\,u_n^{(k)} + c_2$$

to update the second component of our iterate. This more accurate value should then be used to update $u_3^{(k+1)}$, and so on.

The upshot of these considerations is the *Gauss–Seidel method*

$$u_i^{(k+1)} = t_{i1}\,u_1^{(k+1)} + \;\cdots\; + t_{i,i-1}\,u_{i-1}^{(k+1)} + t_{i,i+1}\,u_{i+1}^{(k)} + \;\cdots\; + t_{in}\,u_n^{(k)} + c_i, \quad i = 1, \ldots, n, \tag{7.51}$$

named after Gauss (as usual!) and the German astronomer/mathematician Philipp von Seidel. At the k^{th} stage of the iteration, we use (7.51) to compute the revised entries $u_1^{(k+1)}, u_2^{(k+1)}, \ldots, u_n^{(k+1)}$ in their numerical order. Once an entry has been updated, the new value is immediately used in all subsequent computations.

Example 7.28. For the linear system

$$3x + y - z = 3, \qquad x - 4y + 2z = -1, \qquad -2x - y + 5z = 2,$$

the Jacobi iteration method was given in (7.39). To construct the corresponding Gauss–Seidel scheme we use updated values of x, y and z as they become available. Explicitly,

$$\begin{aligned}
x^{(k+1)} &= -\tfrac{1}{3}\,y^{(k)} + \tfrac{1}{3}\,z^{(k)} + 1, \\
y^{(k+1)} &= \tfrac{1}{4}\,x^{(k+1)} + \tfrac{1}{2}\,z^{(k)} + \tfrac{1}{4}, \\
z^{(k+1)} &= \tfrac{2}{5}\,x^{(k+1)} + \tfrac{1}{5}\,y^{(k+1)} + \tfrac{2}{5}.
\end{aligned} \tag{7.52}$$

The resulting iterates starting with $\mathbf{u}^{(0)} = \mathbf{0}$ are

$$\mathbf{u}^{(1)} = \begin{pmatrix} 1.0000 \\ .5000 \\ .9000 \end{pmatrix}, \qquad \mathbf{u}^{(2)} = \begin{pmatrix} 1.1333 \\ .9833 \\ 1.0500 \end{pmatrix}, \qquad \mathbf{u}^{(3)} = \begin{pmatrix} 1.0222 \\ 1.0306 \\ 1.0150 \end{pmatrix}, \qquad \mathbf{u}^{(4)} = \begin{pmatrix} .9948 \\ 1.0062 \\ .9992 \end{pmatrix},$$

$$\mathbf{u}^{(5)} = \begin{pmatrix} .9977 \\ .9990 \\ .9989 \end{pmatrix}, \qquad \mathbf{u}^{(6)} = \begin{pmatrix} 1.0000 \\ .9994 \\ .9999 \end{pmatrix}, \qquad \mathbf{u}^{(7)} = \begin{pmatrix} 1.0001 \\ 1.0000 \\ 1.0001 \end{pmatrix}, \qquad \mathbf{u}^{(8)} = \begin{pmatrix} 1.0000 \\ 1.0000 \\ 1.0000 \end{pmatrix},$$

and have converged to the solution, to 4 decimal place accuracy, after only 8 iterations — as opposed to the 11 iterations required by the Jacobi method.

The Gauss–Seidel iteration scheme is particularly suited to implementation on a serial computer, since one can immediately replace each component $u_i^{(k)}$ by its updated value $u_i^{(k+1)}$, thereby also saving on storage in the computer's memory. In contrast, the Jacobi scheme requires us to retain all the old values $\mathbf{u}^{(k)}$ until the new approximation $\mathbf{u}^{(k+1)}$ has been computed. Moreover, Gauss–Seidel typically (although not always) converges faster than Jacobi, making it the iterative algorithm of choice for serial processors. On the other hand, with the advent of parallel processing machines, variants of the parallelizable Jacobi scheme have recently been making a comeback.

What is Gauss–Seidel really up to? Let us rewrite the basic iterative equation (7.51) by multiplying by a_{ii} and moving the terms involving $\mathbf{u}^{(k+1)}$ to the left hand side. In view of the formula (7.43) for the entries of T, the resulting equation is

$$a_{i1}\,u_1^{(k+1)} + \;\cdots\; + a_{i,i-1}\,u_{i-1}^{(k+1)} + a_{ii}\,u_i^{(k+1)} = -\,a_{i,i+1}\,u_{i+1}^{(k)} - \;\cdots\; - a_{in}\,u_n^{(k)} + b_i.$$

In matrix form, taking (7.46) into account, this reads

$$(L + D)\mathbf{u}^{(k+1)} = -\,U\,\mathbf{u}^{(k)} + \mathbf{b}, \tag{7.53}$$

and so can be viewed as a linear system of equations for $\mathbf{u}^{(k+1)}$ with lower triangular coefficient matrix $L + D$. Note that the fixed point of (7.53), namely the solution to

$$(L + D)\,\mathbf{u} = -\,U\,\mathbf{u} + \mathbf{b},$$

coincides with the solution to the original system

$$A\,\mathbf{u} = (L + D + U)\,\mathbf{u} = \mathbf{b}.$$

In other words, the Gauss–Seidel procedure is merely implementing Forward Substitution to solve the lower triangular system (7.53) for the next iterate:

$$\mathbf{u}^{(k+1)} = -\,(L + D)^{-1}U\,\mathbf{u}^{(k)} + (L + D)^{-1}\,\mathbf{b}.$$

The latter is in our more usual iterative form

$$\mathbf{u}^{(k+1)} = \widetilde{T}\,\mathbf{u}^{(k)} + \widetilde{\mathbf{c}}, \qquad \text{where} \qquad \widetilde{T} = -\,(L + D)^{-1}U, \qquad \widetilde{\mathbf{c}} = (L + D)^{-1}\,\mathbf{b}. \tag{7.54}$$

Consequently, the convergence of the Gauss–Seidel iterates is governed by the spectral radius of the coefficient matrix $\widetilde{T}$.

Returning to Example 7.28, we have

$$A = \begin{pmatrix} 3 & 1 & -1 \\ 1 & -4 & 2 \\ -2 & -1 & 5 \end{pmatrix}, \qquad L + D = \begin{pmatrix} 3 & 0 & 0 \\ 1 & -4 & 0 \\ -2 & -1 & 5 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & 1 & -1 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix}.$$

Therefore, the Gauss–Seidel matrix is

$$\widetilde{T} = -(L + D)^{-1} U = \begin{pmatrix} 0 & -.3333 & .3333 \\ 0 & -.0833 & .5833 \\ 0 & -.1500 & .2500 \end{pmatrix}.$$

Its eigenvalues are 0 and $.0833 \pm .2444\,\mathrm{i}$, and hence its spectral radius is $\rho(\widetilde{T}) \approx .2582$. This is roughly the square of the Jacobi spectral radius of .5, which tell us that the Gauss–Seidel iterations will converge about twice as fast to the solution. This can be verified by more extensive computations. Although examples can be constructed where the Jacobi method converges faster, in many practical situations Gauss–Seidel tends to converge roughly twice as fast as Jacobi.

Completely general conditions guaranteeing convergence of the Gauss–Seidel method are hard to establish. But, like the Jacobi scheme, it is guaranteed to converge when the original coefficient matrix is strictly diagonally dominant.

Theorem 7.29. *If A is strictly diagonally dominant, then the Gauss–Seidel iteration scheme for solving $A\mathbf{u} = \mathbf{b}$ converges.*

Proof: Let $\mathbf{e}^{(k)} = \mathbf{u}^{(k)} - \mathbf{u}^\star$ denote the k^{th} Gauss–Seidel error vector. As in (7.41), the error vectors satisfy the linear iterative system $\mathbf{e}^{(k+1)} = \widetilde{T}\,\mathbf{e}^{(k)}$, but a direct estimate of $\| \widetilde{T} \|_\infty$ is not so easy. Instead, let us write out the linear iterative system in components:

$$e_i^{(k+1)} = t_{i1}\, e_1^{(k+1)} + \ \cdots\ + t_{i,i-1}\, e_{i-1}^{(k+1)} + t_{i,i+1}\, e_{i+1}^{(k)} + \ \cdots\ + t_{in}\, e_n^{(k)}. \tag{7.55}$$

Let

$$m^{(k)} = \| \mathbf{e}^{(k)} \|_\infty = \max\bigl\{ \,| e_1^{(k)} |,\ \ldots\ , | e_n^{(k)} |\, \bigr\} \tag{7.56}$$

denote the ∞ norm of the k^{th} error vector. To prove convergence, $\mathbf{e}^{(k)} \to \mathbf{0}$, it suffices to show that $m^{(k)} \to 0$ as $k \to \infty$. We claim that diagonal dominance of A implies that

$$m^{(k+1)} \leq s\, m^{(k)}, \qquad \text{where} \qquad s = \| T \|_\infty < 1 \tag{7.57}$$

denotes the ∞ matrix norm of the *Jacobi* matrix (not the Gauss–Seidel matrix), which, by (7.49), is less than 1. We infer that $m^{(k)} \leq s^k\, m^{(0)} \to 0$ as $k \to \infty$, demonstrating the theorem.

To prove (7.57), we use induction on $i = 1, \ldots, n$. Our induction hypothesis is

$$| e_j^{(k+1)} | \leq s\, m^{(k)} < m^{(k)} \qquad \text{for} \qquad j = 1, \ldots, i-1.$$

(When $i = 1$, there is no assumption.) Moreover, by (7.56),

$$| e_j^{(k)} | \leq m^{(k)} \qquad \text{for all} \qquad j = 1, \ldots, n.$$

We use these two inequalities to estimate $|\,e_i^{(k+1)}\,|$ from (7.55):

$$\begin{aligned}|\,e_i^{(k+1)}\,| &\le |\,t_{i1}\,|\,|\,e_1^{(k+1)}\,| + \;\cdots\; + |\,t_{i,i-1}\,|\,|\,e_{i-1}^{(k+1)}\,| + |\,t_{i,i+1}\,|\,|\,e_{i+1}^{(k)}\,| + \;\cdots\; + |\,t_{in}\,|\,|\,e_n^{(k)}\,| \\ &\le \big(\,|\,t_{i1}\,| + \;\cdots\; + |\,t_{in}\,|\,\big)\, m^{(k)} \le s\, m^{(k)},\end{aligned}$$

which completes the induction step. As a result, the maximum

$$m^{(k+1)} = \max\big\{\,|\,e_1^{(k+1)}\,|,\;\ldots\;,|\,e_n^{(k+1)}\,|\,\big\} \le s\, m^{(k)}$$

also satisfies the same bound, and hence (7.57) follows. *Q.E.D.*

Example 7.30. For the linear system considered in Example 7.27, the Gauss–Seidel iterations take the form

$$\begin{aligned}x^{(k+1)} &= -\tfrac14\, y^{(k)} - \tfrac14\, w^{(k)} + \tfrac14\,, \\ y^{(k+1)} &= -\tfrac14\, x^{(k+1)} - \tfrac14\, z^{(k)} - \tfrac14\, v^{(k)} + \tfrac12\,, \\ z^{(k+1)} &= -\tfrac14\, y^{(k+1)} - \tfrac14\, w^{(k)} - \tfrac14\,, \\ w^{(k+1)} &= -\tfrac14\, x^{(k+1)} - \tfrac14\, z^{(k+1)} - \tfrac14\, v^{(k)} + \tfrac12\,, \\ v^{(k+1)} &= -\tfrac14\, y^{(k+1)} - \tfrac14\, w^{(k+1)} + \tfrac14\,.\end{aligned}$$

Starting with $x^{(0)} = y^{(0)} = z^{(0)} = w^{(0)} = v^{(0)} = 0$, the Gauss–Seidel iterates converge to the solution $x = -.1, y = .7, z = -.6, w = .7, v = -.1$, to four decimal places in 11 iterations, again roughly twice as fast as the Jacobi scheme. Indeed, the convergence rate is governed by the corresponding Gauss–Seidel matrix $\widetilde{T}$, which is

$$\begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 & 0 \\ 0 & 1 & 4 & 0 & 0 \\ 1 & 0 & 1 & 4 & 0 \\ 0 & 1 & 0 & 1 & 4 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -.2500 & 0 & -.2500 & 0 \\ 0 & .0625 & -.2500 & .0625 & -.2500 \\ 0 & -.0156 & .0625 & -.2656 & .0625 \\ 0 & .0664 & -.0156 & .1289 & -.2656 \\ 0 & -.0322 & .0664 & -.0479 & .1289 \end{pmatrix}.$$

Its spectral radius is $\rho(\widetilde{T}) = .3936$, which is, as in the previous example, approximately the square of the spectral radius of the Jacobi coefficient matrix, which explains the speed up in convergence.

Successive Over–Relaxation (*SOR*)

As we know, the smaller the spectral radius (or matrix norm) of the coefficient matrix, the faster the convergence of the iterative scheme. One of the goals of researchers in numerical linear algebra is to design new methods for accelerating the convergence. In his 1950 thesis, the American mathematician David Young discovered a simple modification of the Jacobi and Gauss–Seidel methods that can, in favorable situations, lead to a dramatic speed up in the rate of convergence. The method, known as *successive over-relaxation*, and often abbreviated as SOR, has become the iterative method of choice in many modern applications, [**13**, **47**]. In this subsection, we provide a brief overview.

In practice, finding the optimal iterative algorithm to solve a given linear system is as hard as solving the system itself. Therefore, researchers have relied on a few tried and true techniques for designing iterative schemes that can be used in the more common applications. Consider a linear algebraic system

$$A\mathbf{u} = \mathbf{b}.$$

Every decomposition

$$A = M - N \tag{7.58}$$

of the coefficient matrix into the difference of two matrices leads to an equivalent system of the form

$$M\,\mathbf{u} = N\,\mathbf{u} + \mathbf{b}. \tag{7.59}$$

Provided that M is nonsingular, we can rewrite the system in the fixed point form

$$\mathbf{u} = M^{-1}N\,\mathbf{u} + M^{-1}\mathbf{b} = T\,\mathbf{u} + \mathbf{c}, \qquad \text{where} \qquad T = M^{-1}N, \quad \mathbf{c} = M^{-1}\mathbf{b}.$$

Now, we are free to choose any such M, which then specifies $N = A - M$ uniquely. However, for the resulting iterative scheme $\mathbf{u}^{(k+1)} = T\,\mathbf{u}^{(k)} + \mathbf{c}$ to be practical we must arrange that

(*a*) $T = M^{-1}N$ is a convergent matrix, and

(*b*) M can be easily inverted.

The second requirement ensures that the iterative equations

$$M\,\mathbf{u}^{(k+1)} = N\,\mathbf{u}^{(k)} + \mathbf{b} \tag{7.60}$$

can be solved for $\mathbf{u}^{(k+1)}$ with minimal computational effort. Typically, this requires that M be either a diagonal matrix, in which case the inversion is immediate, or upper or lower triangular, in which case one employs Back or Forward Substitution to solve for $\mathbf{u}^{(k+1)}$.

With this in mind, we now introduce the SOR method. It relies on a slight generalization of the Gauss–Seidel decomposition (7.53) of the matrix into lower plus diagonal and upper triangular parts. The starting point is to write

$$A = L + D + U = \big[\,L + \alpha\,D\,\big] - \big[\,(\alpha - 1)\,D - U\,\big], \tag{7.61}$$

where $0 \neq \alpha$ is an adjustable scalar parameter. We decompose the system $A\mathbf{u} = \mathbf{b}$ as

$$(L + \alpha\,D)\mathbf{u} = \big[\,(\alpha - 1)\,D - U\,\big]\mathbf{u} + \mathbf{b}. \tag{7.62}$$

It turns out to be slightly more convenient to divide (7.62) through by α and write the resulting iterative system in the form

$$(\omega\,L + D)\mathbf{u}^{(k+1)} = \big[\,(1 - \omega)\,D - \omega\,U\,\big]\mathbf{u}^{(k)} + \omega\,\mathbf{b}, \tag{7.63}$$

where $\omega = 1/\alpha$ is called the *relaxation parameter*. Assuming, as usual, that all diagonal entries of A are nonzero, the matrix $\omega\,L + D$ is an invertible lower triangular matrix, and so we can use Forward Substitution to solve the iterative system (7.63) to recover $\mathbf{u}^{(k+1)}$. The explicit formula for its i^{th} entry is

$$\begin{aligned} u_i^{(k+1)} = \omega\,t_{i1}\,u_1^{(k+1)} + \;\cdots\; + \omega\,t_{i,i-1}\,u_{i-1}^{(k+1)} + (1-\omega)\,u_i^{(k)} + {}& \\ + \omega\,t_{i,i+1}\,u_{i+1}^{(k)} + \;\cdots\; + \omega\,t_{in}\,u_n^{(k)} + \omega\,c_i,& \end{aligned} \tag{7.64}$$

where t_{ij} and c_i denote the original Jacobi values (7.43). As in the Gauss–Seidel approach, we update the entries $u_i^{(k+1)}$ in numerical order $i = 1, \ldots, n$. Thus, to obtain the SOR scheme (7.64), we merely multiply the right hand side of the Gauss–Seidel scheme (7.51) by the adjustable relaxation parameter ω and append the diagonal term $(1-\omega)\, u_i^{(k)}$. In particular, if we set $\omega = 1$, then the SOR method reduces to the Gauss–Seidel method. Choosing $\omega < 1$ leads to an *under-relaxed* method, while $\omega > 1$, known as *over-relaxation*, is the choice that works in most practical instances.

To analyze the SOR scheme in detail, we rewrite (7.63) in the fixed point form

$$\mathbf{u}^{(k+1)} = T_\omega\, \mathbf{u}^{(k)} + \mathbf{c}_\omega, \tag{7.65}$$

where

$$T_\omega = (\omega\, L + D)^{-1} \big[\, (1-\omega)\, D - \omega\, U\, \big], \qquad \mathbf{c}_\omega = (\omega\, L + D)^{-1}\, \omega\, \mathbf{b}. \tag{7.66}$$

The rate of convergence is governed by the spectral radius of the matrix T_ω. The goal is to choose the relaxation parameter ω so as to make the spectral radius of T_ω as small as possible. As we will see, a clever choice of ω can result in a dramatic speed up in the convergence rate. Let us look at an elementary example.

Example 7.31. Consider the matrix $A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$, which we decompose as $A = L + D + U$, where

$$L = \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix}, \qquad D = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix}.$$

Jacobi iteration is based on the coefficient matrix $T = -D^{-1}(L+U) = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix}$. Its spectral radius is $\rho(T) = .5$, and hence the Jacobi scheme takes, on average, roughly $3.3 \approx -1/\log_{10} .5$ iterations to produce each new decimal place in the solution.

The SOR scheme (7.63) takes the explicit form

$$\begin{pmatrix} 2 & 0 \\ -\omega & 2 \end{pmatrix} \mathbf{u}^{(k+1)} = \begin{pmatrix} 2(1-\omega) & \omega \\ 0 & 2\,(1-\omega) \end{pmatrix} \mathbf{u}^{(k)} + \omega\, \mathbf{b},$$

where Gauss–Seidel is the particular case $\omega = 1$. The SOR coefficient matrix is

$$T_\omega = \begin{pmatrix} 2 & 0 \\ -\omega & 2 \end{pmatrix}^{-1} \begin{pmatrix} 2\,(1-\omega) & \omega \\ 0 & 2\,(1-\omega) \end{pmatrix} = \begin{pmatrix} 1-\omega & \frac{1}{2}\,\omega \\ \frac{1}{2}\,\omega(1-\omega) & \frac{1}{4}\,(2-\omega)^2 \end{pmatrix}.$$

To compute the eigenvalues of T_ω, we form its characteristic equation

$$0 = \det(T_\omega - \lambda\, \mathrm{I}\,) = \lambda^2 - \big(\, 2 - 2\omega + \tfrac{1}{4}\,\omega^2 \,\big)\lambda + (1-\omega)^2 = (\lambda + \omega - 1)^2 - \tfrac{1}{4}\,\lambda\,\omega^2. \tag{7.67}$$

Our goal is to choose ω so that

(a) both eigenvalues are less than 1 in modulus, so $|\,\lambda_1\,|, |\,\lambda_2\,| < 1$. This is the minimal requirement for convergence of the method.

(b) the largest eigenvalue (in modulus) is as small as possible. This will give the smallest spectral radius for T_ω and hence the fastest convergence rate.

The product of the two eigenvalues is the determinant,

$$\lambda_1 \lambda_2 = \det T_\omega = (1-\omega)^2.$$

If $\omega \leq 0$ or $\omega \geq 2$, then $\det T_\omega \geq 1$, and hence at least one of the eigenvalues would have modulus larger than 1. Thus, in order to ensure convergence, we must require $0 < \omega < 2$. For Gauss–Seidel, at $\omega = 1$, the eigenvalues are $\lambda_1 = \frac{1}{4}$, $\lambda_2 = 0$, and the spectral radius is $\rho(T_1) = .25$. This is exactly the square of the Jacobi spectral radius, and hence the Gauss–Seidel iterates converge twice as fast; so it only takes, on average, about $-1/\log_{10} .25 = 1.66$ Gauss–Seidel iterations to produce each new decimal place of accuracy. It can be shown that as ω increases above 1, the two eigenvalues move along the real axis towards each other. They coincide when

$$\omega = \omega_\star = 8 - 4\sqrt{3} \;\approx\; 1.07, \qquad \text{at which point} \qquad \lambda_1 = \lambda_2 = \omega_\star - 1 = .07 = \rho(T_\omega),$$

which is the convergence rate of the optimal SOR scheme. Each iteration produces slightly more than one new decimal place in the solution, which represents a significant improvement over the Gauss–Seidel convergence rate. It takes about twice as many Gauss–Seidel iterations (and four times as many Jacobi iterations) to produce the same accuracy as this optimal SOR method.

Of course, in such a simple 2×2 example, it is not so surprising that we can construct the best value for the relaxation parameter by hand. Young was able to find the optimal value of the relaxation parameter for a broad class of matrices that includes most of those arising in the finite difference and finite element numerical solutions to ordinary and partial differential equations. For the matrices in Young's class, the Jacobi eigenvalues occur in signed pairs. If $\pm\mu$ are a pair of eigenvalues for the Jacobi method, then the corresponding eigenvalues of the SOR iteration matrix satisfy the quadratic equation

$$(\lambda + \omega - 1)^2 = \lambda\, \omega^2 \mu^2. \tag{7.68}$$

If $\omega = 1$, so we have standard Gauss–Seidel, then $\lambda^2 = \lambda\mu^2$, and so the eigenvalues are $\lambda = 0$, $\lambda = \mu^2$. The Gauss–Seidel spectral radius is therefore the square of the Jacobi spectral radius, and so (at least for matrices in the Young class) its iterates converge twice as fast. The quadratic equation (7.68) has the same properties as in the 2×2 version (7.67) (which corresponds to the case $\mu = \frac{1}{2}$), and hence the optimal value of ω will be the one at which the two roots are equal:

$$\lambda_1 = \lambda_2 = \omega - 1, \qquad \text{which occurs when} \qquad \omega = \frac{2 - 2\sqrt{1-\mu^2}}{\mu^2} = \frac{2}{1 + \sqrt{1-\mu^2}}\,.$$

Therefore, if $\rho_J = \max|\,\mu\,|$ denotes the spectral radius of the Jacobi method, then the Gauss–Seidel has spectral radius $\rho_{GS} = \rho_J^2$, while the SOR method with optimal relaxation parameter

$$\omega_\star = \frac{2}{1 + \sqrt{1-\rho_J^2}}, \qquad \text{has spectral radius} \qquad \rho_\star = \omega_\star - 1. \tag{7.69}$$

For example, if $\rho_J = .99$, which is rather slow convergence (but common for iterative numerical solution schemes for partial differential equations), then $\rho_{GS} = .9801$, which is twice as fast, but still quite slow, while SOR with $\omega_\star = 1.7527$ has $\rho_\star = .7527$, which is dramatically faster[†]. Indeed, since $\rho_\star \approx (\rho_{GS})^{14} \approx (\rho_J)^{28}$, it takes about 14 Gauss–Seidel (and 28 Jacobi) iterations to produce the same accuracy as one SOR step. It is amazing that such a simple idea can have such a dramatic effect.

[†] More precisely, since the SOR matrix is not diagonalizable, the overall convergence rate is slightly slower than the spectral radius. However, this technical detail does not affect the overall conclusion.

AIMS Lecture Notes 2006

Peter J. Olver

8. Numerical Computation of Eigenvalues

In this part, we discuss some practical methods for computing eigenvalues and eigenvectors of matrices. Needless to say, we completely avoid trying to solve (or even write down) the characteristic polynomial equation. The very basic power method and its variants, which is based on linear iteration, is used to effectively approximate selected eigenvalues. To determine the complete system of eigenvalues and eigenvectors, the remarkable $Q\,R$ algorithm, which relies on the Gram–Schmidt orthogonalization procedure, is the method of choice, and we shall close with a new proof of its convergence.

8.1. The Power Method.

We have already noted the role played by the eigenvalues and eigenvectors in the solution to linear iterative systems. Now we are going to turn the tables, and use the iterative system as a mechanism for approximating the eigenvalues, or, more correctly, selected eigenvalues of the coefficient matrix. The simplest of the resulting computational procedures is known as the *power method*.

We assume, for simplicity, that A is a complete[†] $n \times n$ matrix. Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ denote its eigenvector basis, and $\lambda_1, \ldots, \lambda_n$ the corresponding eigenvalues. As we have learned, the solution to the linear iterative system

$$\mathbf{v}^{(k+1)} = A\,\mathbf{v}^{(k)}, \qquad \mathbf{v}^{(0)} = \mathbf{v}, \tag{8.1}$$

is obtained by multiplying the initial vector $\mathbf{v}$ by the successive powers of the coefficient matrix: $\mathbf{v}^{(k)} = A^k\,\mathbf{v}$. If we write the initial vector in terms of the eigenvector basis

$$\mathbf{v} = c_1\,\mathbf{v}_1 + \;\cdots\; + c_n\,\mathbf{v}_n, \tag{8.2}$$

then the solution takes the explicit form given in Theorem 7.2, namely

$$\mathbf{v}^{(k)} = A^k\,\mathbf{v} = c_1\,\lambda_1^k\,\mathbf{v}_1 + \;\cdots\; + c_n\,\lambda_n^k\,\mathbf{v}_n. \tag{8.3}$$

[†] This is not a very severe restriction. Most matrices are complete. Moreover, perturbations caused by round off and/or numerical inaccuracies will almost inevitably make an incomplete matrix complete.

Suppose further that A has a single dominant *real* eigenvalue, λ_1, that is larger than all others in magnitude, so

$$|\lambda_1| > |\lambda_j| \qquad \text{for all} \qquad j > 1. \tag{8.4}$$

As its name implies, this eigenvalue will eventually dominate the iteration (8.3). Indeed, since

$$|\lambda_1|^k \gg |\lambda_j|^k \qquad \text{for all} \quad j > 1 \quad \text{and all} \quad k \gg 0,$$

the first term in the iterative formula (8.3) will eventually be much larger than the rest, and so, provided $c_1 \neq 0$,

$$\mathbf{v}^{(k)} \ \approx \ c_1 \lambda_1^k \mathbf{v}_1 \qquad \text{for} \qquad k \gg 0.$$

Therefore, the solution to the iterative system (8.1) will, almost always, end up being a multiple of the dominant eigenvector of the coefficient matrix.

To compute the corresponding eigenvalue, we note that the i^{th} entry of the iterate $\mathbf{v}^{(k)}$ is approximated by $v_i^{(k)} \approx c_1 \lambda_1^k v_{1,i}$, where $v_{1,i}$ is the i^{th} entry of the eigenvector $\mathbf{v}_1$. Thus, as long as $v_{1,i} \neq 0$, we can recover the dominant eigenvalue by taking a ratio between selected components of successive iterates:

$$\lambda_1 \ \approx \ \frac{v_i^{(k)}}{v_i^{(k-1)}}, \qquad \text{provided that} \qquad v_i^{(k-1)} \neq 0. \tag{8.5}$$

Example 8.1. Consider the matrix $A = \begin{pmatrix} -1 & 2 & 2 \\ -1 & -4 & -2 \\ -3 & 9 & 7 \end{pmatrix}$. As you can check, its eigenvalues and eigenvectors are

$$\lambda_1 = 3, \qquad \mathbf{v}_1 = \begin{pmatrix} 1 \\ -1 \\ 3 \end{pmatrix}, \qquad \lambda_2 = -2, \qquad \mathbf{v}_2 = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}, \qquad \lambda_3 = 1, \qquad \mathbf{v}_3 = \begin{pmatrix} -1 \\ 1 \\ -2 \end{pmatrix}.$$

Repeatedly multiplying an initial vector $\mathbf{v} = (1, 0, 0)^T$, say, by A results in the iterates $\mathbf{v}^{(k)} = A^k \mathbf{v}$ listed in the accompanying table. The last column indicates the ratio $\lambda^{(k)} = v_1^{(k)} / v_1^{(k-1)}$ between the first components of successive iterates. (One could equally well use the second or third components.) The ratios are converging to the dominant eigenvalue $\lambda_1 = 3$, while the vectors $\mathbf{v}^{(k)}$ are converging to a very large multiple of the corresponding eigenvector $\mathbf{v}_1 = (1, -1, 3)^T$.

The success of the power method lies in the assumption that A has a unique dominant eigenvalue of maximal modulus, which, by definition, equals its spectral radius: $|\lambda_1| = \rho(A)$. The rate of convergence of the method is governed by the ratio $|\lambda_2/\lambda_1|$ between the subdominant and dominant eigenvalues. Thus, the farther the dominant eigenvalue lies away from the rest, the faster the power method converges. We also assumed that the initial vector $\mathbf{v}^{(0)}$ includes a nonzero multiple of the dominant eigenvector, i.e., $c_1 \neq 0$. As we do not know the eigenvectors, it is not so easy to guarantee this in advance, although one must be quite unlucky to make such a poor choice of initial vector. (Of course, the stupid choice $\mathbf{v}^{(0)} = \mathbf{0}$ is not counted.) Moreover, even if c_1 happens to be 0 initially,

k	$\mathbf{v}^{(k)}$			$\lambda^{(k)}$
0	1	0	0	
1	-1	-1	-3	$-1.$
2	-7	11	-27	7.
3	-25	17	-69	3.5714
4	-79	95	-255	3.1600
5	-241	209	-693	3.0506
6	-727	791	-2247	3.0166
7	-2185	2057	-6429	3.0055
8	-6559	6815	-19935	3.0018
9	-19681	19169	-58533	3.0006
10	-59047	60071	-178167	3.0002
11	-177145	175097	-529389	3.0001
12	-531439	535535	-1598415	3.0000

numerical round-off error will typically come to one's rescue, since it will almost inevitably introduce a tiny component of the eigenvector $\mathbf{v}_1$ into some iterate, and this component will eventually dominate the computation. The trick is to wait long enough for it to show up!

Since the iterates of A are, typically, getting either very large — when $\rho(A) > 1$ — or very small — when $\rho(A) < 1$ — the iterated vectors will be increasingly subject to numerical over- or under-flow, and the method may break down before a reasonable approximation is achieved. One way to avoid this outcome is to restrict our attention to unit vectors relative to a given norm, e.g., the Euclidean norm or the ∞ norm, since their entries cannot be too large, and so are less likely to cause numerical errors in the computations. As usual, the unit vector $\mathbf{u}^{(k)} = \| \mathbf{v}^{(k)} \|^{-1} \mathbf{v}^{(k)}$ is obtained by dividing the iterate by its norm; it can be computed directly by the modified iterative scheme

$$\mathbf{u}^{(0)} = \frac{\mathbf{v}^{(0)}}{\| \mathbf{v}^{(0)} \|}, \qquad \text{and} \qquad \mathbf{u}^{(k+1)} = \frac{A\,\mathbf{u}^{(k)}}{\| A\,\mathbf{u}^{(k)} \|}. \tag{8.6}$$

If the dominant eigenvalue $\lambda_1 > 0$ is positive, then $\mathbf{u}^{(k)} \to \mathbf{u}_1$ will converge to one of the two dominant unit eigenvectors (the other is $-\mathbf{u}_1$). If $\lambda_1 < 0$, then the iterates will switch back and forth between the two eigenvectors, so $\mathbf{u}^{(k)} \approx \pm\mathbf{u}_1$. In either case, the dominant eigenvalue λ_1 is obtained as a limiting ratio between nonzero entries of $A\,\mathbf{u}^{(k)}$ and $\mathbf{u}^{(k)}$. If some other sort of behavior is observed, it means that one of our assumptions is not valid; either A has more than one dominant eigenvalue of maximum modulus, e.g., it has a complex conjugate pair of eigenvalues of largest modulus, or it is not complete.

Example 8.2. For the matrix considered in Example 8.1, starting the iterative scheme (8.6) with $\mathbf{u}^{(k)} = (1,0,0)^T$ by A, the resulting unit vectors are tabulated below.

k	$\mathbf{u}^{(k)}$			λ
0	1	0	0	
1	$-.3015$	$-.3015$	$-.9045$	-1.0000
2	$-.2335$	.3669	$-.9005$	7.0000
3	$-.3319$	.2257	$-.9159$	3.5714
4	$-.2788$	.3353	$-.8999$	3.1600
5	$-.3159$	.2740	$-.9084$	3.0506
6	$-.2919$	.3176	$-.9022$	3.0166
7	$-.3080$	.2899	$-.9061$	3.0055
8	$-.2973$	.3089	$-.9035$	3.0018
9	$-.3044$	.2965	$-.9052$	3.0006
10	$-.2996$	.3048	$-.9041$	3.0002
11	$-.3028$	.2993	$-.9048$	3.0001
12	$-.3007$	.3030	$-.9043$	3.0000

The last column, being the ratio between the first components of $A\mathbf{u}^{(k-1)}$ and $\mathbf{u}^{(k-1)}$, again converges to the dominant eigenvalue $\lambda_1 = 3$.

Variants of the power method for computing the other eigenvalues of the matrix are explored in the exercises.

8.2. The QR Algorithm.

The most popular scheme for simultaneously approximating all the eigenvalues of a matrix A is the remarkable QR algorithm, first proposed in 1961 by Francis, [**18**], and Kublanovskaya, [**31**]. The underlying idea is simple, but surprising. The first step is to factor the matrix

$$A = A_0 = Q_0 R_0$$

into a product of an orthogonal matrix Q_0 and a positive (i.e., with all positive entries along the diagonal) upper triangular matrix R_0 by using the Gram–Schmidt orthogonalization procedure. Next, multiply the two factors together *in the wrong order*! The result is the new matrix

$$A_1 = R_0 Q_0.$$

We then repeat these two steps. Thus, we next factor

$$A_1 = Q_1 R_1$$

using the Gram–Schmidt process, and then multiply the factors in the reverse order to produce

$$A_2 = R_2 Q_2.$$

The complete algorithm can be written as

$$A = Q_0 R_0, \qquad A_{k+1} = R_k Q_k = Q_{k+1} R_{k+1}, \qquad k = 0, 1, 2, \ldots\,, \tag{8.7}$$

where Q_k, R_k come from the previous step, and the subsequent orthogonal matrix Q_{k+1} and positive upper triangular matrix R_{k+1} are computed by using the numerically stable form of the Gram–Schmidt algorithm.

The astonishing fact is that, for many matrices A, the iterates $A_k \longrightarrow V$ converge to an upper triangular matrix V whose diagonal entries are the eigenvalues of A. Thus, after a sufficient number of iterations, say $k^\star$, the matrix $A_{k^\star}$ will have very small entries below the diagonal, and one can read off a complete system of (approximate) eigenvalues along its diagonal. For each eigenvalue, the computation of the corresponding eigenvector can be done by solving the appropriate homogeneous linear system, or by applying the shifted inverse power method.

Example 8.3. Consider the matrix $A = \begin{pmatrix} 2 & 1 \\ 2 & 3 \end{pmatrix}$. The initial Gram–Schmidt factorization $A = Q_0 R_0$ yields

$$Q_0 = \begin{pmatrix} .7071 & -.7071 \\ .7071 & .7071 \end{pmatrix}, \qquad R_0 = \begin{pmatrix} 2.8284 & 2.8284 \\ 0 & 1.4142 \end{pmatrix}.$$

These are multiplied in the reverse order to give

$$A_1 = R_0 Q_0 = \begin{pmatrix} 4 & 0 \\ 1 & 1 \end{pmatrix}.$$

We refactor $A_1 = Q_1 R_1$ via Gram–Schmidt, and then reverse multiply to produce

$$Q_1 = \begin{pmatrix} .9701 & -.2425 \\ .2425 & .9701 \end{pmatrix}, \qquad R_1 = \begin{pmatrix} 4.1231 & .2425 \\ 0 & .9701 \end{pmatrix},$$

$$A_2 = R_1 Q_1 = \begin{pmatrix} 4.0588 & -.7647 \\ .2353 & .9412 \end{pmatrix}.$$

The next iteration yields

$$Q_2 = \begin{pmatrix} .9983 & -.0579 \\ .0579 & .9983 \end{pmatrix}, \qquad R_2 = \begin{pmatrix} 4.0656 & -.7090 \\ 0 & .9839 \end{pmatrix},$$

$$A_3 = R_2 Q_2 = \begin{pmatrix} 4.0178 & -.9431 \\ .0569 & .9822 \end{pmatrix}.$$

Continuing in this manner, after 9 iterations we find, to four decimal places,

$$Q_9 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad R_9 = \begin{pmatrix} 4 & -1 \\ 0 & 1 \end{pmatrix}, \qquad A_{10} = R_9 Q_9 = \begin{pmatrix} 4 & -1 \\ 0 & 1 \end{pmatrix}.$$

The eigenvalues of A, namely 4 and 1, appear along the diagonal of A_{10}. Additional iterations produce very little further change, although they can be used for increasing the accuracy of the computed eigenvalues.

If the original matrix A happens to be symmetric and positive definite, then the limiting matrix $A_k \longrightarrow V = \Lambda$ is, in fact, the diagonal matrix containing the eigenvalues of A. Moreover, if, in this case, we recursively define

$$S_k = S_{k-1} Q_k = Q_0 Q_1 \ \cdots\ Q_{k-1} Q_k, \tag{8.8}$$

then $S_k \longrightarrow S$ have, as their limit, an orthogonal matrix whose columns are the orthonormal eigenvector basis of A.

Example 8.4. Consider the symmetric matrix $A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 3 & -1 \\ 0 & -1 & 6 \end{pmatrix}$. The initial $A = Q_0 R_0$ factorization produces

$$S_0 = Q_0 = \begin{pmatrix} .8944 & -.4082 & -.1826 \\ .4472 & .8165 & .3651 \\ 0 & -.4082 & .9129 \end{pmatrix}, \qquad R_0 = \begin{pmatrix} 2.2361 & 2.2361 & -.4472 \\ 0 & 2.4495 & -3.2660 \\ 0 & 0 & 5.1121 \end{pmatrix},$$

and so

$$A_1 = R_0 Q_0 = \begin{pmatrix} 3.0000 & 1.0954 & 0 \\ 1.0954 & 3.3333 & -2.0870 \\ 0 & -2.0870 & 4.6667 \end{pmatrix}.$$

We refactor $A_1 = Q_1 R_1$ and reverse multiply to produce

$$Q_1 = \begin{pmatrix} .9393 & -.2734 & -.2071 \\ .3430 & .7488 & .5672 \\ 0 & -.6038 & .7972 \end{pmatrix}, \qquad S_1 = S_0 Q_1 = \begin{pmatrix} .7001 & -.4400 & -.5623 \\ .7001 & .2686 & .6615 \\ -.1400 & -.8569 & .4962 \end{pmatrix},$$

$$R_1 = \begin{pmatrix} 3.1937 & 2.1723 & -.7158 \\ 0 & 3.4565 & -4.3804 \\ 0 & 0 & 2.5364 \end{pmatrix}, \qquad A_2 = R_1 Q_1 = \begin{pmatrix} 3.7451 & 1.1856 & 0 \\ 1.1856 & 5.2330 & -1.5314 \\ 0 & -1.5314 & 2.0219 \end{pmatrix}.$$

Continuing in this manner, after 10 iterations we find

$$Q_{10} = \begin{pmatrix} 1.0000 & -.0067 & 0 \\ .0067 & 1.0000 & .0001 \\ 0 & -.0001 & 1.0000 \end{pmatrix}, \qquad S_{10} = \begin{pmatrix} .0753 & -.5667 & -.8205 \\ .3128 & -.7679 & .5591 \\ -.9468 & -.2987 & .1194 \end{pmatrix},$$

$$R_{10} = \begin{pmatrix} 6.3229 & .0647 & 0 \\ 0 & 3.3582 & -.0006 \\ 0 & 0 & 1.3187 \end{pmatrix}, \qquad A_{11} = \begin{pmatrix} 6.3232 & .0224 & 0 \\ .0224 & 3.3581 & -.0002 \\ 0 & -.0002 & 1.3187 \end{pmatrix}.$$

After 20 iterations, the process has completely settled down, and

$$Q_{20} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad S_{20} = \begin{pmatrix} .0710 & -.5672 & -.8205 \\ .3069 & -.7702 & .5590 \\ -.9491 & -.2915 & .1194 \end{pmatrix},$$

$$R_{20} = \begin{pmatrix} 6.3234 & .0001 & 0 \\ 0 & 3.3579 & 0 \\ 0 & 0 & 1.3187 \end{pmatrix}, \qquad A_{21} = \begin{pmatrix} 6.3234 & 0 & 0 \\ 0 & 3.3579 & 0 \\ 0 & 0 & 1.3187 \end{pmatrix}.$$

The eigenvalues of A appear along the diagonal of A_{21}, while the columns of S_{20} are the corresponding orthonormal eigenvector basis, listed in the same order as the eigenvalues, both correct to 4 decimal places.

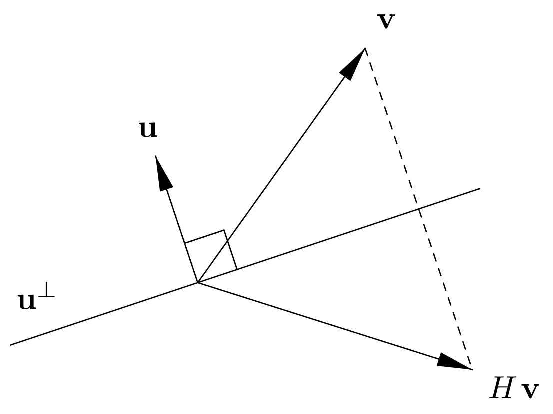


Figure 8.1. Elementary Reflection Matrix.

Tridiagonalization

In practical implementations, the direct $Q\,R$ algorithm often takes too long to provide reasonable approximations to the eigenvalues of large matrices. Fortunately, the algorithm can be made much more efficient by a simple preprocessing step. The key observation is that the $Q\,R$ algorithm preserves the class of symmetric tridiagonal matrices, and, moreover, like Gaussian Elimination, is much faster when applied to this class of matrices.

Consider the *Householder* or *elementary reflection matrix*

$$H = \mathrm{I} - 2\,\mathbf{u}\,\mathbf{u}^T \tag{8.9}$$

in which $\mathbf{u}$ is a unit vector (in the Euclidean norm). The matrix H represents a reflection of vectors through the orthogonal complement to $\mathbf{u}$, as illustrated in Figure 8.1. It is easy to shoe that H is a symmetric orthogonal matrix, and so

$$H^T = H, \qquad H^2 = \mathrm{I}\,, \qquad H^{-1} = H. \tag{8.10}$$

The proof is straightforward: symmetry is immediate, while

$$H\,H^T = H^2 = (\,\mathrm{I} - 2\,\mathbf{u}\,\mathbf{u}^T)\,(\,\mathrm{I} - 2\,\mathbf{u}\,\mathbf{u}^T) = \mathrm{I} - 4\,\mathbf{u}\,\mathbf{u}^T + 4\,\mathbf{u}\,(\mathbf{u}^T\mathbf{u})\,\mathbf{u}^T = \mathrm{I}$$

since, by assumption, $\mathbf{u}^T\mathbf{u} = \|\,\mathbf{u}\,\|^2 = 1$.

In Householder's approach to the $Q\,R$ factorization, we were able to convert the matrix A to upper triangular form R by a sequence of elementary reflection matrices. Unfortunately, this procedure does not preserve the eigenvalues of the matrix — the diagonal entries of R are *not* the eigenvalues — and so we need to be a bit more clever here.

Lemma 8.5. *If $H = \mathrm{I} - 2\,\mathbf{u}\,\mathbf{u}^T$ is an elementary reflection matrix, with $\mathbf{u}$ a unit vector, then A and $B = H\,A\,H$ are similar matrices and hence have the same eigenvalues.*

Proof: According to (8.10), $H^{-1} = H$, and hence $B = H^{-1}A\,H$ is similar to A. *Q.E.D.*

Given a symmetric $n \times n$ matrix A, our goal is to devise a similar tridiagonal matrix by applying a sequence of Householder reflections. We begin by setting

$$\mathbf{x}_1 = \begin{pmatrix} 0 \\ a_{21} \\ a_{31} \\ \vdots \\ a_{n1} \end{pmatrix}, \qquad \mathbf{y}_1 = \begin{pmatrix} 0 \\ \pm r_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \qquad \text{where} \qquad r_1 = \| \mathbf{x}_1 \| = \| \mathbf{y}_1 \|,$$

so that $\mathbf{x}_1$ contains all the off-diagonal entries of the first column of A. Let

$$H_1 = \mathrm{I} - 2\,\mathbf{u}_1\,\mathbf{u}_1^T, \qquad \text{where} \qquad \mathbf{u}_1 = \frac{\mathbf{x}_1 - \mathbf{y}_1}{\| \mathbf{x}_1 - \mathbf{y}_1 \|}$$

be the corresponding elementary reflection matrix that maps $\mathbf{x}_1$ to $\mathbf{y}_1$. Either $\pm$ sign in the formula for $\mathbf{y}_1$ works in the algorithm; a good choice is to set it to be the opposite of the sign of the entry a_{21}, which helps minimize the possible effects of round-off error when computing the unit vector $\mathbf{u}_1$. By direct computation,

$$A_2 = H_1\,A\,H_1 = \begin{pmatrix} a_{11} & r_1 & 0 & \ldots & 0 \\ r_1 & \widetilde{a}_{22} & \widetilde{a}_{23} & \ldots & \widetilde{a}_{2n} \\ 0 & \widetilde{a}_{32} & \widetilde{a}_{33} & \ldots & \widetilde{a}_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \widetilde{a}_{n2} & \widetilde{a}_{n3} & \ldots & \widetilde{a}_{nn} \end{pmatrix} \tag{8.11}$$

for certain $\widetilde{a}_{ij}$; the explicit formulae are not needed. Thus, by a single Householder transformation, we convert A into a similar matrix A_2 whose first row and column are in tridiagonal form. We repeat the process on the lower right $(n-1) \times (n-1)$ submatrix of A_2. We set

$$\mathbf{x}_2 = \begin{pmatrix} 0 \\ 0 \\ \widetilde{a}_{32} \\ \widetilde{a}_{42} \\ \vdots \\ \widetilde{a}_{n2} \end{pmatrix}, \qquad \mathbf{y}_1 = \begin{pmatrix} 0 \\ 0 \\ \pm r_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \qquad \text{where} \qquad r_2 = \| \mathbf{x}_2 \| = \| \mathbf{y}_2 \|,$$

and the $\pm$ sign is chosen to be the opposite of that of $\widetilde{a}_{32}$. Setting

$$H_2 = \mathrm{I} - 2\,\mathbf{u}_2\,\mathbf{u}_2^T, \qquad \text{where} \qquad \mathbf{u}_2 = \frac{\mathbf{x}_2 - \mathbf{y}_2}{\| \mathbf{x}_2 - \mathbf{y}_2 \|}\,,$$

we construct the similar matrix

$$A_3 = H_2\,A_2\,H_2 = \begin{pmatrix} a_{11} & r_1 & 0 & 0 & \ldots & 0 \\ r_1 & \widetilde{a}_{22} & r_2 & 0 & \ldots & 0 \\ 0 & r_2 & \widehat{a}_{33} & \widehat{a}_{34} & \ldots & \widehat{a}_{3n} \\ 0 & 0 & \widehat{a}_{43} & \widehat{a}_{44} & \ldots & \widehat{a}_{4n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \widehat{a}_{n3} & \widehat{a}_{n4} & \ldots & \widehat{a}_{nn} \end{pmatrix}.$$

whose first two rows and columns are now in tridiagonal form. The remaining steps in the algorithm should now be clear. Thus, the final result is a tridiagonal matrix $T = A_n$ that has the *same eigenvalues* as the original symmetric matrix A. Let us illustrate the method by an example.

Example 8.6. To tridiagonalize $A = \begin{pmatrix} 4 & 1 & -1 & 2 \\ 1 & 4 & 1 & -1 \\ -1 & 1 & 4 & 1 \\ 2 & -1 & 1 & 4 \end{pmatrix}$, we begin with its first column. We set $\mathbf{x}_1 = \begin{pmatrix} 0 \\ 1 \\ -1 \\ 2 \end{pmatrix}$, so that $\mathbf{y}_1 = \begin{pmatrix} 0 \\ \sqrt{6} \\ 0 \\ 0 \end{pmatrix} \approx \begin{pmatrix} 0 \\ 2.4495 \\ 0 \\ 0 \end{pmatrix}$. Therefore, the unit vector is $\mathbf{u}_1 = \dfrac{\mathbf{x}_1 - \mathbf{y}_1}{\|\,\mathbf{x}_1 - \mathbf{y}_1\,\|} = \begin{pmatrix} 0 \\ .8391 \\ -.2433 \\ .4865 \end{pmatrix}$, with corresponding Householder matrix

$$H_1 = \mathrm{I} - 2\,\mathbf{u}_1\,\mathbf{u}_1^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -.4082 & .4082 & -.8165 \\ 0 & .4082 & .8816 & .2367 \\ 0 & -.8165 & .2367 & .5266 \end{pmatrix}.$$

Thus,

$$A_2 = H_1\,A\,H_1 = \begin{pmatrix} 4.0000 & -2.4495 & 0 & 0 \\ -2.4495 & 2.3333 & -.3865 & -.8599 \\ 0 & -.3865 & 4.9440 & -.1246 \\ 0 & -.8599 & -.1246 & 4.7227 \end{pmatrix}.$$

In the next phase, $\mathbf{x}_2 = \begin{pmatrix} 0 \\ 0 \\ -.3865 \\ -.8599 \end{pmatrix}$, $\mathbf{y}_2 = \begin{pmatrix} 0 \\ 0 \\ -.9428 \\ 0 \end{pmatrix}$, so $\mathbf{u}_2 = \begin{pmatrix} 0 \\ 0 \\ -.8396 \\ -.5431 \end{pmatrix}$, and

$$H_2 = \mathrm{I} - 2\,\mathbf{u}_2\,\mathbf{u}_2^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -.4100 & -.9121 \\ 0 & 0 & -.9121 & .4100 \end{pmatrix}.$$

The resulting matrix

$$T = A_3 = H_2\,A_2\,H_2 = \begin{pmatrix} 4.0000 & -2.4495 & 0 & 0 \\ -2.4495 & 2.3333 & .9428 & 0 \\ 0 & .9428 & 4.6667 & 0 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

is now in tridiagonal form.

Since the final tridiagonal matrix T has the same eigenvalues as A, we can apply the QR algorithm to T to approximate the common eigenvalues. (The eigenvectors must then be computed separately, e.g., by the shifted inverse power method.) If $A = A_1$ is tridiagonal, so are all the iterates $A_2, A_3, \ldots$. Moreover, far fewer arithmetic operations are required. For instance, in the preceding example, after we apply 20 iterations of the $Q\,R$ algorithm directly to T, the upper triangular factor has become

$$R_{20} = \begin{pmatrix} 6.0000 & -.0065 & 0 & 0 \\ 0 & 4.5616 & 0 & 0 \\ 0 & 0 & 5.0000 & 0 \\ 0 & 0 & 0 & .4384 \end{pmatrix}.$$

The eigenvalues of T, and hence also of A, appear along the diagonal, and are correct to 4 decimal places.

Finally, even if A is not symmetric, one can still apply the same sequence of Householder transformations to simplify it. The final result is no longer tridiagonal, but rather a similar *upper Hessenberg matrix*, which means that all entries below the subdiagonal are zero, but those above the superdiagonal are not necessarily zero. For instance, a 5×5 upper Hessenberg matrix looks like

$$\begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix},$$

where the starred entries can be anything. It can be proved that the $Q\,R$ algorithm maintains the upper Hessenberg form, and, while not as efficient as in the tridiagonal case, still yields a significant savings in computational effort required to find the common eigenvalues. Further details and analysis can be found in [**13**, **43**, **48**].

AIMS Lecture Notes 2006

Peter J. Olver

9. Numerical Solution of Algebraic Systems

In this part, we discuss basic iterative methods for solving systems of algebraic equations. By far the most common is a vector-valued version of Newton's Method, which will form our primary object of study.

9.1. Vector–Valued Iteration.

Extending the scalar analysis to vector-valued iterative systems is not especially difficult. We will build on our experience with linear iterative systems.

We begin by fixing a norm $\| \cdot \|$ on $\mathbb{R}^n$. Since we will also be computing the associated matrix norm $\| A \|$, as defined in Theorem 7.13, it may be more convenient for computations to adopt either the 1 or the ∞ norms rather than the standard Euclidean norm.

We begin by defining the vector-valued counterpart of the basic linear convergence condition (2.21).

Definition 9.1. A function $\mathbf{g}\colon \mathbb{R}^n \to \mathbb{R}^n$ is a *contraction* at a point $\mathbf{u}^\star \in \mathbb{R}^n$ if there exists a constant $0 \le \sigma < 1$ such that

$$\| \mathbf{g}(\mathbf{u}) - \mathbf{g}(\mathbf{u}^\star) \| \ \le\ \sigma \, \| \mathbf{u} - \mathbf{u}^\star \| \tag{9.1}$$

for all $\mathbf{u}$ sufficiently close to $\mathbf{u}^\star$, i.e., $\| \mathbf{u} - \mathbf{u}^\star \| < \delta$ for some fixed $\delta > 0$.

Remark: The notion of a contraction depends on the underlying choice of matrix norm. Indeed, the linear function $\mathbf{g}(\mathbf{u}) = A\mathbf{u}$ if and only if $\| A \| < 1$, which implies that A is a convergent matrix. While every convergent matrix satisfies $\| A \| < 1$ in *some* matrix norm, and hence defines a contraction relative to that norm, it may very well have $\| A \| > 1$ in a particular norm, violating the contaction condition; see (7.31) for an explicit example.

Theorem 9.2. *If* $\mathbf{u}^\star = \mathbf{g}(\mathbf{u}^\star)$ *is a fixed point for the discrete dynamical system* (2.1) *and* $\mathbf{g}$ *is a contraction at* $\mathbf{u}^\star$, *then* $\mathbf{u}^\star$ *is an asymptotically stable fixed point.*

Proof: The proof is a copy of the last part of the proof of Theorem 2.6. We write

$$\| \mathbf{u}^{(k+1)} - \mathbf{u}^\star \| = \| \mathbf{g}(\mathbf{u}^{(k)}) - \mathbf{g}(\mathbf{u}^\star) \| \le \sigma \, \| \mathbf{u}^{(k)} - \mathbf{u}^\star \|,$$

using the assumed estimate (9.1). Iterating this basic inequality immediately demonstrates that

$$\| \mathbf{u}^{(k)} - \mathbf{u}^\star \| \;\leq\; \sigma^k \, \| \mathbf{u}^{(0)} - \mathbf{u}^\star \| \qquad \text{for} \qquad k = 0,1,2,3,\ldots\,. \tag{9.2}$$

Since $\sigma < 1$, the right hand side tends to 0 as $k \to \infty$, and hence $\mathbf{u}^{(k)} \to \mathbf{u}^\star$. *Q.E.D.*

In most interesting situations, the function $\mathbf{g}$ is differentiable, and so can be approximated by its first order Taylor polynomial

$$\mathbf{g}(\mathbf{u}) \approx \mathbf{g}(\mathbf{u}^\star) + \mathbf{g}'(\mathbf{u}^\star)\,(\mathbf{u} - \mathbf{u}^\star) = \mathbf{u}^\star + \mathbf{g}'(\mathbf{u}^\star)\,(\mathbf{u} - \mathbf{u}^\star). \tag{9.3}$$

Here

$$\mathbf{g}'(\mathbf{u}) = \begin{pmatrix} \dfrac{\partial g_1}{\partial u_1} & \dfrac{\partial g_1}{\partial u_2} & \cdots & \dfrac{\partial g_1}{\partial u_n} \\ \dfrac{\partial g_2}{\partial u_1} & \dfrac{\partial g_2}{\partial u_2} & \cdots & \dfrac{\partial g_2}{\partial u_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial g_n}{\partial u_1} & \dfrac{\partial g_n}{\partial u_2} & \cdots & \dfrac{\partial g_n}{\partial u_n} \end{pmatrix}, \tag{9.4}$$

denotes the $n \times n$ *Jacobian matrix* of the vector-valued function $\mathbf{g}$, whose entries are the partial derivatives of its individual components. Since $\mathbf{u}^\star$ is fixed, the the right hand side of (9.3) is an affine function of $\mathbf{u}$. Moreover, $\mathbf{u}^\star$ remains a fixed point of the affine approximation. Proposition 7.25 tells us that iteration of the affine function will converge to the fixed point if and only if its coefficient matrix, namely $\mathbf{g}'(\mathbf{u}^\star)$, is a convergent matrix, meaning that its spectral radius $\rho(\mathbf{g}'(\mathbf{u}^\star)) < 1$.

Theorem 9.3. *Let $\mathbf{u}^\star$ be a fixed point for the discrete dynamical system $\mathbf{u}^{(k+1)} = \mathbf{g}(\mathbf{u}^{(k)})$. If the Jacobian matrix norm $\| \mathbf{g}'(\mathbf{u}^\star) \| < 1$, then $\mathbf{g}$ is a contraction at $\mathbf{u}^\star$, and hence the fixed point $\mathbf{u}^\star$ is asymptotically stable.*

Proof: The first order Taylor expansion of $\mathbf{g}(\mathbf{u})$ at the fixed point $\mathbf{u}^\star$ takes the form

$$\mathbf{g}(\mathbf{u}) = \mathbf{g}(\mathbf{u}^\star) + \mathbf{g}'(\mathbf{u}^\star)\,(\mathbf{u} - \mathbf{u}^\star) + R(\mathbf{u} - \mathbf{u}^\star), \tag{9.5}$$

where the remainder term satisfies

$$\lim_{\mathbf{u} \to \mathbf{u}^\star} \frac{R(\mathbf{u} - \mathbf{u}^\star)}{\| \mathbf{u} - \mathbf{u}^\star \|} \;=\; 0.$$

Let $\varepsilon > 0$ be such that

$$\sigma = \| \mathbf{g}'(\mathbf{u}^\star) \| + \varepsilon < 1.$$

Choose $0 < \delta < 1$ such that $\| R(\mathbf{u} - \mathbf{u}^\star) \| \leq \varepsilon \, \| \mathbf{u} - \mathbf{u}^\star \|$ whenever $\| \mathbf{u} - \mathbf{u}^\star \| \leq \delta$. For such $\mathbf{u}$, we have, by the Triangle Inequality,

$$\begin{aligned} \| \mathbf{g}(\mathbf{u}) - \mathbf{g}(\mathbf{u}^\star) \| \;&\leq\; \| \mathbf{g}'(\mathbf{u}^\star)\,(\mathbf{u} - \mathbf{u}^\star) \| + \| R(\mathbf{u} - \mathbf{u}^\star) \| \\ &\leq\; \big(\| \mathbf{g}'(\mathbf{u}^\star) \| + \varepsilon \big) \, \| \mathbf{u} - \mathbf{u}^\star \| = \sigma \, \| \mathbf{u} - \mathbf{u}^\star \|, \end{aligned}$$

which establishes the contraction inequality (9.1). *Q.E.D.*

Corollary 9.4. *If the Jacobian matrix* $\mathbf{g}'(\mathbf{u}^\star)$ *is a convergent matrix, meaning that its spectral radius satisfies* $\rho\big(\,\mathbf{g}'(\mathbf{u}^\star)\,\big) < 1$, *then* $\mathbf{u}^\star$ *is an asymptotically stable fixed point.*

Proof: Corollary 7.23 assures us that $\|\,\mathbf{g}'(\mathbf{u}^\star)\,\| < 1$ in some matrix norm. Using this norm, the result immediately follows from the theorem. *Q.E.D.*

Theorem 9.3 tells us that initial values $\mathbf{u}^{(0)}$ that are sufficiently near a stable fixed point $\mathbf{u}^\star$ are guaranteed to converge to it. In the linear case, closeness of the initial data to the fixed point was not, in fact, an issue; all stable fixed points are, in fact, globally stable. For nonlinear iteration, it is of critical importance, and one does not typically expect iteration starting with far away initial data to converge to the desired fixed point. An interesting (and difficult) problem is to determine the so-called *basin of attraction* of a stable fixed point, defined as the set of all initial data that ends up converging to it. As in the elementary logistic map (2.22), initial values that lie outside a basin of attraction can lead to divergent iterates, periodic orbits, or even exhibit chaotic behavior. The full range of possible phenomena is a topic of contemporary research in dynamical systems theory, [**42**], and in numerical analysis, [**1**].

Example 9.5. Consider the function

$$\mathbf{g}(u,v) = \begin{pmatrix} -\frac{1}{4}\,u^3 + \frac{9}{8}\,u + \frac{1}{4}\,v^3 \\ \frac{3}{4}\,v - \frac{1}{2}\,uv \end{pmatrix}.$$

There are four (real) fixed points; stability is determined by the size of the eigenvalues of the Jacobian matrix

$$\mathbf{g}'(u,v) = \begin{pmatrix} \frac{9}{8} - \frac{3}{4}\,u^2 & -\frac{1}{2}\,v \\ \frac{3}{4}\,v^2 & \frac{3}{4} - \frac{1}{2}\,u \end{pmatrix}$$

at each of the fixed points. The results are summarized in the following table:

fixed point	$\mathbf{u}_1^\star = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$	$\mathbf{u}_2^\star = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$	$\mathbf{u}_3^\star = \begin{pmatrix} -\frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$	$\mathbf{u}_4^\star = \begin{pmatrix} -\frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$
Jacobian matrix	$\begin{pmatrix} \frac{9}{8} & 0 \\ 0 & \frac{3}{4} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{4} & 0 \\ 0 & \frac{3}{4} - \frac{1}{2\sqrt{2}} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{4} & 0 \\ 0 & \frac{3}{4} + \frac{1}{2\sqrt{2}} \end{pmatrix}$	$\begin{pmatrix} \frac{15}{16} & -\frac{1}{4} \\ \frac{3}{16} & 1 \end{pmatrix}$
eigenvalues	1.125, .75	.75, .396447	1.10355, .75	$.96875 \pm .214239\,\mathrm{i}$
spectral radius	1.125	.75	1.10355	.992157

Thus, $\mathbf{u}_2^\star$ and $\mathbf{u}_4^\star$ are stable fixed points, whereas $\mathbf{u}_1^\star$ and $\mathbf{u}_3^\star$ are both unstable. Indeed, starting with $\mathbf{u}^{(0)} = (\,.5, .5\,)^T$, it takes 24 iterates to converge to $\mathbf{u}_2^\star$ with 4 significant decimal digits, whereas starting with $\mathbf{u}^{(0)} = (\,-.7, .7\,)^T$, it takes 1049 iterates to converge to within 4 digits of $\mathbf{u}_4^\star$; the slower convergence rate is predicted by the larger Jacobian

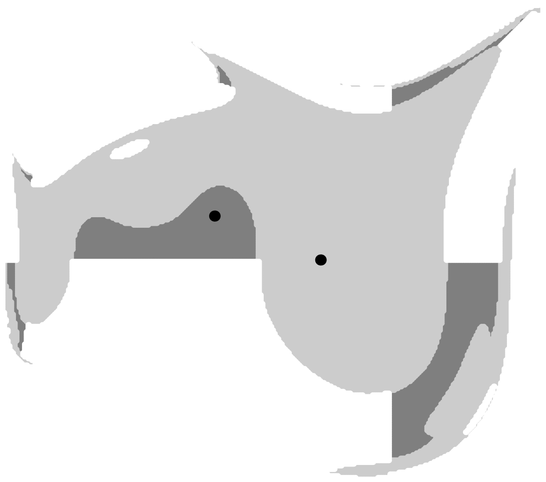


Figure 9.1. Basins of Attraction.

spectral radius. The two basins of attraction are plotted in Figure 9.1. The stable fixed points are indicated by black dots. The light gray region contains $\mathbf{u}_2^\star$ and indicates all the points that converge to it; the darker gray indicates points converging, more slowly, to $\mathbf{u}_4^\star$. All other initial points, except $\mathbf{u}_1^\star$ and $\mathbf{u}_3^\star$, have rapidly unbounded iterates: $\| \mathbf{u}^{(k)} \| \to \infty$.

The smaller the spectral radius or matrix norm of the Jacobian matrix at the fixed point, the faster the nearby iterates will converge to it. As in the scalar case, quadratic convergence will occur when the Jacobian matrix $\mathbf{g}'(\mathbf{u}^\star) = \mathrm{O}$ is the zero matrix, i.e., *all* first order partial derivatives of the components of $\mathbf{g}$ vanish at the fixed point. The quadratic convergence estimate

$$\| \mathbf{u}^{(k+1)} - \mathbf{u}^\star \| \ \leq \ \tau \, \| \mathbf{u}^{(k)} - \mathbf{u}^\star \|^2 \tag{9.6}$$

follows from the second order Taylor expansion of $\mathbf{g}(\mathbf{u})$ at the fixed point.

Of course, in practice we don't know the norm or spectral radius of the Jacobian matrix $\mathbf{g}'(\mathbf{u}^\star)$ because we don't know where the fixed point is. This apparent difficulty can be easily circumvented by requiring that $\| \mathbf{g}'(\mathbf{u}) \| < 1$ for all $\mathbf{u}$ — or, at least, for all $\mathbf{u}$ in a domain Ω containing the fixed point. In fact, this hypothesis can be used to prove the exitence and uniqueness of asymptotically stable fixed points. Rather than work with the Jacobian matrix, let us return to the contraction condition (9.1), but now imposed uniformly on an entire domain.

Definition 9.6. A function $\mathbf{g}\colon \mathbb{R}^n \to \mathbb{R}^n$ is called a *contraction mapping* on a domain $\Omega \subset \mathbb{R}^n$ if

(a) it maps Ω to itself, so $\mathbf{g}(\mathbf{u}) \in \Omega$ whenever $\mathbf{u} \in \Omega$, and

(b) there exists a *constant* $0 \leq \sigma < 1$ such that

$$\| \mathbf{g}(\mathbf{u}) - \mathbf{g}(\mathbf{v}) \| \ \leq \ \sigma \, \| \mathbf{u} - \mathbf{v} \| \qquad \text{for all} \qquad \mathbf{u}, \mathbf{v} \in \Omega. \tag{9.7}$$

In other words, applying a contraction mapping reduces the mutual distance between points. So, as its name indicates, a contraction mapping effectively shrinks the size of its

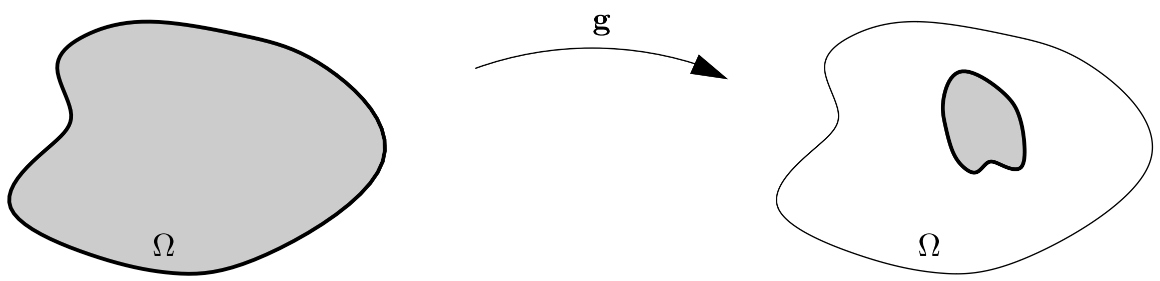


Figure 9.2. A Contraction Mapping.

domain; see Figure 9.2. As the iterations proceed, the successive image domains become smaller and smaller. If the original domain is closed and bounded, then it is forced to shrink down to a single point, which is the unique fixed point of the iterative system. This follows from the *Contraction Mapping Theorem*

Theorem 9.7. *If* $\mathbf{g}$ *is a contraction mapping on a closed bounded domain* $\Omega \subset \mathbb{R}^n$*, then* $\mathbf{g}$ *admits a unique fixed point* $\mathbf{u}^\star \in \Omega$*. Moreover, starting with any initial point* $\mathbf{u}^{(0)} \in \Omega$*, the iterates* $\mathbf{u}^{(k+1)} = \mathbf{g}(\mathbf{u}^{(k)})$ *necessarily converge to the fixed point:* $\mathbf{u}^{(k)} \to \mathbf{u}^\star$*.*

In particular, if $\| \mathbf{g}'(\mathbf{u}) \| < 1$ for all $\mathbf{u} \in \Omega$, then the conclusions of the Contraction Mapping Theorem 9.7 hold.

9.2. Solution of Algebraic Systems.

There is no direct universal solution method for nonlinear systems comparable to Gaussian elimination. Numerical solution techniques rely almost exclusively on iterative algorithms. This section presents the principal methods for numerically approximating the solution(s) to a nonlinear system. We shall only discuss general purpose algorithms; specialized methods for solving particular classes of equations, e.g., polynomial equations, can be found in numerical analysis texts, e.g., [**5**, **7**, **43**]. Of course, the most important specialized methods — those designed for solving linear systems — will continue to play a critical role, even in the nonlinear regime.

We concentrate on the "regular" case when the system contains the same number of equations as unknowns:

$$f_1(u_1, \ldots, u_n) = 0, \qquad \ldots \qquad f_n(u_1, \ldots, u_n) = 0. \tag{9.8}$$

We will rewrite the system in vector form

$$\mathbf{f}(\mathbf{u}) = \mathbf{0}, \tag{9.9}$$

where $\mathbf{f}: \mathbb{R}^n \to \mathbb{R}^n$ is a vector-valued function of n variables. In practice, we do not necessarily require that $\mathbf{f}$ be defined on all of $\mathbb{R}^n$, although this does simplify the exposition.

We shall only consider solutions that are separated from any others. More formally:

Definition 9.8. A solution $\mathbf{u}^\star$ to a system $\mathbf{f}(\mathbf{u}) = \mathbf{0}$ is called *isolated* if there exists $\delta > 0$ such that $\mathbf{f}(\mathbf{u}) \neq \mathbf{0}$ for all $\mathbf{u}$ satisfying $0 < \| \mathbf{u} - \mathbf{u}^\star \| < \delta$.

Example 9.9. Consider the planar equation

$$x^2 + y^2 = (x^2 + y^2)^2.$$

Rewriting the equation in polar coordinates as

$$r = r^2 \qquad \text{or} \qquad r(r-1) = 0,$$

we immediately see that the solutions consist of the origin $x = y = 0$ and all points on the unit circle $r^2 = x^2 + y^2 = 1$. Only the origin is an isolated solution, since every solution lying on the circle has plenty of other points on the circle that lie arbitrarily close to it.

Typically, solutions to a system of n equations in n unknowns are isolated, although this is not always true. For example, if A is a singular $n \times n$ matrix, then the solutions to the homogeneous linear system $A\mathbf{u} = \mathbf{0}$ form a nontrivial subspace, and so are not isolated. Nonlinear systems with non-isolated solutions can similarly be viewed as exhibiting some form of degeneracy. In general, the numerical computation of non-isolated solutions, e.g., solving the implicit equations for a curve or surface, is a much more difficult problem, and we will not attempt to discuss these issues in this introductory presentation. (However, our continuation approach to the Kepler equation in Example 2.20 indicates how one might proceed in such situations.)

In the case of a single scalar equation, the simple roots, meaning those for which $f'(u^\star) \neq 0$, are the easiest to compute. In higher dimensions, the role of the derivative of the function is played by the Jacobian matrix (9.4), and this motivates the following definition.

Definition 9.10. A solution $\mathbf{u}^\star$ to a system $\mathbf{f}(\mathbf{u}) = \mathbf{0}$ is called *nonsingular* if the associated Jacobian matrix is nonsingular there: $\det \mathbf{f}'(\mathbf{u}^\star) \neq 0$.

Note that the Jacobian matrix is square if and only if the system has the same number of equations as unknowns, which is thus one of the requirements for a solution to be nonsingular in our sense. Moreover, the Inverse Function Theorem from multivariable calculus, [**2**, **34**], implies that a nonsingular solution is necessarily isolated.

Theorem 9.11. *Every nonsingular solution* $\mathbf{u}^\star$ *to a system* $\mathbf{f}(\mathbf{u}) = \mathbf{0}$ *is isolated.*

Being the multivariate counterparts of simple roots also means that nonsingular solutions of systems are the most amenable to practical computation. Computing non-isolated solutions, as well as isolated solutions with a singular Jacobian matrix, is a considerable challenge, and practical algorithms remain much less well developed. For this reason, we focus exclusively on numerical solution techniques for nonsingular solutions.

Now, let us turn to numerical solution techniques. The first remark is that, unlike the scalar case, proving existence of a solution to a system of equations is often a challenging issue. There is no counterpart to the Intermediate Value Lemma 2.12 for vector-valued functions. It is not hard to find vector-valued functions whose entries take on both positive and negative values, but admit no solutions. This precludes any simple analog of the Bisection Method for nonlinear systems in more than one unknown.

On the other hand, Newton's Method can be straightforwardly adapted to compute nonsingular solutions to systems of equations, and is *the* most widely used method for this purpose. The derivation proceeds in very similar manner to the scalar case. First, we replace the system (9.9) by a fixed point system

$$\mathbf{u} = \mathbf{g}(\mathbf{u}) \tag{9.10}$$

having the same solutions. By direct analogy with (2.33), any (reasonable) fixed point method will take the form

$$\mathbf{g}(\mathbf{u}) = \mathbf{u} - L(\mathbf{u})\,\mathbf{f}(\mathbf{u}), \tag{9.11}$$

where $L(\mathbf{u})$ is an $n \times n$ matrix-valued function. Clearly, if $\mathbf{f}(\mathbf{u}) = \mathbf{0}$ then $\mathbf{g}(\mathbf{u}) = \mathbf{u}$; conversely, if $\mathbf{g}(\mathbf{u}) = \mathbf{u}$, then $L(\mathbf{u})\,\mathbf{f}(\mathbf{u}) = \mathbf{0}$. If we further require that the matrix $L(\mathbf{u})$ be nonsingular, i.e., $\det L(\mathbf{u}) \neq 0$, then every fixed point of the iterator (9.11) will be a solution to the system (9.9) and vice versa.

According to Theorem 9.3, the speed of convergence (if any) of the iterative method

$$\mathbf{u}^{(k+1)} = \mathbf{g}(\mathbf{u}^{(k)}) \tag{9.12}$$

is governed by the matrix norm (or, more precisely, the spectral radius) of the Jacobian matrix $\mathbf{g}'(\mathbf{u}^\star)$ at the fixed point. In particular, if

$$\mathbf{g}'(\mathbf{u}^\star) = \mathrm{O} \tag{9.13}$$

is the zero matrix, then the method converges quadratically fast. Let's figure out how this can be arranged. Computing the derivative using the matrix version of the Leibniz rule for the derivative of a matrix product, we find

$$\mathbf{g}'(\mathbf{u}^\star) = \mathrm{I} - L(\mathbf{u}^\star)\,\mathbf{f}'(\mathbf{u}^\star), \tag{9.14}$$

where I is the $n \times n$ identity matrix. (Fortunately, all the terms that involve derivatives of the entries of $L(\mathbf{u})$ go away since $\mathbf{f}(\mathbf{u}^\star) = \mathbf{0}$ by assumption.) Therefore, the quadratic convergence criterion (9.13) holds if and only if

$$L(\mathbf{u}^\star)\,\mathbf{f}'(\mathbf{u}^\star) = \mathrm{I}, \qquad \text{and hence} \qquad L(\mathbf{u}^\star) = \mathbf{f}'(\mathbf{u}^\star)^{-1} \tag{9.15}$$

should be the inverse of the Jacobian matrix of $\mathbf{f}$ at the solution, which, fortuitously, was already assumed to be nonsingular.

As in the scalar case, we don't know the solution $\mathbf{u}^\star$, but we can arrange that condition (9.15) holds by setting

$$L(\mathbf{u}) = \mathbf{f}'(\mathbf{u})^{-1}$$

everywhere — or at least everywhere that $\mathbf{f}$ has a nonsingular Jacobian matrix. The resulting fixed point system

$$\mathbf{u} = \mathbf{g}(\mathbf{u}) = \mathbf{u} - \mathbf{f}'(\mathbf{u})^{-1}\,\mathbf{f}(\mathbf{u}), \tag{9.16}$$

leads to the quadratically convergent *Newton iteration scheme*

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} - \mathbf{f}'(\mathbf{u}^{(k)})^{-1}\,\mathbf{f}(\mathbf{u}^{(k)}). \tag{9.17}$$

All it requires is that we guess an initial value $\mathbf{u}^{(0)}$ that is sufficiently close to the desired solution $\mathbf{u}^\star$. We are then guaranteed that the iterates $\mathbf{u}^{(k)}$ converge quadratically fast to $\mathbf{u}^\star$.

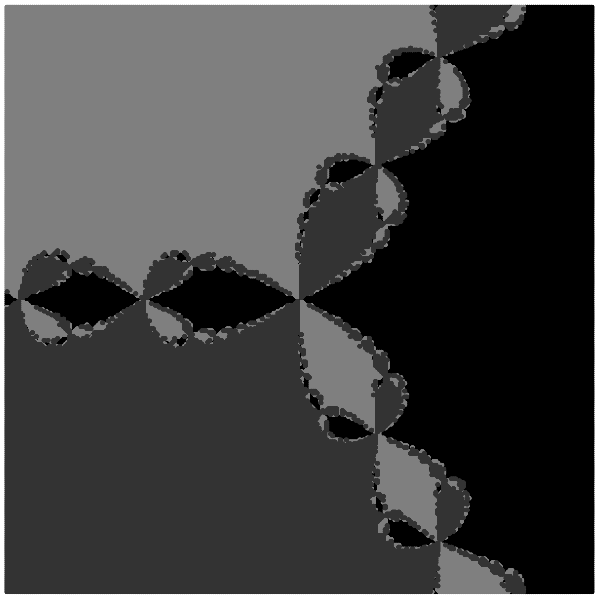


Figure 9.3. Computing the Cube Roots of Unity by Newton's Method.

Theorem 9.12. *Let* $\mathbf{u}^\star$ *be a nonsingular solution to the system* $\mathbf{f}(\mathbf{u}) = \mathbf{0}$*. Then, provided* $\mathbf{u}^{(0)}$ *is sufficiently close to* $\mathbf{u}^\star$*, the Newton iteration scheme* (9.17) *converges at a quadratic rate to the solution:* $\mathbf{u}^{(k)} \to \mathbf{u}^\star$*.*

Example 9.13. Consider the pair of simultaneous cubic equations

$$f_1(u,v) = u^3 - 3uv^2 - 1 = 0, \qquad f_2(u,v) = 3u^2 v - v^3 = 0. \tag{9.18}$$

It is not difficult to prove that there are precisely three solutions:

$$\mathbf{u}_1^\star = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad \mathbf{u}_2^\star = \begin{pmatrix} -.5 \\ .866025\ldots \end{pmatrix}, \qquad \mathbf{u}_3^\star = \begin{pmatrix} -.5 \\ -.866025\ldots \end{pmatrix}. \tag{9.19}$$

The Newton scheme relies on the Jacobian matrix

$$\mathbf{f}'(\mathbf{u}) = \begin{pmatrix} 3u^2 - 3v^2 & -6uv \\ 6uv & 3u^2 - 3v^2 \end{pmatrix}.$$

Since $\det \mathbf{f}'(\mathbf{u}) = 9(u^2+v^2)$ is non-zero except at the origin, all three solutions are non-singular, and hence, for a sufficiently close initial value, Newton's Method will converge to the nearby solution. We explicitly compute the inverse Jacobian matrix:

$$\mathbf{f}'(\mathbf{u})^{-1} = \frac{1}{9(u^2+v^2)} \begin{pmatrix} 3u^2 - 3v^2 & 6uv \\ -6uv & 3u^2 - 3v^2 \end{pmatrix}.$$

Hence, in this particular example, the Newton iterator (9.16) is

$$\mathbf{g}(\mathbf{u}) = \begin{pmatrix} u \\ v \end{pmatrix} - \frac{1}{9(u^2+v^2)} \begin{pmatrix} 3u^2 - 3v^2 & 6uv \\ -6uv & 3u^2 - 3v^2 \end{pmatrix} \begin{pmatrix} u^3 - 3uv^2 - 1 \\ 3u^2 v - v^3 \end{pmatrix}.$$

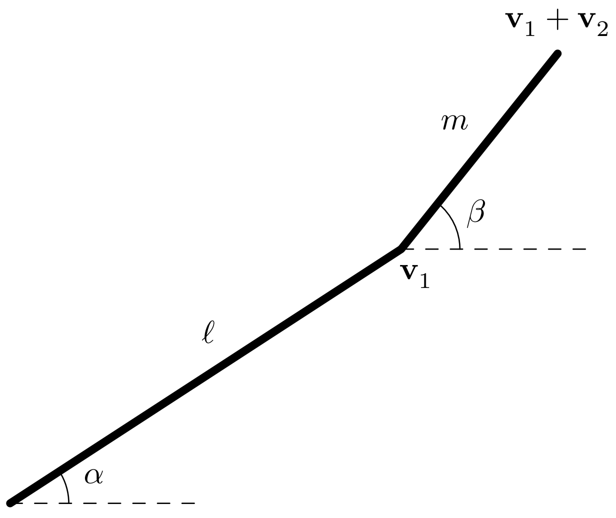


Figure 9.4. Robot Arm.

A complete diagram of the three basins of attraction, consisting of points whose Newton iterates converge to each of the three roots, has a remarkably complicated, fractal-like structure, as illustrated in Figure 9.3. In this plot, the x and y coordinates run from -1.5 to 1.5. The points in the black region all converge to $\mathbf{u}_1^\star$; those in the light gray region all converge to $\mathbf{u}_2^\star$; while those in the dark gray region all converge to $\mathbf{u}_3^\star$. The closer one is to the root, the sooner the iterates converge. On the interfaces between the basins of attraction are points for which the Newton iterates fail to converge, but exhibit a random, chaotic behavior. However, round-off errors will cause such iterates to fall into one of the basins, making it extremely difficult to observe such behavio over the long run.

Remark: The alert reader may notice that in this example, we are in fact merely computing the cube roots of unity, i.e., equations (9.18) are the real and imaginary parts of the complex equation $z^3 = 1$ when $z = u + \mathrm{i}\, v$.

Example 9.14. A robot arm consists of two rigid rods that are joined end-to-end to a fixed point in the plane, which we take as the origin $\mathbf{0}$. The arms are free to rotate, and the problem is to configure them so that the robot's hand ends up at the prescribed position $\mathbf{a} = (\,a, b\,)^T$. The first rod has length ℓ and makes an angle α with the horizontal, so its end is at position $\mathbf{v}_1 = (\,\ell\cos\alpha, \ell\sin\alpha\,)^T$. The second rod has length m and makes an angle β with the horizontal, and so is represented by the vector $\mathbf{v}_2 = (\,m\cos\beta, m\sin\beta\,)^T$. The hand at the end of the second arm is at position $\mathbf{v}_1 + \mathbf{v}_2$, and the problem is to find values for the angles α, β so that $\mathbf{v}_1 + \mathbf{v}_2 = \mathbf{a}$; see Figure 9.4. To this end, we need to solve the system of equations

$$\ell\cos\alpha + m\cos\beta = a, \qquad \ell\sin\alpha + m\sin\beta = b, \tag{9.20}$$

for the angles α, β.

To find the solution, we shall apply Newton's Method. First, we compute the Jacobian matrix of the system with respect to α, β, which is

$$\mathbf{f}'(\alpha,\beta) = \begin{pmatrix} -\,\ell\sin\alpha & -\,m\sin\beta \\ \ell\cos\alpha & m\cos\beta \end{pmatrix},$$

with inverse

$$\mathbf{f}'(\alpha,\beta)^{-1} = \frac{1}{\ell m \sin(\beta-\alpha)} \begin{pmatrix} -\ell\sin\alpha & m\sin\beta \\ -\ell\cos\alpha & m\cos\beta \end{pmatrix}.$$

As a result, the Newton iteration equation (9.17) has the explicit form

$$\begin{pmatrix} \alpha^{(k+1)} \\ \beta^{(k+1)} \end{pmatrix} = \begin{pmatrix} \alpha^{(k)} \\ \beta^{(k)} \end{pmatrix} - \\ - \frac{1}{\ell m \sin(\beta^{(k)}-\alpha^{(k)})} \begin{pmatrix} -\ell\cos\alpha^{(k)} & m\sin\beta^{(k)} \\ -\ell\cos\alpha^{(k)} & m\sin\beta^{(k)} \end{pmatrix} \begin{pmatrix} \ell\cos\alpha^{(k)} + m\cos\beta^{(k)} - a \\ \ell\sin\alpha^{(k)} + m\sin\beta^{(k)} - b \end{pmatrix}.$$

when running the iteration, one must be careful to avoid points at which $\alpha^{(k)} - \beta^{(k)} = 0$ or π, i.e., where the robot arm has straightened out.

As an example, let us assume that the rods have lengths $\ell = 2$, $m = 1$, and the desired location of the hand is at $\mathbf{a} = (1,1)^T$. We start with an initial guess of $\alpha^{(0)} = 0$, $\beta^{(0)} = \frac{1}{2}\pi$, so the first rod lies along the x–axis and the second is perpendicular. The first few Newton iterates are given in the accompanying table. The first column is the iterate number k; the second and third columns indicate the angles $\alpha^{(k)}$, $\beta^{(k)}$ of the rods. The fourth and fifth give the position $(x^{(k)}, y^{(k)})^T$ of the joint or elbow, while the final two indicate the position $(z^{(k)}, w^{(k)})^T$ of the robot's hand.

k	$\alpha^{(k)}$	$\beta^{(k)}$	$x^{(k)}$	$y^{(k)}$	$z^{(k)}$	$w^{(k)}$
0	.0000	1.5708	2.0000	.0000	2.0000	1.0000
1	.0000	2.5708	2.0000	.0000	1.1585	.5403
2	.3533	2.8642	1.8765	.6920	.9147	.9658
3	.2917	2.7084	1.9155	.5751	1.0079	.9948
4	.2987	2.7176	1.9114	.5886	1.0000	1.0000
5	.2987	2.7176	1.9114	.5886	1.0000	1.0000

Observe that the robot has rapidly converged to one of the two possible configurations. (Can you figure out what the second equilibrium is?) In general, convergence depends on the choice of initial configuration, and the Newton iterates do not always settle down to a fixed point. For instance, if $\| \mathbf{a} \| > \ell + m$, there is no possible solution, since the arms are too short for the hand to reach to desired location; thus, no choice of initial conditions will lead to a convergent scheme and the robot arm flaps around in a chaotic manner.

Now that we have gained a little experience with Newton's Method for systems of equations, some supplementary remarks are in order. As we learned, except perhaps in very low-dimensional situations, one should not directly invert a matrix, but rather use Gaussian elimination, or, in favorable situations, a linear iterative scheme, e.g., Jacobi, Gauss–Seidel or even SOR. So a better strategy is to leave the Newton system (9.17) in unsolved, implicit form

$$\mathbf{f}'(\mathbf{u}^{(k)})\,\mathbf{v}^{(k)} = -\,\mathbf{f}(\mathbf{u}^{(k)}), \qquad \mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \mathbf{v}^{(k)}. \tag{9.21}$$

Given the iterate $\mathbf{u}^{(k)}$, we compute the Jacobian matrix $\mathbf{f}'(\mathbf{u}^{(k)})$ and the right hand side $-\mathbf{f}(\mathbf{u}^{(k)})$, and then use our preferred linear systems solver to find $\mathbf{v}^{(k)}$. Adding $\mathbf{u}^{(k)}$ to the result immediately yields the updated approximation $\mathbf{u}^{(k+1)}$ to the solution.

The main bottleneck in the implementation of the Newton scheme, particularly for large systems, is solving the linear system in (9.21). The coefficient matrix $\mathbf{f}'(\mathbf{u}^{(k)})$ must be recomputed at each step of the iteration, and hence knowing the solution to the k^{th} linear system does not appear to help us solve the subsequent system. Pereforming a complete Gaussian elimination at every step will tend to slow down the algorithm, particularly in high dimensional situations involving many equations in many unknowns.

One simple dodge for speeding up the computation is to note that, once we start converging, $\mathbf{u}^{(k)}$ will be very close to $\mathbf{u}^{(k-1)}$ and so we will probably not go far wrong by using $\mathbf{f}'(\mathbf{u}^{(k-1)})$ in place of the updated Jacobian matrix $\mathbf{f}'(\mathbf{u}^{(k)})$. Since we have already solved the linear system with coefficient matrix $\mathbf{f}'(\mathbf{u}^{(k-1)})$, we know its $L\,U$ factorization, and hence can use Forward and Back Substitution to quickly solve the modified system

$$\mathbf{f}'(\mathbf{u}^{(k-1)})\,\mathbf{v}^{(k+1)} = -\,\mathbf{f}(\mathbf{u}^{(k)}), \qquad \mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \mathbf{v}^{(k)}. \tag{9.22}$$

If $\mathbf{u}^{(k+1)}$ is still close to $\mathbf{u}^{(k-1)}$, we can continue to use $\mathbf{f}'(\mathbf{u}^{(k-1)})$ as the coefficient matrix when proceeding on to the next iterate $\mathbf{u}^{(k+2)}$. We proceed in this manner until there has been a notable change in the iterates, at which stage we can revert to solving the correct, unmodified linear system (9.21) by Gaussian Elimination. This strategy may dramatically reduce the total amount of computation required to approximate the solution to a prescribed accuracy. The down side is that this *quasi-Newton scheme* is only linearly convergent, and so does not home in on the root as fast as the unmodified implementation. The user needs to balance the trade-off between speed of convergence versus amount of time needed to solve the linear system at each step in the process. See [**43**] for further discussion.

AIMS Lecture Notes 2006

Peter J. Olver

10. Numerical Solution of Ordinary Differential Equations

This part is concerned with the numerical solution of initial value problems for systems of ordinary differential equations. We will introduce the most basic one-step methods, beginning with the most basic Euler scheme, and working up to the extremely popular Runge–Kutta fourth order method that can be successfully employed in most situations. We end with a brief discussion of stiff differential equations, which present a more serious challenge to numerical analysts.

10.1. First Order Systems of Ordinary Differential Equations.

Let us begin by reviewing the theory of ordinary differential equations. Many physical applications lead to higher order systems of ordinary differential equations, but there is a simple reformulation that will convert them into equivalent first order systems. Thus, we do not lose any generality by restricting our attention to the first order case throughout. Moreover, numerical solution schemes for higher order initial value problems are entirely based on their reformulation as first order systems.

First Order Systems

A *first order system of ordinary differential equations* has the general form

$$\frac{du_1}{dt} = F_1(t,u_1,\ldots,u_n), \qquad \cdots \qquad \frac{du_n}{dt} = F_n(t,u_1,\ldots,u_n). \tag{10.1}$$

The unknowns $u_1(t),\ldots,u_n(t)$ are scalar functions of the real variable t, which usually represents time. We shall write the system more compactly in vector form

$$\frac{d\mathbf{u}}{dt} = \mathbf{F}(t,\mathbf{u}), \tag{10.2}$$

where $\mathbf{u}(t) = (\,u_1(t),\ldots,u_n(t)\,)^T$, and $\mathbf{F}(t,\mathbf{u}) = (\,F_1(t,u_1,\ldots,u_n),\ldots,F_n(t,u_1,\ldots,u_n)\,)^T$ is a vector-valued function of $n+1$ variables. By a *solution* to the differential equation, we mean a vector-valued function $\mathbf{u}(t)$ that is defined and continuously differentiable on an interval $a < t < b$, and, moreover, satisfies the differential equation on its interval of definition. Each solution $\mathbf{u}(t)$ serves to parametrize a curve $C \subset \mathbb{R}^n$, also known as a *trajectory* or *orbit* of the system.

In this part, we shall concentrate on initial value problems for such first order systems. The general initial conditions are

$$u_1(t_0) = a_1, \qquad u_2(t_0) = a_2, \qquad \cdots \qquad u_n(t_0) = a_n, \tag{10.3}$$

or, in vectorial form,

$$\mathbf{u}(t_0) = \mathbf{a} \tag{10.4}$$

Here t_0 is a prescribed initial time, while the vector $\mathbf{a} = (a_1, a_2, \ldots, a_n)^T$ fixes the initial position of the desired solution. In favorable situations, as described below, the initial conditions serve to uniquely specify a solution to the differential equations — at least for nearby times. The general issues of existence and uniquenss of solutions will be addressed in the following section.

A system of differential equations is called *autonomous* if the right hand side does not explicitly depend upon the time t, and so takes the form

$$\frac{d\mathbf{u}}{dt} = \mathbf{F}(\mathbf{u}). \tag{10.5}$$

One important class of autonomous first order systems are the steady state fluid flows. Here $\mathbf{F}(\mathbf{u}) = \mathbf{v}$ represents the fluid velocity vector field at the position $\mathbf{u}$. The solution $\mathbf{u}(t)$ to the initial value problem (10.5, 4) describes the motion of a fluid particle that starts at position $\mathbf{a}$ at time t_0. The differential equation tells us that the fluid velocity at each point on the particle's trajectory matches the prescribed vector field.

An *equilibrium solution* is constant: $\mathbf{u}(t) \equiv \mathbf{u}^\star$ for all t. Thus, its derivative must vanish, $d\mathbf{u}/dt \equiv \mathbf{0}$, and hence, every equilibrium solution arises as a solution to the system of algebraic equations

$$\mathbf{F}(\mathbf{u}^\star) = \mathbf{0} \tag{10.6}$$

prescribed by the vanishing of the right hand side of the system (10.5).

Example 10.1. Although a population of people, animals, or bacteria consists of individuals, the aggregate behavior can often be effectively modeled by a dynamical system that involves continuously varying variables. As first proposed by the English economist Thomas Malthus in 1798, the population of a species grows, roughly, in proportion to its size. Thus, the number of individuals $N(t)$ at time t satisfies a first order differential equation of the form

$$\frac{dN}{dt} = \rho N, \tag{10.7}$$

where the proportionality factor $\rho = \beta - \delta$ measures the rate of growth, namely the difference between the birth rate $\beta \geq 0$ and the death rate $\delta \geq 0$. Thus, if births exceed deaths, $\rho > 0$, and the population increases, whereas if $\rho < 0$, more individuals are dying and the population shrinks.

In the very simplest model, the growth rate ρ is assumed to be independent of the population size, and (10.7) reduces to the simple linear ordinary differential equation. The solutions satisfy the Malthusian exponential growth law $N(t) = N_0 e^{\rho t}$, where $N_0 = N(0)$ is the initial population size. Thus, if $\rho > 0$, the population grows without limit, while if

$\rho < 0$, the population dies out, so $N(t) \to 0$ as $t \to \infty$, at an exponentially fast rate. The Malthusian population model provides a reasonably accurate description of the behavior of an isolated population in an environment with unlimited resources.

In a more realistic scenario, the growth rate will depend upon the size of the population as well as external environmental factors. For example, in the presence of limited resources, relatively small populations will increase, whereas an excessively large population will have insufficient resources to survive, and so its growth rate will be negative. In other words, the growth rate $\rho(N) > 0$ when $N < N^\star$, while $\rho(N) < 0$ when $N > N^\star$, where the *carrying capacity* $N^\star > 0$ depends upon the resource availability. The simplest class of functions that satifies these two inequalities are of the form $\rho(N) = \lambda(N^\star - N)$, where $\lambda > 0$ is a positive constant. This leads us to the nonlinear population model

$$\frac{dN}{dt} = \lambda\, N\, (N^\star - N). \tag{10.8}$$

In deriving this model, we assumed that the environment is not changing over time; a dynamical environment would require a more complicated non-autonomous differential equation.

Before analyzing the solutions to the nonlinear population model, let us make a preliminary change of variables, and set $u(t) = N(t)/N^\star$, so that u represents the size of the population in proportion to the *carrying capacity* $N^\star$. A straightforward computation shows that $u(t)$ satisfies the so-called *logistic differential equation*

$$\frac{du}{dt} = \lambda\, u\,(1-u), \qquad\qquad u(0) = u_0, \tag{10.9}$$

where we assign the initial time to be $t_0 = 0$ for simplicity. The logistic differential equation can be viewed as the continuous counterpart of the logistic map (2.22). However, unlike its discrete namesake, the logistic differential equation is quite sedate, and its solutions easily understood.

First, there are two equilibrium solutions: $u(t) \equiv 0$ and $u(t) \equiv 1$, obtained by setting the right hand side of the equation equal to zero. The first represents a nonexistent population with no individuals and hence no reproduction. The second equilibrium solution corresponds to a static population $N(t) \equiv N^\star$ that is at the ideal size for the environment, so deaths exactly balance births. In all other situations, the population size will vary over time.

To integrate the logistic differential equation, we proceed as above, first writing it in the separated form

$$\frac{du}{u(1-u)} = \lambda\, dt.$$

Integrating both sides, and using partial fractions,

$$\lambda\, t + k = \int \frac{du}{u(1-u)} = \int \left[\, \frac{1}{u} + \frac{1}{1-u} \,\right] du = \log \left| \, \frac{u}{1-u} \, \right| ,$$

where k is a constant of integration. Therefore

$$\frac{u}{1-u} = c\, e^{\lambda\, t}, \qquad \text{where} \qquad c = \pm\, e^{k}.$$

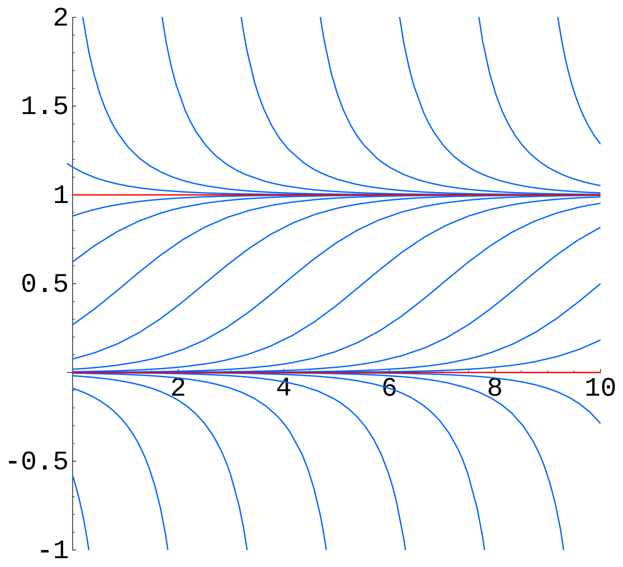


Figure 10.1. Solutions to $u' = u(1-u)$.

Solving for u, we deduce the solution

$$u(t) = \frac{c\,e^{\lambda t}}{1 + c\,e^{\lambda t}}\,. \tag{10.10}$$

The constant of integration is fixed by the initial condition. Solving the algebraic equation

$$u_0 = u(0) = \frac{c}{1+c} \qquad \text{yields} \qquad c = \frac{u_0}{1-u_0}\,.$$

Substituting the result back into the solution formula (10.10) and simplifying, we find

$$u(t) = \frac{u_0\,e^{\lambda t}}{1 - u_0 + u_0\,e^{\lambda t}}\,. \tag{10.11}$$

The resulting solutions are illustrated in Figure 10.1. Interestingly, while the equilibrium solutions are not covered by the integration method, they reappear in the final solution formula, corresponding to initial data $u_0 = 0$ and $u_0 = 1$ respectively. However, this is a lucky accident, and cannot be anticipated in more complicated situations.

When using the logistic equation to model population dynamics, the initial data is assumed to be positive, $u_0 > 0$. As time $t \to \infty$, the solution (10.11) tends to the equilibrium value $u(t) \to 1$ — which corresponds to $N(t) \to N^\star$ approaching the carrying capacity in the original population model. For small initial values $u_0 \ll 1$ the solution initially grows at an exponential rate λ, corresponding to a population with unlimited resources. However, as the population increases, the gradual lack of resources tends to slow down the growth rate, and eventually the population saturates at the equilibrium value. On the other hand, if $u_0 > 1$, the population is too large to be sustained by the available resources, and so dies off until it reaches the same saturation value. If $u_0 = 0$, then the solution remains at equilibrium $u(t) \equiv 0$. Finally, when $u_0 < 0$, the solution only exists for a finite amount of time, with

$$u(t) \longrightarrow -\infty \qquad \text{as} \qquad t \longrightarrow t^\star = \frac{1}{\lambda} \log\left(1 - \frac{1}{u_0}\right).$$

Of course, this final case does appear in the physical world, since we cannot have a negative population!

Example 10.2. A *predator-prey system* is a simplified ecological model of two species: the predators which feed on the prey. For example, the predators might be lions roaming the Serengeti and the prey zebra. We let $u(t)$ represent the number of prey, and $v(t)$ the number of predators at time t. Both species obey a population growth model of the form (10.7), and so the dynamical equations can be written as

$$\frac{du}{dt} = \rho u, \qquad \frac{dv}{dt} = \sigma v,$$

where the growth rates ρ, σ may depend upon the other species. The more prey, i.e., the larger u is, the faster the predators reproduce, while a lack of prey will cause them to die off. On the other hand, the more predators, the faster the prey are consumed and the slower their net rate of growth.

If we assume that the environment has unlimited resources for the prey, which, barring drought, is probably valid in the case of the zebras, then the simplest model that incorporates these assumptions is the *Lotka–Volterra system*

$$\frac{du}{dt} = \alpha u - \delta u v, \qquad \frac{dv}{dt} = -\beta v + \gamma u v, \tag{10.12}$$

corresponding to growth rates $\rho = \alpha - \delta v$, $\sigma = -\beta + \gamma u$. The parameters $\alpha, \beta, \gamma, \delta > 0$ are all positive, and their precise values will depend upon the species involved and how they interact, as indicated by field data, combined with, perhaps, educated guesses. In particular, α represents the unrestrained growth rate of the prey in the absence of predators, while $-\beta$ represents the rate that the predators die off in the absence of their prey. The nonlinear terms model the interaction of the two species: the rate of increase in the predators is proportional to the number of available prey, while the rate of decrese in the prey is proportional to the number of predators. The initial conditions $u(t_0) = u_0$, $v(t_0) = v_0$ represent the initial populations of the two species.

We will discuss the integration of the Lotka–Volterra system (10.12) below. Here, let us content ourselves with determining the possible equilibria. Setting the right hand sides of the system to zero leads to the nonlinear algebraic system

$$0 = \alpha u - \delta u v = u(\alpha - \delta v), \qquad 0 = -\beta v + \gamma u v = v(-\beta + \gamma u).$$

Thus, there are two distinct equilibria, namely

$$u_1^\star = v_1^\star = 0, \qquad u_2^\star = \beta/\gamma, \quad v_2^\star = \alpha/\delta.$$

The first is the uninteresting (or, rather catastropic) situation where there are no animals — no predators and no prey. The second is a nontrivial solution in which both populations maintain a steady value, for which the birth rate of the prey is precisely sufficient to continuously feed the predators. Is this a feasible solution? Or, to state the question more mathematically, is this a stable equilibrium? We shall develop the tools to answer this question below.

Higher Order Systems

A wide variety of physical systems are modeled by nonlinear systems of differential equations depending upon second and, occasionally, even higher order derivatives of the unknowns. But there is an easy device that will reduce any higher order ordinary differential equation or system to an equivalent first order system. "Equivalent" means that each solution to the first order system uniquely corresponds to a solution to the higher order equation and vice versa. The upshot is that, for all practical purposes, one only needs to analyze first order systems. Moreover, the vast majority of numerical solution algorithms are designed for first order systems, and so to numerically integrate a higher order equation, one must place it into an equivalent first order form.

We have already encountered the main idea in our discussion of the phase plane approach to second order scalar equations

$$\frac{d^2u}{dt^2} = F\left(t, u, \frac{du}{dt}\right). \tag{10.13}$$

We introduce a new dependent variable $v = \dfrac{du}{dt}$. Since $\dfrac{dv}{dt} = \dfrac{d^2u}{dt^2}$, the functions u, v satisfy the equivalent first order system

$$\frac{du}{dt} = v, \qquad \frac{dv}{dt} = F(t, u, v). \tag{10.14}$$

Conversely, it is easy to check that if $\mathbf{u}(t) = \left(\, u(t), v(t)\,\right)^T$ is any solution to the first order system, then its first component $u(t)$ defines a solution to the scalar equation, which establishes their equivalence. The basic initial conditions $u(t_0) = u_0$, $v(t_0) = v_0$, for the first order system translate into a pair of initial conditions $u(t_0) = u_0$, $\dot{u}(t_0) = v_0$, specifying the value of the solution and its first order derivative for the second order equation.

Similarly, given a third order equation

$$\frac{d^3u}{dt^3} = F\left(t, u, \frac{du}{dt}, \frac{d^2u}{dt^2}\right),$$

we set

$$v = \frac{du}{dt}, \qquad w = \frac{dv}{dt} = \frac{d^2u}{dt^2}.$$

The variables u, v, w satisfy the equivalent first order system

$$\frac{du}{dt} = v, \qquad \frac{dv}{dt} = w, \qquad \frac{dw}{dt} = F(t, u, v, w).$$

The general technique should now be clear.

Example 10.3. The forced *van der Pol equation*

$$\frac{d^2u}{dt^2} + (u^2 - 1)\,\frac{du}{dt} + u = f(t) \tag{10.15}$$

arises in the modeling of an electrical circuit with a triode whose resistance changes with the current. It also arises in certain chemical reactions and wind-induced motions of structures. To convert the van der Pol equation into an equivalent first order system, we set $v = du/dt$, whence

$$\frac{du}{dt} = v, \qquad\qquad \frac{dv}{dt} = f(t) - (u^2 - 1)\,v - u, \tag{10.16}$$

is the equivalent phase plane system.

Example 10.4. The Newtonian equations for a mass m moving in a potential force field are a second order system of the form

$$m\,\frac{d^2\mathbf{u}}{dt^2} = -\nabla F(\mathbf{u})$$

in which $\mathbf{u}(t) = (\,u(t), v(t), w(t)\,)^T$ represents the position of the mass and $F(\mathbf{u}) = F(u, v, w)$ the potential function. In components,

$$m\,\frac{d^2u}{dt^2} = -\,\frac{\partial F}{\partial u}\,, \qquad m\,\frac{d^2v}{dt^2} = -\,\frac{\partial F}{\partial v}\,, \qquad m\,\frac{d^2w}{dt^2} = -\,\frac{\partial F}{\partial w}\,. \tag{10.17}$$

For example, a planet moving in the sun's gravitational field satisfies the Newtonian system for the gravitational potential

$$F(\mathbf{u}) = -\,\frac{\alpha}{\|\,\mathbf{u}\,\|} = -\,\frac{\alpha}{\sqrt{u^2 + v^2 + w^2}}\,, \tag{10.18}$$

where α depends on the masses and the universal gravitational constant. (This simplified model ignores all interplanetary forces.) Thus, the mass' motion in such a gravitational force field follows the solution to the second order Newtonian system

$$m\,\frac{d^2\mathbf{u}}{dt^2} = -\,\nabla F(\mathbf{u}) = -\,\frac{\alpha\,\mathbf{u}}{\|\,\mathbf{u}\,\|^3} = \frac{\alpha}{(u^2 + v^2 + w^2)^{3/2}}\begin{pmatrix} u \\ v \\ w \end{pmatrix}.$$

The same system of ordinary differential equations describes the motion of a charged particle in a Coulomb electric force field, where the sign of α is positive for attracting opposite charges, and negative for repelling like charges.

To convert the second order Newton equations into a first order system, we set $\mathbf{v} = \dot{\mathbf{u}}$ to be the mass' velocity vector, with components

$$p = \frac{du}{dt}\,, \qquad q = \frac{dv}{dt}\,, \qquad r = \frac{dw}{dt}\,,$$

and so

$$\frac{du}{dt} = p, \qquad \frac{dv}{dt} = q, \qquad \frac{dw}{dt} = r, \tag{10.19}$$
$$\frac{dp}{dt} = -\frac{1}{m}\,\frac{\partial F}{\partial u}\,(u,v,w), \qquad \frac{dq}{dt} = -\frac{1}{m}\,\frac{\partial F}{\partial v}\,(u,v,w), \qquad \frac{dr}{dt} = -\frac{1}{m}\,\frac{\partial F}{\partial w}\,(u,v,w).$$

One of Newton's greatest acheivements was to solve this system in the case of the central gravitational potential (10.18), and thereby confirm the validity of Kepler's laws of planetary motion.

10.2. Existence, Uniqueness, and Continuous Dependence.

It goes without saying that there is no general analytical method that will solve all differential equations. Indeed, even relatively simple first order, scalar, non-autonomous ordinary differential equations cannot be solved in closed form. For example, the solution to the particular *Riccati equation*

$$\frac{du}{dt} = u^2 + t \tag{10.20}$$

cannot be written in terms of elementary functions, although there is a solution formula that relies on Airy functions. The *Abel equation*

$$\frac{du}{dt} = u^3 + t \tag{10.21}$$

fares even worse, since its general solution cannot be written in terms of even standard special functions — although power series solutions can be tediously ground out term by term. Understanding when a given differential equation can be solved in terms of elementary functions or known special functions is an active area of contemporary research, [**6**]. In this vein, we cannot resist mentioning that the most important class of exact solution techniques for differential equations are those based on symmetry. An introduction can be found in the author's graduate level monograph [**37**]; see also [**8**, **26**].

Existence

Before worrying about how to solve a differential equation, either analytically, qualitatively, or numerically, it behooves us to try to resolve the core mathematical issues of existence and uniqueness. First, does a solution exist? If, not, it makes no sense trying to find one. Second, is the solution uniquely determined? Otherwise, the differential equation probably has scant relevance for physical applications since we cannot use it as a predictive tool. Since differential equations inevitably have lots of solutions, the only way in which we can deduce uniqueness is by imposing suitable initial (or boundary) conditions.

Unlike partial differential equations, which must be treated on a case-by-case basis, there are complete general answers to both the existence and uniqueness questions for initial value problems for systems of ordinary differential equations. (Boundary value problems are more subtle.) While obviously important, we will not take the time to present the proofs of these fundamental results, which can be found in most advanced textbooks on the subject, including [**4**, **22**, **25**, **27**].

Let us begin by stating the Fundamental Existence Theorem for initial value problems associated with first order systems of ordinary differential equations.

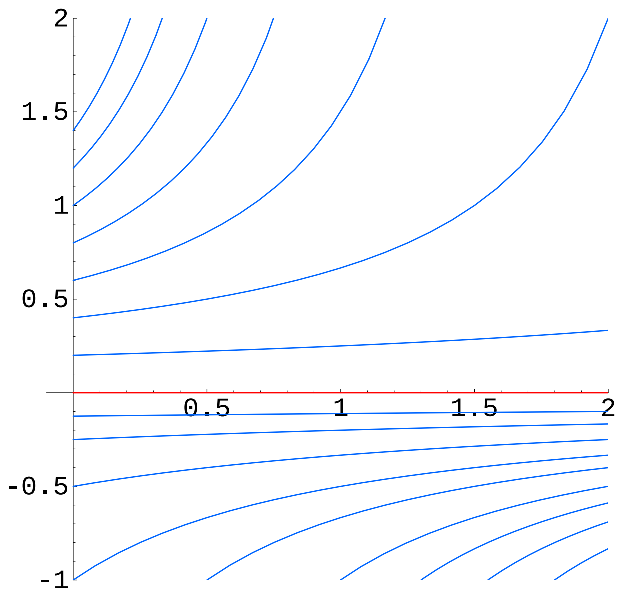


Figure 10.2. Solutions to $\dot{u} = u^2$.

Theorem 10.5. *Let $\mathbf{F}(t, \mathbf{u})$ be a continuous function. Then the initial value problem*[†]

$$\frac{d\mathbf{u}}{dt} = \mathbf{F}(t, \mathbf{u}), \qquad\qquad \mathbf{u}(t_0) = \mathbf{a}, \tag{10.22}$$

admits a solution $\mathbf{u} = \mathbf{f}(t)$ that is, at least, defined for nearby times, i.e., when $|\, t - t_0 \,| < \delta$ for some $\delta > 0$.

Theorem 10.5 guarantees that the solution to the initial value problem exists — at least for times sufficiently close to the initial instant t_0. This may be the most that can be said, although in many cases the maximal interval $\alpha < t < \beta$ of existence of the solution might be much larger — possibly infinite, $-\infty < t < \infty$, resulting in a *global solution*. The interval of existence of a solution typically depends upon both the equation and the particular initial data.

Example 10.6. Consider the autonomous initial value problem

$$\frac{du}{dt} = u^2, \qquad\qquad u(t_0) = u_0. \tag{10.23}$$

To solve the differential equation, we rewrite it in the separated form

$$\frac{du}{u^2} = dt, \qquad \text{and then integrate both sides:} \qquad -\frac{1}{u} = \int \frac{du}{u^2} = t + k.$$

[†] If $\mathbf{F}(t, \mathbf{u})$ is only defined on a subdomain $\Omega \subset \mathbb{R}^{n+1}$, then we must assume that the point $(t_0, \mathbf{a}) \in \Omega$ specifying the initial conditions belongs to its domain of definition.

Solving the resulting algebraic equation for u, we deduce the solution formula

$$u = -\frac{1}{t+k} \,. \tag{10.24}$$

To specify the integration constant k, we evaluate u at the initial time t_0; this implies

$$u_0 = -\frac{1}{t_0+k}\,, \qquad \text{so that} \qquad k = -\frac{1}{u_0} - t_0.$$

Therefore, the solution to the initial value problem is

$$u = \frac{u_0}{1-u_0(t-t_0)} \,. \tag{10.25}$$

Figure 10.2 shows the graphs of some typical solutions.

As t approaches the critical value $t^\star = t_0 + 1/u_0$ from below, the solution "blows up", meaning $u(t) \to \infty$ as $t \to t^\star$. The blow-up time $t^\star$ depends upon the initial data — the larger $u_0 > 0$ is, the sooner the solution goes off to infinity. If the initial data is negative, $u_0 < 0$, the solution is well-defined for all $t > t_0$, but has a singularity in the past, at $t^\star = t_0 + 1/u_0 < t_0$. The only solution that exists for all positive and negative time is the constant solution $u(t) \equiv 0$, corresponding to the initial condition $u_0 = 0$.

Thus, even though its right hand side is defined everywhere, the solutions to the scalar initial value problem (10.23) only exist up until time $1/u_0$, and so, the larger the initial data, the shorter the time of existence. In this example, the only global solution is the equilibrium solution $u(t) \equiv 0$. It is worth noting that this short-term existence phenomenon does not appear in the linear regime, where, barring singularities in the equation itself, solutions to a linear ordinary differential equation are guaranteed to exist for all time.

In practice, one always extends a solutions to its maximal interval of existence. The Existence Theorem 10.5 implies that there are only two possible ways in whcih a solution cannot be extended beyond a time $t^\star$: Either

(*i*) the solution becomes unbounded: $\| \mathbf{u}(t) \| \to \infty$ as $t \to t^\star$, or

(*ii*) if the right hand side $F(t, \mathbf{u})$ is only defined on a subset $\Omega \subset \mathbb{R}^{n+1}$, then the solution $\mathbf{u}(t)$ reaches the boundary $\partial\Omega$ as $t \to t^\star$.

If neither occurs in finite time, then the solution is necessarily global. In other words, a solution to an ordinary differential equation cannot suddenly vanish into thin air.

Remark: The existence theorem can be readily adapted to any higher order system of ordinary differential equations through the method of converting it into an equivalent first order system by introducing additional variables. The appropriate initial conditions guaranteeing existence are induced from those of the corresponding first order system, as in the second order example (10.13) discussed above.

Uniqueness and Smoothness

As important as existence is the question of uniqueness. Does the initial value problem have more than one solution? If so, then we cannot use the differential equation to predict

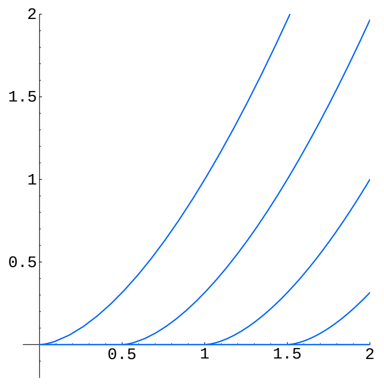


Figure 10.3. Solutions to the Differential Equation $\dot{u} = \frac{5}{3}\, u^{2/5}$.

the future behavior of the system from its current state. While continuity of the right hand side of the differential equation will guarantee that a solution exists, it is not quite sufficient to ensure uniqueness of the solution to the initial value problem. The difficulty can be appreciated by looking at an elementary example.

Example 10.7. Consider the nonlinear initial value problem

$$\frac{du}{dt} = \frac{5}{3}\, u^{2/5}, \qquad u(0) = 0. \tag{10.26}$$

Since the right hand side is a continuous function, Theorem 10.5 assures us of the existence of a solution — at least for t close to 0. This autonomous scalar equation can be easily solved by the usual method:

$$\int \frac{3}{5}\, \frac{du}{u^{2/5}} = u^{3/5} = t + c, \qquad \text{and so} \qquad u = (t+c)^{5/3}.$$

Substituting into the initial condition implies that $c = 0$, and hence $u(t) = t^{5/3}$ is a solution to the initial value problem.

On the other hand, since the right hand side of the differential equation vanishes at $u = 0$, the constant function $u(t) \equiv 0$ is an equilibrium solution to the differential equation. (Here is an example where the integration method fails to recover the equilibrium solution.) Moreover, the equilibrium solution has the same initial value $u(0) = 0$. Therefore, we have constructed two different solutions to the initial value problem (10.26). Uniqueness is *not* valid! Worse yet, there are, in fact, an *infinite* number of solutions to the initial value problem. For *any* $a > 0$, the function

$$u(t) = \begin{cases} 0, & 0 \le t \le a, \\ (t-a)^{5/3}, & t \ge a, \end{cases} \tag{10.27}$$

is differentiable everywhere, even at $t = a$. (Why?) Moreover, it satisfies both the differential equation and the initial condition, and hence defines a solution to the initial value problem. Several of these solutions are plotted in Figure 10.3.

Thus, to ensure uniqueness of solutions, we need to impose a more stringent condition, beyond mere continuity. The proof of the following basic uniqueness theorem can be found in the above references.

Theorem 10.8. *If $\mathbf{F}(t,\mathbf{u}) \in \mathrm{C}^1$ is continuously differentiable, then there exists one and only one solution*[†] *to the initial value problem* (10.22).

Thus, the difficulty with the differential equation (10.26) is that the function $F(u) = \frac{5}{3}\,u^{2/5}$, although continuous everywhere, is not differentiable at $u = 0$, and hence the Uniqueness Theorem 10.8 does not apply. On the other hand, $F(u)$ is continuously differentiable away from $u = 0$, and so any nonzero initial condition $u(t_0) = u_0 \neq 0$ will produce a unique solution — for as long as it remains away from the problematic value $u = 0$.

Blanket Hypothesis: From now on, all differential equations must satisfy the uniqueness criterion that their right hand side is continuously differentiable.

While continuous differentiability is sufficient to guarantee uniqueness of solutions, the smoother the right hand side of the system, the smoother the solutions. Specifically:

Theorem 10.9. *If $\mathbf{F} \in \mathrm{C}^n$ for $n \geq 1$, then any solution to the system $\dot{\mathbf{u}} = \mathbf{F}(t,\mathbf{u})$ is of class $\mathbf{u} \in \mathrm{C}^{n+1}$. If $\mathbf{F}(t,\mathbf{u})$ is an analytic function, then all solutions $\mathbf{u}(t)$ are analytic.*

One important consequence of uniqueness is that the solution trajectories of an autonomous system do not vary over time.

Proposition 10.10. *Consider an autonomous system $\dot{\mathbf{u}} = \mathbf{F}(\mathbf{u})$ whose right hand side $\mathbf{F} \in \mathrm{C}^1$. If $\mathbf{u}(t)$ is the solution to with initial condition $\mathbf{u}(t_0) = \mathbf{u}_0$, then the solution to the initial value problem $\widetilde{\mathbf{u}}(t_1) = \mathbf{u}_0$ is $\widetilde{\mathbf{u}}(t) = \mathbf{u}(t - t_1 + t_0)$.*

Note that the two solutions $\mathbf{u}(t)$ and $\widetilde{\mathbf{u}}(t)$ parametrize the *same* curve in $\mathbb{R}^n$, differing only by an overall "phase shift", $t_1 - t_0$, in their parametrizations. Thus, all solutions passing through the point $\mathbf{u}_0$ follow the same trajectory, irrespective of the time they arrive there. Indeed, not only is the trajectory the same, but the solutions have identical speeds at each point along the trajectory curve. For instance, if the right hand side of the system represents the velocity vector field of steady state fluid flow, Proposition 10.10 implies that the stream lines — the paths followed by the individual fluid particles — do not change in time, even though the fluid itself is in motion. This, indeed, is the meaning of the term "steady state" in fluid mechanics.

Continuous Dependence

In a real-world applications, initial conditions are almost never known exactly. Rather, experimental and physical errors will only allow us to say that their values are approximately equal to those in our mathematical model. Thus, to retain physical relevance, we need to be sure that small errors in our initial measurements do not induce a large change in the solution. A similar argument can be made for any physical parameters, e.g., masses,

[†] As noted earlier, we extend all solutions to their maximal interval of existence.

charges, spring stiffnesses, frictional coefficients, etc., that appear in the differential equation itself. A slight change in the parameters should not have a dramatic effect on the solution.

Mathematically, what we are after is a criterion of *continuous dependence* of solutions upon both initial data and parameters. Fortunately, the desired result holds without any additional assumptions, beyond requiring that the parameters appear continuously in the differential equation. We state both results in a single theorem.

Theorem 10.11. *Consider an initial value problem problem*

$$\frac{d\mathbf{u}}{dt} = \mathbf{F}(t, \mathbf{u}, \boldsymbol{\mu}), \qquad\qquad \mathbf{u}(t_0) = \mathbf{a}(\boldsymbol{\mu}), \tag{10.28}$$

in which the differential equation and/or the initial conditions depend continuously on one or more parameters $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_k)$. Then the unique[†] *solution $\mathbf{u}(t, \boldsymbol{\mu})$ depends continuously upon the parameters.*

Example 10.12. Let us look at a perturbed version

$$\frac{du}{dt} = \alpha\, u^2, \qquad\qquad u(0) = u_0 + \varepsilon,$$

of the initial value problem that we considered in Example 10.6. We regard ε as a small perturbation of our original initial data u_0, and α as a variable parameter in the equation. The solution is

$$u(t, \varepsilon) = \frac{u_0 + \varepsilon}{1 - \alpha\, (u_0 + \varepsilon)\, t}\,. \tag{10.29}$$

Note that, where defined, this is a continuous function of both parameters α, ε. Thus, a small change in the initial data, or in the equation, produces a small change in the solution — at least for times near the initial time.

Continuous dependence *does not* preclude nearby solutions from eventually becoming far apart. Indeed, the blow-up time $t^\star = 1/\big[\,\alpha\, (u_0 + \varepsilon)\,\big]$ for the solution (10.29) depends upon both the initial data and the parameter in the equation. Thus, as we approach the singularity, solutions that started out very close to each other will get arbitrarily far apart; see Figure 10.2 for an illustration.

An even simpler example is the linear model of exponential growth $\dot{u} = \alpha u$ when $\alpha > 0$. A very tiny change in the initial conditions has a negligible short term effect upon the solution, but over longer time intervals, the differences between the two solutions will be dramatic. Thus, the "sensitive dependence" of solutions on initial conditions already appears in very simple linear equations. For similar reasons, sontinuous dependence does *not* prevent solutions from exhibiting chaotic behavior. Further development of these ideas can be found in [**1**, **14**] and elsewhere.

[†] We continue to impose our blanket uniqueness hypothesis.

10.3. Numerical Methods.

Since we have no hope of solving the vast majority of differential equations in explicit, analytic form, the design of suitable numerical algorithms for accurately approximating solutions is essential. The ubiquity of differential equations throughout mathematics and its applications has driven the tremendous research effort devoted to numerical solution schemes, some dating back to the beginnings of the calculus. Nowadays, one has the luxury of choosing from a wide range of excellent software packages that provide reliable and accurate results for a broad range of systems, at least for solutions over moderately long time periods. However, all of these packages, and the underlying methods, have their limitations, and it is essential that one be able to to recognize when the software is working as advertised, and when it produces spurious results! Here is where the theory, particularly the classification of equilibria and their stability properties, as well as first integrals and Lyapunov functions, can play an essential role. Explicit solutions, when known, can also be used as test cases for tracking the reliability and accuracy of a chosen numerical scheme.

In this section, we survey the most basic numerical methods for solving initial value problems. For brevity, we shall only consider so-called single step schemes, culminating in the very popular and versatile fourth order Runge–Kutta Method. This should only serve as a extremely basic introduction to the subject, and many other important and useful methods can be found in more specialized texts, [**21**, **28**]. It goes without saying that some equations are more difficult to accurately approximate than others, and a variety of more specialized techniques are employed when confronted with a recalcitrant system. But all of the more advanced developments build on the basic schemes and ideas laid out in this section.

Euler's Method

The key issues confronting the numerical analyst of ordinary differential equations already appear in the simplest first order ordinary differential equation. Our goal is to calculate a decent approxiomation to the (unique) solution to the initial value problem

$$\frac{du}{dt} = F(t,u), \qquad\qquad u(t_0) = u_0. \tag{10.30}$$

To keep matters simple, we will focus our attention on the scalar case; however, all formulas and results written in a manner that can be readily adapted to first order systems — just replace the scalar functions $u(t)$ and $F(t,u)$ by vector-valued functions $\mathbf{u}$ and $\mathbf{F}(t,\mathbf{u})$ throughout. (The time t, of course, remains a scalar.) Higher order ordinary differential equations are inevitably handled by first converting them into an equivalent first order system, as discussed in Section 10.1, and then applying the numerical scheme.

The very simplest numerical solution method is named after Leonhard Euler — although Newton and his contemporaries were well aware of such a simple technique. Euler's Method is rarely used in practice because much more efficient and accurate techniques can be implemented with minimal additional work. Nevertheless, the method lies at the core of the entire subject, and must be thoroughly understood before progressing on to the more sophisticated algorithms that arise in real-world computations.

Starting at the initial time t_0, we introduce successive *mesh points* (or sample times)

$$t_0 < t_1 < t_2 < t_3 < \ \cdots \ ,$$

continuing on until we reach a desired final time $t_n = t^\star$. The mesh points should be fairly closely spaced. To keep the analysis as simple as possible, we will always use a uniform *step size*, and so

$$h = t_{k+1} - t_k > 0, \tag{10.31}$$

does not depend on k and is assumed to be relatively small. This assumption serves to simplify the analysis, and does not significantly affect the underlying ideas. For a uniform step size, the k^{th} mesh point is at $t_k = t_0 + k\,h$. More sophisticated *adaptive* methods, in which the step size is adjusted in order to maintain accuracy of the numerical solution, can be found in more specialized texts, e.g., [**21**, **28**]. Our numerical algorithm will recursively compute approximations $u_k \approx u(t_k)$, for $k = 0, 1, 2, 3, \ldots$, to the sampled values of the solution $u(t)$ at the chosen mesh points. Our goal is to make the *error* $E_k = u_k - u(t_k)$ in the approximation at each time t_k as small as possible. If required, the values of the solution $u(t)$ between mesh points may be computed by a subsequent interpolation procedure, e.g., the cubic splines of Section 13.3.

As you learned in first year calculus, the simplest approximation to a (continuously differentiable) function $u(t)$ is provided by its tangent line or first order Taylor polynomial. Thus, near the mesh point t_k

$$u(t) \;\approx\; u(t_k) + (t - t_k)\,\frac{du}{dt}(t_k) \;=\; u(t_k) + (t - t_k)\,F(t_k, u(t_k)),$$

in which we replace the derivative du/dt of the solution by the right hand side of the governing differential equation (10.30). In particular, the approximate value of the solution at the subsequent mesh point is

$$u(t_{k+1}) \;\approx\; u(t_k) + (t_{k+1} - t_k)\,F(t_k, u(t_k)). \tag{10.32}$$

This simple idea forms the basis of Euler's Method.

Since in practice we only know the approximation u_k to the value of $u(t_k)$ at the current mesh point, we are forced to replace $u(t_k)$ by its approximation u_k in the preceding formula. We thereby convert (10.32) into the iterative scheme

$$u_{k+1} = u_k + (t_{k+1} - t_k)\,F(t_k, u_k). \tag{10.33}$$

In particular, when based on a uniform step size (10.31), *Euler's Method* takes the simple form

$$u_{k+1} = u_k + h\,F(t_k, u_k). \tag{10.34}$$

As sketched in Figure 10.4, the method starts off approximating the solution reasonably well, but gradually loses accuracy as the errors accumulate.

Euler's Method is the simplest example of a *one-step* numerical scheme for integrating an ordinary differential equation. This refers to the fact that the succeeding approximation, $u_{k+1} \approx u(t_{k+1})$, depends only upon the current value, $u_k \approx u(t_k)$, which is one mesh point or "step" behind.

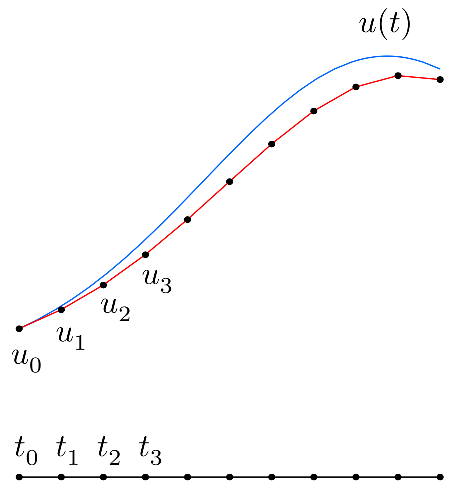


Figure 10.4. Euler's Method.

To begin to understand how Euler's Method works in practice, let us test it on a problem we know how to solve, since this will allow us to precisely monitor the resulting errors in our numerical approximation to the solution.

Example 10.13. The simplest "nontrivial" initial value problem is

$$\frac{du}{dt} = u, \qquad\qquad u(0) = 1,$$

whose solution is, of course, the exponential function $u(t) = e^t$. Since $F(t,u) = u$, Euler's Method (10.34) with a fixed step size $h > 0$ takes the form

$$u_{k+1} = u_k + h\,u_k = (1+h)\,u_k.$$

This is a linear iterative equation, and hence easy to solve:

$$u_k = (1+h)^k u_0 = (1+h)^k$$

is our proposed approximation to the solution $u(t_k) = e^{t_k}$ at the mesh point $t_k = k\,h$. Therefore, the Euler scheme to solve the differential equation, we are effectively approximating the exponential by a power function:

$$e^{t_k} = e^{k\,h} \;\approx\; (1+h)^k$$

When we use simply t to indicate the mesh time $t_k = k\,h$, we recover, in the limit, a well-known calculus formula:

$$e^t = \lim_{h\to 0}\,(1+h)^{t/h} = \lim_{k\to\infty}\left(1+\frac{t}{k}\right)^k. \tag{10.35}$$

A reader familiar with the computation of compound interest will recognize this particular approximation. As the time interval of compounding, h, gets smaller and smaller, the amount in the savings account approaches an exponential.

How good is the resulting approximation? The *error*

$$E(t_k) = E_k = u_k - e^{t_k}$$

measures the difference between the true solution and its numerical approximation at time $t = t_k = k\,h$. Let us tabulate the error at the particular times $t = 1, 2$ and 3 for various values of the step size h. The actual solution values are

$$e^1 = e = 2.718281828\ldots\,, \qquad e^2 = 7.389056096\ldots\,, \qquad e^3 = 20.085536912\ldots\,.$$

In this case, the numerical approximation always underestimates the true solution.

h	$E(1)$	$E(2)$	$E(3)$
.1	$-.125$	$-.662$	-2.636
.01	$-.0134$	$-.0730$	$-\ \ .297$
.001	$-.00135$	$-.00738$	$-\ \ .0301$
.0001	$-.000136$	$-.000739$	$-\ \ .00301$
.00001	$-.0000136$	$-.0000739$	$-\ \ .000301$

Some key observations:

- For a fixed step size h, the further we go from the initial point $t_0 = 0$, the larger the magnitude of the error.
- On the other hand, the smaller the step size, the smaller the error at a fixed value of t. The trade-off is that more steps, and hence more computational effort[†] is required to produce the numerical approximation. For instance, we need $k = 10$ steps of size $h = .1$, but $k = 1000$ steps of size $h = .001$ to compute an approximation to $u(t)$ at time $t = 1$.
- The error is more or less in proportion to the step size. Decreasing the step size by a factor of $\frac{1}{10}$ decreases the error by a similar amount, but simultaneously increases the amount of computation by a factor of 10.

The final observation indicates that the Euler Method is of *first order*, which means that the error depends *linearly*[‡] on the step size h. More specifically, at a fixed time t, the error is bounded by

$$|\,E(t)\,| = |\,u_k - u(t)\,| \leq C(t)\,h, \qquad \text{when} \qquad t = t_k = k\,h, \tag{10.36}$$

for some positive $C(t) > 0$ that depends upon the time, and the initial condition, but not on the step size.

[†] In this case, there happens to be an explicit formula for the numerical solution which can be used to bypass the iterations. However, in almost any other situation, one cannot compute the approximation u_k without having first determined the intermediate values $u_0, \ldots, u_{k-1}$.

[‡] See the discussion of the order of iterative methods in Section 2.1 for motivation.

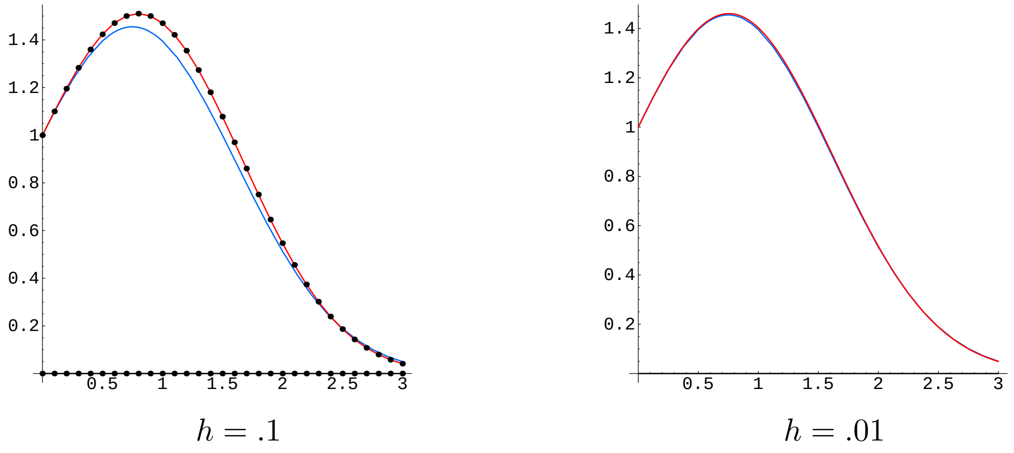


Figure 10.5. Euler's Method for $\dot{u} = \left(1 - \frac{4}{3}\,t\right) u$.

Example 10.14. The solution to the initial value problem

$$\frac{du}{dt} = \left(1 - \tfrac{4}{3}\,t\right) u, \qquad u(0) = 1, \tag{10.37}$$

is found by the method of separation of variables:

$$u(t) = \exp\left(t - \tfrac{2}{3}\,t^2\right). \tag{10.38}$$

Euler's Method leads to the iterative numerical scheme

$$u_{k+1} = u_k + h\left(1 - \tfrac{4}{3}\,t_k\right) u_k, \qquad u_0 = 1.$$

In Figure 10.5 we compare the graphs of the actual and numerical solutions for step sizes $h = .1$ and $.01$. In the former plot, we expliticly show the mesh points, but not in the latter, since they are too dense; moreover, the graphs of the numerical and true solutions are almost indistinguishable at this resolution.

The following table lists the numerical errors $E(t_k) = u_k - u(t_k)$ between the computed and actual solution values

$$u(1) = 1.395612425\ldots\,, \qquad u(2) = .513417119\ldots\,, \qquad u(3) = .049787068\ldots\,,$$

for several different step sizes:

h	$E(1)$	$E(2)$	$E(3)$
.1000	.07461761	.03357536	$-.00845267$
.0100	.00749258	.00324416	$-.00075619$
.0010	.00074947	.00032338	$-.00007477$
.0001	.00007495	.00003233	$-.00000747$

As in the previous example, each decrease in step size by a factor of 10 leads to one additional decimal digit of accuracy in the computed solution.

Taylor Methods

In general, the order of a numerical solution method governs both the accuracy of its approximations and the speed at which they converge to the true solution as the step size is decreased. Although the Euler Method is simple and easy to implement, it is only a first order scheme, and therefore of limited utility in serious computations. So, the goal is to devise simple numerical methods that enjoy a much higher order of accuracy.

Our derivation of the Euler Method was based on a first order Taylor approximation to the solution. So, an evident way to design a higher order method is to employ a higher order Taylor approximation. The Taylor series expansion for the solution $u(t)$ at the succeeding mesh point $t_{k+1} = t_k + h$ has the form

$$u(t_{k+1}) = u(t_k + h) = u(t_k) + h\,\frac{du}{dt}(t_k) + \frac{h^2}{2}\,\frac{d^2u}{dt^2}(t_k) + \frac{h^3}{6}\,\frac{d^3u}{dt^3}(t_k) + \;\cdots\;. \tag{10.39}$$

As we just saw, we can evaluate the first derivative term through use of the underlying differential equation:

$$\frac{du}{dt} = F(t,u). \tag{10.40}$$

The second derivative term can be found by differentiating the equation with respect to t. Invoking the chain rule[†],

$$\begin{aligned} \frac{d^2u}{dt^2} = \frac{d}{dt}\,\frac{du}{dt} = \frac{d}{dt}\,F(t,u(t)) &= \frac{\partial F}{\partial t}(t,u) + \frac{\partial F}{\partial u}(t,u)\,\frac{du}{dt} \\ &= \frac{\partial F}{\partial t}(t,u) + \frac{\partial F}{\partial u}(t,u)\;F(t,u) \equiv F^{(2)}(t,u). \end{aligned} \tag{10.41}$$

This operation is known as the *total derivative*, indicating that that we must treat the second variable u as a function of t when differentiating. Substituting (10.40–41) into (10.39) and truncating at order h^2 leads to the *Second Order Taylor Method*

$$\begin{aligned} u_{k+1} &= u_k + h\,F(t_k,u_k) + \frac{h^2}{2}\,F^{(2)}(t_k,u_k) \\ &= u_k + h\,F(t_k,u_k) + \frac{h^2}{2}\,\left(\frac{\partial F}{\partial t}(t_k,u_k) + \frac{\partial F}{\partial u}(t_k,u_k)\;F(t_k,u_k)\right), \end{aligned} \tag{10.42}$$

in which, as before, we replace the solution value $u(t_k)$ by its computed approximation u_k. The resulting method is of second order, meaning that the error function satisfies the quadratic error estimate

$$|\,E(t)\,| = |\,u_k - u(t)\,| \le C(t)\,h^2 \qquad \text{when} \qquad t = t_k = k\,h. \tag{10.43}$$

[†] We assume throughout that F has as many continuous derivatives as needed.

Example 10.15. Let us explicitly formulate the second order Taylor Method for the initial value problem (10.37). Here

$$\begin{aligned}
\frac{du}{dt} &= F(t,u) = \left(1 - \tfrac{4}{3}t\right)u, \\
\frac{d^2u}{dt^2} &= \frac{d}{dt}\,F(t,u) = -\tfrac{4}{3}u + \left(1 - \tfrac{4}{3}t\right)\frac{du}{dt} = -\tfrac{4}{3}u + \left(1 - \tfrac{4}{3}t\right)^2 u,
\end{aligned}$$

and so (10.42) becomes

$$u_{k+1} = u_k + h\left(1 - \tfrac{4}{3}t_k\right)u_k + \tfrac{1}{2}h^2\left[-\tfrac{4}{3}u_k + \left(1 - \tfrac{4}{3}t_k\right)^2 u_k\right], \qquad u_0 = 1.$$

The following table lists the errors between the values computed by the second order Taylor scheme and the actual solution values, as given in Example 10.14.

h	$E(1)$	$E(2)$	$E(3)$
.100	.00276995	−.00133328	.00027753
.010	.00002680	−.00001216	.00000252
.001	.00000027	−.00000012	.00000002

In accordance with the quadratic error estimate (10.43), a decrease in the step size by a factor of $\frac{1}{10}$ leads in an increase in accuracy of the solution by a factor $\frac{1}{100}$, i.e., an increase in 2 significant decimal places in the numerical approximation of the solution.

Higher order Taylor Methods can be readily established by including further terms in the expansion (10.39). However, they are rarely used in practice, for two reasons:

- Owing to their dependence upon the partial derivatives of $F(t,u)$, the right hand side of the differential equation needs to be rather smooth.
- Even worse, the explicit formulae become exceedingly complicated, even for relatively simple functions $F(t,u)$. Efficient evaluation of the multiplicity of terms in the Taylor approximation and avoidance of round off errors become significant concerns.

As a result, mathematicians soon abandoned the Taylor series approach, and began to look elsewhere for high order, efficient integration methods.

Error Analysis

Before pressing on, we need to engage in a more serious discussion of the error in a numerical scheme. A general *one-step* numerical method can be written in the form

$$u_{k+1} = G(h, t_k, u_k), \tag{10.44}$$

where G is a prescribed function of the current approximate solution value $u_k \approx u(t_k)$, the time t_k, and the step size $h = t_{k+1} - t_k$, which, for illustrative purposes, we assume to be fixed. (We leave the discussion of *multi-step methods*, in which G could also depend upon the earlier values $u_{k-1}, u_{k-2}, \ldots$, to more advanced texts, e.g., [**21**, **28**].)

In any numerical integration scheme there are, in general, three sources of error.

- The first is the *local error* committed in the current step of the algorithm. Even if we have managed to compute a completely accurate value of the solution $u_k = u(t_k)$ at time t_k, the numerical approximation scheme (10.44) is almost certainly not exact, and will therefore introduce an error into the next computed value $u_{k+1} \approx u(t_{k+1})$. Round-off errors, resulting from the finite precision of the computer arithmetic, will also contribute to the local error.
- The second is due to the error that is already present in the current approximation $u_k \approx u(t_k)$. The local errors tend to accumulate as we continue to run the iteration, and the net result is the *global error*, which is what we actually observe when compaing the numerical apporximation with the exact solution.
- Finally, if the initial condition $u_0 \approx u(t_0)$ is not computed accurately, this *initial error* will also make a contribution. For example, if $u(t_0) = \pi$, then we introduce some initial error by using a decimal approximation, say $\pi \approx 3.14159$.

The third error source will, for simplicity, be ignored in our discussion, i.e., we will assume $u_0 = u(t_0)$ is exact. Further, for simplicity we will assume that round-off errors do not play any significant role — although one must always keep them in mind when analyzing the computation. Since the global error is entirely due to the accumulation of successive local errors, we must first understand the local error in detail.

To measure the local error in going from t_k to t_{k+1}, we compare the exact solution value $u(t_{k+1})$ with its numerical approximation (10.44) under the assumption that the current computed value is correct: $u_k = u(t_k)$. Of course, in practice this is never the case, and so the local error is an artificial quantity. Be that as it may, in most circumstances the local error is (*a*) easy to estimate, and, (*b*) provides a reliable guide to the global accuracy of the numerical scheme. To estimate the local error, we assume that the step size h is small and approximate the solution $u(t)$ by its Taylor expansion[†]

$$\begin{aligned} u(t_{k+1}) &= u(t_k) + h\,\frac{du}{dt}(t_k) + \frac{h^2}{2}\,\frac{d^2u}{dt^2}(t_k) + \cdots \\ &= u_k + h\,F(t_k,u_k) + \frac{h^2}{2}\,F^{(2)}(t_k,u_k) + \cdots . \end{aligned} \tag{10.45}$$

In the second expression, we have employed (10.41) and its higher order analogs to evaluate the derivative terms, and then invoked our local accuracy assumption to replace $u(t_k)$ by u_k. On the other hand, a direct Taylor expansion, in h, of the numerical scheme produces

$$u_{k+1} = G(h,t_k,u_k) = G(0,t_k,u_k) + h\,\frac{\partial G}{\partial h}\,(0,t_k,u_k) + \frac{h^2}{2}\,\frac{\partial^2 G}{\partial h^2}\,(0,t_k,u_k) + \cdots . \tag{10.46}$$

The local error is obtained by comparing these two Taylor expansions.

Definition 10.16. A numerical integration method is of *order* n if the Taylor expansions (10.45, 46) of the exact and numerical solutions agree up to order h^n.

[†] In our analysis, we assume that the differential equation, and hence the solution, has sufficient smoothness to justify the relevant Taylor approximation.

For example, the Euler Method

$$u_{k+1} = G(h, t_k, u_k) = u_k + h F(t_k, u_k),$$

is already in the form of a Taylor expansion — that has no terms involving $h^2, h^3, \ldots$. Comparing with the exact expansion (10.45), we see that the constant and order h terms are the same, but the order h^2 terms differ (unless $F^{(2)} \equiv 0$). Thus, according to the definition, the Euler Method is a first order method. Similarly, the Taylor Method (10.42) is a second order method, because it was explicitly designed to match the constant, h and h^2 terms in the Taylor expansion of the solution. The general Taylor Method of order n sets $G(h, t_k, u_k)$ to be exactly the order n Taylor polynomial, differing from the full Taylor expansion at order h^{n+1}.

Under fairly general hypotheses, it can be proved that, if the numerical scheme has order n as measured by the local error, then the *global error* is bounded by a multiple of h^n. In other words, assuming no round-off or initial error, the computed value u_k and the solution at time t_k can be bounded by

$$|\, u_k - u(t_k) \,| \leq M\, h^n, \tag{10.47}$$

where the constant $M > 0$ may depend on the time t_k and the particular solution $u(t)$. The error bound (10.47) serves to justify our numerical observations. For a method of order n, decreasing the step size by a factor of $\frac{1}{10}$ will decrease the overall error by a factor of about 10^{-n}, and so, roughly speaking, we anticipate gaining an additional n digits of accuracy — at least up until the point that round-off errors begin to play a role. Readers interested in a more complete error analysis of numerical integration schemes should consult a specialized text, e.g., [**21**, **28**].

The bottom line is the higher its order, the more accurate the numerical scheme, and hence the larger the step size that can be used to produce the solution to a desired accuracy. However, this must be balanced with the fact that higher order methods inevitably require more computational effort at each step. If the total amount of computation has also decreased, then the high order method is to be preferred over a simpler, lower order method. Our goal now is to find another route to the design of higher order methods that avoids the complications inherent in a direct Taylor expansion.

An Equivalent Integral Equation

The secret to the design of higher order numerical algorithms is to replace the differential equation by an equivalent integral equation. By way of motivation, recall that, in general, differentiation is a badly behaved process; a reasonable function can have an unreasonable derivative. On the other hand, integration ameliorates; even quite nasty functions have relatively well-behaved integrals. For the same reason, accurate numerical integration is relatively painless, whereas numerical differentiation should be avoided unless necessary. While we have not dealt directly with integral equations in this text, the subject has been extensively developed by mathematicians, [**10**], and has many important physical applications.

Conversion of an initial value problem (10.30) to an integral equation is straightforward. We integrate both sides of the differential equation from the initial point t_0 to a variable time t. The Fundamental Theorem of Calculus is used to explicitly evaluate the left hand integral:

$$u(t) - u(t_0) = \int_{t_0}^{t} \dot{u}(s)\, ds = \int_{t_0}^{t} F(s, u(s))\, ds.$$

Rearranging terms, we arrive at the key result.

Lemma 10.17. *The solution $u(t)$ to the the integral equation*

$$u(t) = u(t_0) + \int_{t_0}^{t} F(s, u(s))\, ds \tag{10.48}$$

coincides with the solution to the initial value problem $\dfrac{du}{dt} = F(t,u)$, $u(t_0) = u_0$.

Proof: Our derivation already showed that the solution to the initial value problem satisfies the integral equation (10.48). Conversely, suppose that $u(t)$ solves the integral equation. Since $u(t_0) = u_0$ is constant, the Fundamental Theorem of Calculus tells us that the derivative of the right hand side of (10.48) is equal to the integrand, so $\dfrac{du}{dt} = F(t, u(t))$. Moreover, at $t = t_0$, the upper and lower limits of the integral coincide, and so it vanishes, whence $u(t) = u(t_0) = u_0$ has the correct initial conditions. *Q.E.D.*

Observe that, unlike the differential equation, the integral equation (10.48) requires no additional initial condition — it is automatically built into the equation. The proofs of the fundamental existence and uniqueness Theorems 10.5 and 10.8 for ordinary differential equations are, in fact, based on the integral equation reformulation of the initial value problem; see [**22**, **25**] for details.

The integral equation reformulation is equally valid for systems of first order ordinary differential equations. As noted above, $\mathbf{u}(t)$ and $\mathbf{F}(t, \mathbf{u}(t))$ become vector-valued functions. Integrating a vector-valued function is accomplished by integrating its individual components. Complete details are left to the exercises.

Implicit and Predictor–Corrector Methods

From this point onwards, we shall abandon the original initial value problem, and turn our attention to numerically solving the equivalent integral equation (10.48). Let us rewrite the integral equation, starting at the mesh point t_k instead of t_0, and integrating until time $t = t_{k+1}$. The result is the basic integral formula

$$u(t_{k+1}) = u(t_k) + \int_{t_k}^{t_{k+1}} F(s, u(s))\, ds \tag{10.49}$$

that (implicitly) computes the value of the solution at the subsequent mesh point. Comparing this formula with the Euler Method

$$u_{k+1} = u_k + h\, F(t_k, u_k), \qquad \text{where} \qquad h = t_{k+1} - t_k,$$

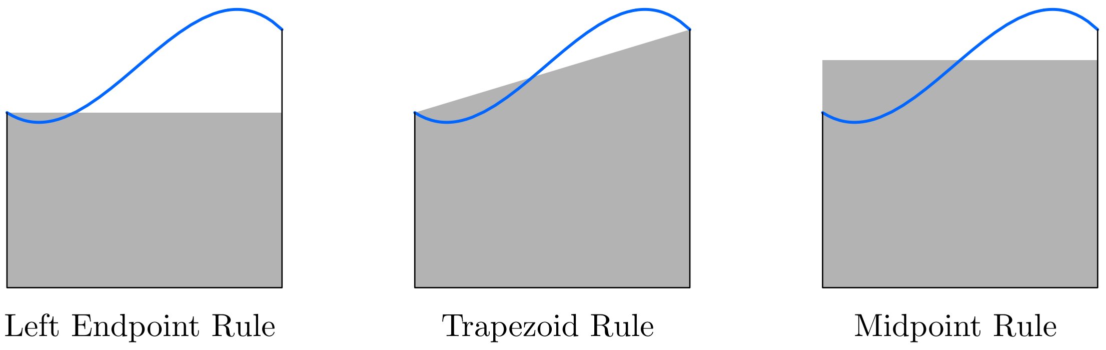


Figure 10.6. Numerical Integration Methods.

and assuming for the moment that $u_k = u(t_k)$ is exact, we discover that we are merely approximating the integral by

$$\int_{t_k}^{t_{k+1}} F(s,u(s))\,ds \;\approx\; h\,F(t_k,u(t_k)). \tag{10.50}$$

This is the Left Endpoint Rule for numerical integration — that approximates the area under the curve $g(t) = F(t,u(t))$ between $t_k \le t \le t_{k+1}$ by the area of a rectangle whose height $g(t_k) = F(t_k,u(t_k)) \approx F(t_k,u_k)$ is prescribed by the left-hand endpoint of the graph. As indicated in Figure 10.6, this is a reasonable, but not especially accurate method of numerical integration.

In first year calculus, you no doubt encountered much better methods of approximating the integral of a function. One of these is the *Trapezoid Rule*, which approximates the integral of the function $g(t)$ by the area of a trapezoid obtained by connecting the two points $(t_k, g(t_k))$ and $(t_{k+1}, g(t_{k+1}))$ on the graph of g by a straight line, as in the second Figure 10.6. Let us therefore try replacing (10.50) by the more accurate trapezoidal approximation

$$\int_{t_k}^{t_{k+1}} F(s,u(s))\,ds \;\approx\; \tfrac{1}{2}\,h\,\big[\,F(t_k,u(t_k)) + F(t_{k+1},u(t_{k+1}))\,\big]. \tag{10.51}$$

Substituting this approximation into the integral formula (10.49), and replacing the solution values $u(t_k), u(t_{k+1})$ by their numerical approximations, leads to the (hopefully) more accurate numerical scheme

$$u_{k+1} = u_k + \tfrac{1}{2}\,h\,\big[\,F(t_k,u_k) + F(t_{k+1},u_{k+1})\,\big], \tag{10.52}$$

known as the *Trapezoid Method*. It is an *implicit scheme*, since the updated value u_{k+1} appears on both sides of the equation, and hence is only defined implicitly.

Example 10.18. Consider the differential equation $\dot u = \big(1 - \frac{4}{3}\,t\big)\,u$ studied in Examples 10.14 and 10.15. The Trapezoid Method with a fixed step size h takes the form

$$u_{k+1} = u_k + \tfrac{1}{2}\,h\,\big[\,\big(1 - \tfrac{4}{3}\,t_k\big)\,u_k + \big(1 - \tfrac{4}{3}\,t_{k+1}\big)\,u_{k+1}\,\big].$$

In this case, we can explicit solve for the updated solution value, leading to the recursive formula

$$u_{k+1} = \frac{1 + \frac{1}{2}h\left(1 - \frac{4}{3}t_k\right)}{1 - \frac{1}{2}h\left(1 - \frac{4}{3}t_{k+1}\right)}\,u_k = \frac{1 + \frac{1}{2}h - \frac{2}{3}h\,t_k}{1 - \frac{1}{2}h + \frac{2}{3}h\,(t_k + h)}\,u_k. \tag{10.53}$$

Implementing this scheme for three different step sizes gives the following errors between the computed and true solutions at times $t = 1, 2, 3$.

h	$E(1)$	$E(2)$	$E(3)$
.100	−.00133315	.00060372	−.00012486
.010	−.00001335	.00000602	−.00000124
.001	−.00000013	.00000006	−.00000001

The numerical data indicates that the Trapezoid Method is of second order. For each reduction in step size by $\frac{1}{10}$, the accuracy in the solution increases by, roughly, a factor of $\frac{1}{100} = \frac{1}{10^2}$; that is, the numerical solution acquires two additional accurate decimal digits.

The main difficulty with the Trapezoid Method (and any other implicit scheme) is immediately apparent. The updated approximate value for the solution u_{k+1} appears on both sides of the equation (10.52). Only for very simple functions $F(t,u)$ can one expect to solve (10.52) explicitly for u_{k+1} in terms of the known quantities t_k, u_k and $t_{k+1} = t_k + h$. The alternative is to employ a numerical equation solver, such as the bisection algorithm or Newton's Method, to compute u_{k+1}. In the case of Newton's Method, one would use the current approximation u_k as a first guess for the new approximation u_{k+1} — as in the continuation method discussed in Example 2.20. The resulting scheme requires some work to program, but can be effective in certain situations.

An alternative, less involved strategy is based on the following far-reaching idea. We already know a half-way decent approximation to the solution value u_{k+1} — namely that provided by the more primitive Euler scheme

$$\widetilde{u}_{k+1} = u_k + h\,F(t_k, u_k). \tag{10.54}$$

Let's use this estimated value in place of u_{k+1} on the right hand side of the implicit equation (10.52). The result

$$\begin{aligned} u_{k+1} &= u_k + \tfrac{1}{2}h\left[\,F(t_k,u_k) + F(t_k + h, \widetilde{u}_{k+1})\,\right] \\ &= u_k + \tfrac{1}{2}h\left[\,F(t_k,u_k) + F\big(\,t_k + h, u_k + h\,F(t_k,u_k)\,\big)\,\right]. \end{aligned} \tag{10.55}$$

is known as the *Improved Euler Method*. It is a completely explicit scheme since there is no need to solve any equation to find the updated value u_{k+1}.

Example 10.19. For our favorite equation $\dot{u} = \left(1 - \frac{4}{3}t\right)u$, the Improved Euler Method begins with the Euler approximation

$$\widetilde{u}_{k+1} = u_k + h\left(1 - \tfrac{4}{3}t_k\right)u_k,$$

and then replaces it by the improved value

$$\begin{aligned} u_{k+1} &= u_k + \tfrac{1}{2}\,h\,\big[\,\big(1-\tfrac{4}{3}\,t_k\,\big)\,u_k + \big(1-\tfrac{4}{3}\,t_{k+1}\,\big)\,\widetilde{u}_{k+1}\,\big] \\ &= u_k + \tfrac{1}{2}\,h\,\big[\,\big(1-\tfrac{4}{3}\,t_k\,\big)\,u_k + \big(1-\tfrac{4}{3}\,(t_k+h)\,\big)\,\big(\,u_k + h\,\big(1-\tfrac{4}{3}\,t_k\,\big)\,u_k\,\big)\,\big] \\ &= \Big[\,\big(1-\tfrac{2}{3}\,h^2\,\big)\,\big[\,1+h\,\big(1-\tfrac{4}{3}\,t_k\,\big)\,\big] + \tfrac{1}{2}\,h^2\,\big(1-\tfrac{4}{3}\,t_k\,\big)^2\,\Big]\;u_k. \end{aligned}$$

Implementing this scheme leads to the following errors in the numerical solution at the indicated times. The Improved Euler Method performs comparably to the fully implicit scheme (10.53), and significantly better than the original Euler Method.

h	$E(1)$	$E(2)$	$E(3)$
.100	$-.00070230$	.00097842	.00147748
.010	$-.00000459$	.00001068	.00001264
.001	$-.00000004$	.00000011	.00000012

The Improved Euler Method is the simplest of a large family of so-called *predictor–corrector algorithms*. In general, one begins a relatively crude method — in this case the Euler Method — to *predict* a first approximation $\widetilde{u}_{k+1}$ to the desired solution value u_{k+1}. One then employs a more sophisticated, typically implicit, method to *correct* the original prediction, by replacing the required update u_{k+1} on the right hand side of the implicit scheme by the less accurate prediction $\widetilde{u}_{k+1}$. The resulting explicit, corrected value u_{k+1} will, provided the method has been designed with due care, be an improved approximation to the true solution.

The numerical data in Example 10.19 indicates that the Improved Euler Method is of second order since each reduction in step size by $\frac{1}{10}$ improves the solution accuracy by, roughly, a factor of $\frac{1}{100}$. To verify this prediction, we expand the right hand side of (10.55) in a Taylor series in h, and then compare, term by term, with the solution expansion (10.45). First,

$$F\big(\,t_k+h, u_k + h\,F(t_k,u_k)\,\big) = F + h\big(\,F_t + F\,F_u\,\big) + \tfrac{1}{2}\,h^2\big(\,F_{tt} + 2\,F\,F_{tu} + F^2\,F_{uu}\,\big) + \;\cdots\;,$$

where all the terms involving F and its partial derivatives on the right hand side are evaluated at t_k, u_k. Substituting into (10.55), we find

$$u_{k+1} = u_k + h\,F + \tfrac{1}{2}\,h^2\big(\,F_t + F\,F_u\,\big) + \tfrac{1}{4}\,h^2\big(\,F_{tt} + 2\,F\,F_{tu} + F^2\,F_{uu}\,\big) + \;\cdots\;. \qquad (10.56)$$

The two Taylor expansions (10.45) and (10.56) agree in their order $1, h$ and h^2 terms, but differ at order h^3. This confirms our experimental observation that the Improved Euler Method is of second order.

We can design a range of numerical solution schemes by implementing alternative numerical approximations to the basic integral equation (10.49). For example, the Midpoint

Rule approximates the integral of the function $g(t)$ by the area of the rectangle whose height is the value of the function at the midpoint:

$$\int_{t_k}^{t_{k+1}} g(s)\,ds \;\approx\; h\,g\left(t_k+\tfrac{1}{2}\,h\right), \qquad \text{where} \qquad h=t_{k+1}-t_k. \tag{10.57}$$

See Figure 10.6 for an illustration. The Midpoint Rule is known to have the same order of accuracy as the Trapezoid Rule, [**2**, **7**]. Substituting into (10.49) leads to the approximation

$$u_{k+1}=u_k+\int_{t_k}^{t_{k+1}} F(s,u(s))\,ds \;\approx\; u_k+h\,F\left(t_k+\tfrac{1}{2}\,h, u\left(t_k+\tfrac{1}{2}\,h\right)\right).$$

Of course, we don't know the value of the solution $u\left(t_k+\frac{1}{2}\,h\right)$ at the midpoint, but can predict it through a straightforward adaptation of the basic Euler approximation:

$$u\left(t_k+\tfrac{1}{2}\,h\right) \;\approx\; u_k+\tfrac{1}{2}\,h\,F(t_k,u_k).$$

The result is the *Midpoint Method*

$$u_{k+1}=u_k+h\,F\left(t_k+\tfrac{1}{2}\,h, u_k+\tfrac{1}{2}\,h\,F(t_k,u_k)\right). \tag{10.58}$$

A comparison of the terms in the Taylor expansions of (10.45), (10.58) reveals that the Midpoint Method is also of second order.

Runge–Kutta Methods

The Improved Euler and Midpoint Methods are the most elementary incarnations of a general class of numerical schemes for ordinary differential equations that were first systematically studied by the German mathematicians Carle Runge and Martin Kutta in the late nineteenth century. Runge–Kutta Methods are by far the most popular and powerful general-purpose numerical methods for integrating ordinary differential equations. While not appropriate in all possible situations, Runge–Kutta schemes are surprisingly robust, performing efficiently and accurately in a wide variety of problems. Barring significant complications, they are the method of choice in most basic applications. They comprise the engine that powers most computer software for solving general initial value problems for systems of ordinary differential equations.

The most general *Runge–Kutta Method* takes the form

$$u_{k+1}=u_k+h\sum_{i=1}^{m} c_i\,F(t_{i,k},u_{i,k}), \tag{10.59}$$

where m counts the number of *terms* in the method. Each $t_{i,k}$ denotes a point in the k^{th} mesh interval, so $t_k \le t_{i,k} \le t_{k+1}$. The second argument $u_{i,k} \approx u(t_{i,k})$ should be viewed as an approximation to the solution at the point $t_{i,k}$, and so is computed by a simpler Runge–Kutta scheme of the same general format. There is a lot of flexibility in the design of the method, through choosing the coefficients c_i, the times $t_{i,k}$, as well as the scheme (and all parameters therein) used to compute each of the intermediate approximations $u_{i,k}$. As always, the *order* of the method is fixed by the power of h to which the Taylor expansions of the numerical method (10.59) and the actual solution (10.45) agree. Clearly, the more terms we include in the Runge–Kutta formula (10.59), the more free parameters available to match terms in the solution's Taylor series, and so the higher the potential order of the method. Thus, the goal is to arrange the parameters so that the method has a high order of accuracy, while, simultaneously, avoiding unduly complicated, and hence computationally costly, formulae.

Both the Improved Euler and Midpoint Methods are instances of a family of two term Runge–Kutta Methods

$$\begin{aligned} u_{k+1} &= u_k + h\left[\, a\,F(t_k,u_k) + b\,F\big(\,t_{k,2},u_{k,2}\,\big)\,\right] \\ &= u_k + h\left[\, a\,F(t_k,u_k) + b\,F\big(\,t_k+\lambda h, u_k + \lambda\, h\, F(t_k,u_k)\,\big)\,\right], \end{aligned} \tag{10.60}$$

based on the current mesh point, so $t_{k,1} = t_k$, and one intermediate point $t_{k,2} = t_k + \lambda h$ with $0 \le \lambda \le 1$. The basic Euler Method is used to approximate the solution value

$$u_{k,2} = u_k + \lambda\, h\, F(t_k,u_k)$$

at $t_{k,2}$. The Improved Euler Method sets $a = b = \frac12$ and $\lambda = 1$, while the Midpoint Method corresponds to $a = 0$, $b = 1$, $\lambda = \frac12$. The range of possible values for a, b and λ is found by matching the Taylor expansion

$$\begin{aligned} u_{k+1} &= u_k + h\left[\, a\,F(t_k,u_k) + b\,F\big(\,t_k+\lambda h, u_k + \lambda\, h\, F(t_k,u_k)\,\big)\,\right] \\ &= u_k + h\,(a+b)\,F(t_k,u_k) + h^2\, b\,\lambda\,\left[\,\frac{\partial F}{\partial t}\,(t_k,u_k) + F(t_k,u_k)\,\frac{\partial F}{\partial u}\,(t_k,u_k)\right] + \;\cdots\;. \end{aligned}$$

(in powers of h) of the right hand side of (10.60) with the Taylor expansion (10.45) of the solution, namely

$$u(t_{k+1}) = u_k + h\,F(t_k,u_k) + \frac{h^2}{2}\,\left[\,F_t(t_k,u_k) + F(t_k,u_k)\,F_u(t_k,u_k)\,\right] + \;\cdots\;,$$

to as high an order as possible. First, the constant terms, u_k, are the same. For the order h and order h^2 terms to agree, we must have, respectively,

$$a + b = 1, \qquad\qquad b\,\lambda = \tfrac12 .$$

Therefore, setting

$$a = 1-\mu, \qquad b = \mu, \qquad \text{and} \qquad \lambda = \frac{1}{2\,\mu}\,, \qquad \text{where } \mu \text{ is arbitrary}^{\dagger},$$

leads to the following family of two term, second order Runge–Kutta Methods:

$$u_{k+1} = u_k + h\,\left[\,(1-\mu)\,F(t_k,u_k) + \mu\,F\left(\,t_k + \frac{h}{2\,\mu}, u_k + \frac{h}{2\,\mu}\,F(t_k,u_k)\right)\right]. \tag{10.61}$$

† Although we should restrict $\mu \ge \frac12$ in order that $0 \le \lambda \le 1$.

The case $\mu = \frac{1}{2}$ corresponds to the Improved Euler Method (10.55), while $\mu = 1$ yields the Midpoint Method (10.58). Unfortunately, none of these methods are able to match all of the third order terms in the Taylor expansion for the solution, and so we are left with a one-parameter family of two step Runge–Kutta Methods, all of second order, that include the Improved Euler and Midpoint schemes as particular instances. The methods with $\frac{1}{2} \le \mu \le 1$ all perform more or less comparably, and there is no special reason to prefer one over the other.

To construct a third order Runge–Kutta Method, we need to take at least $m \ge 3$ terms in (10.59). A rather intricate computation (best done with the aid of computer algebra) will produce a range of valid schemes; the results can be found in [**21**, **28**]. The algebraic manipulations are rather tedious, and we leave a complete discussion of the available options to a more advanced treatment. In practical applications, a particularly simple fourth order, four term formula has become the most used. The method, often abbreviated as RK4, takes the form

$$u_{k+1} = u_k + \frac{h}{6}\left[\, F(t_k, u_k) + 2\,F(t_{2,k}, u_{2,k}) + 2\,F(t_{3,k}, u_{3,k}) + F(t_{4,k}, u_{4,k})\,\right], \tag{10.62}$$

where the times and function values are successively computed according to the following procedure:

$$\begin{aligned} t_{2,k} &= t_k + \tfrac{1}{2}\,h, & u_{2,k} &= u_k + \tfrac{1}{2}\,h\,F(t_k, u_k),\\ t_{3,k} &= t_k + \tfrac{1}{2}\,h, & u_{3,k} &= u_k + \tfrac{1}{2}\,h\,F(t_{2,k}, u_{2,k}),\\ t_{4,k} &= t_k + h, & u_{4,k} &= u_k + h\,F(t_{3,k}, u_{3,k}). \end{aligned} \tag{10.63}$$

The four term RK4 scheme (10.62–63) is, in fact, a fourth order method. This is confirmed by demonstrating that the Taylor expansion of the right hand side of (10.62) in powers of h matches all of the terms in the Taylor series for the solution (10.45) up to and including those of order h^4, and hence the local error is of order h^5. This is not a computation for the faint-hearted — bring lots of paper and erasers, or, better yet, a good computer algebra package! The RK4 scheme is but one instance of a large family of fourth order, four term Runge–Kutta Methods, and by far the most popular owing to its relative simplicity.

Example 10.20. Application of the RK4 Method (10.62–63) to our favorite initial value problem (10.37) leads to the following errors at the indicated times:

h	$E(1)$	$E(2)$	$E(3)$
.100	-1.944×10^{-7}	1.086×10^{-6}	4.592×10^{-6}
.010	-1.508×10^{-11}	1.093×10^{-10}	3.851×10^{-10}
.001	-1.332×10^{-15}	-4.741×10^{-14}	1.932×10^{-14}

The accuracy is phenomenally good — much better than any of our earlier numerical schemes. Each decrease in the step size by a factor of $\frac{1}{10}$ results in about 4 additional decimal digits of accuracy in the computed solution, in complete accordance with its status as a fourth order method.

Actually, it is not entirely fair to compare the accuracy of the methods using the same step size. Each iteration of the RK4 Method requires four evaluations of the function $F(t,u)$, and hence takes the same computational effort as four Euler iterations, or, equivalently, two Improved Euler iterations. Thus, the more revealing comparison would be between RK4 at step size h, Euler at step size $\frac{1}{4}h$, and Improved Euler at step size $\frac{1}{2}h$, as these involve roughly the same amount of computational effort. The resulting errors $E(1)$ at time $t=1$ are listed in the following table.

Thus, even taking computational effort into account, the Runge–Kutta Method continues to outperform its rivals. At a step size of .1, it is almost as accurate as the Improved Euler Method with step size .0005, and hence 200 times as much computation, while the Euler Method would require a step size of approximately $.24 \times 10^{-6}$, and would be $4,000,000$ times as slow as Runge–Kutta! With a step size of .001, RK4 computes a solution value that is near the limits imposed by machine accuracy (in single precision arithmetic). The superb performance level and accuracy of the RK4 Method immediately explains its popularity for a broad range of applications.

h	Euler	Improved Euler	Runge–Kutta 4
.1	1.872×10^{-2}	-1.424×10^{-4}	-1.944×10^{-7}
.01	1.874×10^{-3}	-1.112×10^{-6}	-1.508×10^{-11}
.001	1.870×10^{-4}	-1.080×10^{-8}	-1.332×10^{-15}

Example 10.21. As noted earlier, by writing the function values as vectors $\mathbf{u}_k \approx \mathbf{u}(t_k)$, one can immediately use all of the preceding methods to integrate initial value problems for first order systems of ordinary differential equations $\dot{\mathbf{u}} = \mathbf{F}(\mathbf{u})$. Consider, by way of example, the Lotka–Volterra system

$$\frac{du}{dt} = 2u - uv, \qquad \frac{dv}{dt} = -9v + 3uv. \tag{10.64}$$

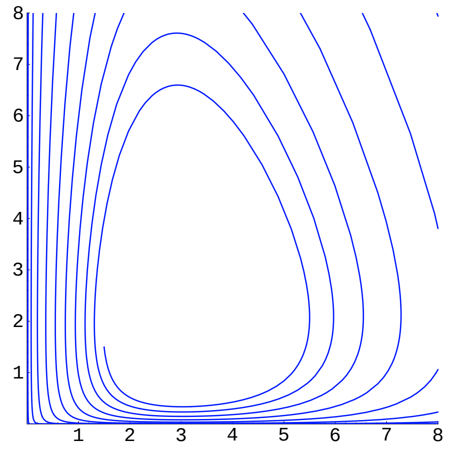
To find a numerical solution, we write $\mathbf{u} = (u, v)^T$ for the solution vector, while $\mathbf{F}(\mathbf{u}) = (2u - uv, -9v + 3uv)^T$ is the right hand side of the system. The Euler Method with step size h is given by

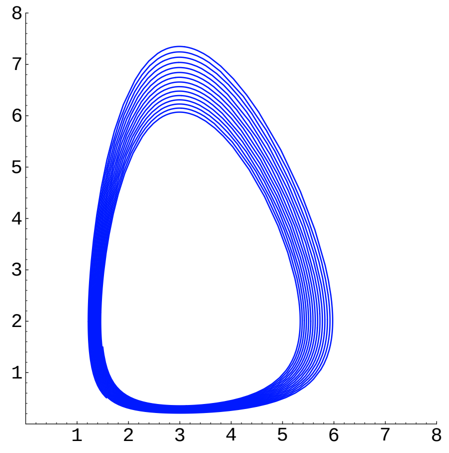
$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + h\,\mathbf{F}(\mathbf{u}^{(k)}),$$

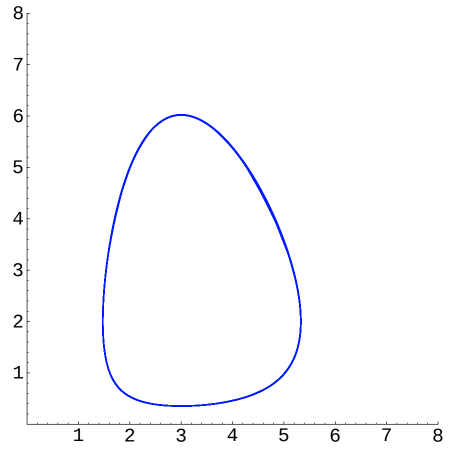
or, explicitly, as a first order nonlinear iterative system

$$u^{(k+1)} = u^{(k)} + h\,(2u^{(k)} - u^{(k)}v^{(k)}), \qquad v^{(k+1)} = v^{(k)} + h\,(-9v^{(k)} + 3u^{(k)}v^{(k)}).$$

The Improved Euler and Runge–Kutta schemes are implemented in a similar fashion. Phase portraits of the three numerical algorithms starting with initial conditions $u^{(0)} = v^{(0)} = 1.5$, and up to time $t = 25$ in the case of the Euler Method, and $t = 50$ for the other


Euler Method, $h = .01$


Euler Method, $h = .001$


Improved Euler Method, $h = .01$

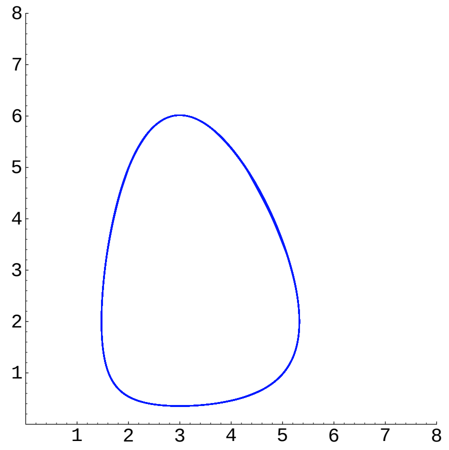

RK4 Method, $h = .01$

Figure 10.7. Numerical Solutions of Predator–Prey Model.

two, appear in Figure 10.7. In fact, the solution is supposed to travel periodically around a closed curve, which is the level set

$$I(u,v) = 9\log u - 3u + 2\log v - v = I(1.5, 1.5) = -1.53988$$

of the first integral. The Euler Method spirals away from the exact periodic solution, whereas the Improved Euler and RK4 Methods perform rather well. Since we do not have an analytic formula for the solution, we cannot measure the exact error in the methods. However, the known first integral is supposed to remain constant on the solution trajectories, and so one means of monitoring the accuracy of the solution is by the variation in the numerical values of $I(u^{(k)}, v^{(k)})$. These are graphed in Figure 10.8; the Improved Euler keeps the vlue within .0005, while for the RK4 solution, only the fifth decimal place in the value of the first integral experiences any change over the indicated time period. Of course, the ononger one continues to integrate, the more error will gradually creep into the

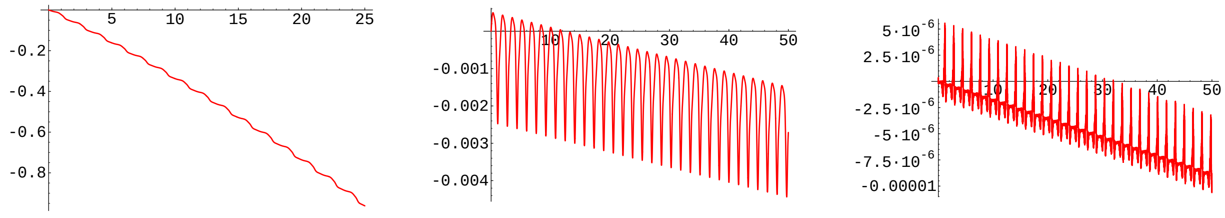


Figure 10.8. Numerical Evaluation of Lotka–Volterra First Integral.

numerical solution. Still, for most practical purposes, the RK4 solution is indistinguishable from the exact solution.

In practical implementations, it is important to monitor the accuracy of the numerical solution, so to gauge when to abandon an insufficiently precise computation. Since accuracy is dependent upon the step size h, one may try adjusting h so as stay within a preassigned error. *Adaptive methods*, allow one to change the step size during the course of the computation, in response to some estimation of the overall error. Insufficiently accurate numerical solutions would necessitate a suitable reduction in step size (or increase in the order of the scheme). On the other hand, if the solution is more accurate than the application requires, one could increase the step size so as to reduce the total amount of computational effort.

How might one decide when a method is giving inaccurate results, since one presumably does not know the true solution and so has nothing to directly test the numerical approximation against? One useful idea is to integrate the differential equation using two different numerical schemes, usually of different orders of accuracy, and then compare the results. If the two solution values are reasonably close, then one is usually safe in assuming that the methods are both giving accurate results, while in the event that they differ beyond some preassigned tolerance, then one needs to re-evaluate the step size. The required adjustment to the step size relies on a more detailed analysis of the error terms. Several well-studied methods are employed in practical situations; the most popular is the Runge–Kutta–Fehlberg Method, which combines a fourth and a fifth order Runge–Kutta scheme for error control. Details can be found in more advanced treatments of the subject, e.g., [**21**, **28**].

Stiff Differential Equations

While the fourth order Runge–Kutta Method with a sufficiently small step size will successfully integrate a broad range of differential equations — at least over not unduly long time intervals — it does occasionally experience unexpected difficulties. While we have not developed sufficiently sophisticated analytical tools to conduct a thorough analysis, it will be instructive to look at why a breakdown might occur in a simpler context.

Example 10.22. The elementary linear initial value problem

$$\frac{du}{dt} = -250\,u, \qquad u(0) = 1, \tag{10.65}$$

is an instructive and sobering example. The explicit solution is easily found; it is a very rapidly decreasing exponential: $u(t) = e^{-250\,t}$.

$$u(t) = e^{-250\,t} \qquad \text{with} \qquad u(1) \approx 2.69 \times 10^{-109}.$$

The following table gives the result of approximating the solution value $u(1) \approx 2.69 \times 10^{-109}$ at time $t = 1$ using three of our numerical integration schemes for various step sizes:

h	Euler	Improved Euler	RK4
.1	6.34×10^{13}	3.99×10^{24}	2.81×10^{41}
.01	4.07×10^{17}	1.22×10^{21}	1.53×10^{-19}
.001	1.15×10^{-125}	6.17×10^{-108}	2.69×10^{-109}

The results are not misprints! When the step size is .1, the computed solution values are perplexingly large, and appear to represent an exponentially growing solution — the complete opposite of the rapidly decaying true solution. Reducing the step size beyond a critical threshold suddenly transforms the numerical solution to an exponentially decaying function. Only the fourth order RK4 Method with step size $h = .001$ — and hence a total of $1,000$ steps — does a reasonable job at approximating the correct value of the solution at $t = 1$.

You may well ask, what on earth is going on? The solution couldn't be simpler — why is it so difficult to compute it? To understand the basic issue, let us analyze how the Euler Method handles such simple differential equations. Consider the initial value problem

$$\frac{du}{dt} = \lambda\, u, \qquad u(0) = 1, \tag{10.66}$$

which has an exponential solution

$$u(t) = e^{\lambda\, t}. \tag{10.67}$$

As in Example 10.13, the Euler Method with step size h relies on the iterative scheme

$$u_{k+1} = (1 + \lambda\, h)\, u_k, \qquad u_0 = 1,$$

with solution

$$u_k = (1 + \lambda\, h)^k. \tag{10.68}$$

If $\lambda > 0$, the exact solution (10.67) is exponentially growing. Since $1 + \lambda\, h > 1$, the numerical iterates are also growing, albeit at a somewhat slower rate. In this case, there is no inherent surprise with the numerical approximation procedure — in the short run it gives fairly accurate results, but eventually trails behind the exponentially growing solution.

On the other hand, if $\lambda < 0$, then the exact solution is exponentially decaying and positive. But now, if $\lambda\, h < -2$, then $1 + \lambda\, h < -1$, and the iterates (10.68) grow exponentially fast in magnitude, with alternating signs. In this case, the numerical solution is nowhere close to the true solution; this explains the previously observed pathological behavior. If $-1 < 1 + \lambda h < 0$, the numerical solutions decay in magnitude, but continue to alternate between positive and negative values. Thus, to correctly model the qualitative features of the solution and obtain a numerically respectable approximation, we need to choose the step size h so as to ensure that $0 < 1 + \lambda h$, and hence $h < -1/\lambda$ when $\lambda < 0$. For the value $\lambda = -250$ in the example, then, we must choose $h < \frac{1}{250} = .004$ in order that the Euler Method give a reasonable numerical answer. A similar, but more complicated analysis applies to any of the Runge–Kutta schemes.

Thus, the numerical methods for ordinary differential equations exhibit a form of conditional stability. Paradoxically, the larger negative λ is — and hence the faster the solution tends to a trivial zero equilibrium — the *more* difficult and expensive the numerical integration. The system (10.65) is the simplest example of what is known as a *stiff differential equation*. In general, an equation or system is stiff if it has one or more very rapidly decaying solutions. In the case of autonomous (constant coefficient) linear systems $\dot{\mathbf{u}} = A\,\mathbf{u}$, stiffness occurs whenever the coefficient matrix A has an eigenvalue with a large negative real part: $\operatorname{Re} \lambda \ll 0$, resulting in a very rapidly decaying eigensolution. It only takes one such eigensolution to render the equation stiff, and ruin the numerical computation of even the well behaved solutions! Curiously, the component of the actual solution corresponding to such large negative eigenvalues is almost irrelevant, as it becomes almost instanteously tiny. However, the presence of such an eigenvalue continues to render the numerical solution to the system very difficult, even to the point of exhausting any available computing resources. Stiff equations require more sophisticated numerical procedures to integrate, and we refer the reader to [**21**, **28**] for details.

AIMS Lecture Notes 2006

Peter J. Olver

11. Numerical Solution of the Heat and Wave Equations

In this part, we study numerical solution methodss for the two most important equations of one-dimensional continuum dynamics. The *heat equation* models the diffusion of thermal energy in a body; here, we treat the case of a one-dimensional bar. The *wave equation* describes vibrations and waves in continuous media, including sound waves, water waves, elastic waves, electromagnetic waves, and so on. For simplicity, we restrict our attention to the case of waves in a one-dimensional medium, e.g., a string, bar, or column of air.

We begin with a general discussion of finite difference formulae for numerically approximating derivatives of functions. The basic *finite difference scheme* is obtained by replacing the derivatives in the equation by the appropriate numerical differentiation formulae. However, there is no guarantee that the resulting numerical scheme will accurately approximate the true solution, and further analysis is required to elicit bona fide, convergent numerical algorithms. In dynamical problems, the finite difference schemes replace the partial differential equation by an iterative linear matrix system, and the analysis of convergence relies on the methods covered in Section 7.1.

We will only introduce the most basic algorithms, leaving more sophisticated variations and extensions to a more thorough treatment, which can be found in numerical analysis texts, e.g., [**5**, **7**, **28**].

11.1. Finite Differences.

In general, to approximate the derivative of a function at a point, say $f'(x)$ or $f''(x)$, one constructs a suitable combination of sampled function values at nearby points. The underlying formalism used to construct these approximation formulae is known as the *calculus of finite differences*. Its development has a long and influential history, dating back to Newton. The resulting *finite difference numerical methods* for solving differential equations have extremely broad applicability, and can, with proper care, be adapted to most problems that arise in mathematics and its many applications.

The simplest finite difference approximation is the ordinary *difference quotient*

$$\frac{u(x+h)-u(x)}{h} \approx u'(x), \tag{11.1}$$

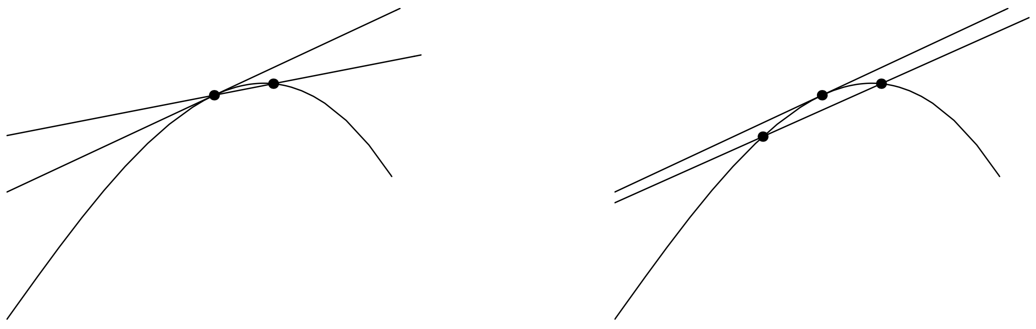


Figure 11.1. Finite Difference Approximations.

used to approximate the first derivative of the function $u(x)$. Indeed, if u is differentiable at x, then $u'(x)$ is, by definition, the limit, as $h \to 0$ of the finite difference quotients. Geometrically, the difference quotient equals the slope of the secant line through the two points $\bigl(x, u(x)\bigr)$ and $\bigl(x+h, u(x+h)\bigr)$ on the graph of the function. For small h, this should be a reasonably good approximation to the slope of the tangent line, $u'(x)$, as illustrated in the first picture in Figure 11.1.

How close an approximation is the difference quotient? To answer this question, we assume that $u(x)$ is at least twice continuously differentiable, and examine the first order Taylor expansion

$$u(x+h) = u(x) + u'(x)\,h + \tfrac{1}{2}\,u''(\xi)\,h^2. \tag{11.2}$$

We have used the Cauchy formula for the remainder term, in which ξ represents some point lying between x and $x+h$. The *error* or difference between the finite difference formula and the derivative being approximated is given by

$$\frac{u(x+h)-u(x)}{h} \;-\; u'(x) = \tfrac{1}{2}\,u''(\xi)\,h. \tag{11.3}$$

Since the error is proportional to h, we say that the finite difference quotient (11.3) is a *first order* approximation. When the precise formula for the error is not so important, we will write

$$u'(x) = \frac{u(x+h)-u(x)}{h} + \mathrm{O}(h). \tag{11.4}$$

The "big Oh" notation $\mathrm{O}(h)$ refers to a term that is proportional to h, or, more rigorously, bounded by a constant multiple of h as $h \to 0$.

Example 11.1. Let $u(x) = \sin x$. Let us try to approximate $u'(1) = \cos 1 = 0.5403023\ldots$ by computing finite difference quotients

$$\cos 1 \;\approx\; \frac{\sin(1+h) - \sin 1}{h}\,.$$

The result for different values of h is listed in the following table.

h	1	.1	.01	.001	.0001
approximation	0.067826	0.497364	0.536086	0.539881	0.540260
error	−0.472476	−0.042939	−0.004216	−0.000421	−0.000042

We observe that reducing the step size by a factor of $\frac{1}{10}$ reduces the size of the error by approximately the same factor. Thus, to obtain 10 decimal digits of accuracy, we anticipate needing a step size of about $h = 10^{-11}$. The fact that the error is more of less proportional to the step size confirms that we are dealing with a first order numerical approximation.

To approximate higher order derivatives, we need to evaluate the function at more than two points. In general, an approximation to the n^{th} order derivative $u^{(n)}(x)$ requires at least $n+1$ distinct sample points. For simplicity, we shall only use equally spaced points, leaving the general case to the exercises.

For example, let us try to approximate $u''(x)$ by sampling u at the particular points x, $x+h$ and $x-h$. Which combination of the function values $u(x-h), u(x), u(x+h)$ should be used? The answer to such a question can be found by consideration of the relevant Taylor expansions

$$\begin{aligned} u(x+h) &= u(x) + u'(x)\,h + u''(x)\,\frac{h^2}{2} + u'''(x)\,\frac{h^3}{6} + \mathrm{O}(h^4), \\ u(x-h) &= u(x) - u'(x)\,h + u''(x)\,\frac{h^2}{2} - u'''(x)\,\frac{h^3}{6} + \mathrm{O}(h^4), \end{aligned} \tag{11.5}$$

where the error terms are proportional to h^4. Adding the two formulae together gives

$$u(x+h) + u(x-h) = 2\,u(x) + u''(x)\,h^2 + \mathrm{O}(h^4).$$

Rearranging terms, we conclude that

$$u''(x) = \frac{u(x+h) - 2\,u(x) + u(x-h)}{h^2} + \mathrm{O}(h^2), \tag{11.6}$$

The result is known as the *centered finite difference approximation* to the second derivative of a function. Since the error is proportional to h^2, this is a second order approximation.

Example 11.2. Let $u(x) = e^{x^2}$, with $u''(x) = (4x^2+2)\,e^{x^2}$. Let us approximate $u''(1) = 6\,e = 16.30969097\ \ldots$ by using the finite difference quotient (11.6):

$$6\,e \approx \frac{e^{(1+h)^2} - 2\,e + e^{(1-h)^2}}{h^2}.$$

The results are listed in the following table.

h	1	.1	.01	.001	.0001
approximation	50.16158638	16.48289823	16.31141265	16.30970819	16.30969115
error	33.85189541	0.17320726	0.00172168	0.00001722	0.00000018

Each reduction in step size by a factor of $\frac{1}{10}$ reduces the size of the error by a factor of $\frac{1}{100}$ and results in a gain of two new decimal digits of accuracy, confirming that the finite difference approximation is of second order.

However, this prediction is not completely borne out in practice. If we take[†] $h = .00001$ then the formula produces the approximation 16.3097002570, with an error of 0.0000092863 — which is *less* accurate that the approximation with $h = .0001$. The problem is that round-off errors have now begun to affect the computation, and underscores the difficulty with numerical differentiation. Finite difference formulae involve dividing very small quantities, which can induce large numerical errors due to round-off. As a result, while they typically produce reasonably good approximations to the derivatives for moderately small step sizes, to achieve high accuracy, one must switch to a higher precision. In fact, a similar comment applied to the previous Example 11.1, and our expectations about the error were not, in fact, fully justified as you may have discovered if you tried an extremely small step size.

Another way to improve the order of accuracy of finite difference approximations is to employ more sample points. For instance, if the first order approximation (11.4) to the first derivative based on the two points x and $x+h$ is not sufficiently accurate, one can try combining the function values at three points x, $x+h$ and $x-h$. To find the appropriate combination of $u(x-h), u(x), u(x+h)$, we return to the Taylor expansions (11.5). To solve for $u'(x)$, we subtract[†] the two formulae, and so

$$u(x+h) - u(x-h) = 2\,u'(x)\,h + u'''(x)\,\frac{h^3}{3} + \mathrm{O}(h^4).$$

Rearranging the terms, we are led to the well-known *centered difference formula*

$$u'(x) = \frac{u(x+h) - u(x-h)}{2\,h} + \mathrm{O}(h^2), \tag{11.7}$$

which is a second order approximation to the first derivative. Geometrically, the centered difference quotient represents the slope of the secant line through the two points $\bigl(x-h, u(x-h)\bigr)$ and $\bigl(x+h, u(x+h)\bigr)$ on the graph of u centered symmetrically about the point x. Figure 11.1 illustrates the two approximations; the advantages in accuracy in the centered difference version are graphically evident. Higher order approximations can be found by evaluating the function at yet more sample points, including, say, $x+2\,h, x-2\,h$, etc.

Example 11.3. Return to the function $u(x) = \sin x$ considered in Example 11.1. The centered difference approximation to its derivative $u'(1) = \cos 1 = 0.5403023\ \ldots$ is

$$\cos 1 \;\approx\; \frac{\sin(1+h) - \sin(1-h)}{2\,h}\,.$$

The results are tabulated as follows:

[†] This next computation depends upon the computer's precision; here we used single precision in MATLAB.

[†] The terms $\mathrm{O}(h^4)$ do *not* cancel, since they represent potentially different multiples of h^4.

h	.1	.01	.001	.0001
approximation	0.53940225217	0.54029330087	0.54030221582	0.54030230497
error	-0.00090005370	-0.00000900499	-0.00000009005	-0.00000000090

As advertised, the results are much more accurate than the one-sided finite difference approximation used in Example 11.1 at the same step size. Since it is a second order approximation, each reduction in the step size by a factor of $\frac{1}{10}$ results in two more decimal places of accuracy.

Many additional finite difference approximations can be constructed by similar manipulations of Taylor expansions, but these few very basic ones will suffice for our subsequent purposes. In the following subsection, we apply the finite difference formulae to develop numerical solution schemes for the heat and wave equations.

11.2. Numerical Algorithms for the Heat Equation.

Consider the heat equation

$$\frac{\partial u}{\partial t} = \gamma \, \frac{\partial^2 u}{\partial x^2} \,, \qquad 0 < x < \ell, \qquad t \geq 0, \tag{11.8}$$

representing a homogeneous diffusion process of, sqy, heat in bar of length ℓ and constant thermal diffusivity $\gamma > 0$. The solution $u(t,x)$ represents the temperature in the bar at time $t \geq 0$ and position $0 \leq x \leq \ell$. To be concrete, we will impose time-dependent Dirichlet boundary conditions

$$u(t,0) = \alpha(t), \qquad u(t,\ell) = \beta(t), \qquad t \geq 0, \tag{11.9}$$

specifying the temperature at the ends of the bar, along with the initial conditions

$$u(0,x) = f(x), \qquad 0 \leq x \leq \ell, \tag{11.10}$$

specifying the bar's initial temperature distribution. In order to effect a numerical approximation to the solution to this initial-boundary value problem, we begin by introducing a *rectangular mesh* consisting of points (t_i, x_j) with

$$0 = x_0 < x_1 < \cdots < x_n = \ell \qquad \text{and} \qquad 0 = t_0 < t_1 < t_2 < \cdots \,.$$

For simplicity, we maintain a uniform mesh spacing in both directions, with

$$h = x_{j+1} - x_j = \frac{\ell}{n} \,, \qquad k = t_{i+1} - t_i,$$

representing, respectively, the spatial mesh size and the time step size. It will be essential that we do *not* a priori require the two to be the same. We shall use the notation

$$u_{i,j} \approx u(t_i, x_j) \qquad \text{where} \qquad t_i = i\,k, \qquad x_j = j\,h, \tag{11.11}$$

to denote the numerical approximation to the solution value at the indicated mesh point.

As a first attempt at designing a numerical method, we shall use the simplest finite difference approximations to the derivatives. The second order space derivative is approximated by (11.6), and hence

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2}(t_i, x_j) &\approx \frac{u(t_i, x_{j+1}) - 2\,u(t_i, x_j) + u(t_i, x_{j-1})}{h^2} + \mathrm{O}(h^2) \\ &\approx \frac{u_{i,j+1} - 2\,u_{i,j} + u_{i,j-1}}{h^2} + \mathrm{O}(h^2), \end{aligned} \tag{11.12}$$

where the error in the approximation is proportional to h^2. Similarly, the one-sided finite difference approximation (11.4) is used for the time derivative, and so

$$\frac{\partial u}{\partial t}(t_i, x_j) \approx \frac{u(t_{i+1}, x_j) - u(t_i, x_j)}{k} + \mathrm{O}(k) \approx \frac{u_{i+1,j} - u_{i,j}}{k} + \mathrm{O}(k), \tag{11.13}$$

where the error is proportion to k. In practice, one should try to ensure that the approximations have similar orders of accuracy, which leads us to choose

$$k \approx h^2.$$

Assuming $h < 1$, this requirement has the important consequence that the time steps must be *much* smaller than the space mesh size.

Remark: At this stage, the reader might be tempted to replace (11.13) by the second order central difference approximation (11.7). However, this produces significant complications, and the resulting numerical scheme is not practical.

Replacing the derivatives in the heat equation (11.14) by their finite difference approximations (11.12), (11.13), and rearranging terms, we end up with the linear system

$$u_{i+1,j} = \mu\, u_{i,j+1} + (1 - 2\mu) u_{i,j} + \mu\, u_{i,j-1}, \qquad \begin{aligned} &i = 0, 1, 2, \ldots\,, \\ &j = 1, \ldots, n-1, \end{aligned} \tag{11.14}$$

in which

$$\mu = \frac{\gamma\, k}{h^2}\,. \tag{11.15}$$

The resulting numerical scheme takes the form of an iterative linear system for the solution values $u_{i,j} \approx u(t_i, x_j)$, $j = 1, \ldots, n-1$, at each time step t_i.

The initial condition (11.10) means that we should initialize our numerical data by sampling the initial temperature at the mesh points:

$$u_{0,j} = f_j = f(x_j), \qquad j = 1, \ldots, n-1. \tag{11.16}$$

Similarly, the boundary conditions (11.9) require that

$$u_{i,0} = \alpha_i = \alpha(t_i), \qquad u_{i,n} = \beta_i = \beta(t_i), \qquad i = 0, 1, 2, \ldots\,. \tag{11.17}$$

For consistency, we should assume that the initial and boundary conditions agree at the corners of the domain:

$$f_0 = f(0) = u(0,0) = \alpha(0) = \alpha_0, \qquad f_n = f(\ell) = u(0,\ell) = \beta(0) = \beta_0.$$

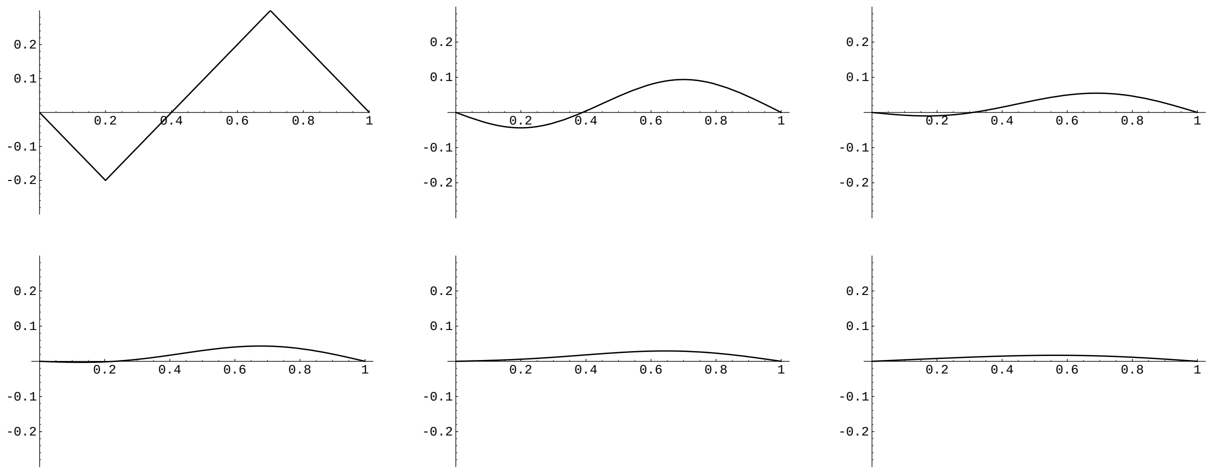


Figure 11.2. A Solution to the Heat Equation.

The three equations (11.14–17) completely prescribe the numerical approximation algorithm for solving the initial-boundary value problem (11.8–10).

Let us rewrite the scheme in a more transparent matrix form. First, let

$$\mathbf{u}^{(i)} = \big(u_{i,1}, u_{i,2}, \ldots, u_{i,n-1} \big)^T \approx \big(u(t_i, x_1), u(t_i, x_2), \ldots, u(t_i, x_{n-1}) \big)^T \tag{11.18}$$

be the vector whose entries are the numerical approximations to the solution values at time t_i at the *interior* nodes. We omit the boundary nodes $x_0 = 0,\ x_n = \ell$, since those values are fixed by the boundary conditions (11.9). Then (11.14) assumes the compact vectorial form

$$\mathbf{u}^{(i+1)} = A\,\mathbf{u}^{(i)} + \mathbf{b}^{(i)}, \tag{11.19}$$

where

$$A = \begin{pmatrix} 1-2\mu & \mu & & & & \\ \mu & 1-2\mu & \mu & & & \\ & \mu & 1-2\mu & \mu & & \\ & & \mu & \ddots & \ddots & \\ & & & \ddots & \ddots & \mu \\ & & & & \mu & 1-2\mu \end{pmatrix}, \qquad \mathbf{b}^{(i)} = \begin{pmatrix} \mu\,\alpha_i \\ 0 \\ 0 \\ \vdots \\ 0 \\ \mu\,\beta_i \end{pmatrix}. \tag{11.20}$$

The coefficient matrix A is symmetric and tridiagonal. The contributions (11.17) of the boundary nodes appear in the vector $\mathbf{b}^{(i)}$. This numerical method is known as an *explicit scheme* since each iterate is computed directly without relying on solving an auxiliary equation — unlike the implicit schemes to be discussed below.

Example 11.4. Let us fix the diffusivity $\gamma = 1$ and the bar length $\ell = 1$. Consider the initial temperature profile

$$u(0,x) = f(x) = \begin{cases} -x, & 0 \le x \le \frac{1}{5}, \\ x - \frac{2}{5}, & \frac{1}{5} \le x \le \frac{7}{10}, \\ 1 - x, & \frac{7}{10} \le x \le 1, \end{cases} \tag{11.21}$$

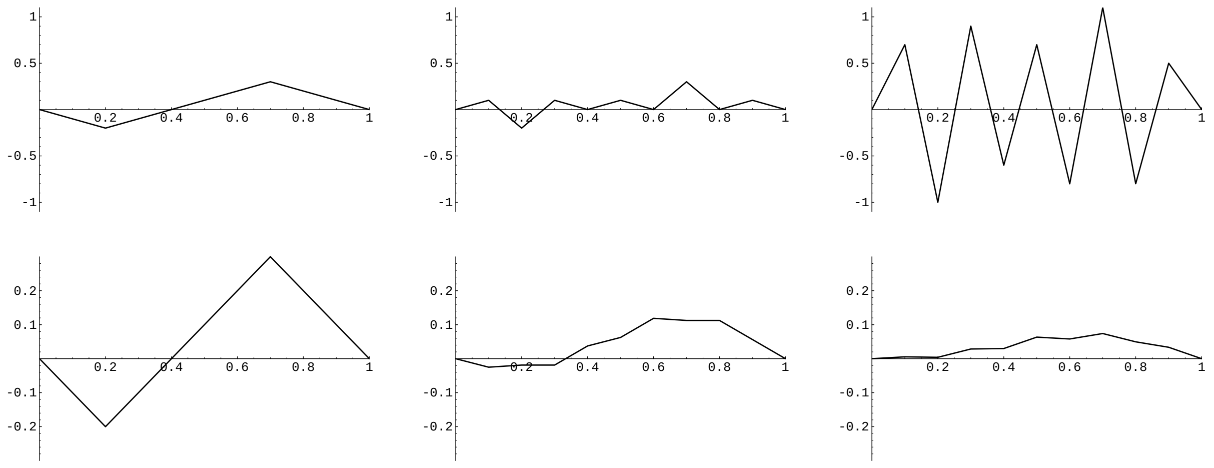


Figure 11.3. Numerical Solutions for the Heat Equation Based on the Explicit Scheme.

on a bar of length 1, plotted in the first graph in Figure 11.2. The solution is plotted at the successive times $t = ., .02, .04, \ldots, .1$. Observe that the corners in the initial data are immediately smoothed out. As time progresses, the solution decays, at an exponential rate of $\pi^2 \approx 9.87$, to a uniform, zero temperature, which is the equilibrium temperature distribution for the homogeneous Dirichlet boundary conditions. As the solution decays to thermal equilibrium, it also assumes the progressively more symmetric shape of a single sine arc, of exponentially decreasing amplitude.

In our numerical solution, we take the spatial step size $h = .1$. In Figure 11.3 we compare two (slightly) different time step sizes on the same initial data as used in (11.21). The first sequence uses the time step $k = h^2 = .01$ and plots the solution at times $t = 0., .02, .04$. The solution is already starting to show signs of instability, and indeed soon thereafter becomes completely wild. The second sequence takes $k = .005$ and plots the solution at times $t = 0., .025, .05$. (Note that the two sequences of plots have different vertical scales.) Even though we are employing a rather coarse mesh, the numerical solution is not too far away from the true solution to the initial value problem, which can be found in Figure 11.2.

In light of this calculation, we need to understand why our scheme sometimes gives reasonable answers but at other times utterly fails. To this end, let us specialize to homogeneous boundary conditions

$$u(t,0) = 0 = u(t,\ell), \qquad \text{whereby} \qquad \alpha_i = \beta_i = 0 \qquad \text{for all} \qquad i = 0,1,2,3,\ldots\,, \tag{11.22}$$

and so (11.19) reduces to a homogeneous, linear iterative system

$$\mathbf{u}^{(i+1)} = A\,\mathbf{u}^{(i)}. \tag{11.23}$$

According to Proposition 7.8, all solutions will converge to zero, $\mathbf{u}^{(i)} \to \mathbf{0}$ — as they are supposed to (why?) — if and only if A is a convergent matrix. But convergence depends upon the step sizes. Example 11.4 is indicating that for mesh size $h = .1$, the time step

$k = .01$ yields a non-convergent matrix, while $k = .005$ leads to a convergent matrix and a valid numerical scheme.

As we learned in Theorem 7.11, the convergence property of a matrix is fixed by its spectral radius, i.e., its largest eigenvalue in magnitude. There is, in fact, an explicit formula for the eigenvalues of the particular tridiagonal matrix in our numerical scheme, which follows from the following general result.

Lemma 11.5. *The eigenvalues of an $(n-1)\times(n-1)$ tridiagonal matrix all of whose diagonal entries are equal to a and all of whose sub- and super-diagonal entries are equal to b are*

$$\lambda_k = a + 2b\cos\frac{\pi k}{n}\,, \qquad\qquad k = 1,\ldots,n-1. \tag{11.24}$$

Proof: The corresponding eigenvectors are

$$\mathbf{v}_k = \left(\sin\frac{k\pi}{n}\,,\ \ \sin\frac{2k\pi}{n}\,,\ \ \ldots\ \ \sin\frac{nk\pi}{n}\right)^T.$$

Indeed, the $j^{\rm th}$ entry of the eigenvalue equation $A\mathbf{v}_k = \lambda_k\mathbf{v}_k$ reads

$$\sin\frac{jk\pi}{n} \ +\ b\left(\sin\frac{(j-1)k\pi}{n} + \sin\frac{(j+1)k\pi}{n}\right) = \left(a + 2b\cos\frac{k\pi}{n}\right)\sin\frac{jk\pi}{n}\,,$$

which follows from the trigonometric identity

$$\sin\alpha + \sin\beta = 2\,\cos\frac{\alpha-\beta}{2}\,\sin\frac{\alpha+\beta}{2}\,. \qquad\qquad Q.E.D.$$

In our particular case, $a = 1 - 2\mu$ and $b = \mu$, and hence the eigenvalues of the matrix A given by (11.20) are

$$\lambda_k = 1 - 2\mu + 2\mu\cos\frac{\pi k}{n}\,, \qquad\qquad k = 1,\ldots,n-1.$$

Since the cosine term ranges between -1 and $+1$, the eigenvalues satisfy

$$1 - 4\mu < \lambda_k < 1.$$

Thus, assuming that $0 < \mu \le \frac{1}{2}$ guarantees that all $|\,\lambda_k\,| < 1$, and hence A is a convergent matrix. In this way, we have deduced the basic stability criterion

$$\mu = \frac{\gamma k}{h^2} \le \frac{1}{2}\,, \qquad \text{or} \qquad k \le \frac{h^2}{2\gamma}\,. \tag{11.25}$$

With some additional analytical work, [**28**], it can be shown that this is sufficient to conclude that the numerical scheme (11.14–17) converges to the true solution to the initial-boundary value problem for the heat equation.

Since not all choices of space and time steps lead to a convergent scheme, the numerical method is called *conditionally stable*. The convergence criterion (11.25) places a severe restriction on the time step size. For instance, if we have $h = .01$, and $\gamma = 1$, then we can only use a time step size $k \le .00005$, which is minuscule. It would take an inordinately large number of time steps to compute the value of the solution at even a moderate times, e.g., $t = 1$. Moreover, owing to the limited accuracy of computers, the propagation of round-off errors might then cause a significant reduction in the overall accuracy of the final solution values.

An unconditionally stable method — one that does not restrict the time step — can be constructed by using the backwards difference formula

$$\frac{\partial u}{\partial t}(t_i, x_j) \approx \frac{u(t_i, x_j) - u(t_{i-1}, x_j)}{k} + \mathrm{O}(h^k) \tag{11.26}$$

to approximate the temporal derivative. Substituting (11.26) and the same approximation (11.12) for u_{xx} into the heat equation, and then replacing i by $i+1$, leads to the iterative system

$$u_{i+1,j} - \mu\left(u_{i+1,j+1} - 2u_{i+1,j} + u_{i+1,j-1}\right) = u_{i,j}, \qquad \begin{array}{l} i = 0, 1, 2, \ldots\,, \\ j = 1, \ldots, n-1, \end{array} \tag{11.27}$$

where the parameter $\mu = \gamma\, k/h^2$ is as above. The initial and boundary conditions also have the same form (11.16), (11.17). The system can be written in the matrix form

$$\widehat{A}\,\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + \mathbf{b}^{(i+1)}, \tag{11.28}$$

where $\widehat{A}$ is obtained from the matrix A in (11.20) by replacing μ by $-\mu$. This defines an *implicit method* since we have to solve a tridiagonal linear system at each step in order to compute the next iterate $\mathbf{u}^{(i+1)}$. However, as we learned in Section 4.5, tridiagonal systems can be solved very rapidly, and so speed does not become a significant issue in the practical implementation of this implicit scheme.

Let us look at the convergence properties of the implicit scheme. For homogeneous Dirichlet boundary conditions (11.22), the system takes the form

$$\mathbf{u}^{(i+1)} = \widehat{A}^{-1}\,\mathbf{u}^{(i)},$$

and the convergence is now governed by the eigenvalues of $\widehat{A}^{-1}$. Lemma 11.5 tells us that the eigenvalues of $\widehat{A}$ are

$$\lambda_k = 1 + 2\mu - 2\mu\cos\frac{\pi k}{n}\,, \qquad k = 1, \ldots, n-1.$$

As a result, its inverse $\widehat{A}^{-1}$ has eigenvalues

$$\frac{1}{\lambda_k} = \frac{1}{1 + 2\mu\left(1 - \cos\dfrac{\pi k}{n}\right)}\,, \qquad k = 1, \ldots, n-1.$$

Since $\mu > 0$, the latter are *always* less than 1 in absolute value, and so $\widehat{A}$ is always a convergent matrix. The implicit scheme (11.28) is convergent for any choice of step sizes h, k, and hence *unconditionally stable*.

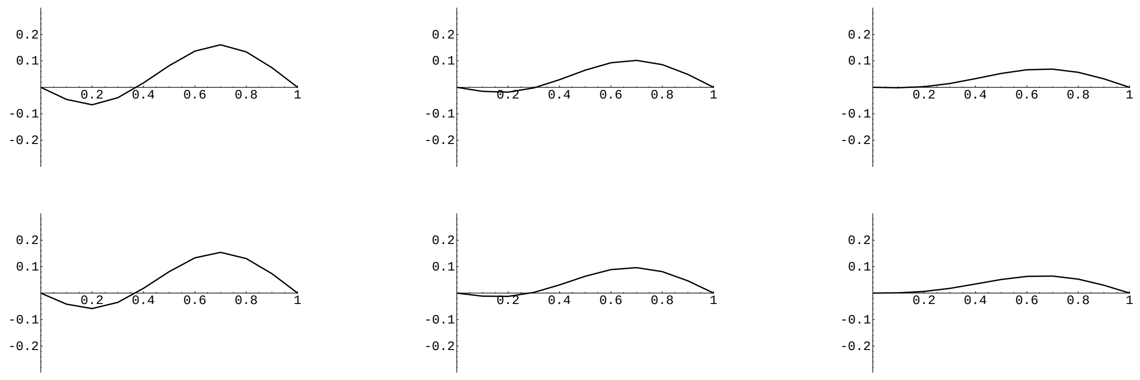


Figure 11.4. Numerical Solutions for the Heat Equation Based on the Implicit Scheme.

Example 11.6. Consider the same initial-boundary value problem considered in Example 11.4. In Figure 11.4, we plot the numerical solutions obtained using the implicit scheme. The initial data is not displayed, but we graph the numerical solutions at times $t = .2, .4, .6$ with a mesh size of $h = .1$. On the top line, we use a time step of $k = .01$, while on the bottom $k = .005$. Unlike the explicit scheme, there is very little difference between the two — both come much closer to the actual solution than the explicit scheme. Indeed, even significantly larger time steps give reasonable numerical approximations to the solution.

Another popular numerical scheme is the *Crank–Nicolson method*

$$u_{i+1,j} - u_{i,j} = \frac{\mu}{2}\left(u_{i+1,j+1} - 2\,u_{i+1,j} + u_{i+1,j-1} + u_{i,j+1} - 2\,u_{i,j} + u_{i,j-1} \right). \tag{11.29}$$

which can be obtained by averaging the explicit and implicit schemes (11.14, 27). We can write the iterative system in matrix form

$$B\,\mathbf{u}^{(i+1)} = C\,\mathbf{u}^{(i)} + \tfrac{1}{2}\left(\mathbf{b}^{(i)} + \mathbf{b}^{(i+1)} \right),$$

where

$$B = \begin{pmatrix} 1+\mu & -\frac{1}{2}\mu & & \\ -\frac{1}{2}\mu & 1+\mu & -\frac{1}{2}\mu & \\ & -\frac{1}{2}\mu & \ddots & \ddots \\ & & \ddots & \ddots \end{pmatrix}, \qquad C = \begin{pmatrix} 1-\mu & \frac{1}{2}\mu & & \\ \frac{1}{2}\mu & 1-\mu & \frac{1}{2}\mu & \\ & \frac{1}{2}\mu & \ddots & \ddots \\ & & \ddots & \ddots \end{pmatrix}. \tag{11.30}$$

Convergence is governed by the generalized eigenvalues of the tridiagonal matrix pair B, C, or, equivalently, the eigenvalues of the product $B^{-1}C$, which are

$$\lambda_k = \frac{1 - \mu\left(1 - \cos\dfrac{\pi k}{n}\right)}{1 + \mu\left(1 - \cos\dfrac{\pi k}{n}\right)}, \qquad k = 1, \ldots, n-1. \tag{11.31}$$

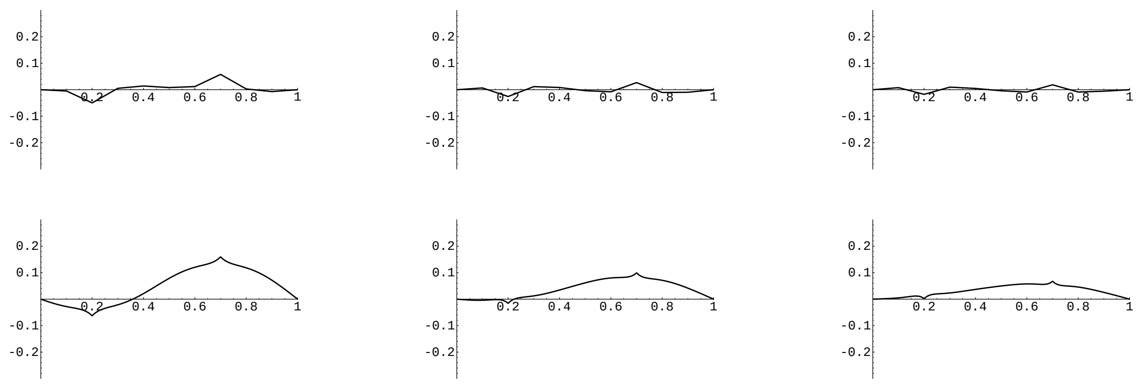


Figure 11.5. Numerical Solutions for the Heat Equation Based on the Crank–Nicolson Scheme.

Since $\mu > 0$, all of the eigenvalues are strictly less than 1 in absolute value, and so the Crank–Nicolson scheme is also unconditionally stable. A detailed analysis will show that the errors are of the order of k^2 and h^2, and so it is reasonable to choose the time step to have the same order of magnitude as the space step, $k \approx h$. This gives the Crank–Nicolson scheme one advantage over the previous two methods. However, applying it to the initial value problem considered earlier points out a significant weakness. Figure 11.5 shows the result of running the scheme on the initial data (11.21). The top row has space and time step sizes $h = k = .1$, and does a rather poor job replicating the solution. The second row uses $h = k = .01$, and performs better except near the corners where an annoying and incorrect local time oscillation persists as the solution decays. Indeed, since most of its eigenvalues are near -1, the Crank–Nicolson scheme does not do a good job of damping out the high frequency modes that arise from small scale features, including discontinuities and corners in the initial data. On the other hand, most of the eigenvalues of the fully implicit scheme are near zero, and it tends to handle the high frequency modes better, losing out to Crank–Nicolson when the data is smooth. Thus, a good strategy is to first evolve using the implicit scheme until the small scale noise is dissipated away, and then switch to Crank–Nicolson to use a much larger time step for final the large scale changes.

11.3. Numerical Solution Methods for the Wave Equation.

Let us now look at some numerical solution techniques for the wave equation. Although this is in a sense unnecessary, owing to the explicit d'Alembert solution formula, the experience we gain in designing workable schemes will serve us well in more complicated situations, including inhomogeneous media, and higher dimensional problems, when analytic solution formulas are no longer available.

Consider the wave equation

$$\frac{\partial^2 u}{\partial t^2} = c^2 \, \frac{\partial^2 u}{\partial x^2} \,, \qquad\qquad 0 < x < \ell, \qquad t \geq 0, \tag{11.32}$$

modeling vibrations of a homogeneous bar of length ℓ with constant wave speed $c > 0$. We impose Dirichlet boundary conditions

$$u(t,0) = \alpha(t), \qquad u(t,\ell) = \beta(t), \qquad t \geq 0. \tag{11.33}$$

and initial conditions

$$u(0,x) = f(x), \qquad \frac{\partial u}{\partial t}(0,x) = g(x), \qquad 0 \leq x \leq \ell. \tag{11.34}$$

We adopt the same uniformly spaced mesh

$$t_i = i\,k, \qquad x_j = j\,h, \qquad \text{where} \qquad h = \frac{\ell}{n}\,.$$

In order to discretize the wave equation, we replace the second order derivatives by their standard finite difference approximations (11.6), namely

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2}(t_i,x_j) &\approx \frac{u(t_{i+1},x_j) - 2\,u(t_i,x_j) + u(t_{i-1},x_j)}{k^2} + \mathrm{O}(h^2), \\ \frac{\partial^2 u}{\partial x^2}(t_i,x_j) &\approx \frac{u(t_i,x_{j+1}) - 2\,u(t_i,x_j) + u(t_i,x_{j-1})}{h^2} + \mathrm{O}(k^2), \end{aligned} \tag{11.35}$$

Since the errors are of orders of k^2 and h^2, we anticipate to be able to choose the space and time step sizes of comparable magnitude:

$$k \approx h.$$

Substituting the finite difference formulae (11.35) into the partial differential equation (11.32), and rearranging terms, we are led to the iterative system

$$u_{i+1,j} = \sigma^2\,u_{i,j+1} + 2\,(1-\sigma^2)\,u_{i,j} + \sigma^2\,u_{i,j-1} - u_{i-1,j}, \qquad \begin{array}{l} i = 1,2,\ldots\,, \\ j = 1,\ldots,n-1, \end{array} \tag{11.36}$$

for the numerical approximations $u_{i,j} \approx u(t_i,x_j)$ to the solution values at the mesh points. The positive parameter

$$\sigma = \frac{c\,k}{h} > 0 \tag{11.37}$$

depends upon the wave speed and the ratio of space and time step sizes. The boundary conditions (11.33) require that

$$u_{i,0} = \alpha_i = \alpha(t_i), \qquad u_{i,n} = \beta_i = \beta(t_i), \qquad i = 0,1,2,\ldots\,. \tag{11.38}$$

This allows us to rewrite the system in matrix form

$$\mathbf{u}^{(i+1)} = B\,\mathbf{u}^{(i)} - \mathbf{u}^{(i-1)} + \mathbf{b}^{(i)}, \tag{11.39}$$

where

$$B = \begin{pmatrix} 2\,(1-\sigma^2) & \sigma^2 & & & \\ \sigma^2 & 2\,(1-\sigma^2) & \sigma^2 & & \\ & \sigma^2 & \ddots & \ddots & \\ & & \ddots & \ddots & \sigma^2 \\ & & & \sigma^2 & 2\,(1-\sigma^2) \end{pmatrix}, \quad \mathbf{u}^{(j)} = \begin{pmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{n-2,j} \\ u_{n-1,j} \end{pmatrix}, \quad \mathbf{b}^{(j)} = \begin{pmatrix} \sigma^2\,\alpha_j \\ 0 \\ \vdots \\ 0 \\ \sigma^2\,\beta_j \end{pmatrix}. \tag{11.40}$$

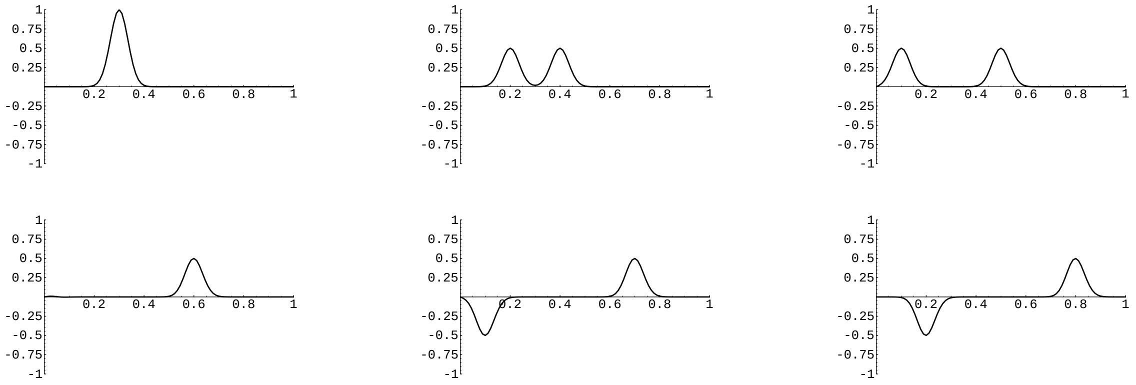


Figure 11.6. Numerically Stable Waves.

The entries of $\mathbf{u}^{(i)}$ are, as in (11.18), the numerical approximations to the solution values at the *interior* nodes. Note that the system (11.39) is a second order iterative scheme, since computing the next iterate $\mathbf{u}^{(i+1)}$ requires the value of the preceding two, $\mathbf{u}^{(i)}$ and $\mathbf{u}^{(i-1)}$.

The one difficulty is getting the method started. We know $\mathbf{u}^{(0)}$ since $u_{0,j} = f_j = f(x_j)$ is determined by the initial position. However, we also need to find $\mathbf{u}^{(1)}$ with entries $u_{1,j} \approx u(k, x_j)$ at time $t_1 = k$ in order launch the iteration, but the initial velocity $u_t(0,x) = g(x)$ prescribes the derivatives $u_t(0,x_j) = g_j = g(x_j)$ at time $t_0 = 0$ instead. One way to resolve this difficult would be to utilize the finite difference approximation

$$g_j = \frac{\partial u}{\partial t}(0,x_j) \approx \frac{u(k,x_j) - u(0,x_j)}{k} \approx \frac{u_{1,j} - g_j}{k} \tag{11.41}$$

to compute the required values

$$u_{1,j} = f_j + k\, g_j.$$

However, the approximation (11.41) is only accurate to order k, whereas the rest of the scheme has error proportional to k^2. Therefore, we would introduce an unacceptably large error at the initial step.

To construct an initial approximation to $\mathbf{u}^{(1)}$ with error on the order of k^2, we need to analyze the local error in more detail. Note that, by Taylor's theorem,

$$\frac{u(k,x_j) - u(0,x_j)}{k} \approx \frac{\partial u}{\partial t}(0,x_j) + \frac{k}{2}\,\frac{\partial^2 u}{\partial t^2}(0,x_j) = \frac{\partial u}{\partial t}(0,x_j) + \frac{c^2\, k}{2}\,\frac{\partial^2 u}{\partial x^2}(0,x_j),$$

where the error is now of order k^2, and, in the final equality, we have used the fact that u is a solution to the wave equation. Therefore, we find

$$\begin{aligned} u(k,x_j) &\approx u(0,x_j) + k\,\frac{\partial u}{\partial t}(0,x_j) + \frac{c^2\,k^2}{2}\,\frac{\partial^2 u}{\partial x^2}(0,x_j) \\ &= f(x_j) + k\,g(x_j) + \frac{c^2\,k^2}{2}\,f''(x_j) \approx f_j + k\,g_j + \frac{c^2\,k^2}{2\,h^2}(f_{j+1} - 2\,f_j + f_{j-1}), \end{aligned}$$

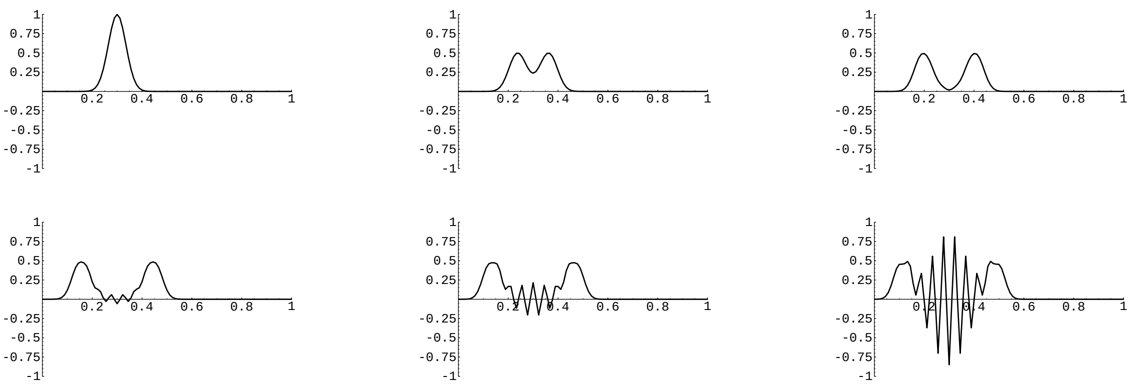


Figure 11.7. Numerically Unstable Waves.

where we can use the finite difference approximation (11.6) for the second derivative of $f(x)$ if no explicit formula is known. Therefore, when we initiate the scheme by setting

$$u_{1,j} = \tfrac{1}{2}\,\sigma^2\, f_{j+1} + (1-\sigma^2) f_j + \tfrac{1}{2}\,\sigma^2\, f_{j-1} + k\, g_j, \tag{11.42}$$

or, in matrix form,

$$\mathbf{u}^{(0)} = \mathbf{f}, \qquad\qquad \mathbf{u}^{(1)} = \tfrac{1}{2}\, B\,\mathbf{u}^{(0)} + k\,\mathbf{g} + \tfrac{1}{2}\,\mathbf{b}^{(0)}, \tag{11.43}$$

we will have maintained the desired order k^2 (and h^2) accuracy.

Example 11.7. Consider the particular initial value problem

$$u_{tt} = u_{xx}, \qquad \begin{array}{cc} u(0,x) = e^{-400\,(x-.3)^2}, \qquad u_t(0,x) = 0, & 0 \le x \le 1, \\ u(t,0) = u(1,0) = 0, & t \ge 0, \end{array}$$

subject to homogeneous Dirichlet boundary conditions on the interval $[0,1]$. The initial data is a fairly concentrated single hump centered at $x = .3$, and we expect it to split into two half sized humps, which then collide with the ends. Let us choose a space discretization consisting of 90 equally spaced points, and so $h = \frac{1}{90} = .0111\ldots$. If we choose a time step of $k = .01$, whereby $\sigma = .9$, then we get reasonably accurate solution over a fairly long time range, as plotted in Figure 11.6 at times $t = 0, .1, .2, \ldots, .5$. On the other hand, if we double the time step, setting $k = .02$, so $\sigma = 1.8$, then, as plotted in Figure 11.7 at times $t = 0, .05, .1, .14, .16, .18$, we observe an instability eventually creeping into the picture that eventually overwhelms the numerical solution. Thus, the numerical scheme appears to only be conditionally stable.

The stability analysis of this numerical scheme proceeds as follows. We first need to recast the second order iterative system (11.39) into a first order system. In analogy with Example 7.4, this is accomplished by introducing the vector $\mathbf{z}^{(i)} = \begin{pmatrix} \mathbf{u}^{(i)} \\ \mathbf{u}^{(i-1)} \end{pmatrix} \in \mathbb{R}^{2n-2}$. Then

$$\mathbf{z}^{(i+1)} = C\,\mathbf{z}^{(i)} + \mathbf{c}^{(i)}, \qquad \text{where} \qquad C = \begin{pmatrix} B & -\mathrm{I} \\ \mathrm{I} & \mathrm{O} \end{pmatrix}. \tag{11.44}$$

Therefore, the stability of the method will be determined by the eigenvalues of the coefficient matrix C. The eigenvector equation $C\mathbf{z} = \lambda\,\mathbf{z}$, where $\mathbf{z} = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}$, can be written out in its individual components:

$$B\mathbf{u} - \mathbf{v} = \lambda\,\mathbf{u}, \qquad \mathbf{u} = \lambda\,\mathbf{v}.$$

Substituting the second equation into the first, we find

$$(\lambda B - \lambda^2 - 1)\,\mathbf{v} = \mathbf{0}, \qquad \text{or} \qquad B\mathbf{v} = \left(\lambda + \frac{1}{\lambda}\right)\mathbf{v}.$$

The latter equation implies that $\mathbf{v}$ is an eigenvector of B with $\lambda + \lambda^{-1}$ the corresponding eigenvalue. The eigenvalues of the tridiagonal matrix B are governed by Lemma 11.5, in which $a = 2(1-\sigma^2)$ and $b = \sigma^2$, and hence are

$$\lambda + \frac{1}{\lambda} = 2\left(1 - \sigma^2 + \sigma^2 \cos\frac{\pi k}{n}\right), \qquad k = 1, \ldots, n-1.$$

Multiplying both sides by λ leads to a quadratic equation for the eigenvalues,

$$\lambda^2 - 2\,a_k\,\lambda + 1 = 0, \qquad \text{where} \qquad 1 - 2\sigma^2 < a_k = 1 - \sigma^2 + \sigma^2\cos\frac{\pi k}{n} < 1. \tag{11.45}$$

Each pair of solutions to these $n-1$ quadratic equations, namely

$$\lambda_k^{\pm} = a_k \pm \sqrt{a_k^2 - 1}\,, \tag{11.46}$$

yields two eigenvalues of the matrix C. If $a_k > 1$, then one of the two eigenvalues will be larger than one in magnitude, which means that the linear iterative system has an exponentially growing mode, and so $\|\,\mathbf{u}^{(i)}\,\| \to \infty$ as $i \to \infty$ for almost all choices of initial data. This is clearly incompatible with the wave equation solution that we are trying to approximate, which is periodic and hence remains bounded. On the other hand, if $|\,a_k\,| < 1$, then the eigenvalues (11.46) are complex numbers of modulus 1, indicated stability (but not convergence) of the matrix C. Therefore, in view of (11.45), we should require that

$$\sigma = \frac{c\,k}{h} < 1, \qquad \text{or} \qquad k < \frac{h}{c}\,, \tag{11.47}$$

which places a restriction on the relative sizes of the time and space steps. We conclude that the numerical scheme is conditionally stable.

The stability criterion (11.47) is known as the *Courant condition*, and can be assigned a simple geometric interpretation. Recall that the wave speed c is the slope of the characteristic lines for the wave equation. The Courant condition requires that the *mesh slope*, which is defined to be the ratio of the space step size to the time step size, namely h/k, must be strictly greater than the characteristic slope c. A signal starting at a mesh point (t_i, x_j) will reach positions $x_j \pm k/c$ at the next time $t_{i+1} = t_i + k$, which are still between the mesh points x_{j-1} and x_{j+1}. Thus, characteristic lines that start at a mesh point are not allowed to reach beyond the neighboring mesh points at the next time step.

For instance, in Figure 11.8, the wave speed is $c = 1.25$. The first figure has equal mesh spacing $k = h$, and does not satisfy the Courant condition (11.47), whereas the

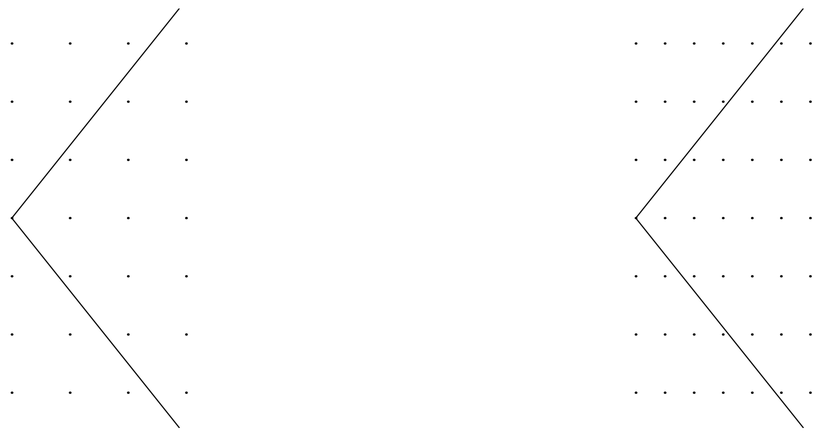


Figure 11.8. The Courant Condition.

second figure has $k = \frac{1}{2}\,h$, which does. Note how the characteristic lines starting at a given mesh point have progressed beyond the neighboring mesh points after one time step in the first case, but not in the second.

AIMS Lecture Notes 2006

Peter J. Olver

12. Minimization

In this part, we will introduce and solve the most basic mathematical optimization problem: minimize a quadratic function depending on several variables. This will require a short introduction to positive definite matrices. Assuming the coefficient matrix of the quadratic terms is positive definite, the minimizer can be found by solving an associated linear algebraic system. With the solution in hand, we are able to treat a wide range of applications, including least squares fitting of data, interpolation, as well as the finite element method for solvilng boundary value problems for differential equations.

12.1. Positive Definite Matrices.

Minimization of functions of several variables relies on an extremely important class of symmetric matrices.

Definition 12.1. An $n \times n$ matrix K is called *positive definite* if it is symmetric, $K^T = K$, and satisfies the positivity condition

$$\mathbf{x}^T K \mathbf{x} > 0 \qquad \text{for all vectors} \qquad \mathbf{0} \neq \mathbf{x} \in \mathbb{R}^n. \tag{12.1}$$

We will sometimes write $K > 0$ to mean that K is a symmetric, positive definite matrix.

Warning: The condition $K > 0$ does *not* mean that all the entries of K are positive. There are many positive definite matrices that have some negative entries; see Example 12.2 below. Conversely, many symmetric matrices with all positive entries are not positive definite!

Remark: Although some authors allow non-symmetric matrices to be designated as positive definite, we will *only* say that a matrix is positive definite when it is symmetric. But, to underscore our convention and remind the casual reader, we will often include the superfluous adjective "symmetric" when speaking of positive definite matrices.

Given any symmetric matrix K, the homogeneous quadratic polynomial

$$q(\mathbf{x}) = \mathbf{x}^T K \mathbf{x} = \sum_{i,j=1}^{n} k_{ij}\, x_i\, x_j, \tag{12.2}$$

is known as a *quadratic form* on $\mathbb{R}^n$. The quadratic form is called *positive definite* if

$$q(\mathbf{x}) > 0 \qquad \text{for all} \qquad 0 \neq \mathbf{x} \in \mathbb{R}^n. \tag{12.3}$$

Thus, a quadratic form is positive definite if and only if its coefficient matrix is.

Example 12.2. Even though the symmetric matrix $K = \begin{pmatrix} 4 & -2 \\ -2 & 3 \end{pmatrix}$ has two negative entries, it is, nevertheless, a positive definite matrix. Indeed, the corresponding quadratic form

$$q(\mathbf{x}) = \mathbf{x}^T K\,\mathbf{x} = 4\,x_1^2 - 4\,x_1\,x_2 + 3\,x_2^2 = (2\,x_1 - x_2)^2 + 2\,x_2^2 \geq 0$$

is a sum of two non-negative quantities. Moreover, $q(\mathbf{x}) = 0$ if and only if both $2\,x_1 - x_2 = 0$ and $x_2 = 0$, which implies $x_1 = 0$ also. This proves $q(\mathbf{x}) > 0$ for all $\mathbf{x} \neq \mathbf{0}$, and hence K is indeed a positive definite matrix.

On the other hand, despite the fact that $K = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$ has all positive entries, it is *not* a positive definite matrix. Indeed, writing out

$$q(\mathbf{x}) = \mathbf{x}^T K\,\mathbf{x} = x_1^2 + 4\,x_1\,x_2 + x_2^2,$$

we find, for instance, that $q(1,-1) = -2 < 0$, violating positivity. These two simple examples should be enough to convince the reader that the problem of determining whether a given symmetric matrix is or is not positive definite is not completely elementary.

Example 12.3. By definition, a general symmetric 2×2 matrix $K = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$ is positive definite if and only if the associated quadratic form satisfies

$$q(\mathbf{x}) = a\,x_1^2 + 2\,b\,x_1\,x_2 + c\,x_2^2 > 0 \qquad \text{for all} \qquad \mathbf{x} \neq \mathbf{0}. \tag{12.4}$$

Analytic geometry tells us that this is the case if and only if

$$a > 0, \qquad\qquad a\,c - b^2 > 0, \tag{12.5}$$

i.e., the quadratic form has positive leading coefficient and positive determinant (or negative discriminant).

A practical test of positive definiteness comes from the following result, whose proof is based on Gaussian Elimination, [**38**].

Theorem 12.4. *A symmetric matrix K is positive definite if and only if it is regular and has all positive pivots.*

In other words, a square matrix K is positive definite if and only if it can be factored $K = L\,D\,L^T$, where L is special lower triangular and D is diagonal with all positive diagonal entries. Indeed, we cna then write the associated quadratic form as a sum of squares

$$\begin{aligned} q(\mathbf{x}) &= \mathbf{x}^T K\,\mathbf{x} = \mathbf{x}^T L\,D\,L^T\mathbf{x} = (L^T\mathbf{x})^T D\,(L^T\mathbf{x}) \\ &= \mathbf{y}^T D\,\mathbf{y} = d_1\,y_1^2 + \;\cdots\; + d_n\,y_n^2, \end{aligned} \qquad \text{where} \qquad \mathbf{y} = L^T\mathbf{x}. \tag{12.6}$$

Furthermore, the resulting diagonal quadratic form is positive definite, $\mathbf{y}^T D\,\mathbf{y} > 0$ for all $\mathbf{y} \neq \mathbf{0}$ if and only if all the pivots are positive, $d_i > 0$. Invertibility of L^T tells us that $\mathbf{y} = \mathbf{0}$ if and only if $\mathbf{x} = \mathbf{0}$, and hence, positivity of the pivots is equivalent to positive definiteness of the original quadratic form: $q(\mathbf{x}) > 0$ for all $\mathbf{x} \neq 0$.

Example 12.5. Consider the symmetric matrix $K = \begin{pmatrix} 1 & 2 & -1 \\ 2 & 6 & 0 \\ -1 & 0 & 9 \end{pmatrix}$. Gaussian Elimination produces the factors

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 1 & 1 \end{pmatrix}, \qquad D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 6 \end{pmatrix}, \qquad L^T = \begin{pmatrix} 1 & 2 & -1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$

in its factorization $K = L\,D\,L^T$. Since the pivots — the diagonal entries $1, 2$ and 6 in D — are all positive, Theorem 12.4 implies that K is positive definite, which means that the associated quadratic form satisfies

$$q(\mathbf{x}) = x_1^2 + 4\,x_1 x_2 - 2\,x_1 x_3 + 6\,x_2^2 + 9\,x_3^2 > 0, \qquad \text{for all} \qquad \mathbf{x} = (\,x_1, x_2, x_3\,)^T \neq \mathbf{0}.$$

Indeed, the $L\,D\,L^T$ factorization implies that $q(\mathbf{x})$ can be explicitly written as a sum of squares:

$$q(\mathbf{x}) = x_1^2 + 4\,x_1 x_2 - 2\,x_1 x_3 + 6\,x_2^2 + 9\,x_3^2 = y_1^2 + 2\,y_2^2 + 6\,y_3^2, \tag{12.7}$$

where

$$y_1 = x_1 + 2\,x_2 - x_3, \qquad y_2 = x_2 + x_3, \qquad y_3 = x_3,$$

are the entries of $\mathbf{y} = L^T \mathbf{x}$. Positivity of the coefficients of the y_i^2 (which are the pivots) implies that $q(\mathbf{x})$ is positive definite.

Slightly more generally, a quadratic form and its associated symmetric coefficient matrix are called *positive semi-definite* if

$$q(\mathbf{x}) = \mathbf{x}^T K\,\mathbf{x} \geq 0 \qquad \text{for all} \qquad \mathbf{x} \in \mathbb{R}^n. \tag{12.8}$$

A positive semi-definite matrix may have *null directions*, meaning non-zero vectors $\mathbf{z}$ such that $q(\mathbf{z}) = \mathbf{z}^T K\,\mathbf{z} = \mathbf{0}$. Clearly, any nonzero vector $\mathbf{z}$ such that $K\,\mathbf{z} = \mathbf{0}$ defines a null direction, but there may be others. A positive definite matrix is not allowed to have null directions, and so $\ker K = \{\mathbf{0}\}$. Therefore:

Proposition 12.6. *If K is positive definite, then K is nonsingular.*

The converse, however, is *not* valid; many symmetric, nonsingular matrices fail to be positive definite.

Example 12.7. The matrix $K = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$ is positive semi-definite, but not positive definite. Indeed, the associated quadratic form

$$q(\mathbf{x}) = \mathbf{x}^T K\,\mathbf{x} = x_1^2 - 2\,x_1 x_2 + x_2^2 = (x_1 - x_2)^2 \geq 0$$

is a perfect square, and so clearly non-negative. However, the elements of $\ker K$, namely the scalar multiples of the vector $(\,1, 1\,)^T$, define null directions: $q(c, c) = 0$.

In a similar fashion, a quadratic form $q(\mathbf{x}) = \mathbf{x}^T K\,\mathbf{x}$ and its associated symmetric matrix K are called *negative semi-definite* if $q(\mathbf{x}) \le 0$ for all $\mathbf{x}$ and *negative definite* if $q(\mathbf{x}) < 0$ for all $\mathbf{x} \neq \mathbf{0}$. A quadratic form is called *indefinite* if it is neither positive nor negative semi-definite; equivalently, there exist points $\mathbf{x}_+$ where $q(\mathbf{x}_+) > 0$ and points $\mathbf{x}_-$ where $q(\mathbf{x}_-) < 0$. Details can be found in the exercises.

Gram Matrices

Symmetric matrices whose entries are given by inner products of elements of an inner product space will appear throughout this text. They are named after the nineteenth century Danish mathematician Jorgen Gram — not the metric mass unit!

Definition 12.8. Let A be an $m \times n$ matrix Then the $n \times n$ matrix

$$K = A^T A \tag{12.9}$$

is known as the associated *Gram matrix*.

Example 12.9. If

$$A = \begin{pmatrix} 1 & 3 \\ 2 & 0 \\ -1 & 6 \end{pmatrix}, \qquad \text{then} \qquad K = A^T A = \begin{pmatrix} 1 & 2 & -1 \\ 3 & 0 & 6 \end{pmatrix} \begin{pmatrix} 1 & 3 \\ 2 & 0 \\ -1 & 6 \end{pmatrix} = \begin{pmatrix} 6 & -3 \\ -3 & 45 \end{pmatrix}.$$

The resulting matrix is positive definite owing to the following result.

Theorem 12.10. *All Gram matrices are positive semi-definite. The Gram matrix $K = A^T A$ is positive definite if and only if* $\ker A = \{0\}$.

Proof: To prove positive (semi-)definiteness of K, we need to examine the associated quadratic form

$$q(\mathbf{x}) = \mathbf{x}^T K\mathbf{x} = \mathbf{x}^T A^T A\,\mathbf{x} = (A\,\mathbf{x})^T A\,\mathbf{x} = \|\,A\,\mathbf{x}\,\|^2 \ge 0,$$

for all $\mathbf{x} \in \mathbb{R}^n$. Moreover, it equals 0 if and only if $A\,\mathbf{x} = \mathbf{0}$, and so if A has trivial kernel, this requires $\mathbf{x} = \mathbf{0}$, and hence $q(\mathbf{x}) = 0$ if and only if $\mathbf{x} = \mathbf{0}$. Thus, in this case, $q(\mathbf{x})$ and K are positive definite. *Q.E.D.*

More generally, if $C > 0$ is any symmetric, positive definite $m \times m$ matrix, then we define the *weighted Gram matrix*

$$K = A^T C\,A. \tag{12.10}$$

Theorem 12.10 also holds as stated for weighted Gram matrices. In the majority of applications, $C = \operatorname{diag}(c_1, \ldots, c_m)$ is a diagonal positive definite matrix, which requires it to have strictly positive diagonal entries $c_i > 0$.

Example 12.11. Returning to the situation of Example 12.9, let $C = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 5 \end{pmatrix}$ be a diagonal positive definite matrix. Then the corresponding weighted Gram matrix

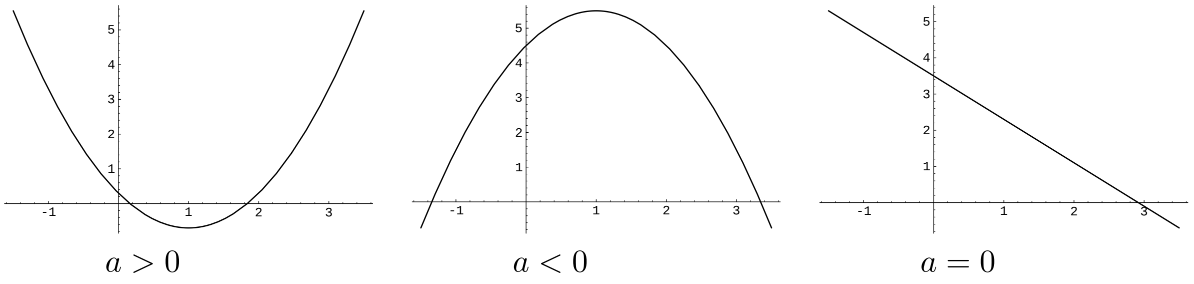


Figure 12.1. Parabolas.

(12.10) is

$$\widetilde{K} = A^T C\, A = \begin{pmatrix} 1 & 2 & -1 \\ 3 & 0 & 6 \end{pmatrix} \begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 5 \end{pmatrix} \begin{pmatrix} 1 & 3 \\ 2 & 0 \\ -1 & 6 \end{pmatrix} = \begin{pmatrix} 16 & -21 \\ -21 & 207 \end{pmatrix},$$

which is again positive definite.

12.2. Minimization of Quadratic Functions.

The simplest algebraic equations are linear systems. As such, they must be thoroughly understood before venturing into the far more complicated nonlinear realm. For minimization problems, the starting point is the quadratic function. (Linear functions do not have minima — think of the function $f(x) = \alpha x + \beta$ whose graph is a straight line.) In this section, we shall see how the problem of minimizing a general quadratic function of n variables can be solved by linear algebra techniques.

Let us begin by reviewing the very simplest example — minimizing a scalar quadratic function

$$p(x) = a x^2 + 2 b x + c \tag{12.11}$$

over all possible values of $x \in \mathbb{R}$. If $a > 0$, then the graph of p is a parabola pointing upwards, and so there exists a unique minimum value. If $a < 0$, the parabola points downwards, and there is no minimum (although there is a maximum). If $a = 0$, the graph is a straight line, and there is neither minimum nor maximum — except in the trivial case when $b = 0$ also, and the function $p(x) = c$ is constant, with every x qualifying as a minimum and a maximum. The three nontrivial possibilities are sketched in Figure 12.1.

In the case $a > 0$, the minimum can be found by calculus. The *critical points* of a function, which are candidates for minima (and maxima), are found by setting its derivative to zero. In this case, differentiating, and solving

$$p'(x) = 2 a x + 2 b = 0,$$

we conclude that the only possible minimum value occurs at

$$x^\star = -\,\frac{b}{a}, \qquad \text{where} \qquad p(x^\star) = c - \frac{b^2}{a}\,. \tag{12.12}$$

Of course, one must check that this critical point is indeed a minimum, and not a maximum or inflection point. The second derivative test will show that $p''(x^\star) = 2a > 0$, and so $x^\star$ is at least a local minimum.

A more instructive approach to this problem — and one that only requires elementary algebra — is to "complete the square". We rewrite

$$p(x) = a\left(x + \frac{b}{a}\right)^2 + \frac{ac - b^2}{a}. \tag{12.13}$$

If $a > 0$, then the first term is always ≥ 0, and, moreover, attains its minimum value 0 only at $x^\star = -b/a$. The second term is constant, and so is unaffected by the value of x. Thus, the global minimum of $p(x)$ is at $x^\star = -b/a$. Moreover, its minimal value equals the constant term, $p(x^\star) = (ac - b^2)/a$, thereby reconfirming and strengthening the calculus result in (12.12).

Now that we have the one-variable case firmly in hand, let us turn our attention to the more substantial problem of minimizing quadratic functions of several variables. Thus, we seek to minimize a (real) *quadratic function*

$$p(\mathbf{x}) = p(x_1, \ldots, x_n) = \sum_{i,j=1}^{n} k_{ij}\, x_i\, x_j - 2 \sum_{i=1}^{n} f_i\, x_i + c, \tag{12.14}$$

depending on n variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$. The coefficients k_{ij}, f_i and c are all assumed to be real. Moreover, we can assume, without loss of generality, that the coefficients of the quadratic terms are symmetric: $k_{ij} = k_{ji}$. Note that $p(\mathbf{x})$ is more general than a quadratic form (12.2) in that it also contains linear and constant terms. We seek a global minimum, and so the variables $\mathbf{x}$ are allowed to vary over all of $\mathbb{R}^n$.

Let us begin by rewriting the quadratic function (12.14) in a more compact matrix notation:

$$p(\mathbf{x}) = \mathbf{x}^T K\, \mathbf{x} - 2\,\mathbf{x}^T \mathbf{f} + c, \tag{12.15}$$

in which $K = (k_{ij})$ is a symmetric $n \times n$ matrix, $\mathbf{f}$ is a constant vector, and c is a constant scalar. We first note that in the simple scalar case (12.11), we needed to impose the condition that the quadratic coefficient a is *positive* in order to obtain a (unique) minimum. The corresponding condition for the multivariable case is that the quadratic coefficient matrix K be *positive definite*. This key assumption enables us to establish a general minimization criterion.

Theorem 12.12. *If K is a symmetric, positive definite matrix, then the quadratic function* (12.15) *has a unique minimizer, which is the solution to the linear system*

$$K\,\mathbf{x} = \mathbf{f}, \qquad \textit{namely} \qquad \mathbf{x}^\star = K^{-1}\mathbf{f}. \tag{12.16}$$

The minimum value of $p(\mathbf{x})$ is equal to any of the following expressions:

$$p(\mathbf{x}^\star) = p(K^{-1}\mathbf{f}) = c - \mathbf{f}^T K^{-1}\mathbf{f} = c - \mathbf{f}^T\mathbf{x}^\star = c - (\mathbf{x}^\star)^T K\,\mathbf{x}^\star. \tag{12.17}$$

Proof: First recall that, by Proposition 12.6, positive definiteness implies that K is a nonsingular matrix, and hence the linear system (12.16) does have a unique solution $\mathbf{x}^\star = K^{-1}\mathbf{f}$. Then, for any $\mathbf{x} \in \mathbb{R}^n$, we can write

$$\begin{aligned} p(\mathbf{x}) &= \mathbf{x}^T K\mathbf{x} - 2\,\mathbf{x}^T\mathbf{f} + c = \mathbf{x}^T K\mathbf{x} - 2\,\mathbf{x}^T K\mathbf{x}^\star + c \\ &= (\mathbf{x}-\mathbf{x}^\star)^T K(\mathbf{x}-\mathbf{x}^\star) + \bigl[\, c - (\mathbf{x}^\star)^T K\mathbf{x}^\star \,\bigr], \end{aligned} \tag{12.18}$$

where we used the symmetry of $K = K^T$ to identify $\mathbf{x}^T K\mathbf{x}^\star = (\mathbf{x}^\star)^T K\mathbf{x}$. The first term in the final expression has the form $\mathbf{y}^T K\mathbf{y}$, where $\mathbf{y} = \mathbf{x} - \mathbf{x}^\star$. Since we assumed that K is positive definite, we know that $\mathbf{y}^T K\mathbf{y} > 0$ for all $\mathbf{y} \neq \mathbf{0}$. Thus, the first term achieves its minimum value, namely 0, if and only if $\mathbf{0} = \mathbf{y} = \mathbf{x} - \mathbf{x}^\star$. Moreover, since $\mathbf{x}^\star$ is fixed, the second term does not depend on $\mathbf{x}$. Therefore, the minimum of $p(\mathbf{x})$ occurs at $\mathbf{x} = \mathbf{x}^\star$ and its minimum value $p(\mathbf{x}^\star)$ is equal to the constant term. The alternative expressions in (12.17) follow from simple substitutions. *Q.E.D.*

Example 12.13. Consider the problem of minimizing the quadratic function

$$p(x_1, x_2) = 4x_1^2 - 2x_1x_2 + 3x_2^2 + 3x_1 - 2x_2 + 1$$

over all (real) x_1, x_2. We first write the function in our matrix form (12.15), so

$$p(x_1,x_2) = (\, x_1 \;\; x_2 \,) \begin{pmatrix} 4 & -1 \\ -1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 2\,(\, x_1 \;\; x_2 \,) \begin{pmatrix} -\frac{3}{2} \\ 1 \end{pmatrix} + 1,$$

whereby

$$K = \begin{pmatrix} 4 & -1 \\ -1 & 3 \end{pmatrix}, \qquad \mathbf{f} = \begin{pmatrix} -\frac{3}{2} \\ 1 \end{pmatrix}. \tag{12.19}$$

(Pay attention to the overall factor of -2 in front of the linear terms.) According to Theorem 12.12, to find the minimum we must solve the linear system

$$\begin{pmatrix} 4 & -1 \\ -1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -\frac{3}{2} \\ 1 \end{pmatrix}. \tag{12.20}$$

Applying the usual Gaussian Elimination algorithm, only one row operation is required to place the coefficient matrix in upper triangular form:

$$\left(\begin{array}{cc|c} 4 & -1 & -\frac{3}{2} \\ -1 & 3 & 1 \end{array}\right) \longmapsto \left(\begin{array}{cc|c} 4 & -1 & -\frac{3}{2} \\ 0 & \frac{11}{4} & \frac{5}{8} \end{array}\right).$$

The coefficient matrix is regular as no row interchanges were required, and its two pivots, namely $4, \frac{11}{4}$, are both positive. Thus, by Theorem 12.4, $K > 0$ and hence $p(x_1, x_2)$ really does have a minimum, obtained by applying Back Substitution to the reduced system:

$$\mathbf{x}^\star = \begin{pmatrix} x_1^\star \\ x_2^\star \end{pmatrix} = \begin{pmatrix} -\frac{7}{22} \\ \frac{5}{22} \end{pmatrix} \approx \begin{pmatrix} -.31818 \\ .22727 \end{pmatrix}. \tag{12.21}$$

The quickest way to compute the minimal value

$$p(\mathbf{x}^\star) = p\left(-\tfrac{7}{22}, \tfrac{5}{22}\right) = \tfrac{13}{44} \approx .29546$$

is to use the second formula in (12.17).

It is instructive to compare the algebraic solution method with the minimization procedure you learned in multi-variable calculus, cf. [**2**, **34**]. The *critical points* of $p(x_1, x_2)$ are found by setting both partial derivatives equal to zero:

$$\frac{\partial p}{\partial x_1} = 8x_1 - 2x_2 + 3 = 0, \qquad \frac{\partial p}{\partial x_2} = -2x_1 + 6x_2 - 2 = 0.$$

If we divide by an overall factor of 2, these are precisely the *same* linear equations we already constructed in (12.20). Thus, not surprisingly, the calculus approach leads to the same minimizer (12.21). To check whether $\mathbf{x}^\star$ is a (local) minimum, we need to apply the second derivative test. In the case of a function of several variables, this requires analyzing the *Hessian matrix*, which is the symmetric matrix of second order partial derivatives

$$H = \begin{pmatrix} \dfrac{\partial^2 p}{\partial x_1^2} & \dfrac{\partial^2 p}{\partial x_1 \partial x_2} \\ \dfrac{\partial^2 p}{\partial x_1 \partial x_2} & \dfrac{\partial^2 p}{\partial x_2^2} \end{pmatrix} = \begin{pmatrix} 8 & -2 \\ -2 & 6 \end{pmatrix} = 2K,$$

which is exactly twice the quadratic coefficient matrix (12.19). If the Hessian matrix is positive definite — which we already know in this case — then the critical point is indeed a (local) minimum.

Thus, the calculus and algebraic approaches to this minimization problem lead, as they must, to identical results. However, the algebraic method is *more* powerful, because it immediately produces the *unique*, *global* minimum, whereas, barring additional work, calculus can only guarantee that the critical point is a local minimum. Moreover, the proof of the calculus local minimization criterion — that the Hessian matrix be positive definite at the critical point — relies, in fact, on the algebraic solution to the quadratic minimization problem! In summary: minimization of quadratic functions is a problem in linear algebra, while minimizing more complicated functions requires the full force of multivariable calculus.

The most efficient method for producing a minimum of a quadratic function $p(\mathbf{x})$ on $\mathbb{R}^n$, then, is to first write out the symmetric coefficient matrix K and the vector $\mathbf{f}$. Solving the system $K\mathbf{x} = \mathbf{f}$ will produce the minimizer $\mathbf{x}^\star$ *provided* $K > 0$ — which should be checked during the course of the procedure using the criteria of Theorem 12.4, that is, making sure that no row interchanges are used and all the pivots are positive.

Example 12.14. Let us minimize the quadratic function

$$p(x, y, z) = x^2 + 2xy + xz + 2y^2 + yz + 2z^2 + 6y - 7z + 5.$$

This has the matrix form (12.15) with

$$K = \begin{pmatrix} 1 & 1 & \frac{1}{2} \\ 1 & 2 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 2 \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \qquad \mathbf{f} = \begin{pmatrix} 0 \\ -3 \\ \frac{7}{2} \end{pmatrix}, \qquad c = 5.$$

Gaussian Elimination produces the $L\,D\,L^T$ factorization

$$K = \begin{pmatrix} 1 & 1 & \frac{1}{2} \\ 1 & 2 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{7}{4} \end{pmatrix} \begin{pmatrix} 1 & 1 & \frac{1}{2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The pivots, i.e., the diagonal entries of D, are all positive, and hence K is positive definite. Theorem 12.12 then guarantees that $p(x,y,z)$ has a unique minimizer, which is found by solving the linear system $K\,\mathbf{x} = \mathbf{f}$. The solution is then quickly obtained by forward and back substitution:

$$x^\star = 2, \qquad y^\star = -3, \qquad z^\star = 2, \qquad \text{with} \qquad p(x^\star,y^\star,z^\star) = p(2,-3,2) = -11.$$

Theorem 12.12 solves the general quadratic minimization problem when the quadratic coefficient matrix is positive definite. If K is not positive definite, then the quadratic function (12.15) does not have a minimum, apart from one exceptional situation.

Theorem 12.15. *If K is positive definite, then the quadratic function $p(\mathbf{x}) = \mathbf{x}^T K\,\mathbf{x} - 2\,\mathbf{x}^T\mathbf{f} + c$ has a unique global minimizer $\mathbf{x}^\star$ satisfying $K\,\mathbf{x}^\star = \mathbf{f}$. If K is only positive semi-definite, and $\mathbf{f} \in \operatorname{rng} K$, then every solution to the linear system $K\,\mathbf{x}^\star = \mathbf{f}$ is a global minimum of $p(\mathbf{x})$, but the minimum is not unique since $p(\mathbf{x}^\star + \mathbf{z}) = p(\mathbf{x}^\star)$ for any null vector $\mathbf{z} \in \ker K$. In all other cases, $p(\mathbf{x})$ has no global minimum.*

AIMS Lecture Notes 2006

Peter J. Olver

13. Approximation and Interpolation

We will now apply our minimization results to the interpolation and least squares fitting of data and functions.

13.1. Least Squares.

Linear systems with more equations than unknowns typically do not have solutions. In such situations, the least squares solution to a linear system is one means of getting as close as one can to an actual solution.

Definition 13.1. A *least squares solution* to a linear system of equations

$$A\,\mathbf{x} = \mathbf{b} \tag{13.1}$$

is a vector $\mathbf{x}^\star \in \mathbb{R}^n$ that minimizes the Euclidean norm $\|\, A\,\mathbf{x} - \mathbf{b}\,\|$.

If the system (13.1) actually has a solution, then it is automatically the least squares solution. Thus, the concept of least squares solution is new only when the system does not have a solution.

To find the least squares solution, we need to minimize the quadratic function

$$\begin{aligned}\|\, A\,\mathbf{x} - \mathbf{b}\,\|^2 &= (A\,\mathbf{x} - \mathbf{b})^T(A\,\mathbf{x} - \mathbf{b}) = (\mathbf{x}^T A^T - \mathbf{b}^T)(A\,\mathbf{x} - \mathbf{b}) \\ &= \mathbf{x}^T A^T A\,\mathbf{x} - 2\,\mathbf{x}^T A^T\mathbf{b} + \mathbf{b}^T\mathbf{b} = \mathbf{x}^T K\mathbf{x} - 2\,\mathbf{x}^T\mathbf{f} + c,\end{aligned}$$

where

$$K = A^T A, \qquad \mathbf{f} = A^T\mathbf{b}, \qquad c = \|\,\mathbf{b}\,\|^2. \tag{13.2}$$

According to Theorem 12.10, the Gram matrix $K = A^T A$ is positive definite if and only if $\ker A = \{0\}$. In this case, Theorem 12.12 supplies us with the solution to this minimization problem.

Theorem 13.2. *Assume that* $\ker A = \{\mathbf{0}\}$. *Set* $K = A^T A$ *and* $\mathbf{f} = A^T\mathbf{b}$. *Then the least squares solution to the linear system* $A\,\mathbf{x} = \mathbf{b}$ *is the unique solution* $\mathbf{x}^\star$ *to the so-called* normal equations

$$K\,\mathbf{x} = \mathbf{f} \qquad \textit{or, explicitly,} \qquad (A^T A)\,\mathbf{x} = A^T\mathbf{b}, \tag{13.3}$$

namely

$$\mathbf{x}^\star = (A^T A)^{-1} A^T\mathbf{b}. \tag{13.4}$$

The least squares error is

$$\| A\mathbf{x}^\star - \mathbf{b} \|^2 = \| \mathbf{b} \|^2 - \mathbf{f}^T \mathbf{x}^\star = \| \mathbf{b} \|^2 - \mathbf{b}^T A (A^T A)^{-1} A^T \mathbf{b}. \tag{13.5}$$

Note that the normal equations (13.3) can be simply obtained by multiplying the original system $A\mathbf{x} = \mathbf{b}$ on both sides by A^T. In particular, if A is square and invertible, then $(A^T A)^{-1} = A^{-1}(A^T)^{-1}$, and so (13.4) reduces to $\mathbf{x} = A^{-1}\mathbf{b}$, while the two terms in the error formula (13.5) cancel out, producing zero error. In the rectangular case — when inversion is *not* allowed — (13.4) gives a *new* formula for the solution to a compatible linear system $A\mathbf{x} = \mathbf{b}$.

Example 13.3. Consider the linear system

$$\begin{aligned} x_1 + 2x_2 \qquad &= 1, \\ 3x_1 - x_2 + x_3 &= 0, \\ -x_1 + 2x_2 + x_3 &= -1, \\ x_1 - x_2 - 2x_3 &= 2, \\ 2x_1 + x_2 - x_3 &= 2, \end{aligned}$$

consisting of 5 equations in 3 unknowns. The coefficient matrix and right hand side are

$$A = \begin{pmatrix} 1 & 2 & 0 \\ 3 & -1 & 1 \\ -1 & 2 & 1 \\ 1 & -1 & -2 \\ 2 & 1 & -1 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 1 \\ 0 \\ -1 \\ 2 \\ 2 \end{pmatrix}.$$

A direct application of Gaussian Elimination shows that the system is incompatible — it has no solution. Of course, to apply the least squares method, we are not required to check this in advance. If the system has a solution, it is the least squares solution too, and the least squares method will find it.

To form the normal equations (13.3), we compute

$$K = A^T A = \begin{pmatrix} 16 & -2 & -2 \\ -2 & 11 & 2 \\ -2 & 2 & 7 \end{pmatrix}, \qquad \mathbf{f} = A^T \mathbf{b} = \begin{pmatrix} 8 \\ 0 \\ -7 \end{pmatrix}.$$

Solving the 3×3 system $K\mathbf{x} = \mathbf{f}$ by Gaussian Elimination, we find

$$\mathbf{x} = K^{-1}\mathbf{f} \approx (\, .4119, .2482, -.9532\,)^T$$

to be the least squares solution to the system. The least squares error is

$$\| \mathbf{b} - A\mathbf{x}^\star \| \;\approx\; \| (\, -.0917, .0342, .1313, .0701, .0252\,)^T \| \;\approx\; .1799,$$

which is reasonably small — indicating that the system is, roughly speaking, not too incompatible.

13.2. Data Fitting and Interpolation.

One of the most important applications of the least squares minimization process is to the fitting of data points. Suppose we are running an experiment in which we measure a certain time-dependent physical quantity. At time t_i we make the measurement y_i, and thereby obtain a set of, say, m data points

$$(t_1, y_1), \qquad (t_2, y_2), \qquad \ldots \qquad (t_m, y_m). \tag{13.6}$$

Suppose our theory indicates that the data points are supposed to all lie on a single line

$$y = \alpha + \beta t, \tag{13.7}$$

whose precise form — meaning its coefficients α, β — is to be determined. For example, a police car is interested in clocking the speed of a vehicle by using measurements of its relative distance at several times. Assuming that the vehicle is traveling at constant speed, its position at time t will have the linear form (13.7), with β, the velocity, and α, the initial position, to be determined. Experimental error will almost inevitably make this impossible to achieve exactly, and so the problem is to find the straight line (13.7) that "best fits" the measured data and then use its slope to estimate the vehicle's velocity.

The *error* between the measured value y_i and the sample value predicted by the function (13.7) at $t = t_i$ is

$$e_i = y_i - (\alpha + \beta t_i), \qquad\qquad i = 1, \ldots, m.$$

We can write this system of equations in matrix form as

$$\mathbf{e} = \mathbf{y} - A\,\mathbf{x},$$

where

$$\mathbf{e} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}, \qquad \text{while} \qquad A = \begin{pmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_m \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}. \tag{13.8}$$

We call $\mathbf{e}$ the *error vector* and $\mathbf{y}$ the *data vector*. The coefficients α, β of our desired function (13.7) are the unknowns, forming the entries of the column vector $\mathbf{x}$.

If we could fit the data exactly, so $y_i = \alpha + \beta t_i$ for all i, then each $e_i = 0$, and we could solve $A\mathbf{x} = \mathbf{y}$ for the coefficients α, β. In the language of linear algebra, the data points all lie on a straight line if and only if $\mathbf{y} \in \operatorname{rng} A$. If the data points are not collinear, then we seek the straight line that minimizes the total squared error or Euclidean norm

$$\text{Error} = \|\,\mathbf{e}\,\| = \sqrt{e_1^2 + \;\cdots\; + e_m^2}\,.$$

Pictorially, referring to Figure 13.1, the errors are the vertical distances from the points to the line, and we are seeking to minimize the square root of the sum of the squares of

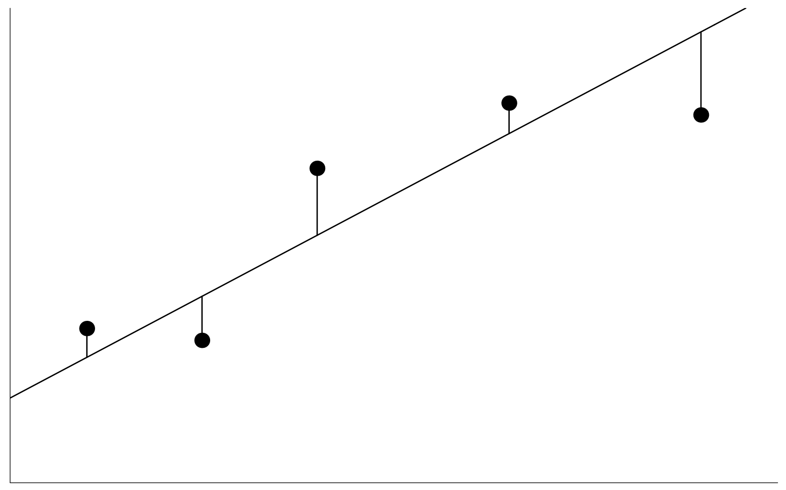


Figure 13.1. Least Squares Approximation of Data by a Straight Line.

the individual errors[†], hence the term *least squares*. In other words, we are looking for the coefficient vector $\mathbf{x} = (\,\alpha, \beta\,)^T$ that minimizes the Euclidean norm of the error vector

$$\|\,\mathbf{e}\,\| = \|\,A\,\mathbf{x} - \mathbf{y}\,\|. \tag{13.9}$$

Thus, we recover the problem of characterizing the least squares solution to the linear system $A\,\mathbf{x} = \mathbf{y}$.

Theorem 13.2 prescribes the solution to this least squares minimization problem. We form the normal equations

$$(A^T A)\,\mathbf{x} = A^T\mathbf{y}, \qquad \text{with solution} \qquad \mathbf{x}^\star = (A^T A)^{-1}A^T\mathbf{y}. \tag{13.10}$$

Invertibility of the Gram matrix $K = A^T A$ relies on the assumption that the matrix A has linearly independent columns. For the particular matrix in (13.8), linear independence of its two columns requires that not all the t_i's be equal, i.e., we must measure the data at at least two distinct times. Note that this restriction does not preclude measuring some of the data at the same time, e.g., by repeating the experiment. However, choosing *all* the t_i's to be the same is a silly data fitting problem. (Why?)

[†] This choice of minimization may strike the reader as a little odd. Why not just minimize the sum of the absolute value of the errors, i.e., the 1 norm $\|\,\mathbf{e}\,\|_1 = |\,e_1\,| + \cdots + |\,e_n\,|$ of the error vector, or minimize the maximal error, i.e., the ∞ norm $\|\,\mathbf{e}\,\|_\infty = \max\{\,|\,e_1\,|, \ldots, |\,e_n\,|\,\}$? Or, even better, why minimize the vertical distance to the line? The perpendicular distance from each data point to the line might strike you as a better measure of error. The answer is that, although each of these alternative minimization criteria is interesting and potentially useful, they all lead to *nonlinear* minimization problems, and so are much harder to solve! The least squares minimization problem can be solved by linear algebra, and so, purely on the grounds of simplicity, is the method of choice in most applications. Moreover, one needs to fully understand the linear problem before diving into more treacherous nonlinear waters.

Under this assumption, we then compute

$$A^T A = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ t_1 & t_2 & \ldots & t_m \end{pmatrix} \begin{pmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_m \end{pmatrix} = \begin{pmatrix} m & \sum t_i \\ \sum t_i & \sum (t_i)^2 \end{pmatrix} = m \begin{pmatrix} 1 & \overline{t} \\ \overline{t} & \overline{t^2} \end{pmatrix},$$

$$A^T \mathbf{y} = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ t_1 & t_2 & \ldots & t_m \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum t_i y_i \end{pmatrix} = m \begin{pmatrix} \overline{y} \\ \overline{t\,y} \end{pmatrix}, \tag{13.11}$$

where the overbars, namely

$$\overline{t} = \frac{1}{m} \sum_{i=1}^m t_i, \qquad \overline{y} = \frac{1}{m} \sum_{i=1}^m y_i, \qquad \overline{t^2} = \frac{1}{m} \sum_{i=1}^m t_i^2, \qquad \overline{t\,y} = \frac{1}{m} \sum_{i=1}^m t_i y_i, \tag{13.12}$$

denote the *average* sample values of the indicated variables.

Warning: The average of a product is *not* equal to the product of the averages! In particular,

$$\overline{t^2} \neq (\overline{t}\,)^2, \qquad \overline{t\,y} \neq \overline{t}\,\overline{y}.$$

Substituting (13.11) into the normal equations (13.10), and canceling the common factor of m, we find that we have only to solve a pair of linear equations

$$\alpha + \overline{t}\,\beta = \overline{y}, \qquad \overline{t}\,\alpha + \overline{t^2}\,\beta = \overline{t\,y},$$

for the coefficients:

$$\alpha = \overline{y} - \overline{t}\,\beta, \qquad \beta = \frac{\overline{t\,y} - \overline{t}\,\overline{y}}{\overline{t^2} - (\overline{t}\,)^2} = \frac{\sum (t_i - \overline{t}\,)\, y_i}{\sum (t_i - \overline{t}\,)^2}\,. \tag{13.13}$$

Therefore, the best (in the least squares sense) straight line that fits the given data is

$$y = \beta\,(t - \overline{t}\,) + \overline{y}, \tag{13.14}$$

where the line's slope β is given in (13.13).

Example 13.4. Suppose the data points are given by the table

t_i	0	1	3	6
y_i	2	3	7	12

To find the least squares line, we construct

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 3 \\ 1 & 6 \end{pmatrix}, \qquad A^T = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & 6 \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} 2 \\ 3 \\ 7 \\ 12 \end{pmatrix}.$$

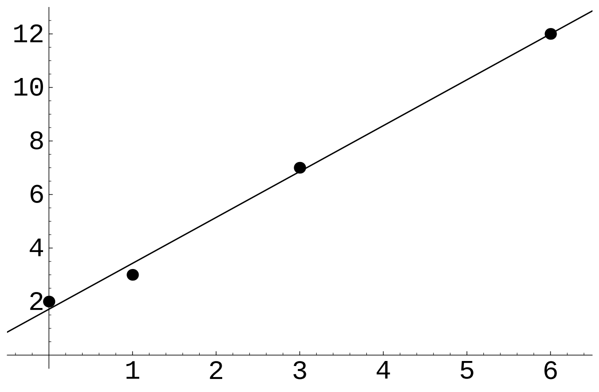


Figure 13.2. Least Squares Line.

Therefore

$$A^T A = \begin{pmatrix} 4 & 10 \\ 10 & 46 \end{pmatrix}, \qquad A^T \mathbf{y} = \begin{pmatrix} 24 \\ 96 \end{pmatrix}.$$

The normal equations (13.10) reduce to

$$4\alpha + 10\beta = 24, \qquad 10\alpha + 46\beta = 96, \qquad \text{so} \qquad \alpha = \tfrac{12}{7}, \qquad \beta = \tfrac{12}{7}.$$

Therefore, the best least squares fit to the data is the straight line

$$y = \tfrac{12}{7} + \tfrac{12}{7}\,t.$$

Alternatively, one can compute this formula directly from (13.13–14). As you can see in Figure 13.2, the least squares line does a fairly good job of approximating the data points.

Example 13.5. Suppose we are given a sample of an unknown radioactive isotope. At time t_i we measure, using a Geiger counter, the amount m_i of radioactive material in the sample. The problem is to determine the initial amount of material along with the isotope's half life. If the measurements were exact, we would have $m(t) = m_0 e^{\beta t}$, where $m_0 = m(0)$ is the initial mass, and $\beta < 0$ the decay rate. The half-life is given by $t^\star = \beta^{-1} \log 2$.

As it stands this is *not* a linear least squares problem. But it can be easily converted to the proper form by taking logarithms:

$$y(t) = \log m(t) = \log m_0 + \beta t = \alpha + \beta t.$$

We can thus do a linear least squares fit on the logarithms $y_i = \log m_i$ of the radioactive mass data at the measurement times t_i to determine the best values for β and $\alpha = \log m_0$.

Polynomial Approximation and Interpolation

The basic least squares philosophy has a variety of different extensions, all interesting and all useful. First, we can replace the straight line (13.7) by a parabola defined by a quadratic function

$$y = \alpha + \beta t + \gamma t^2. \tag{13.15}$$

For example, Newton's theory of gravitation says that (in the absence of air resistance) a falling object obeys the parabolic law (13.15), where $\alpha = h_0$ is the initial height, $\beta = v_0$ is the initial velocity, and $\gamma = -\frac{1}{2}g$ is minus one half the gravitational constant. Suppose we observe a falling body on a new planet, and measure its height y_i at times t_i. Then we can approximate its initial height, initial velocity and gravitational acceleration by finding the parabola (13.15) that best fits the data. Again, we characterize the least squares fit by minimizing the sum of the squares of the individual errors $e_i = y_i - y(t_i)$.

The method can evidently be extended to a completely general polynomial function

$$y(t) = \alpha_0 + \alpha_1 t + \cdots + \alpha_n t^n \tag{13.16}$$

of degree n. The total least squares error between the data and the sample values of the function is equal to

$$\|\mathbf{e}\|^2 = \sum_{i=1}^{m} \big[\, y_i - y(t_i)\,\big]^2 = \|\mathbf{y} - A\mathbf{x}\|^2, \tag{13.17}$$

where

$$A = \begin{pmatrix} 1 & t_1 & t_1^2 & \ldots & t_1^n \\ 1 & t_2 & t_2^2 & \ldots & t_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & t_m^2 & \ldots & t_m^n \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}. \tag{13.18}$$

The coefficient $m \times (n+1)$ coefficient matrix is known as a *Vandermonde matrix*, named after the eighteenth century French mathematician, scientist and musicologist Alexandre–Théophile Vandermonde — despite the fact that it appears nowhere in his four mathematical papers! In particular, if $m = n+1$, then A is square, and so, assuming invertibility, we can solve $A\mathbf{x} = \mathbf{y}$ exactly. In other words, there is no error, and the solution is an *interpolating polynomial*, meaning that it fits the data exactly. A proof of the following result can be found at the end of this section.

Lemma 13.6. *If $t_1, \ldots, t_{n+1}$ are distinct, $t_i \neq t_j$, then the $(n+1)\times(n+1)$ Vandermonde interpolation matrix* (13.18) *is nonsingular.*

This result immediately implies the basic existence theorem for interpolating polynomials.

Theorem 13.7. *Let $t_1, \ldots, t_{n+1}$ be distinct sample points. Then, for any prescribed data $y_1, \ldots, y_{n+1}$, there exists a unique interpolating polynomial of degree $\leq n$ with the prescribed sample values $y(t_i) = y_i$ for all $i = 1, \ldots, n+1$.*

Thus, two points will determine a unique interpolating line, three points a unique interpolating parabola, four points an interpolating cubic, and so on; see Figure 13.3.

The basic ideas of interpolation and least squares fitting of data can be applied to approximate complicated mathematical functions by much simpler polynomials. Such approximation schemes are used in all numerical computations. Your computer or calculator is only able to add, subtract, multiply and divide. Thus, when you ask it to compute $\sqrt{t}$ or

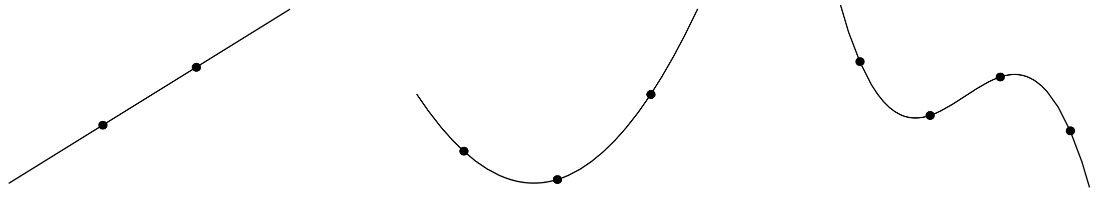


Figure 13.3. Interpolating Polynomials.

e^t or $\cos t$ or any other non-rational function, the program must rely on an approximation scheme based on polynomials[†]. In the "dark ages" before computers, one would consult precomputed tables of values of the function at particular data points. If one needed a value at a nontabulated point, then some form of polynomial interpolation would be used to accurately approximate the intermediate value.

Example 13.8. Suppose that we would like to compute reasonably accurate values for the exponential function e^t for values of t lying in the interval $0 \le t \le 1$ by approximating it by a quadratic polynomial

$$p(t) = \alpha + \beta t + \gamma t^2. \tag{13.19}$$

If we choose 3 points, say $t_1 = 0, t_2 = .5, t_3 = 1$, then there is a unique quadratic polynomial (13.19) that interpolates e^t at the data points, i.e.,

$$p(t_i) = e^{t_i} \qquad \text{for} \qquad i = 1, 2, 3.$$

In this case, the coefficient matrix (13.18), namely

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & .5 & .25 \\ 1 & 1 & 1 \end{pmatrix},$$

is nonsingular. Therefore, we can exactly solve the interpolation equations

$$A\mathbf{x} = \mathbf{y}, \qquad \text{where} \qquad \mathbf{y} = \begin{pmatrix} e^{t_1} \\ e^{t_2} \\ e^{t_3} \end{pmatrix} = \begin{pmatrix} 1. \\ 1.64872 \\ 2.71828 \end{pmatrix}$$

is the data vector, which we assume we already know. The solution

$$\mathbf{x} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} 1. \\ .876603 \\ .841679 \end{pmatrix}$$

[†] Actually, one could also allow interpolation and approximation by rational functions, a subject known as *Padé approximation theory*, [**3**].

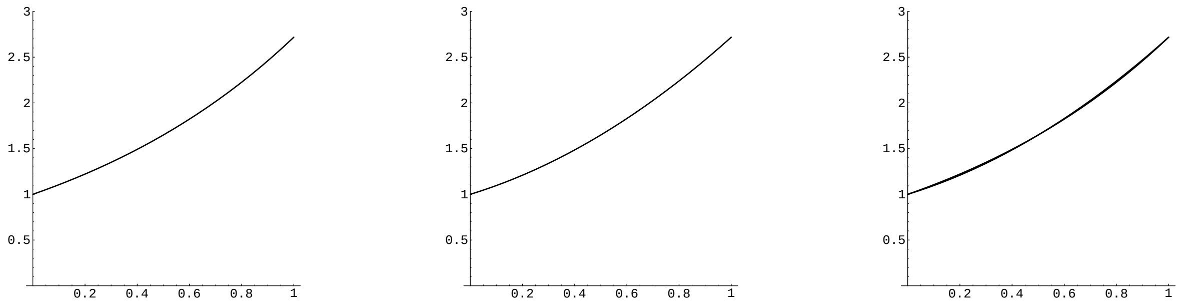


Figure 13.4. Quadratic Interpolating Polynomial for e^t.

yields the interpolating polynomial

$$p(t) = 1 + .876603\, t + .841679\, t^2. \tag{13.20}$$

It is the unique quadratic polynomial that agrees with e^t at the three specified data points. See Figure 13.4 for a comparison of the graphs; the first graph shows e^t, the second $p(t)$, and the third lays the two graphs on top of each other. Even with such a primitive interpolation scheme, the two functions are quite close. The maximum error or L^∞ norm of the difference is

$$\| \, e^t - p(t) \, \|_\infty = \max \left\{ \, | \, e^t - p(t) \, | \; \Big| \; 0 \le t \le 1 \, \right\} \approx .01442,$$

with the largest deviation occurring at $t \approx .796$.

There is, in fact, an explicit formula for the interpolating polynomial that is named after the influential eighteenth century Italo–French mathematician Joseph–Louis Lagrange. Suppose we know the solutions $\mathbf{x}_1, \ldots, \mathbf{x}_{n+1}$ to the particular interpolation systems

$$A\,\mathbf{x}_k = \mathbf{e}_k, \qquad k = 1, \ldots, n+1, \tag{13.21}$$

where $\mathbf{e}_1, \ldots, \mathbf{e}_{n+1}$ are the standard basis vectors of $\mathbb{R}^{n+1}$. Then the solution to

$$A\,\mathbf{x} = \mathbf{y} = y_1\,\mathbf{e}_1 + \;\cdots\; + y_{n+1}\,\mathbf{e}_{n+1}$$

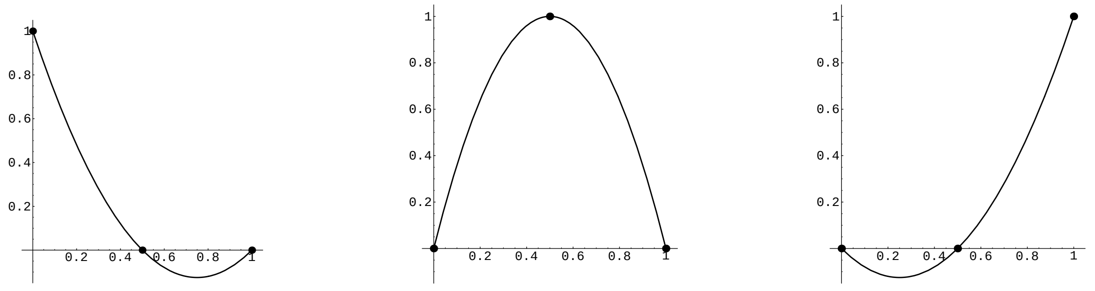
is given by the superposition formula

$$\mathbf{x} = y_1\,\mathbf{x}_1 + \;\cdots\; + y_{n+1}\,\mathbf{x}_{n+1}.$$

The particular interpolation equation (13.21) corresponds to the interpolation data $\mathbf{y} = \mathbf{e}_k$, meaning that $y_k = 1$, while $y_i = 0$ at all points t_i with $i \ne k$. If we can find the $n+1$ particular interpolating polynomials that realize this very special data, we can use superposition to construct the general interpolating polynomial.

Theorem 13.9. *Given distinct sample points* $t_1, \ldots, t_{n+1}$, *the* k^{th} Lagrange interpolating polynomial *is given by*

$$L_k(t) = \frac{(t - t_1) \;\cdots\; (t - t_{k-1})(t - t_{k+1}) \;\cdots\; (t - t_{n+1})}{(t_k - t_1) \;\cdots\; (t_k - t_{k-1})(t_k - t_{k+1}) \;\cdots\; (t_k - t_{n+1})}\,, \qquad k = 1, \ldots, n+1. \tag{13.22}$$



$L_1(t)$ $L_2(t)$ $L_3(t)$

Figure 13.5. Lagrange Interpolating Polynomials for the Points $0, .5, 1$.

It is the unique polynomial of degree n that satisfies

$$L_k(t_i) = \begin{cases} 1, & i = k, \\ 0, & i \neq k, \end{cases} \qquad i, k = 1, \ldots, n+1. \tag{13.23}$$

Proof: The uniqueness of the Lagrange interpolating polynomial is an immediate consequence of Theorem 13.7. To show that (13.22) is the correct formula, we note that when $t = t_i$ for any $i \neq k$, the factor $(t - t_i)$ in the numerator of $L_k(t)$ vanishes, while the denominator is not zero since the points are distinct. On the other hand, when $t = t_k$, the numerator and denominator are equal, and so $L_k(t_k) = 1$. *Q.E.D.*

Theorem 13.10. *If $t_1, \ldots, t_{n+1}$ are distinct, then the polynomial of degree $\leq n$ that interpolates the associated data $y_1, \ldots, y_{n+1}$ is*

$$p(t) = y_1 L_1(t) + \cdots + y_{n+1} L_{n+1}(t). \tag{13.24}$$

Proof: We merely compute

$$p(t_k) = y_1 L_1(t_k) + \cdots + y_k L_k(t) + \cdots + y_{n+1} L_{n+1}(t_k) = y_k,$$

where, according to (13.23), every summand except the kth is zero. *Q.E.D.*

Example 13.11. For example, the three quadratic Lagrange interpolating polynomials for the values $t_1 = 0, t_2 = \frac{1}{2}, t_3 = 1$ used to interpolate e^t in Example 13.8 are

$$\begin{aligned} L_1(t) &= \frac{\left(t - \frac{1}{2}\right)(t-1)}{\left(0 - \frac{1}{2}\right)(0-1)} = 2t^2 - 3t + 1, \\ L_2(t) &= \frac{(t-0)(t-1)}{\left(\frac{1}{2} - 0\right)\left(\frac{1}{2} - 1\right)} = -4t^2 + 4t, \\ L_3(t) &= \frac{(t-0)\left(t - \frac{1}{2}\right)}{(1-0)\left(1 - \frac{1}{2}\right)} = 2t^2 - t. \end{aligned} \tag{13.25}$$

Thus, we can rewrite the quadratic interpolant (13.20) to e^t as

$$\begin{aligned} y(t) &= L_1(t) + e^{1/2} L_2(t) + e\, L_3(t) \\ &= (2t^2 - 3t + 1) + 1.64872(-4t^2 + 4t) + 2.71828(2t^2 - t). \end{aligned}$$

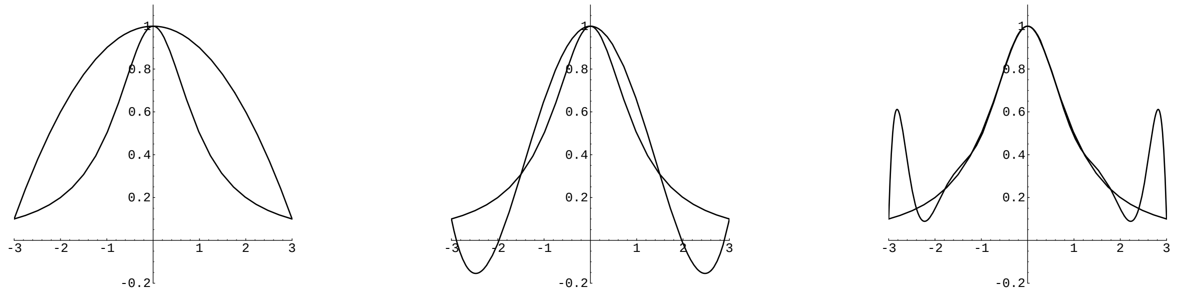


Figure 13.6. Degree 2, 4 and 10 Interpolating Polynomials for $1/(1+t^2)$.

We stress that this is the *same* interpolating polynomial — we have merely rewritten it in the more transparent Lagrange form.

You might expect that the higher the degree, the more accurate the interpolating polynomial. This expectation turns out, unfortunately, not to be uniformly valid. While low degree interpolating polynomials are usually reasonable approximants to functions, high degree interpolants are not only more expensive to compute, but can be rather badly behaved, particularly near the ends of the interval. For example, Figure 13.6 displays the degree $2, 4$ and 10 interpolating polynomials for the function $1/(1+t^2)$ on the interval $-3 \le t \le 3$ using equally spaced data points. Note the rather poor approximation of the function near the ends of the interval. Higher degree interpolants fare even worse, although the bad behavior becomes more and more concentrated near the endpoints. As a consequence, high degree polynomial interpolation tends not to be used in practical applications. Better alternatives rely on least squares approximants by low degree polynomials, to be described next, and interpolation by piecewise cubic splines, a topic that will be discussed in depth later.

If we have $m > n+1$ data points, then, usually, there is no degree n polynomial that fits all the data, and so we must switch over to a least squares approximation. The first requirement is that the associated $m \times (n+1)$ interpolation matrix (13.18) has rank $n+1$; this follows from Lemma 13.6, provided that at least $n+1$ of the values $t_1, \ldots, t_m$ are distinct. Thus, given data at $m \ge n+1$ different sample points $t_1, \ldots, t_m$, we can uniquely determine the best least squares polynomial of degree n that fits the data by solving the normal equations (13.10).

Example 13.12. Let us return to the problem of approximating the exponential function e^t. If we use more than three data points, but still require a quadratic polynomial, then we can no longer interpolate exactly, and must devise a least squares approximant. For instance, using five equally spaced sample points $t_1 = 0$, $t_2 = .25$, $t_3 = .5$, $t_4 = .75$, $t_5 = 1$, the coefficient matrix and sampled data vector (13.18) are

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & .25 & .0625 \\ 1 & .5 & .25 \\ 1 & .75 & .5625 \\ 1 & 1 & 1 \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} 1. \\ 1.28403 \\ 1.64872 \\ 2.11700 \\ 2.71828 \end{pmatrix}.$$

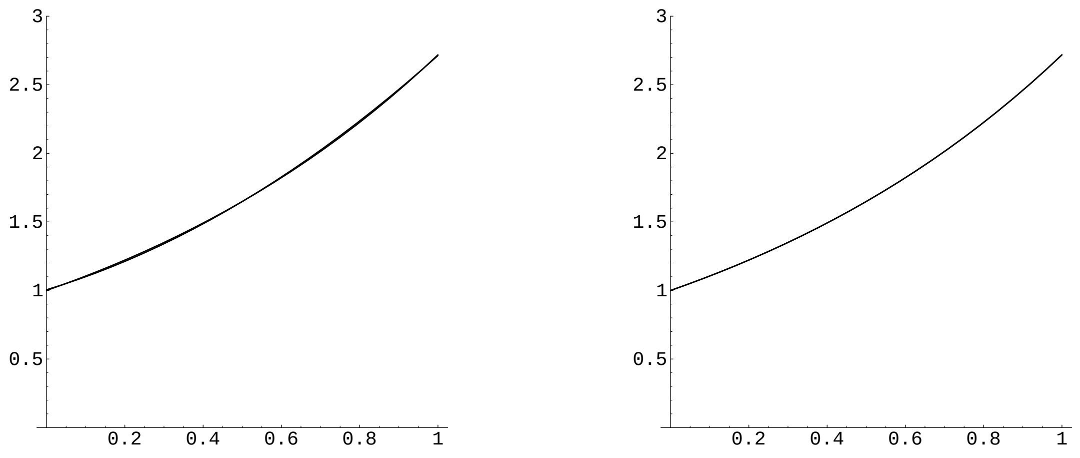


Figure 13.7. Quadratic Approximant and Quartic Interpolant for e^t.

The solution to the normal equations (13.3), with

$$K = A^T A = \begin{pmatrix} 5. & 2.5 & 1.875 \\ 2.5 & 1.875 & 1.5625 \\ 1.875 & 1.5625 & 1.38281 \end{pmatrix}, \qquad \mathbf{f} = A^T \mathbf{y} = \begin{pmatrix} 8.76803 \\ 5.45140 \\ 4.40153 \end{pmatrix},$$

is

$$\mathbf{x} = K^{-1}\mathbf{f} = (\,1.00514, .864277, .843538\,)^T .$$

This leads to the quadratic least squares approximant

$$p_2(t) = 1.00514 + .864277\,t + .843538\,t^2.$$

On the other hand, the quartic interpolating polynomial

$$p_4(t) = 1 + .998803\,t + .509787\,t^2 + .140276\,t^3 + .069416\,t^4$$

is found directly from the data values as above. The quadratic polynomial has a maximal error of $\approx .011$ over the interval $[0,1]$ — slightly better than the quadratic interpolant — while the quartic has a significantly smaller maximal error: $\approx .0000527$. (In this case, high degree interpolants are not ill behaved.) See Figure 13.7 for a comparison of the graphs.

Proof of Lemma 13.6: We will establish the rather striking LU factorization of the transposed Vandermonde matrix $V = A^T$, which will immediately prove that, when $t_1, \ldots, t_{n+1}$ are distinct, both V and A are nonsingular matrices. The 4×4 case is instructive for understanding the general pattern. Applying regular Gaussian Elimination, we find the explicit LU factorization

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ t_1 & t_2 & t_3 & t_4 \\ t_1^2 & t_2^2 & t_3^2 & t_4^2 \\ t_1^3 & t_2^3 & t_3^3 & t_4^3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ t_1 & 1 & 0 & 0 \\ t_1^2 & t_1 + t_2 & 1 & 0 \\ t_1^3 & t_1^2 + t_1 t_2 + t_2^2 & t_1 + t_2 + t_3 & 1 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & t_2 - t_1 & t_3 - t_1 & t_4 - t_1 \\ 0 & 0 & (t_3 - t_1)(t_3 - t_2) & (t_4 - t_1)(t_4 - t_2) \\ 0 & 0 & 0 & (t_4 - t_1)(t_4 - t_2)(t_4 - t_3) \end{pmatrix}.$$

In the general $(n+1) \times (n+1)$ case, the individual entries of the matrices appearing in factorization $V = LU$ are

$$v_{ij} = t_j^{i-1}, \qquad i, j = 1, \ldots, n+1, \tag{13.26}$$

$$\ell_{ij} = \sum_{1 \le k_1 \le \cdots \le k_{i-j} \le j} t_{k_1} t_{k_2} \cdots t_{k_{i-j}}, \quad 1 \le j < i \le n+1, \qquad \begin{aligned} \ell_{ii} &= 1, \quad i = 1, \ldots, n+1, \\ \ell_{ij} &= 0, \quad 1 \le i < j \le n+1, \end{aligned}$$

$$u_{ij} = \prod_{k=1}^{i} (t_j - t_k), \qquad 1 < i \le j \le n+1, \qquad \begin{aligned} u_{1j} &= 1, \quad j = 1, \ldots, n+1, \\ u_{ij} &= 0, \quad 1 \le j < i \le n+1. \end{aligned}$$

Full details of the proof that $V = LU$ can be found in [**20**, **41**]. (Surprisingly, as far as we know, these are the first places this factorization appears in the literature.) The entries of L lying below the diagonal are known as the *complete monomial polynomials* since ℓ_{ij} is obtained by summing, with unit coefficients, all monomials of degree $i-j$ in the j variables $t_1, \ldots, t_j$. The entries of U appearing on or above the diagonal are known as the *Newton difference polynomials*. In particular, if $t_1, \ldots, t_n$ are distinct, so $t_i \ne t_j$ for $i \ne j$, all entries of U lying on or above the diagonal are nonzero. In this case, V has all nonzero pivots, and is a regular, hence nonsingular matrix. *Q.E.D.*

Approximation and Interpolation by General Functions

There is nothing special about polynomial functions in the preceding approximation scheme. For example, suppose we were interested in finding the best trigonometric approximation

$$y = \alpha_1 \cos t + \alpha_2 \sin t$$

to a given set of data. Again, the least squares error takes the same form $\| \mathbf{y} - A \mathbf{x} \|^2$ as in (13.17), where

$$A = \begin{pmatrix} \cos t_1 & \sin t_1 \\ \cos t_2 & \sin t_2 \\ \vdots & \vdots \\ \cos t_m & \sin t_m \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}.$$

Thus, the columns of A are the sampled values of the functions $\cos t, \sin t$. The key is that the unspecified parameters — in this case α_1, α_2 — occur *linearly* in the approximating function. Thus, the most general case is to approximate the data (13.6) by a linear combination

$$y(t) = \alpha_1 h_1(t) + \alpha_2 h_2(t) + \cdots + \alpha_n h_n(t)$$

of prescribed functions $h_1(x), \ldots, h_n(x)$. The least squares error is, as always, given by

$$\text{Error} = \sqrt{\sum_{i=1}^{m} \bigl(y_i - y(t_i) \bigr)^2} = \| \mathbf{y} - A\mathbf{x} \|,$$

where the sample matrix A, the vector of unknown coefficients $\mathbf{x}$, and the data vector $\mathbf{y}$ are

$$A = \begin{pmatrix} h_1(t_1) & h_2(t_1) & \ldots & h_n(t_1) \\ h_1(t_2) & h_2(t_2) & \ldots & h_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(t_m) & h_2(t_m) & \ldots & h_n(t_m) \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}. \tag{13.27}$$

If A is square and nonsingular, then we can find an interpolating function of the prescribed form by solving the linear system

$$A\mathbf{x} = \mathbf{y}. \tag{13.28}$$

A particularly important case is provided by the $2n+1$ trigonometric functions

$$1, \qquad \cos x, \qquad \sin x, \qquad \cos 2x, \qquad \sin 2x, \qquad \ldots \qquad \cos nx, \qquad \sin nx.$$

Interpolation on $2n+1$ equally spaced data points on the interval $[0, 2\pi]$ leads to the Discrete Fourier Transform, used in signal processing, data transmission, and compression.

If there are more than n data points, then we cannot, in general, interpolate exactly, and must content ourselves with a least squares approximation that minimizes the error at the sample points as best it can. The least squares solution to the interpolation equations (13.28) is found by solving the associated normal equations $K\mathbf{x} = \mathbf{f}$, where the (i,j) entry of $K = A^T A$ is m times the average sample value of the product of $h_i(t)$ and $h_j(t)$, namely

$$k_{ij} = m\, \overline{h_i(t)\, h_j(t)} = \sum_{\kappa=1}^{m} h_i(t_\kappa)\, h_j(t_\kappa), \tag{13.29}$$

whereas the i^{th} entry of $\mathbf{f} = A^T \mathbf{y}$ is

$$f_i = m\, \overline{h_i(t)\, y} = \sum_{\kappa=1}^{m} h_i(t_\kappa)\, y_\kappa. \tag{13.30}$$

The one issue is whether the columns of the sample matrix A are linearly independent. This is more subtle than the polynomial case covered by Lemma 13.6. Linear independence of the sampled function vectors is, in general, more restrictive than merely requiring the functions themselves to be linearly independent.

If the parameters do not occur linearly in the functional formula, then we cannot use linear algebra to effect a least squares approximation. For example, one cannot determine the frequency ω, the amplitude r, *and* the phase shift δ of the general trigonometric approximation

$$y = c_1 \cos \omega t + c_2 \sin \omega t = r \cos(\omega t + \delta)$$

that minimizes the least squares error at the sample points. Approximating data by such a function constitutes a *nonlinear* minimization problem.

Weighted Least Squares

Another extension to the basic least squares method is to introduce weights in the measurement of the error. Suppose some of the data is known to be more reliable or more significant than others. For example, measurements at an earlier time may be more accurate, or more critical to the data fitting problem, than later measurements. In that situation, we should penalize any errors in the earlier measurements and downplay errors in the later data.

In general, this requires the introduction of a positive weight $c_i > 0$ associated to each data point (t_i, y_i); the larger the weight, the more vital the error. For a straight line approximation $y = \alpha + \beta t$, the *weighted least squares error* is defined as

$$\text{Error} = \sqrt{\sum_{i=1}^{m} c_i e_i^2} = \sqrt{\sum_{i=1}^{m} c_i \left[y_i - (\alpha + \beta t_i) \right]^2}\,.$$

Let us rewrite this formula in matrix form. Let $C = \operatorname{diag}(c_1, \ldots, c_m)$ denote the diagonal *weight matrix*. Note that $C > 0$ is positive definite, since all the weights are positive. The least squares error,

$$\text{Error} = \sqrt{\mathbf{e}^T C \mathbf{e}} = \| \mathbf{e} \|,$$

is then the norm of the error vector $\mathbf{e}$ with respect to the weighted inner product $\langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v}^T C \mathbf{w}$. Since $\mathbf{e} = \mathbf{y} - A\mathbf{x}$,

$$\begin{aligned} \| \mathbf{e} \|^2 = \| A\mathbf{x} - \mathbf{y} \|^2 &= (A\mathbf{x} - \mathbf{y})^T C (A\mathbf{x} - \mathbf{y}) \\ &= \mathbf{x}^T A^T C A \mathbf{x} - 2\mathbf{x}^T A^T C \mathbf{y} + \mathbf{y}^T C \mathbf{y} = \mathbf{x}^T K \mathbf{x} - 2\mathbf{x}^T \mathbf{f} + c, \end{aligned} \tag{13.31}$$

where

$$K = A^T C A, \qquad \mathbf{f} = A^T C \mathbf{y}, \qquad c = \mathbf{y}^T C \mathbf{y} = \| \mathbf{y} \|^2.$$

Note that K is the weighted Gram matrix derived in (12.10), and so is positive definite provided A has linearly independent columns or, equivalently, has rank n.

Theorem 13.13. *Suppose A is an $m \times n$ matrix with linearly independent columns. Suppose $C > 0$ is any positive definite $m \times m$ matrix. Then, the quadratic function* (13.31) *giving the weighted least squares error has a unique minimizer, which is the solution to the* weighted normal equations

$$A^T C A \mathbf{x} = A^T C \mathbf{y}, \qquad \textit{so that} \qquad \mathbf{x} = (A^T C A)^{-1} A^T C \mathbf{y}. \tag{13.32}$$

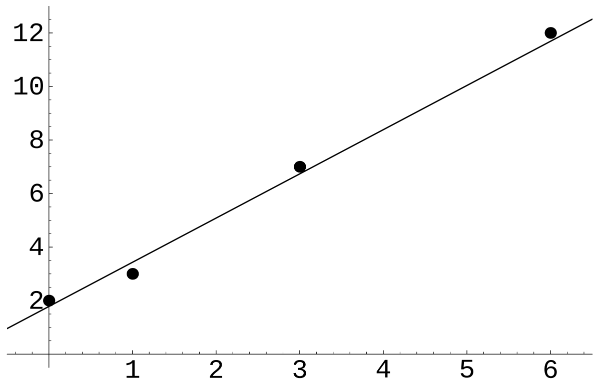


Figure 13.8. Weighted Least Squares Line.

In brief, the weighted least squares solution is obtained by multiplying both sides of the original system $A\mathbf{x} = \mathbf{y}$ by the matrix $A^T C$. The derivation of this result allows $C > 0$ to be *any* positive definite matrix. In applications, the off-diagonal entries of C can be used to weight cross-correlation terms in the data, although this extra freedom is rarely used in practice.

Example 13.14. In Example 13.4, we fit the following data

t_i	0	1	3	6
y_i	2	3	7	12
c_i	3	2	$\frac{1}{2}$	$\frac{1}{4}$

with an unweighted least squares line. Now we shall assign the weights listed in the last row of the table for the error at each sample point. Thus, errors in the first two data values carry more weight than the latter two. To find the weighted least squares line $y = \alpha + \beta t$ that best fits the data, we compute

$$A^T C A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & 6 \end{pmatrix} \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 3 \\ 1 & 6 \end{pmatrix} = \begin{pmatrix} \frac{23}{4} & 5 \\ 5 & \frac{31}{2} \end{pmatrix},$$

$$A^T C \mathbf{y} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & 6 \end{pmatrix} \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} 2 \\ 3 \\ 7 \\ 12 \end{pmatrix} = \begin{pmatrix} \frac{37}{2} \\ \frac{69}{2} \end{pmatrix}.$$

Thus, the weighted normal equations (13.32) reduce to

$$\tfrac{23}{4}\alpha + 5\beta = \tfrac{37}{2}, \qquad 5\alpha + \tfrac{31}{2}\beta = \tfrac{69}{2}, \qquad \text{so} \qquad \alpha = 1.7817, \qquad \beta = 1.6511.$$

Therefore, the least squares fit to the data under the given weights is

$$y = 1.7817 + 1.6511\,t,$$

as plotted in Figure 13.8.

13.3. Splines.

Polynomials are but one of the options for interpolating data points by smooth functions. In pre–CAD (computer aided design) draftsmanship, a *spline* was a long, thin, flexible strip of wood that was used to draw a smooth curve through prescribed points. The points were marked by small pegs, and the spline rested on the pegs. The mathematical theory of splines was first developed in the 1940's by the Romanian mathematician Isaac Schoenberg as an attractive alternative to polynomial interpolation and approximation. Splines have since become ubiquitous in numerical analysis, in geometric modeling, in design and manufacturing, in computer graphics and animation, and in many other applications.

We suppose that the spline coincides with the graph of a function $y = u(x)$. The pegs are fixed at the prescribed data points $(x_0, y_0), \ldots, (x_n, y_n)$, and this requires $u(x)$ to satisfy the interpolation conditions

$$u(x_j) = y_j, \qquad\qquad j = 0, \ldots, n. \tag{13.33}$$

The *mesh points* $x_0 < x_1 < x_2 < \cdots < x_n$ are distinct and labeled in increasing order. The spline is modeled as an elastic beam, [**38**], and so

$$u(x) = a_j + b_j\,(x - x_j) + c_j\,(x - x_j)^2 + d_j\,(x - x_j)^3, \qquad \begin{array}{c} x_j \le x \le x_{j+1}, \\ j = 0, \ldots, n-1, \end{array} \tag{13.34}$$

is a piecewise cubic function — meaning that, between successive mesh points, it is a cubic polynomial, but not necessarily the same cubic on each subinterval. The fact that we write the formula (13.34) in terms of $x - x_j$ is merely for computational convenience.

Our problem is to determine the coefficients

$$a_j, \qquad b_j, \qquad c_j, \qquad d_j, \qquad\qquad j = 0, \ldots, n-1.$$

Since there are n subintervals, there are a total of $4n$ coefficients, and so we require $4n$ equations to uniquely prescribe them. First, we need the spline to satisfy the interpolation conditions (13.33). Since it is defined by a different formula on each side of the mesh point, this results in a total of $2n$ conditions:

$$\begin{aligned} u(x_j^+) &= a_j = y_j, \\ u(x_{j+1}^-) &= a_j + b_j\,h_j + c_j\,h_j^2 + d_j\,h_j^3 = y_{j+1}, \end{aligned} \qquad j = 0, \ldots, n-1, \tag{13.35}$$

where we abbreviate the length of the $j^{\rm th}$ subinterval by

$$h_j = x_{j+1} - x_j.$$

The next step is to require that the spline be as smooth as possible. The interpolation conditions (13.35) guarantee that $u(x)$ is continuous. The condition $u(x) \in \mathrm{C}^1$ be continuously differentiable requires that $u'(x)$ be continuous at the interior mesh points $x_1, \ldots, x_{n-1}$, which imposes the $n - 1$ additional conditions

$$b_j + 2\,c_j\,h_j + 3\,d_j\,h_j^2 = u'(x_{j+1}^-) = u'(x_{j+1}^+) = b_{j+1}, \qquad j = 0, \ldots, n-2. \tag{13.36}$$

To make $u \in \mathrm{C}^2$, we impose $n-1$ further conditions

$$2\,c_j + 6\,d_j\,h_j = u''(x_{j+1}^-) = u''(x_{j+1}^+) = 2\,c_{j+1}, \qquad j = 0, \ldots, n-2, \tag{13.37}$$

to ensure that u'' is continuous at the mesh points. We have now imposed a total of $4n-2$ conditions, namely (13.35–37), on the $4n$ coefficients. The two missing constraints will come from boundary conditions at the two endpoints, namely x_0 and x_n. There are three common types:

(i) *Natural boundary conditions*: $u''(x_0) = u''(x_n) = 0$, whereby

$$c_0 = 0, \qquad c_{n-1} + 3\,d_{n-1}\,h_{n-1} = 0. \tag{13.38}$$

Physically, this models a simply supported spline that rests freely on the first and last pegs.

(ii) *Clamped boundary conditions*: $u'(x_0) = \alpha,\ \ u'(x_n) = \beta$, where α, β, which could be 0, are fixed by the user. This requires

$$b_0 = \alpha, \qquad b_{n-1} + 2\,c_{n-1}\,h_{n-1} + 3\,d_{n-1}\,h_{n-1}^2 = \beta. \tag{13.39}$$

This corresponds to clamping the spline at prescribed angles at each end.

(iii) *Periodic boundary conditions*: $u'(x_0) = u'(x_n),\ \ u''(x_0) = u''(x_n)$, so that

$$b_0 = b_{n-1} + 2\,c_{n-1}\,h_{n-1} + 3\,d_{n-1}\,h_{n-1}^2, \qquad c_0 = c_{n-1} + 3\,d_{n-1}\,h_{n-1}. \tag{13.40}$$

If we also require that the end interpolation values agree,

$$u(x_0) = y_0 = y_n = u(x_n), \tag{13.41}$$

then the resulting spline will be a periodic C^2 function, so $u(x+p) = u(x)$ with $p = x_n - x_0$ for all x. The periodic case is used to draw smooth closed curves; see below.

Theorem 13.15. *Suppose we are given mesh points $a = x_0 < x_1 < \cdots < x_n = b$, and corresponding data values $y_0, y_1, \ldots, y_n$, along with one of the three kinds of boundary conditions* (13.38), (13.39), *or* (13.40). *Then there exists a unique piecewise cubic spline function $u(x) \in \mathrm{C}^2[a, b]$ that interpolates the data, $u(x_0) = y_0, \ldots, u(x_n) = y_n$, and satisfies the boundary conditions.*

Proof: We first discuss the natural case. The clamped case is left as an exercise for the reader, while the slightly harder periodic case will be treated at the end of the section. The first set of equations in (13.35) says that

$$a_j = y_j, \qquad j = 0, \ldots, n-1. \tag{13.42}$$

Next, (13.37–38) imply that

$$d_j = \frac{c_{j+1} - c_j}{3\,h_j}\,. \tag{13.43}$$

This equation also holds for $j = n-1$, provided that we make the convention that[†]

$$c_n = 0.$$

[†] This is merely for convenience; there is no c_n used in the formula for the spline.

We now substitute (13.42–43) into the second set of equations in (13.35), and then solve the resulting equation for

$$b_j = \frac{y_{j+1} - y_j}{h_j} - \frac{(2c_j + c_{j+1})\, h_j}{3}\,. \tag{13.44}$$

Substituting this result and (13.43) back into (13.36), and simplifying, we find

$$h_j\, c_j + 2\,(h_j + h_{j+1})\, c_{j+1} + h_{j+1}\, c_{j+2} = 3 \left[\frac{y_{j+2} - y_{j+1}}{h_{j+1}} - \frac{y_{j+1} - y_j}{h_j} \right] = z_{j+1}, \tag{13.45}$$

where we introduce z_{j+1} as a shorthand for the quantity on the right hand side.

In the case of natural boundary conditions, we have

$$c_0 = 0, \qquad\qquad c_n = 0,$$

and so (13.45) constitutes a tridiagonal linear system

$$A\,\mathbf{c} = \mathbf{z}, \tag{13.46}$$

for the unknown coefficients $\mathbf{c} = \left(c_1, c_2, \ldots, c_{n-1} \right)^T$, with coefficient matrix

$$A = \begin{pmatrix} 2\,(h_0 + h_1) & h_1 & & & & \\ h_1 & 2\,(h_1 + h_2) & h_2 & & & \\ & h_2 & 2\,(h_2 + h_3) & h_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-3} & 2\,(h_{n-3} + h_{n-2}) & h_{n-2} \\ & & & & h_{n-2} & 2\,(h_{n-2} + h_{n-1}) \end{pmatrix} \tag{13.47}$$

and right hand side $\mathbf{z} = \left(z_1, z_2, \ldots, z_{n-1} \right)^T$. Once (13.47) has been solved, we will then use (13.42–44) to reconstruct the other spline coefficients a_j, b_j, d_j.

The key observation is that the coefficient matrix A is *strictly diagonally dominant*, cf. Definition 6.25, because all the $h_j > 0$, and so

$$2\,(h_{j-1} + h_j) > h_{j-1} + h_j.$$

Theorem 6.26 implies that A is nonsingular, and hence the tridiagonal linear system has a unique solution $\mathbf{c}$. This suffices to prove the theorem in the case of natural boundary conditions. *Q.E.D.*

To actually solve the linear system (13.46), we can apply our tridiagonal solution algorithm (4.47). Let us specialize to the most important case, when the mesh points are equally spaced in the interval $[a, b]$, so that

$$x_j = a + j\,h, \qquad \text{where} \qquad h = h_j = \frac{b-a}{n}\,, \qquad\quad j = 0, \ldots, n-1.$$

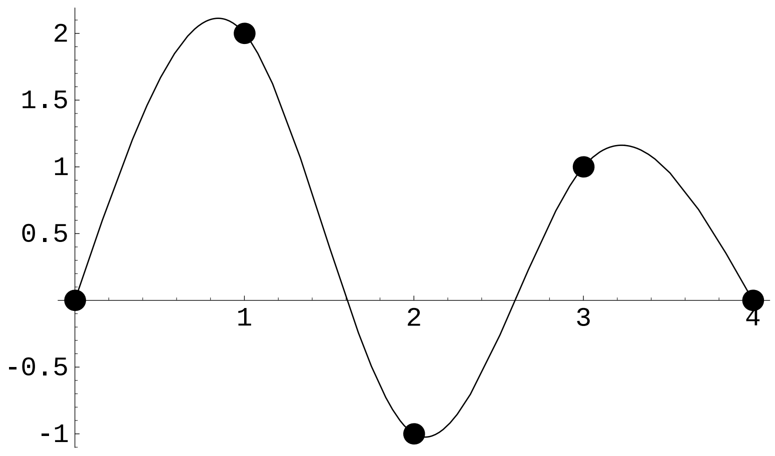


Figure 13.9. A Cubic Spline.

In this case, the coefficient matrix $A = hB$ is equal to h times the tridiagonal matrix

$$B = \begin{pmatrix} 4 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & 1 & 4 & 1 & \\ & & & 1 & 4 & 1 \\ & & & & \ddots & \ddots & \ddots \end{pmatrix}$$

that first appeared in Example 4.26. Its LU factorization takes on an especially simple form, since most of the entries of L and U are essentially the same decimal numbers. This makes the implementation of the Forward and Back Substitution procedures almost trivial.

Figure 13.9 shows a particular example — a natural spline passing through the data points $(0,0)$, $(1,2)$, $(2,-1)$, $(3,1)$, $(4,0)$. As with the Green's function for the beam, the human eye is unable to discern the discontinuities in its third derivatives, and so the graph appears completely smooth, even though it is, in fact, only C^2.

In the periodic case, we set

$$a_{n+k} = a_n, \qquad b_{n+k} = b_n, \qquad c_{n+k} = c_n, \qquad d_{n+k} = d_n, \qquad z_{n+k} = z_n.$$

With this convention, the basic equations (13.42–45) are the same. In this case, the coefficient matrix for the linear system

$$A\,\mathbf{c} = \mathbf{z}, \qquad \text{with} \qquad \mathbf{c} = \left(c_0, c_1, \ldots, c_{n-1}\right)^T, \qquad \mathbf{z} = \left(z_0, z_1, \ldots, z_{n-1}\right)^T,$$

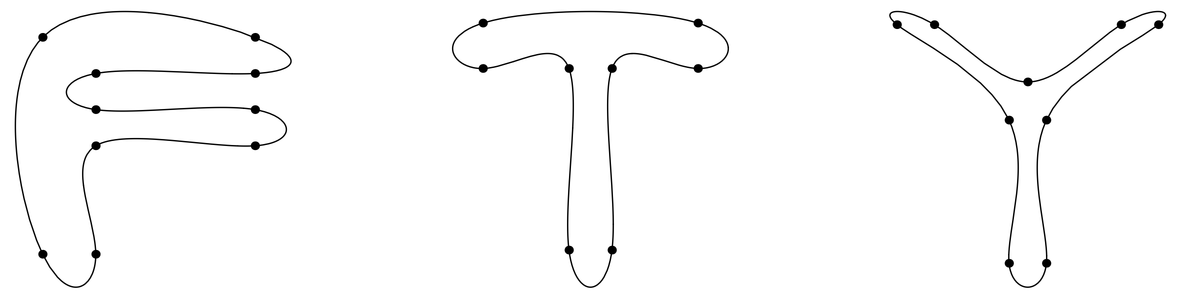


Figure 13.10. Three Sample Spline Letters.

is of *circulant tridiagonal* form:

$$A = \begin{pmatrix} 2\,(h_{n-1}+h_0) & h_0 & & & & h_{n-1} \\ h_0 & 2\,(h_0+h_1) & h_1 & & & \\ & h_1 & 2\,(h_1+h_2) & h_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-3} & 2\,(h_{n-3}+h_{n-2}) & h_{n-2} \\ h_{n-1} & & & & h_{n-2} & 2\,(h_{n-2}+h_{n-1}) \end{pmatrix}. \qquad (13.48)$$

Again A is strictly diagonally dominant, and so there is a unique solution $\mathbf{c}$, from which one reconstructs the spline, proving Theorem 13.15 in the periodic case.

One immediate application of splines is curve fitting in computer aided design and graphics. The basic problem is to draw a smooth parametrized curve $\mathbf{u}(t) = (\,u(t), v(t)\,)^T$ that passes through a set of prescribed data points $\mathbf{x}_k = (\,x_k, y_k\,)^T$ in the plane. We have the freedom to choose the parameter value $t = t_k$ when the curve passes through the k^{th} point; the simplest and most common choice is to set $t_k = k$. We then construct the functions $x = u(t)$ and $y = v(t)$ as cubic splines interpolating the x and y coordinates of the data points, so $u(t_k) = x_k$, $v(t_k) = y_k$. For smooth closed curves, we require that both splines be periodic; for curves with ends, either natural or clamped boundary conditions are used.

Most computer graphics packages include one or more implementations of parametrized spline curves. The same idea also underlies modern font design for laser printing and typography (including the fonts used in this book). The great advantage of spline fonts over their bitmapped counterparts is that they can be readily scaled. Some sample letter shapes parametrized by periodic splines passing through the indicated data points are plotted in Figure 13.10. Better fits can be easily obtained by increasing the number of data points. Various extensions of the basic spline algorithms to space curves and surfaces are an essential component of modern computer graphics, design, and animation, [**15**, **44**].

AIMS Lecture Notes 2006

Peter J. Olver

14. Finite Elements

In this part, we introduce the powerful finite element method for finding numerical approximations to the solutions to boundary value problems involving both ordinary and partial differential equations can be solved by direct integration. The method relies on the characterization of the solution as the minimizer of a suitable quadratic functional. The innovative idea is to restrict the infinite-dimensional minimization principle characterizing the exact solution to a suitably chosen finite-dimensional subspace of the function space. When properly formulated, the solution to the resulting finite-dimensional minimization problem approximates the true minimizer. The finite-dimensional minimizer is found by solving the induced linear algebraic system, using either direct or iterative methods. We begin with one-dimensional boundary value problems involving ordinary differential equations, and, in the final section, show how to adapt the finite element analysis to partial differential equations, specifically the two-dimensional Laplace and Poisson equations.

14.1. Finite Elements for Ordinary Differential Equations.

The characterization of the solution to a linear boundary value problem via a quadratic minimization principle inspires a very powerful and widely used numerical solution scheme, known as the *finite element method*. In this final section, we give a brief introduction to the finite element method in the context of one-dimensional boundary value problems involving ordinary differential equations.

The underlying idea is strikingly simple. We are trying to find the solution to a boundary value problem by minimizing a quadratic functional $\mathcal{P}[u]$ on an infinite-dimensional vector space U. The solution $u_\star \in U$ to this minimization problem is found by solving a differential equation subject to specified boundary conditions. However, minimizing the functional on a *finite-dimensional subspace* $W \subset U$ is a problem in linear algebra, and, moreover, one that we already know how to solve! Of course, restricting the functional $\mathcal{P}[u]$ to the subspace W will not, barring luck, lead to the exact minimizer. Nevertheless, if we choose W to be a sufficiently "large" subspace, the resulting minimizer $w_\star \in W$ may very well provide a reasonable approximation to the actual solution $u_\star \in U$. A rigorous justification of this process, under appropriate hypotheses, requires a full analysis of the finite element method, and we refer the interested reader to [**45**, **49**]. Here we shall concentrate on trying to understand how to apply the method in practice.

To be a bit more explicit, consider the minimization principle

$$\mathcal{P}[u] = \tfrac{1}{2}\, \| \, L[u] \, \|^2 - \langle \, f \, , u \, \rangle \tag{14.1}$$

for the linear system

$$K[u] = f, \qquad \text{where} \qquad K = L^* \circ L,$$

representing our boundary value problem. The norm in (14.1) is typically based on some form of weighted inner product $\langle\!\langle \, v \, , \widetilde{v} \, \rangle\!\rangle$ on the space of strains $v = L[u] \in V$, while the inner product term $\langle \, f \, , u \, \rangle$ is typically (although not necessarily) unweighted on the space of displacements $u \in U$. The linear operator takes the self-adjoint form $K = L^* \circ L$, and must be positive definite — which requires $\ker L = \{0\}$. Without the positivity assumption, the boundary value problem has either no solutions, or infinitely many; in either event, the basic finite element method will not apply.

Rather than try to minimize $\mathcal{P}[u]$ on the entire function space U, we now seek to minimize it on a suitably chosen finite-dimensional subspace $W \subset U$. We begin by selecting a basis[†] $\varphi_1, \ldots, \varphi_n$ of the subspace W. The general element of W is a (uniquely determined) linear combination

$$\varphi(x) = c_1 \varphi_1(x) + \ \cdots \ + c_n \varphi_n(x) \tag{14.2}$$

of the basis functions. Our goal, then, is to determine the coefficients $c_1, \ldots, c_n$ such that $\varphi(x)$ minimizes $\mathcal{P}[\varphi]$ among all such functions. Substituting (14.2) into (14.1) and expanding we find

$$\mathcal{P}[\varphi] = \frac{1}{2} \sum_{i,j=1}^{n} m_{ij}\, c_i\, c_j - \sum_{i=1}^{n} b_i\, c_i = \tfrac{1}{2}\, \mathbf{c}^T M \mathbf{c} - \mathbf{c}^T \mathbf{b}, \tag{14.3}$$

where

(a) $\mathbf{c} = (\, c_1, c_2, \ldots, c_n \,)^T$ is the vector of unknown coefficients in (14.2),

(b) $M = (m_{ij})$ is the symmetric $n \times n$ matrix with entries

$$m_{ij} = \langle\!\langle \, L[\varphi_i] \, , L[\varphi_j] \, \rangle\!\rangle, \qquad i, j = 1, \ldots, n, \tag{14.4}$$

(c) $\mathbf{b} = (\, b_1, b_2, \ldots, b_n \,)^T$ is the vector with entries

$$b_i = \langle \, f \, , \varphi_i \, \rangle, \qquad i = 1, \ldots, n. \tag{14.5}$$

Observe that, once we specify the basis functions φ_i, the coefficients m_{ij} and b_i are all known quantities. Therefore, we have reduced our original problem to a finite-dimensional problem of minimizing the quadratic function (14.3) over all possible vectors $\mathbf{c} \in \mathbb{R}^n$. The coefficient matrix M is, in fact, positive definite, since, by the preceding computation,

$$\mathbf{c}^T M \mathbf{c} = \sum_{i,j=1}^{n} m_{ij}\, c_i\, c_j = \| \, L[c_1 \varphi_1(x) + \ \cdots \ + c_n \varphi_n] \, \|^2 = \| \, L[\varphi] \, \|^2 \ > \ 0 \tag{14.6}$$

[†] In this case, an orthonormal basis is not of any particular help.

as long as $L[\varphi] \neq 0$. Moreover, our positivity assumption implies that $L[\varphi] = 0$ if and only if $\varphi \equiv 0$, and hence (14.6) is indeed positive for all $\mathbf{c} \neq \mathbf{0}$. We can now invoke the original finite-dimensional minimization Theorem 12.12 to conclude that the unique minimizer to (14.3) is obtained by solving the associated linear system

$$M\mathbf{c} = \mathbf{b}. \tag{14.7}$$

Solving (14.7) relies on some form of Gaussian Elimination, or, alternatively, an iterative linear system solver, e.g., Gauss–Seidel or SOR.

This constitutes the basic abstract setting for the finite element method. The main issue, then, is how to effectively choose the finite-dimensional subspace W. Two candidates that might spring to mind are the space $\mathcal{P}^{(n)}$ of polynomials of degree $\leq n$, or the space $\mathcal{T}^{(n)}$ of trigonometric polynomials of degree $\leq n$. However, for a variety of reasons, neither is well suited to the finite element method. One criterion is that the functions in W must satisfy the relevant boundary conditions — otherwise W would not be a subspace of U. More importantly, in order to obtain sufficient accuracy, the linear algebraic system (14.7) will typically be rather large, and so the coefficient matrix M should be as sparse as possible, i.e., have lots of zero entries. Otherwise, computing the solution will be too time-consuming to be of much practical value. Such considerations prove to be of absolutely crucial importance when applying the method to solve boundary value problems for partial differential equations in higher dimensions.

The really innovative contribution of the finite element method is to first (paradoxically) *enlarge* the space U of allowable functions upon which to minimize the quadratic functional $\mathcal{P}[u]$. The governing differential equation requires its solutions to have a certain degree of smoothness, whereas the associated minimization principle typically requires only half as many derivatives. Thus, for second order boundary value problems, $\mathcal{P}[u]$ only involves first order derivatives. It can be rigorously shown that the functional has the *same* minimizing solution, even if one allows (reasonable) functions that fail to have enough derivatives to satisfy the differential equation. Thus, one can try minimizing over subspaces containing fairly "rough" functions. Again, the justification of this method requires some deeper analysis, which lies beyond the scope of this introductory treatment.

For second order boundary value problems, a popular and effective choice of the finite-dimensional subspace is to use continuous, piecewise affine functions. Recall that a function is affine, $f(x) = a\,x + b$, if and only if its graph is a straight line. The function is *piecewise affine* if its graph consists of a finite number of straight line segments; a typical example is plotted in Figure 14.1. Continuity requires that the individual line segments be connected together end to end.

Given a boundary value problem on a bounded interval $[a, b]$, let us fix a finite collection of *mesh points*

$$a = x_0 < x_1 < x_2 < \;\cdots\; < x_{n-1} < x_n = b.$$

The formulas simplify if one uses equally spaced mesh points, but this is not necessary for the method to apply. Let W denote the vector space consisting of all continuous, piecewise affine functions, with corners at the nodes, that satisfy the homogeneous boundary

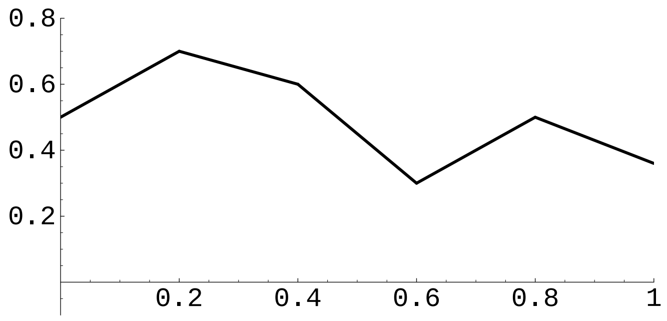


Figure 14.1. A Continuous Piecewise Affine Function.

conditions. To be specific, let us treat the case of Dirichlet (fixed) boundary conditions

$$\varphi(a) = \varphi(b) = 0. \tag{14.8}$$

Thus, on each subinterval

$$\varphi(x) = c_j + b_j(x - x_j), \qquad \text{for} \qquad x_j \le x \le x_{j+1}, \qquad j = 0, \ldots, n-1.$$

Continuity of $\varphi(x)$ requires

$$c_j = \varphi(x_j^+) = \varphi(x_j^-) = c_{j-1} + b_{j-1} h_{j-1}, \qquad j = 1, \ldots, n-1, \tag{14.9}$$

where $h_{j-1} = x_j - x_{j-1}$ denotes the length of the j^{th} subinterval. The boundary conditions (14.8) require

$$\varphi(a) = c_0 = 0, \qquad \varphi(b) = c_{n-1} + h_{n-1} b_{n-1} = 0. \tag{14.10}$$

The function $\varphi(x)$ involves a total of $2n$ unspecified coefficients $c_0, \ldots, c_{n-1}, b_0, \ldots, b_{n-1}$. The continuity conditions (14.9) and the second boundary condition (14.10) uniquely determine the b_j. The first boundary condition specifies c_0, while the remaining $n-1$ coefficients $c_1 = \varphi(x_1), \ldots, c_{n-1} = \varphi(x_{n-1})$ are arbitrary. We conclude that the finite element subspace W has dimension $n-1$, which is the number of interior mesh points.

Remark: Every function $\varphi(x)$ in our subspace has piecewise constant first derivative $w'(x)$. However, the jump discontinuities in $\varphi'(x)$ imply that its second derivative $\varphi''(x)$ has a delta function impulse at each mesh point, and is therefore far from being a solution to the differential equation. Nevertheless, the finite element minimizer $\varphi_\star(x)$ will, in practice, provide a reasonable approximation to the actual solution $u_\star(x)$.

The most convenient basis for W consists of the *hat functions*, which are continuous, piecewise affine functions that interpolate the same basis data as the Lagrange polynomials (13.22), namely

$$\varphi_j(x_k) = \begin{cases} 1, & j = k, \\ 0, & j \ne k, \end{cases} \qquad \text{for} \qquad j = 1, \ldots, n-1, \qquad k = 0, \ldots, n.$$

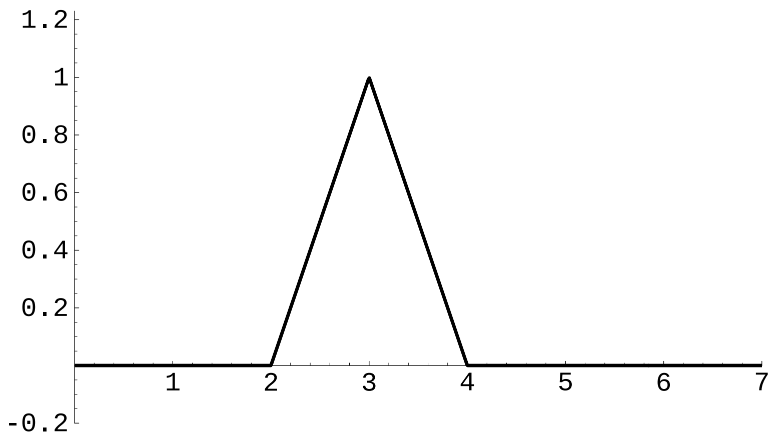


Figure 14.2. A Hat Function.

The graph of a typical hat function appears in Figure 14.2. The explicit formula is easily established:

$$\varphi_j(x) = \begin{cases} \dfrac{x - x_{j-1}}{x_j - x_{j-1}}, & x_{j-1} \le x \le x_j, \\ \dfrac{x_{j+1} - x}{x_{j+1} - x_j}, & x_j \le x \le x_{j+1}, \\ 0, & x \le x_{j-1} \quad \text{or} \quad x \ge x_{j+1}, \end{cases} \qquad j = 1, \ldots, n-1. \tag{14.11}$$

An advantage of using these basis elements is that the resulting coefficient matrix (14.4) turns out to be tridiagonal. Therefore, the tridiagonal Gaussian Elimination algorithm in (4.47) will rapidly produce the solution to the linear system (14.7). Since the accuracy of the finite element solution increases with the number of mesh points, this solution scheme allows us to easily compute very accurate numerical approximations.

Example 14.1. Consider the equilibrium equations

$$K[u] = -\frac{d}{dx}\left(c(x)\,\frac{du}{dx}\right) = f(x), \qquad 0 < x < \ell,$$

for a non-uniform bar subject to homogeneous Dirichlet boundary conditions. In order to formulate a finite element approximation scheme, we begin with the minimization principle based on the quadratic functional

$$\mathcal{P}[u] = \tfrac{1}{2}\,\|\,u'\,\|^2 - \langle\, f\,,u\,\rangle = \int_0^\ell \left[\, \tfrac{1}{2}\, c(x)\, u'(x)^2 - f(x)\, u(x)\,\right] dx.$$

We divide the interval $[0, \ell]$ into n equal subintervals, each of length $h = \ell/n$. The resulting uniform mesh has

$$x_j = j\,h = \frac{j\,\ell}{n}, \qquad j = 0, \ldots, n.$$

The corresponding finite element basis hat functions are explicitly given by

$$\varphi_j(x) = \begin{cases} (x - x_{j-1})/h, & x_{j-1} \le x \le x_j, \\ (x_{j+1} - x)/h, & x_j \le x \le x_{j+1}, \\ 0, & \text{otherwise}, \end{cases} \qquad j = 1, \ldots, n-1. \tag{14.12}$$

The associated linear system (14.7) has coefficient matrix entries

$$m_{ij} = \langle\!\langle\, \varphi_i' \,,\varphi_j' \,\rangle\!\rangle = \int_0^\ell \varphi_i'(x)\,\varphi_j'(x)\,c(x)\,dx, \qquad i,j = 1, \ldots, n-1.$$

Since the function $\varphi_i(x)$ vanishes except on the interval $x_{i-1} < x < x_{i+1}$, while $\varphi_j(x)$ vanishes outside $x_{j-1} < x < x_{j+1}$, the integral will vanish unless $i = j$ or $i = j \pm 1$. Moreover,

$$\varphi_j'(x) = \begin{cases} 1/h, & x_{j-1} \le x \le x_j, \\ -1/h, & x_j \le x \le x_{j+1}, \\ 0, & \text{otherwise}, \end{cases} \qquad j = 1, \ldots, n-1.$$

Therefore, the coefficient matrix has the tridiagonal form

$$M = \frac{1}{h^2} \begin{pmatrix} s_0 + s_1 & -s_1 & & & & \\ -s_1 & s_1 + s_2 & -s_2 & & & \\ & -s_2 & s_2 + s_3 & -s_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -s_{n-3} & s_{n-3} + s_{n-2} & -s_{n-2} \\ & & & & -s_{n-2} & s_{n-2} + s_{n-1} \end{pmatrix}, \tag{14.13}$$

where

$$s_j = \int_{x_j}^{x_{j+1}} c(x)\,dx \tag{14.14}$$

is the total stiffness of the j^{th} subinterval. For example, in the homogeneous case $c(x) \equiv 1$, the coefficient matrix (14.13) reduces to the very special form

$$M = \frac{1}{h} \begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix}. \tag{14.15}$$

The corresponding right hand side has entries

$$\begin{aligned} b_j = \langle\, f \,,\varphi_j \,\rangle &= \int_0^\ell f(x)\,\varphi_j(x)\,dx \\ &= \frac{1}{h}\left[\int_{x_{j-1}}^{x_j} (x - x_{j-1}) f(x)\,dx + \int_{x_j}^{x_{j+1}} (x_{j+1} - x) f(x)\,dx\right]. \end{aligned} \tag{14.16}$$

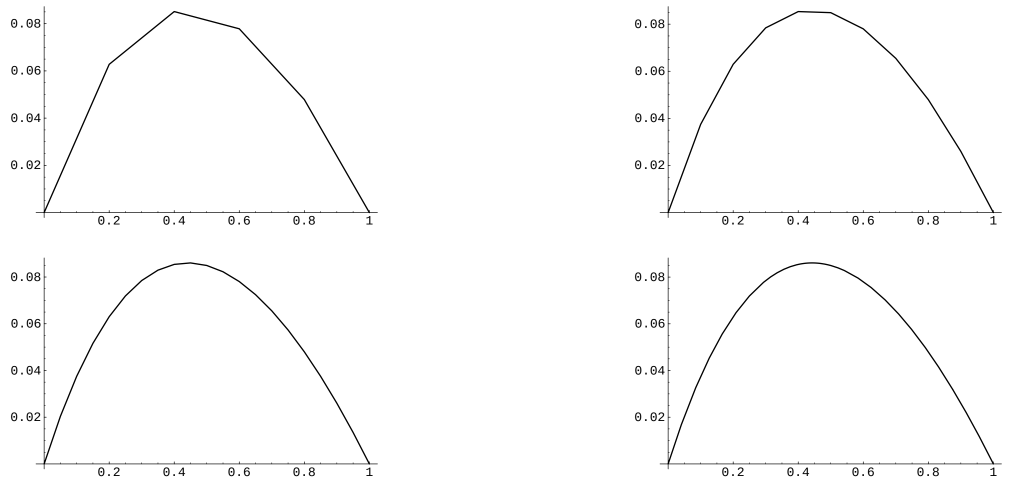


Figure 14.3. Finite Element Solution to (14.19).

In this manner, we have assembled the basic ingredients for determining the finite element approximation to the solution.

In practice, we do not have to explicitly evaluate the integrals (14.14, 16), but may replace them by a suitably close numerical approximation. When $h \ll 1$ is small, then the integrals are taken over small intervals, and we can use the trapezoid rule[†], [**5**, **7**], to approximate them:

$$s_j \;\approx\; \frac{h}{2}\big[\,c(x_j) + c(x_{j+1})\,\big], \qquad\qquad b_j \;\approx\; h\, f(x_j). \tag{14.17}$$

Remark: The j^{th} entry of the resulting finite element system $M\,\mathbf{c} = \mathbf{b}$ is, upon dividing by h, given by

$$-\;\frac{c_{j+1} - 2\,c_j + c_{j-1}}{h^2} \;=\; -\;\frac{u(x_{j+1}) - 2\,u(x_j) + u(x_{j-1})}{h^2} = -f(x_j). \tag{14.18}$$

The left hand side coincides with the standard finite difference approximation to minus the second derivative $-u''(x_j)$ at the mesh point x_j. As a result, for this particular differential equation, the finite element and finite difference numerical solution methods happen to coincide.

Example 14.2. Consider the boundary value problem

$$-\,\frac{d}{dx}\,(x+1)\,\frac{du}{dx} = 1, \qquad\qquad u(0) = 0, \qquad\qquad u(1) = 0. \tag{14.19}$$

[†] One might be tempted use more accurate numerical integration procedures, but the improvement in accuracy of the final answer is not very significant, particularly if the step size h is small.

The explicit solution is easily found by direct integration:

$$u(x) = -x + \frac{\log(x+1)}{\log 2}\,. \tag{14.20}$$

It minimizes the associated quadratic functional

$$\mathcal{P}[u] = \int_0^\ell \left[\, \tfrac{1}{2}\,(x+1)\,u'(x)^2 - u(x) \,\right] dx \tag{14.21}$$

over all possible functions $u \in \mathrm{C}^1$ that satisfy the given boundary conditions. The finite element system (14.7) has coefficient matrix given by (14.13) and right hand side (14.16), where

$$s_j = \int_{x_j}^{x_{j+1}} (1+x)\,dx \; = h\,(1+x_j) + \tfrac{1}{2}\,h^2 = h + h^2\left(j + \frac{1}{2}\right), \qquad b_j = \int_{x_j}^{x_{j+1}} 1\,dx \; = h.$$

The resulting solution is plotted in Figure 14.3. The first three graphs contain, respectively, 5, 10, 20 points in the mesh, so that $h = .2, .1, .05$, while the last plots the exact solution (14.20). Even when computed on rather coarse meshes, the finite element approximation is quite respectable.

Example 14.3. Consider the Sturm–Liouville boundary value problem

$$-u'' + (x+1)u = x\,e^x, \qquad u(0) = 0, \qquad u(1) = 0. \tag{14.22}$$

The solution minimizes the quadratic functional

$$\mathcal{P}[u] = \int_0^1 \left[\, \tfrac{1}{2}\,u'(x)^2 + \tfrac{1}{2}\,(x+1)\,u(x)^2 - e^x\,u(x) \,\right] dx, \tag{14.23}$$

over all functions $u(x)$ that satisfy the boundary conditions. We lay out a uniform mesh of step size $h = 1/n$. The corresponding finite element basis hat functions as in (14.12). The matrix entries are given by[†]

$$m_{ij} = \int_0^1 \left[\, \varphi_i'(x)\,\varphi_j'(x) + (x+1)\,\varphi_i(x)\,\varphi_j(x) \,\right] dx \;\approx\; \begin{cases} \dfrac{2}{h} + \dfrac{2\,h}{3}\,(x_i+1), & i = j, \\[2mm] -\dfrac{1}{h} + \dfrac{h}{6}\,(x_i+1), & |\,i-j\,| = 1, \\[2mm] 0, & \text{otherwise,} \end{cases}$$

while

$$b_i = \langle\, x\,e^x\,, \varphi_i \,\rangle = \int_0^1 x\,e^x\,\varphi_i(x)\,dx \;\approx\; x_i\,e^{x_i}\,h.$$

[†] The integration is made easier by noting that the integrand is zero except on a small subinterval. Since the function $x+1$ (but not φ_i or φ_j) does not vary significantly on this subinterval, it can be approximated by its value $1+x_i$ at a mesh point. A similar simplification is used in the ensuing integral for b_i.

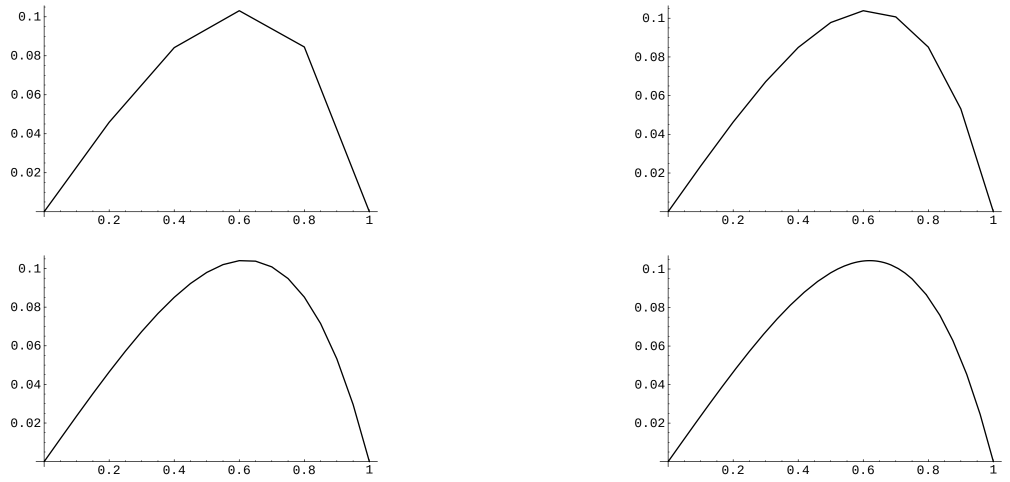


Figure 14.4. Finite Element Solution to (14.22).

The resulting solution is plotted in Figure 14.4. As in the previous figure, the first three graphs contain, respectively, 5, 10, 20 points in the mesh, while the last plots the exact solution, which can be expressed in terms of Airy functions, cf. [**36**].

So far, we have only treated homogeneous boundary conditions. An inhomogeneous boundary value problem does not immediately fit into our framework since the set of functions satisfying the boundary conditions does *not* form a vector space. One way to get around this problem is to replace $u(x)$ by $\widetilde{u}(x) = u(x) - h(x)$, where $h(x)$ is any convenient function that satisfies the boundary conditions. For example, for the inhomogeneous Dirichlet conditions

$$u(a) = \alpha, \qquad\qquad u(b) = \beta,$$

we can subtract off the affine function

$$h(x) = \frac{(\beta - \alpha)x + \alpha b - \beta a}{b - a}.$$

Another option is to choose an appropriate combination of elements at the endpoints:

$$h(x) = \alpha\varphi_0(x) + \beta\varphi_n(x).$$

Linearity implies that the difference $\widetilde{u}(x) = u(x) - h(x)$ satisfies the amended differential equation

$$K[\widetilde{u}] = \widetilde{f}, \qquad\qquad \text{where} \qquad\qquad \widetilde{f} = f - K[h],$$

now supplemented by homogeneous boundary conditions. The modified boundary value problem can then be solved by the standard finite element method. Further details are left as a project for the motivated student.

Finally, one can employ other functions beyond the piecewise affine hat functions (14.11) to span finite element subspace. Another popular choice, which is essential for higher order boundary value problems such as beams, is to use splines. Thus, once we have chosen our mesh points, we can let $\varphi_j(x)$ be the basis B–splines. Since $\varphi_j(x) = 0$

for $x \leq x_{j-2}$ or $x \geq x_{j+2}$, the resulting coefficient matrix (14.4) is *pentadiagonal*, which means $m_{ij} = 0$ whenever $|\, i - j \,| > 2$. Pentadiagonal matrices are not quite as pleasant as their tridiagonal cousins, but are still rather sparse. Positive definiteness of M implies that an iterative solution technique, e.g., SOR, can effectively and rapidly solve the linear system, and thereby produce the finite element spline approximation to the boundary value problem.

14.2. Finite Elements for the Laplace and Poisson Equations.

Finite element methods are also effectively employed to solving boundary value problems for elliptic partial differential equations. In this section, we concentrate on applying these ideas to the two-dimensional Poisson equation. For specificity, we concentrate on the homogeneous Dirichlet boundary value problem.

Theorem 14.4. *The function $u(x,y)$ that minimizes the* Dirichlet integral

$$\tfrac{1}{2} \,\|\, \nabla u \,\|^2 - \langle\, u \,,f \,\rangle = \iint_\Omega \left(\tfrac{1}{2}\, u_x^2 + \tfrac{1}{2}\, u_y^2 - f\, u \right) dx\, dy \tag{14.24}$$

among all C^1 *functions that satisfy the prescribed homogeneous Dirichlet boundary conditions is the solution to the boundary value problem*

$$-\,\Delta u = f \quad \text{in} \quad \Omega \qquad\qquad u = 0 \quad \text{on} \quad \partial\Omega. \tag{14.25}$$

In the finite element approximation, we restrict the Dirichlet functional to a suitably chosen finite-dimensional subspace. As in the one-dimensional situation, the most convenient finite-dimensional subspaces consist of functions that may lack the requisite degree of smoothness that qualifies them as possible solutions to the partial differential equation. Nevertheless, they do provide good approximations to the actual solution. An important practical consideration, impacting the speed of the calculation, is to employ functions with small support. The resulting finite element matrix will then be sparse and the solution to the linear system can be relatively rapidly calculate, usually by application of an iterative numerical scheme such as the Gauss–Seidel or SOR methods discussed in Section 7.4.

Finite Elements and Triangulation

For one-dimensional boundary value problems, the finite element construction rests on the introduction of a mesh $a = x_0 < x_1 < \cdots < x_n = b$ on the interval of definition. The mesh nodes x_k break the interval into a collection of small subintervals. In two-dimensional problems, a *mesh* consists of a finite number of points $\mathbf{x}_k = (x_k, y_k)$, $k = 1, \ldots, m$, known as *nodes*, usually lying inside the domain $\Omega \subset \mathbb{R}^2$. As such, there is considerable freedom in the choice of mesh nodes, and completely uniform spacing is often not possible. We regard the nodes as forming the vertices of a *triangulation* of the domain Ω, consisting of a finite number of small triangles, which we denote by $T_1, \ldots, T_N$. The nodes are split into two categories — *interior nodes* and *boundary nodes*, the latter lying on or close to the boundary of the domain. A curved boundary is approximated by the polygon through the boundary nodes formed by the sides of the triangles lying on the edge of the domain; see Figure 14.5 for a typical example. Thus, in computer implementations of the finite element

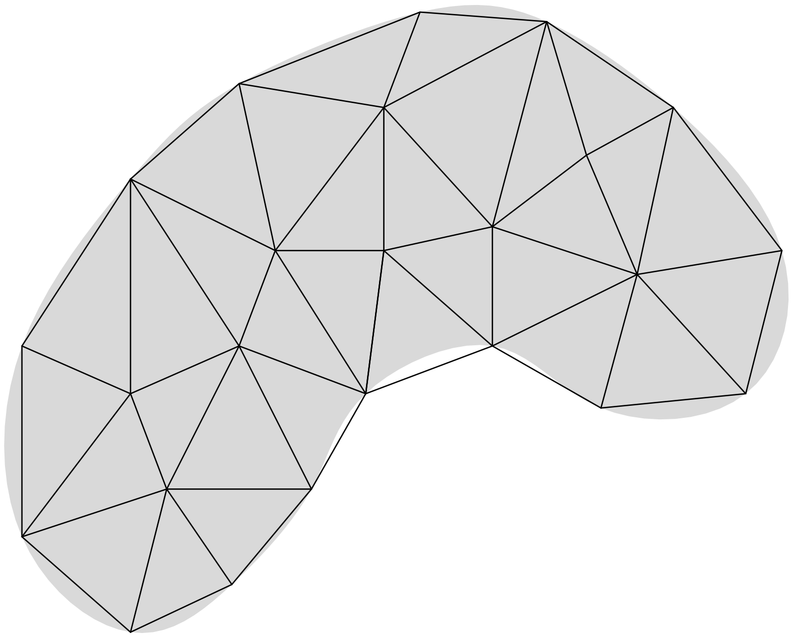


Figure 14.5. Triangulation of a Planar Domain.

method, the first module is a routine that will automatically triangulate a specified domain in some reasonable manner; see below for details on what "reasonable" entails.

As in our one-dimensional finite element construction, the functions $w(x,y)$ in the finite-dimensional subspace W will be continuous and *piecewise affine*. "Piecewise affine" means that, on each triangle, the graph of w is flat, and so has the formula[†]

$$w(x,y) = \alpha^\nu + \beta^\nu x + \gamma^\nu y, \qquad \text{for} \qquad (x,y) \in T_\nu. \tag{14.26}$$

Continuity of w requires that its values on a common edge between two triangles must agree, and this will impose certain compatibility conditions on the coefficients $\alpha^\mu, \beta^\mu, \gamma^\mu$ and $\alpha^\nu, \beta^\nu, \gamma^\nu$ associated with adjacent pairs of triangles T_μ, T_ν. The graph of $z = w(x,y)$ forms a connected polyhedral surface whose triangular faces lie above the triangles in the domain; see Figure 14.5 for an illustration.

The next step is to choose a basis of the subspace of piecewise affine functions for the given triangulation. As in the one-dimensional version, the most convenient basis consists of *pyramid functions* $\varphi_k(x,y)$ which assume the value 1 at a single node $\mathbf{x}_k$, and are zero at all the other nodes; thus

$$\varphi_k(x_i,y_i) = \begin{cases} 1, & i = k, \\ 0, & i \neq k. \end{cases} \tag{14.27}$$

Note that φ_k will be nonzero only on those triangles which have the node $\mathbf{x}_k$ as one of their vertices, and hence the graph of φ_k looks like a pyramid of unit height sitting on a flat plane, as illustrated in Figure 14.7.

[†] Here and subsequently, the index ν is a superscript, not a power!

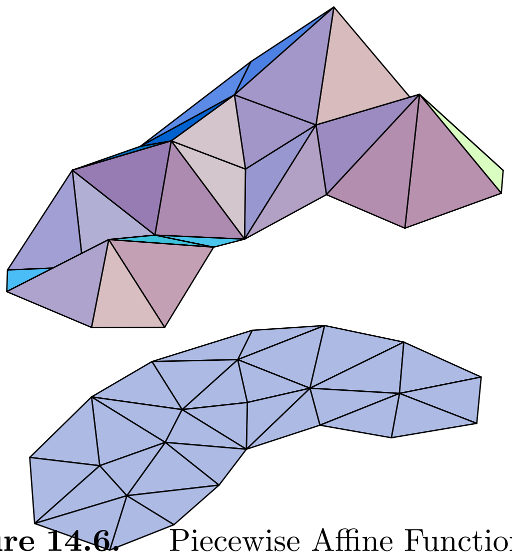


Figure 14.6. Piecewise Affine Function.

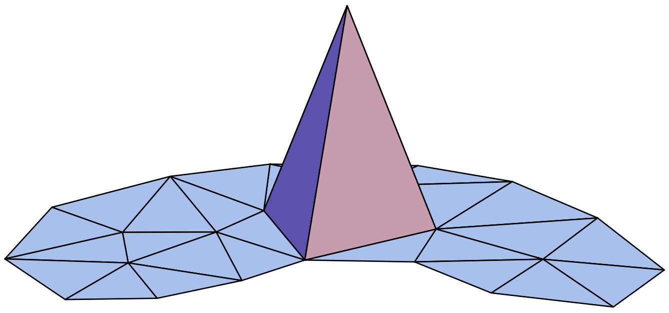


Figure 14.7. Finite Element Pyramid Function.

The pyramid functions $\varphi_k(x,y)$ corresponding to the *interior nodes* $\mathbf{x}_k$ automatically satisfy the homogeneous Dirichlet boundary conditions on the boundary of the domain — or, more correctly, on the polygonal boundary of the triangulated domain, which is supposed to be a good approximation to the curved boundary of the original domain Ω. Thus, the finite-dimensional finite element subspace W is the span of the interior node pyramid functions, and so general element $w \in W$ is a linear combination thereof:

$$w(x,y) = \sum_{k=1}^{n} c_k \, \varphi_k(x,y), \tag{14.28}$$

where the sum ranges over the n interior nodes of the triangulation. Owing to the original specification (14.27) of the pyramid functions, the coefficients

$$c_k = w(x_k, y_k) \;\approx\; u(x_k, y_k), \qquad\qquad k = 1, \ldots, n, \tag{14.29}$$

are the *same* as the values of the finite element approximation $w(x,y)$ at the interior nodes. This immediately implies linear independence of the pyramid functions, since the only linear combination that vanishes at all nodes is the trivial one $c_1 = \cdots = c_n = 0$. Thus, the interior node pyramid functions $\varphi_1, \ldots \varphi_n$ form a basis for finite element subspace W, which therefore has dimension equal to n, the number of interior nodes.

Determining the explicit formulae for the finite element basis functions is not difficult. On one of the triangles T_ν that has $\mathbf{x}_k$ as a vertex, $\varphi_k(x,y)$ will be the unique affine function (14.26) that takes the value 1 at the vertex $\mathbf{x}_k$ and 0 at its other two vertices $\mathbf{x}_l$ and $\mathbf{x}_m$. Thus, we are in need of a formula for an affine function or *element*

$$\omega_k^\nu(x,y) = \alpha_k^\nu + \beta_k^\nu\, x + \gamma_k^\nu\, y, \qquad\qquad (x,y) \in T_\nu, \tag{14.30}$$

that takes the prescribed values

$$\omega_k^\nu(x_i,y_i) = \omega_k^\nu(x_j,y_j) = 0, \qquad\qquad \omega_k^\nu(x_k,y_k) = 1,$$

at three distinct points. These three conditions lead to the linear system

$$\begin{aligned} \omega_k^\nu(x_i,y_i) &= \alpha_k^\nu + \beta_k^\nu\, x_i + \gamma_k^\nu\, y_i = 0, \\ \omega_k^\nu(x_j,y_j) &= \alpha_k^\nu + \beta_k^\nu\, x_j + \gamma_k^\nu\, y_j = 0, \\ \omega_k^\nu(x_k,y_k) &= \alpha_k^\nu + \beta_k^\nu\, x_k + \gamma_k^\nu\, y_k = 1. \end{aligned} \tag{14.31}$$

The solution produces the explicit formulae

$$\alpha_k^\nu = \frac{x_i\,y_j - x_j\,y_i}{\Delta_\nu}\,, \qquad\qquad \beta_k^\nu = \frac{y_i - y_j}{\Delta_\nu}\,, \qquad\qquad \gamma_k^\nu = \frac{x_j - x_i}{\Delta_\nu}\,, \tag{14.32}$$

for the coefficients; the denominator

$$\Delta_\nu = \det \begin{pmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{pmatrix} = \pm\, 2 \,\text{area}\; T_\nu \tag{14.33}$$

is, up to sign, twice the area of the triangle T_ν.

Example 14.5. Consider an isoceles right triangle T with vertices

$$\mathbf{x}_1 = (0,0), \qquad\qquad \mathbf{x}_2 = (1,0), \qquad\qquad \mathbf{x}_3 = (0,1).$$

Using (14.32–33) (or solving the linear systems (14.31) directly), we immediately produce the three affine elements

$$\omega_1(x,y) = 1 - x - y, \qquad\qquad \omega_2(x,y) = x, \qquad\qquad \omega_3(x,y) = y. \tag{14.34}$$

As required, each ω_k equals 1 at the vertex $\mathbf{x}_k$ and is zero at the other two vertices.

The finite element pyramid function is then obtained by piecing together the individual affine elements, whence

$$\varphi_k(x,y) = \begin{cases} \omega_k^\nu(x,y), & \text{if } (x,y) \in T_\nu \text{ which has } \mathbf{x}_k \text{ as a vertex}, \\ 0, & \text{otherwise.} \end{cases} \tag{14.35}$$

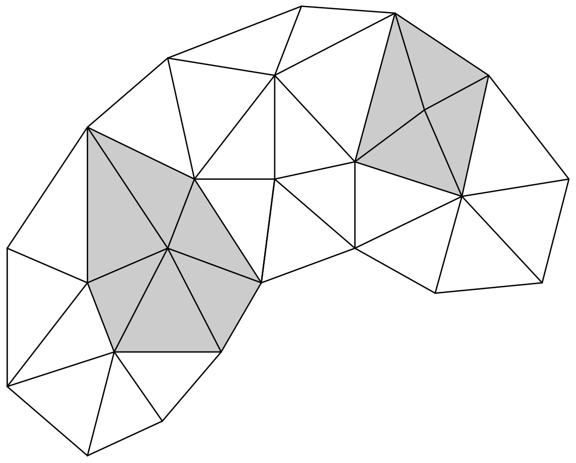


Figure 14.8. Vertex Polygons.

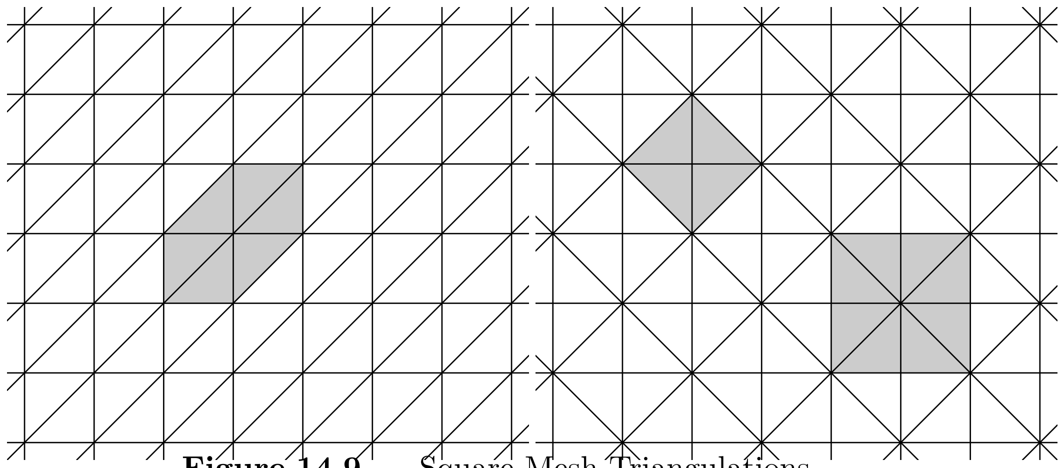


Figure 14.9. Square Mesh Triangulations.

Continuity of $\varphi_k(x,y)$ is assured since the constituent affine elements have the same values at common vertices. The support of the pyramid function (14.35) is the polygon

$$\operatorname{supp}\varphi_k = P_k = \bigcup_{\nu} T_\nu \tag{14.36}$$

consisting of all the triangles T_ν that have the node $\mathbf{x}_k$ as a vertex. In other words, $\varphi_k(x,y) = 0$ whenever $(x,y) \notin P_k$. We will call P_k the k^{th} *vertex polygon*. The node $\mathbf{x}_k$ lies on the interior of its vertex polygon P_k, while the vertices of P_k are all those that are connected to $\mathbf{x}_k$ by a single edge of the triangulation. In Figure 14.8 the shaded regions indicate two of the vertex polygons for the triangulation in Figure 14.5.

Example 14.6. The simplest, and most common triangulations are based on regular meshes. Suppose that the nodes lie on a square grid, and so are of the form $\mathbf{x}_{i,j} = (i\,h + a, j\,h + b)$ where $h > 0$ is the inter-node spacing, and (a,b) represents an overall offset. If we choose the triangles to all have the same orientation, as in the first picture in Figure 14.9, then the vertex polygons all have the same shape, consisting of 6 triangles of total area $3h^2$ — the shaded region. On the other hand, if we choose an alternating, perhaps more æsthetically pleasing triangulation as in the second picture, then there are two types of vertex polygons. The first, consisting of four triangles, has area $2h^2$, while the second, containing 8 triangles, has twice the area, $4h^2$. In practice, there are good reasons to prefer the former triangulation.

In general, in order to ensure convergence of the finite element solution to the true minimizer, one should choose a triangulation with the following properties:

(a) The triangles are not too long and skinny. In other words, their sides should have comparable lengths. In particular, obtuse triangles should be avoided.

(b) The areas of nearby triangles T_ν should not vary too much.

(c) The areas of nearby vertex polygons P_k should also not vary too much.

For adaptive or variable meshes, one might very well have wide variations in area over the entire grid, with small triangles in regions of rapid change in the solution, and large ones in less interesting regions. But, overall, the sizes of the triangles and vertex polygons should not dramatically vary as one moves across the domain.

The Finite Element Equations

We now seek to approximate the solution to the homogeneous Dirichlet boundary value problem by restricting the Dirichlet functional to the selected finite element subspace W. Substituting the formula (14.28) for a general element of W into the quadratic Dirichlet functional (14.24) and expanding, we find

$$\mathcal{P}[w] = \mathcal{P}\left[\sum_{i=1}^{n} c_i\,\varphi_i\right] = \iint_\Omega \left[\left(\sum_{i=1}^{n} c_i\,\nabla\varphi_i\right)^2 - f(x,y)\left(\sum_{i=1}^{n} c_i\,\varphi_i\right)\right] dx\,dy$$
$$= \frac{1}{2}\sum_{i,j=1}^{n} k_{ij}\,c_i\,c_j - \sum_{i=1}^{n} b_i\,c_i = \tfrac{1}{2}\,\mathbf{c}^T K\,\mathbf{c} - \mathbf{b}^T\mathbf{c}.$$

Here, $K = (k_{ij})$ is the symmetric $n\times n$ matrix, while $\mathbf{b} = (\,b_1, b_2, \ldots, b_n\,)^T$ is the vector that have the respective entries

$$k_{ij} = \langle\,\nabla\varphi_i\,,\nabla\varphi_j\,\rangle = \iint_\Omega \nabla\varphi_i\cdot\nabla\varphi_j\,dx\,dy,$$
$$b_i = \langle\,f\,,\varphi_i\,\rangle = \iint_\Omega f\,\varphi_i\,dx\,dy. \tag{14.37}$$

Thus, to determine the finite element approximation, we need to minimize the quadratic function

$$P(\mathbf{c}) = \tfrac{1}{2}\,\mathbf{c}^T K\,\mathbf{c} - \mathbf{b}^T\mathbf{c} \tag{14.38}$$

over all possible choices of coefficients $\mathbf{c} = (\,c_1, c_2, \ldots, c_n\,)^T \in \mathbb{R}^n$, i.e., over all possible function values at the interior nodes. Restricting to the finite element subspace has reduced us to a standard finite-dimensional quadratic minimization problem. First, the coefficient matrix $K > 0$ is positive definite due to the positive definiteness of the original functional;

the proof in Section 14.1 is easily adapted to the present situation. The minimizer is obtained by solving the associated linear system

$$K\mathbf{c} = \mathbf{b}. \tag{14.39}$$

The solution to (14.39) can be effected by either Gaussian elimination or an iterative technique.

To find explicit formulae for the matrix coefficients k_{ij} in (14.37), we begin by noting that the gradient of the affine element (14.30) is equal to

$$\nabla \omega_k^\nu(x,y) = \mathbf{a}_k^\nu = \begin{pmatrix} \beta_k^\nu \\ \gamma_k^\nu \end{pmatrix} = \frac{1}{\Delta_\nu} \begin{pmatrix} y_i - y_j \\ x_j - x_i \end{pmatrix}, \qquad (x,y) \in T_\nu, \tag{14.40}$$

which is a constant vector inside the triangle T_ν, while outside $\nabla \omega_k^\nu = \mathbf{0}$. Therefore,

$$\nabla \varphi_k(x,y) = \begin{cases} \nabla \omega_k^\nu = \mathbf{a}_k^\nu, & \text{if } (x,y) \in T_\nu \text{ which has } \mathbf{x}_k \text{ as a vertex}, \\ \mathbf{0}, & \text{otherwise}, \end{cases} \tag{14.41}$$

reduces to a piecewise constant function on the triangulation. Actually, (14.41) is not quite correct since if (x,y) lies on the boundary of a triangle T_ν, then the gradient does not exist. However, this technicality will not cause any difficulty in evaluating the ensuing integral.

We will approximate integrals over the domain Ω by integrals over the triangles, which relies on our assumption that the polygonal boundary of the triangulation is a reasonably close approximation to the true boundary $\partial\Omega$. In particular,

$$k_{ij} \approx \sum_\nu \iint_{T_\nu} \nabla \varphi_i \cdot \nabla \varphi_j \, dx \, dy \equiv \sum_\nu k_{ij}^\nu. \tag{14.42}$$

Now, according to (14.41), one or the other gradient in the integrand will vanish on the entire triangle T_ν *unless* both $\mathbf{x}_i$ and $\mathbf{x}_j$ are vertices. Therefore, the only terms contributing to the sum are those triangles T_ν that have both $\mathbf{x}_i$ and $\mathbf{x}_j$ as vertices. If $i \neq j$ there are only two such triangles, while if $i = j$ every triangle in the i^{th} vertex polygon P_i contributes. The individual summands are easily evaluated, since the gradients are constant on the triangles, and so, by (14.41),

$$k_{ij}^\nu = \iint_{T_\nu} \mathbf{a}_i^\nu \cdot \mathbf{a}_j^\nu \, dx \, dy = \mathbf{a}_i^\nu \cdot \mathbf{a}_j^\nu \ \text{area}\ T_\nu = \tfrac{1}{2}\, \mathbf{a}_i^\nu \cdot \mathbf{a}_j^\nu \, |\, \Delta_\nu \,|.$$

Let T_ν have vertices $\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k$. Then, by (14.40, 41, 33),

$$\begin{aligned} k_{ij}^\nu &= \frac{1}{2}\, \frac{(y_j - y_k)(y_k - y_i) + (x_k - x_j)(x_i - x_k)}{(\Delta_\nu)^2}\, |\, \Delta_\nu \,| = -\, \frac{(\mathbf{x}_i - \mathbf{x}_k)\cdot(\mathbf{x}_j - \mathbf{x}_k)}{2\, |\, \Delta_\nu \,|}\,, \quad i \neq j, \\ k_{ii}^\nu &= \frac{1}{2}\, \frac{(y_j - y_k)^2 + (x_k - x_j)^2}{(\Delta_\nu)^2}\, |\, \Delta_\nu \,| = \frac{\|\, \mathbf{x}_j - \mathbf{x}_k \,\|^2}{2\, |\, \Delta_\nu \,|}\,. \end{aligned} \tag{14.43}$$

In this manner, each triangle T_ν specifies a collection of 6 different coefficients, $k_{ij}^\nu = k_{ji}^\nu$, indexed by its vertices, and known as the *elemental stiffnesses* of T_ν. Interestingly, the

Figure 14.10. Right and Equilateral Triangles.

elemental stiffnesses depend only on the *angles* of the triangle and not on its size. Thus, similar triangles have the *same* elemental stiffnesses. Indeed, if we denote the angle in T_ν at the vertex $\mathbf{x}_k$ by θ_k^ν, then

$$k_{ii}^\nu = \tfrac{1}{2}\big(\cot\theta_k^\nu + \cot\theta_j^\nu\big), \qquad \text{while} \qquad k_{ij}^\nu = k_{ji}^\nu = -\,\tfrac{1}{2}\,\cot\theta_k^\nu, \quad i \neq j, \tag{14.44}$$

depend only upon the cotangents of the angles.

Example 14.7. The right triangle with vertices $\mathbf{x}_1 = (0,0)$, $\mathbf{x}_2 = (1,0)$, $\mathbf{x}_3 = (0,1)$ has elemental stiffnesses

$$k_{11} = 1, \qquad k_{22} = k_{33} = \tfrac{1}{2}, \qquad k_{12} = k_{21} = k_{13} = k_{31} = -\,\tfrac{1}{2}, \qquad k_{23} = k_{32} = 0. \tag{14.45}$$

The same holds for any other isoceles right triangle, as long as we chose the first vertex to be at the right angle. Similarly, an equilateral triangle has all 60° angles, and so its elemental stiffnesses are

$$\begin{aligned} k_{11} &= k_{22} = k_{33} = \tfrac{1}{\sqrt{3}} \approx .577350, \\ k_{12} &= k_{21} = k_{13} = k_{31} = k_{23} = k_{32} = -\tfrac{1}{2\sqrt{3}} \approx -.288675. \end{aligned} \tag{14.46}$$

Assembling the Elements

The elemental stiffnesses of each triangle will contribute, through the summation (14.42), to the finite element coefficient matrix K. We begin by constructing a larger matrix K^*, which we call the *full finite element matrix*, of size $m \times m$ where m is the total number of nodes in our triangulation, including both interior and boundary nodes. The rows and columns of K^* are labeled by the nodes $\mathbf{x}_i$. Let $K_\nu = (k_{ij}^\nu)$ be the corresponding $m \times m$ matrix containing the elemental stiffnesses k_{ij}^ν of T_ν in the rows and columns indexed by its vertices, and all other entries equal to 0. Thus, K_ν will have (at most) 9 nonzero entries. The resulting $m \times m$ matrices are all summed together over all the triangles,

$$K^* = \sum_{\nu=1}^{N} K_\nu, \tag{14.47}$$

to produce the full finite element matrix, in accordance with (14.42).

The full finite element matrix K^* is too large, since its rows and columns include all the nodes, whereas the finite element matrix K appearing in (14.39) only refers to the n

Figure 14.11. The Oval Plate.

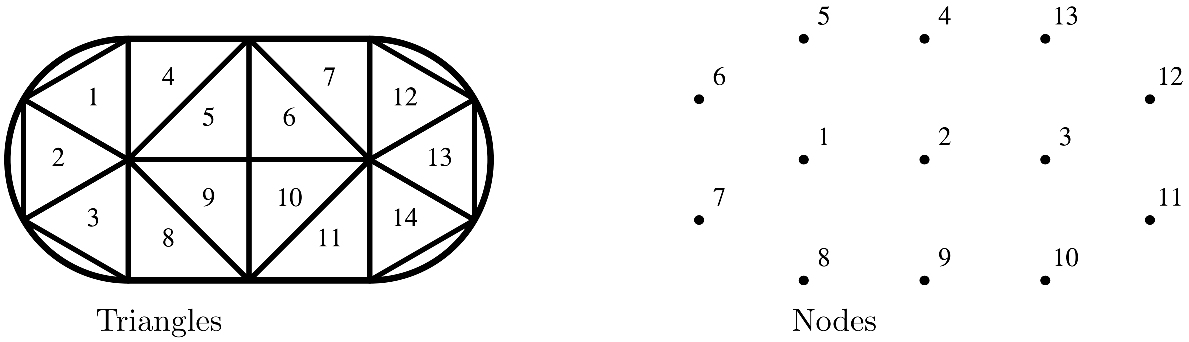


Figure 14.12. A Coarse Triangulation of the Oval Plate.

interior nodes. The *reduced $n \times n$ finite element matrix K* is simply obtained from K^* by deleting all rows and columns indexed by boundary nodes, retaining only the elements k_{ij} when both $\mathbf{x}_i$ and $\mathbf{x}_j$ are interior nodes. For the homogeneous boundary value problem, this is all we require. As we shall see, inhomogeneous boundary conditions are most easily handled by retaining (part of) the full matrix K^*.

The easiest way to digest the construction is by working through a particular example.

Example 14.8. A metal plate has the shape of an oval running track, consisting of a rectangle, with side lengths 1 m by 2 m, and two semicircular disks glued onto its shorter ends, as sketched in Figure 14.11. The plate is subject to a heat source while its edges are held at a fixed temperature. The problem is to find the equilibrium temperature distribution within the plate. Mathematically, we must solve the Poisson equation with Dirichlet boundary conditions, for the equilibrium temperature $u(x,y)$.

Let us describe how to set up the finite element approximation to such a boundary value problem. We begin with a very coarse triangulation of the plate, which will not give particularly accurate results, but does serve to illustrate how to go about assembling the finite element matrix. We divide the rectangular part of the plate into 8 right triangles, while each semicircular end will be approximated by three equilateral triangles. The triangles are numbered from 1 to 14 as indicated in Figure 14.12. There are 13 nodes in all, numbered as in the second figure. Only nodes $1, 2, 3$ are interior, while the boundary nodes are labeled 4 through 13, going counterclockwise around the boundary starting at the top. The full finite element matrix K^* will have size 13×13, its rows and columns labeled by all the nodes, while the reduced matrix K appearing in the finite element equations (14.39) consists of the upper left 3×3 submatrix of K^* corresponding to the three interior nodes.

Each triangle T_ν will contribute the summand K_ν whose values are its elemental stiffnesses, as indexed by its vertices. For example, the first triangle T_1 is equilateral, and so has elemental stiffnesses (14.46). Its vertices are labeled 1, 5, and 6, and therefore we place the stiffnesses (14.46) in the rows and columns numbered $1, 5, 6$ to form the summand

$$K_1 = \begin{pmatrix} .577350 & 0 & 0 & 0 & -.288675 & -.288675 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ -.288675 & 0 & 0 & 0 & .577350 & -.288675 & 0 & 0 & \ldots \\ -.288675 & 0 & 0 & 0 & -.288675 & .577350 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

where all the undisplayed entries in the full 13×13 matrix are 0. The next triangle T_2 has the same equilateral elemental stiffness matrix (14.46), but now its vertices are $1, 6, 7$, and so it will contribute

$$K_2 = \begin{pmatrix} .577350 & 0 & 0 & 0 & 0 & -.288675 & -.288675 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ -.288675 & 0 & 0 & 0 & 0 & .577350 & -.2886750 & 0 & \ldots \\ -.288675 & 0 & 0 & 0 & 0 & -.288675 & .5773500 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Similarly for K_3, with vertices $1, 7, 8$. On the other hand, triangle T_4 is an isoceles right triangle, and so has elemental stiffnesses (14.45). Its vertices are labeled 1, 4, and 5, with

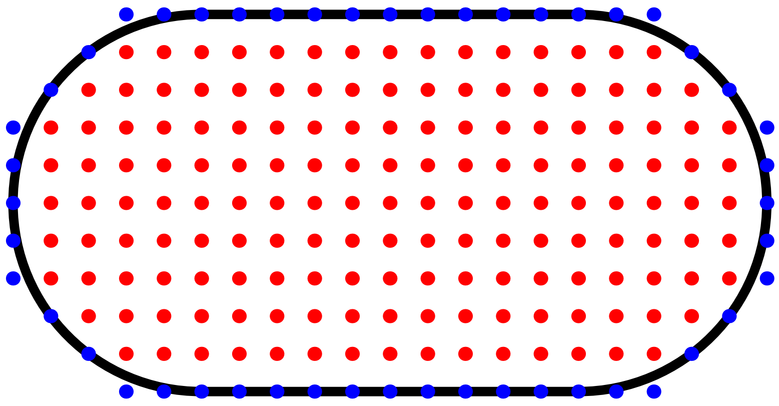


Figure 14.13. A Square Mesh for the Oval Plate.

vertex 5 at the right angle. Therefore, its contribution is

$$K_4 = \begin{pmatrix} .5 & 0 & 0 & 0 & -.5 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & .5 & -.5 & 0 & 0 & 0 & \ldots \\ -.5 & 0 & 0 & -.5 & 1.0 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Continuing in this manner, we assemble 14 contributions $K_1, \ldots, K_{14}$, each with (at most) 9 nonzero entries. The full finite element matrix is the sum

$$K^* = K_1 + K_2 + \;\cdots\; + K_{14}$$

$$= \left(\begin{array}{ccccccc} 3.732 & -1 & 0 & 0 & -.7887 & -.5774 & -.5774 \\ -1 & 4 & -1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3.732 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 2 & -.5 & 0 & 0 \\ -.7887 & 0 & 0 & -.5 & 1.577 & -.2887 & 0 \\ -.5774 & 0 & 0 & 0 & -.2887 & 1.155 & -.2887 \\ -.5774 & 0 & 0 & 0 & 0 & -.2887 & 1.155 \\ -.7887 & 0 & 0 & 0 & 0 & 0 & -.2887 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.7887 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.5774 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.5774 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.7887 & -.5 & 0 & 0 & 0 \end{array} \right. \qquad (14.48)$$

$$\left. \begin{matrix} -.7887 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.7887 & -.5774 & -.5774 & -.7887 \\ 0 & 0 & 0 & 0 & 0 & -.5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -.2887 & 0 & 0 & 0 & 0 & 0 \\ 1.577 & -.5 & 0 & 0 & 0 & 0 \\ -.5 & 2 & -.5 & 0 & 0 & 0 \\ 0 & -.5 & 1.577 & -.2887 & 0 & 0 \\ 0 & 0 & -.2887 & 1.155 & -.2887 & 0 \\ 0 & 0 & 0 & -.2887 & 1.155 & -.2887 \\ 0 & 0 & 0 & 0 & -.2887 & 1.577 \end{matrix} \right) .$$

Since only nodes $1, 2, 3$ are interior nodes, the reduced finite element matrix only uses the upper left 3×3 block of K^*, so

$$K = \begin{pmatrix} 3.732 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 3.732 \end{pmatrix}. \tag{14.49}$$

It is not difficult to directly construct K, bypassing K^* entirely.

For a finer triangulation, the construction is similar, but the matrices become much larger. The procedure can, of course, be automated. Fortunately, if we choose a very regular triangulation, then we do not need to be nearly as meticulous in assembling the stiffness matrices, since many of the entries are the same. The simplest case is when we use a uniform square mesh, and so triangulate the domain into isoceles right triangles. This is accomplished by laying out a relatively dense square grid over the domain $\Omega \subset \mathbb{R}^2$. The interior nodes are the grid points that fall inside the oval domain, while the boundary nodes are all those grid points lying adjacent to one or more of the interior nodes, and are near but not necessarily precisely on the boundary $\partial\Omega$. Figure 14.13 shows the nodes in a square grid with intermesh spacing $h = .2$. While a bit crude in its approximation of the boundary of the domain, this procedure does have the advantage of making the construction of the associated finite element matrix relatively painless.

For such a mesh, all the triangles are isoceles right triangles, with elemental stiffnesses (14.45). Summing the corresponding matrices K_ν over all the triangles, as in (14.47), the rows and columns of K^* corresponding to the interior nodes are seen to all have the same form. Namely, if i labels an interior node, then the corresponding diagonal entry is $k_{ii} = 4$, while the off-diagonal entries $k_{ij} = k_{ji}$, $i \neq j$, are equal to either -1 when node i is adjacent to node j on the grid, and is equal to 0 in all other cases. Node j is allowed to be a boundary node. (Interestingly, the result does not depend on how one orients the pair of triangles making up each square of the grid, which only plays a role in the computation of the right hand side of the finite element equation.) Observe that the same computation applies even to our coarse triangulation. The interior node 2 belongs to all right isoceles triangles, and the corresponding entries in (14.48) are $k_{22} = 4$, and $k_{2j} = -1$ for the four adjacent nodes $j = 1, 3, 4, 9$.

Remark: Interestingly, the coefficient matrix arising from the finite element method on a square (or even rectangular) grid is the same as the coefficient matrix arising from a finite difference solution to the Laplace or Poisson equation. The finite element approach has the advantage of applying to much more general triangulations.

In general, while the finite element matrix K for a two-dimensional boundary value problem is not as nice as the tridiagonal matrices we obtained in our one-dimensional problems, it is still very sparse and, on regular grids, highly structured. This makes solution of the resulting linear system particularly amenable to an iterative matrix solver such as Gauss–Seidel, Jacobi, or, for even faster convergence, successive over-relaxation (SOR).

The Coefficient Vector and the Boundary Conditions

So far, we have been concentrating on assembling the finite element coefficient matrix K. We also need to compute the forcing vector $\mathbf{b} = (\, b_1, b_2, \ldots, b_n \,)^T$ appearing on the right hand side of the fundamental linear equation (14.39). According to (14.37), the entries b_i are found by integrating the product of the forcing function and the finite element basis function. As before, we will approximate the integral over the domain Ω by an integral over the triangles, and so

$$b_i = \iint_\Omega f\,\varphi_i\,dx\,dy \;\approx\; \sum_\nu \iint_{T_\nu} f\,\omega_i^\nu\,dx\,dy \equiv \sum_\nu b_i^\nu. \tag{14.50}$$

Typically, the exact computation of the various triangular integrals is not convenient, and so we resort to a numerical approximation. Since we are assuming that the individual triangles are small, we can adopt a very crude numerical integration scheme. If the function $f(x,y)$ does not vary much over the triangle T_ν — which will certainly be the case if T_ν is sufficiently small — we may approximate $f(x,y) \approx c_i^\nu$ for $(x,y) \in T_\nu$ by a constant. The integral (14.50) is then approximated by

$$b_i^\nu = \iint_{T_\nu} f\,\omega_i^\nu\,dx\,dy \;\approx\; c_i^\nu \iint_{T_\nu} \omega_i^\nu(x,y)\,dx\,dy = \tfrac{1}{3}\,c_i^\nu \,\text{area}\,T_\nu = \tfrac{1}{6}\,c_i^\nu\,|\,\Delta_\nu\,|. \tag{14.51}$$

The formula for the integral of the affine element $\omega_i^\nu(x,y)$ follows from solid geometry. Indeed, it equals the volume under its graph, a tetrahedron of height 1 and base T_ν, as illustrated in Figure 14.14.

How to choose the constant c_i^ν? In practice, the simplest choice is to let $c_i^\nu = f(x_i, y_i)$ be the value of the function at the i^{th} vertex. With this choice, the sum in (14.50) becomes

$$b_i \;\approx\; \sum_\nu \tfrac{1}{3}\,f(x_i,y_i)\,\text{area}\,T_\nu = \tfrac{1}{3}\,f(x_i,y_i)\,\text{area}\,P_i, \tag{14.52}$$

where P_i is the vertex polygon (14.36) corresponding to the node $\mathbf{x}_i$. In particular, for the square mesh with the uniform choice of triangles, as in Example 14.6,

$$\text{area}\,P_i = 3\,h^2 \qquad \text{for all } i, \text{ and so} \qquad b_i \;\approx\; f(x_i,y_i)\,h^2 \tag{14.53}$$

is well approximated by just h^2 times the value of the forcing function at the node. This is the underlying reason to choose the uniform triangulation for the square mesh; the alternating version would give unequal values for the b_i over adjacent nodes, and this would introduce unnecessary errors into the final approximation.

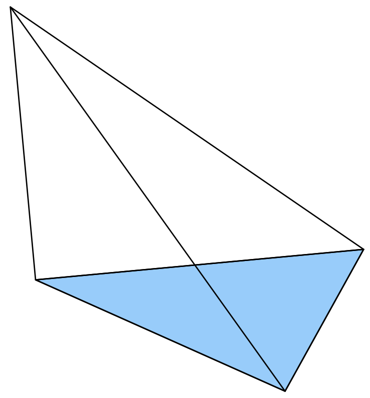


Figure 14.14. Finite Element Tetrahedron.

Example 14.9. For the coarsely triangulated oval plate, the reduced stiffness matrix is (14.49). The Poisson equation

$$-\Delta u = 4$$

models a constant external heat source of magnitude 4° over the entire plate. If we keep the edges of the plate fixed at 0°, then we need to solve the finite element equation $K\mathbf{c} = \mathbf{b}$, where K is the coefficient matrix (14.49), while

$$\mathbf{b} = \tfrac{4}{3}\left(2 + \tfrac{3\sqrt{3}}{4},\ 2,\ 2 + \tfrac{3\sqrt{3}}{4}\right)^T = (\,4.39872, 2.66667, 4.39872\,)^T .$$

The entries of $\mathbf{b}$ are, by (14.52), equal to $4 = f(x_i, y_i)$ times one third the area of the corresponding vertex polygon, which for node 2 is the square consisting of 4 right triangles, each of area $\frac{1}{2}$, whereas for nodes 1 and 3 it consists of 4 right triangles of area $\frac{1}{2}$ plus three equilateral triangles, each of area $\frac{\sqrt{3}}{4}$; see Figure 14.12.

The solution to the final linear system is easily found:

$$\mathbf{c} = (\,1.56724, 1.45028, 1.56724\,)^T .$$

Its entries are the values of the finite element approximation at the three interior nodes. The finite element solution is plotted in the first illustration in Figure 14.15. A more accurate solution, based on a square grid triangulation of size $h = .1$ is plotted in the second figure.

Inhomogeneous Boundary Conditions

So far, we have restricted our attention to problems with homogeneous Dirichlet boundary conditions. The solution to the inhomogeneous Dirichlet problem

$$-\Delta u = f \quad \text{in} \quad \Omega, \qquad\qquad u = h \quad \text{on} \quad \partial\Omega,$$

is also obtained by minimizing the Dirichlet functional (14.24). However, now the minimization takes place over the affine subspace consisting of all functions that satisfy the

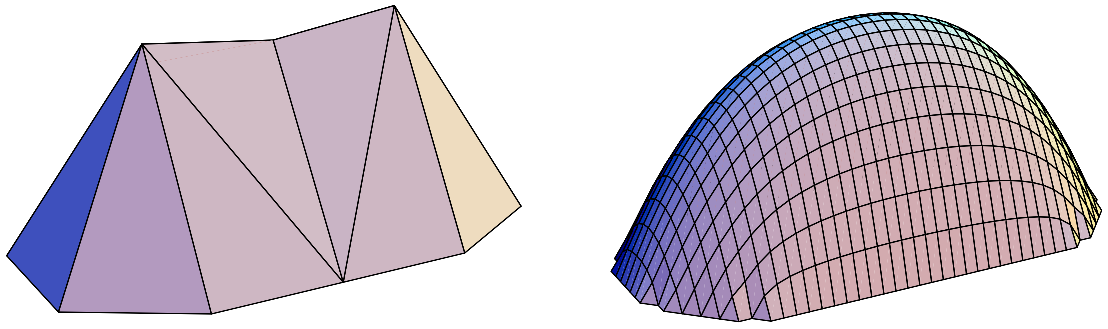


Figure 14.15. Finite Element Solutions to Poisson's Equation for an Oval Plate.

inhomogeneous boundary conditions. It is not difficult to fit this problem into the finite element scheme.

The elements corresponding to the interior nodes of our triangulation remain as before, but now we need to include additional elements to ensure that our approximation satisfies the boundary conditions. Note that if $\mathbf{x}_k$ is a boundary node, then the corresponding *boundary element* $\varphi_k(x,y)$ satisfies the interpolation condition (14.27), and so has the same piecewise affine form (14.35). The corresponding finite element approximation

$$w(x,y) = \sum_{i=1}^{m} c_i\,\varphi_i(x,y), \tag{14.54}$$

has the same form as before, (14.28), but now the sum is over all nodes, both interior and boundary. As before, the coefficients $c_i = w(x_i,y_i) \approx u(x_i,y_i)$ are the values of the finite element approximation at the nodes. Therefore, in order to satisfy the boundary conditions, we require

$$c_j = h(x_j,y_j) \qquad \text{whenever} \qquad \mathbf{x}_j = (x_j,y_j) \qquad \text{is a boundary node.} \tag{14.55}$$

Remark: If the boundary node $\mathbf{x}_j$ does not lie precisely on the boundary $\partial\Omega$, we need to approximate the value $h(x_j,y_j)$ appropriately, e.g., by using the value of $h(x,y)$ at the nearest boundary point $(x,y) \in \partial\Omega$.

The derivation of the finite element equations proceeds as before, but now there are additional terms arising from the nonzero boundary values. Leaving the intervening details to the reader, the final outcome can be written as follows. Let K^* denote the full $m \times m$ finite element matrix constructed as above. The reduced coefficient matrix K is obtained by retaining the rows and columns corresponding to only interior nodes, and so will have size $n \times n$, where n is the number of interior nodes. The *boundary coefficient matrix* $\widetilde{K}$ is the $n \times (m-n)$ matrix consisting of the entries of the interior rows that do not appear in K, i.e., those lying in the columns indexed by the boundary nodes. For instance, in the the coarse triangulation of the oval plate, the full finite element matrix is given in (14.48), and the upper 3×3 subblock is the reduced matrix (14.49). The remaining entries of the

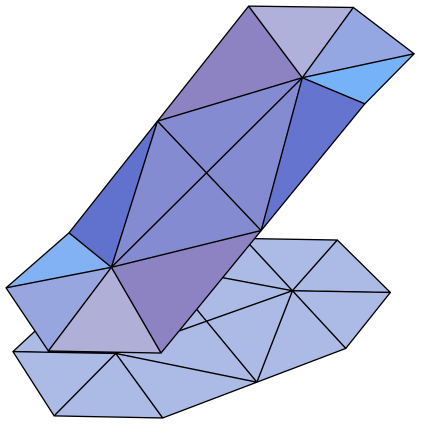
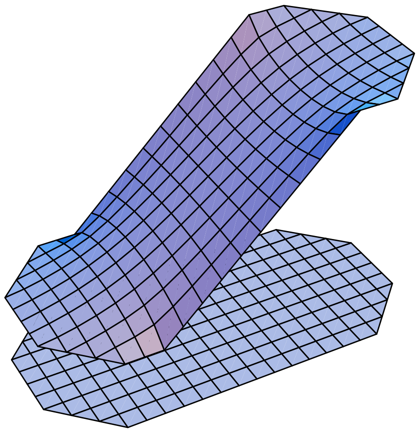



Figure 14.16. Solution to the Dirichlet Problem for the Oval Plate.

first three rows form the boundary coefficient matrix

$$\widetilde{K} = \begin{pmatrix} 0 & -.7887 & -.5774 & -.5774 & -.7887 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -.7887 & -.5774 & -.5774 & -.7887 \end{pmatrix}. \tag{14.56}$$

We similarly split the coefficients c_i of the finite element function (14.54) into two groups. We let $\mathbf{c} \in \mathbb{R}^n$ denote the as yet unknown coefficients c_i corresponding to the values of the approximation at the interior nodes $\mathbf{x}_i$, while $\mathbf{h} \in \mathbb{R}^{m-n}$ will be the vector of boundary values (14.55). The solution to the finite element approximation (14.54) is obtained by solving the associated linear system

$$K\mathbf{c} + \widetilde{K}\,\mathbf{h} = \mathbf{b}, \qquad \text{or} \qquad K\mathbf{c} = \mathbf{f} = \mathbf{b} - \widetilde{K}\,\mathbf{h}. \tag{14.57}$$

Example 14.10. For the oval plate discussed in Example 14.8, suppose the right hand semicircular edge is held at 10°, the left hand semicircular edge at -10°, while the two straight edges have a linearly varying temperature distribution ranging from -10° at the left to 10° at the right, as illustrated in Figure 14.16. Our task is to compute its equilibrium temperature, assuming no internal heat source. Thus, for the coarse triangulation we have the boundary nodes values

$$\mathbf{h} = (\,h_4, \ldots, h_{13}\,)^T = (\,0, -1, -1, -1, -1, 0, 1, 1, 1, 1, 0\,)^T.$$

Using the previously computed formulae (14.49, 56) for the interior coefficient matrix K and boundary coefficient matrix $\widetilde{K}$, we approximate the solution to the Laplace equation by solving (14.57). We are assuming that there is no external forcing function, $f(x,y) \equiv 0$, and so the right hand side is $\mathbf{b} = \mathbf{0}$, and so we must solve $K\mathbf{c} = \mathbf{f} = -\widetilde{K}\,\mathbf{h} = (\,2.18564, 3.6, 7.64974\,)^T$. The finite element function corresponding to the solution $\mathbf{c} = (\,1.06795, 1.8, 2.53205\,)^T$ is plotted in the first illustration in Figure 14.16. Even on such a coarse mesh, the approximation is not too bad, as evidenced by the second illustration, which plots the finite element solution for a square mesh with spacing $h = .2$ between nodes.

References

[**1**] Alligood, K.T., Sauer, T.D., and Yorke, J.A., *Chaos. An Introduction to Dynamical Systems*, Springer-Verlag, New York, 1997.

[**2**] Apostol, T.M., *Calculus*, Blaisdell Publishing Co., Waltham, Mass., 1967–69.

[**3**] Baker, G.A., Jr., and Graves–Morris, P., *Padé Approximants*, Encyclopedia of Mathematics and Its Applications, v. 59, Cambridge University Press, Cambridge, 1996.

[**4**] Birkhoff, G., and Rota, G.–C., *Ordinary Differential Equations*, Blaisdell Publ. Co., Waltham, Mass., 1962.

[**5**] Bradie, B., *A Friendly Introduction to Numerical Analysis*, Prentice–Hall, Inc., Upper Saddle River, N.J., 2006.

[**6**] Bronstein, M., and Lafaille, S., Solutions of linear ordinary differential equations in terms of special functions, *in*: *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, T. Mora, ed., ACM, New York, 2002, pp. 23–28.

[**7**] Burden, R.L., and Faires, J.D., *Numerical Analysis*, Seventh Edition, Brooks/Cole, Pacific Grove, CA, 2001.

[**8**] Cantwell, B.J., *Introduction to Symmetry Analysis*, Cambridge University Press, Cambridge, 2003.

[**9**] Clenshaw, C.W., and Olver, F.W.J., Beyond floating point, *J. Assoc. Comput. Mach.* **31** (1984), 319–328.

[**10**] Courant, R., and Hilbert, D., *Methods of Mathematical Physics*, vol. I, Interscience Publ., New York, 1953.

[**11**] Davidson, K.R., and Donsig, A.P., *Real Analysis with Real Applications*, Prentice–Hall, Inc., Upper Saddle River, N.J., 2002.

[**12**] DeGroot, M.H., and Schervish, M.J., *Probability and Statistics*, 3rd ed., Addison–Wesley, Boston, 2002.

[**13**] Demmel, J.W., *Applied Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.

[**14**] Devaney, R.L., *An Introduction to Chaotic Dynamical Systems*, Addison–Wesley, Redwood City, Calif., 1989.

[**15**] Farin, G.E., *Curves and Surfaces for CAGD: A Practical Guide*, Academic Press, London, 2002.

[**16**] Feigenbaum, M.J., Qualitative universality for a class of nonlinear transformations, *J. Stat. Phys.* **19** (1978), 25–52.

[**17**] Fine, B., and Rosenberger, G., *The Fundamental Theorem of Algebra*, Undergraduate Texts in Mathematics, Springer–Verlag, New York, 1997.

[**18**] Francis, J.G.F., The QR transformation I, II, *Comput. J.* **4** (1961–2), 265–271, 332–345.

[**19**] Gaal, L., *Classical Galois theory*, 4th ed., Chelsea Publ. Co., New York, 1988.

[**20**] Gohberg, I., and Koltracht, I., Triangular factors of Cauchy and Vandermonde matrices, *Integral Eq. Operator Theory* **26** (1996), 46–59.

[**21**] Hairer, E., Nørsett, S.P., and Wanner, G., *Solving Ordinary Differential Equations*, 2nd ed., Springer–Verlag, New York, 1993–1996.

[**22**] Hale, J.K., *Ordinary Differential Equations*, Second Edition, R. E. Krieger Pub. Co., Huntington, N.Y., 1980.

[**23**] Hamming, R.W., *Numerical Methods for Scientists and Engineers*, McGraw–Hill, New York, 1962.

[**24**] Higham, N.J., *Accuracy and Stability of Numerical Algorithms*, Second Edition, SIAM, Philadelphia, 2002.

[**25**] Hirsch, M.W., and Smale, S., *Differential Equations, Dynamical Systems, and Linear Algebra*, Academic Press, New York, 1974.

[**26**] Hydon, P.E., *Symmetry Methods for Differential Equations*, Cambridge Texts in Appl. Math., Cambridge University Press, Cambridge, 2000.

[**27**] Ince, E.L., *Ordinary Differential Equations*, Dover Publ., New York, 1956.

[**28**] Iserles, A., *A First Course in the Numerical Analysis of Differential Equations*, Cambridge University Press, Cambridge, 1996.

[**29**] Jolliffe, I.T., *Principal Component Analysis*, 2nd ed., Springer–Verlag, New York, 2002.

[**30**] Krall, A.M., *Applied Analysis*, D. Reidel Publishing Co., Boston, 1986.

[**31**] Kublanovskaya, V.N., On some algorithms for the solution of the complete eigenvalue problem, *USSR Comput. Math. Math. Phys.* **3** (1961), 637–657.

[**32**] Lanford, O., A computer-assisted proof of the Feigenbaum conjecture, *Bull. Amer. Math. Soc.* **6** (1982), 427–434.

[**33**] Mandelbrot, B.B., *The Fractal Geometry of Nature*, W.H. Freeman, New York, 1983.

[**34**] Marsden, J.E., and Tromba, A.J., *Vector Calculus*, 4th ed., W.H. Freeman, New York, 1996.

[**35**] Moon, F.C., *Chaotic Vibrations*, John Wiley & Sons, New York, 1987.

[**36**] Olver, F.W.J., *Asymptotics and Special Functions*, Academic Press, New York, 1974.

[**37**] Olver, P.J., *Applications of Lie Groups to Differential Equations*, 2nd ed., Graduate Texts in Mathematics, vol. 107, Springer–Verlag, New York, 1993.

[**38**] Olver, P.J., and Shakiban, C., *Applied Linear Algebra*, Prentice–Hall, Inc., Upper Saddle River, N.J., 2005.

[**39**] Olver, P.J., and Shakiban, C., *Applied Mathematics*, Prentice–Hall, Inc., Upper Saddle River, N.J., to appear.

[**40**] Ortega, J.M., *Numerical Analysis; A Second Course*, Academic Press, New York, 1972.

[**41**] Orucc, H., and Phillips, G. M., Explicit factorization of the Vandermonde matrix, *Linear Algebra Appl.* **315** (2000), 113–123.

[**42**] Peitgen, H.-O., and Richter, P.H., *The Beauty of Fractals: Images of Complex Dynamical Systems*,Springer–Verlag, New York, 1986.

[**43**] Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed., Cambridge University Press, Cambridge, 1995.

[**44**] Schumaker, L.L., *Spline Functions: Basic Theory*, John Wiley & Sons, New York, 1981.

[**45**] Strang, G., and Fix, G.J., *An Analysis of the Finite Element Method*, Prentice–Hall, Inc., Englewood Cliffs, N.J., 1973.

[**46**] Tannenbaum, P., *Excursions in Modern Mathematics*, 5th ed., Prentice–Hall, Inc., Upper Saddle River, N.J, 2004.

[**47**] Varga, R.S., *Matrix Iterative Analysis*, 2nd ed., Springer–Verlag, New York, 2000.

[**48**] Watkins, D.S., *Fundamentals of Matrix Computations*, Second Edition,, Wiley–Interscience, New York, 2002.

[**49**] Zienkiewicz, O.C., and Taylor, R.L., *The Finite Element Method*, 4th ed., McGraw–Hill, New York, 1989.