AnT 4.669

Release 2a

Examples

University of Stuttgart
Nonlinear Dynamics Group

April 8, 2003

Contents

Chapter 1

Introduction

1.1 About this document

This document presents some application examples for the `AnT` package.

Chapter 2

Rössler system

2.1 Definition of the system

The Rössler system is a dynamical system continuous in time, defined by the following ordinary differential equation:

$$\begin{aligned} \dot{x} &= -(y+z) \\ \dot{y} &= x + ay \\ \dot{z} &= b + z(x-c) \end{aligned} \tag{2.1}$$

For the investigation within the `AnT` package the system function 2.1 has to be implemented according to the interface for ordinary differential equations[1] and connected to the simulator using the `ODE_Proxy`, as follows:

[1] See Reference Manual, Part II

```
#include "AnT.hpp"

#define a     parameters[0]
#define b     parameters[1]
#define c     parameters[2]
#define X     currentState[0]
#define Y     currentState[1]
#define Z     currentState[2]

bool roessler (const Array<real_t>& currentState,
       const Array<real_t>& parameters,
       Array<real_t>& rhs)
{
  rhs[0] = -(Y + Z);
  rhs[1] = X + a * Y;
  rhs[2] = b + Z * (X - c);

  return true;
}

#undef a
#undef b
#undef c
#undef X
#undef Y
#undef Z

extern "C" {

void connectSystem ()
{
  ODE_Proxy::systemFunction = roessler;
}

}

#endif
```

The part of the initialization file, needed for the simulation of the Rössler system, concerning the dynamical system itself and its simulation, can look like the following:

```
dynamical_system =
{  type = "ode",
   name = "roessler",
   state_space_dimension = 3,
   initial_state = (1.0, 1.1, 1.2 ),
   parameter_space_dimension = 3,
   parameters = { parameter[0] = {value = 0.2,
                                  name = "a"},
                  parameter[1] = {value = 0.15,
                                  name = "b"},
                  parameter[2] = {value = 6.2,
                                  name = "c"}
                },
   number_of_iterations = 250000,

   integration = { method = "rk44",
                   step = 0.001,
                 }
}
```

These settings specify, that the system is an ordinary differential equation with three variables and three parameters. The initial values for the state variables and the values of the parameters are given. Additionally the number of the integration steps, integration method (Runge–Kutta method of the order 4 without step size adaption) and the integration step size (10^{-3} time units) are set.

2.2 A chaotic attractor

Firstly we investigated the Rössler system at fixed parameter settings, $a = 0.2$, $b = 0.15$, $c = 6.2$. It can be shown, that the system possesses a chaotic attractor at these parameter values.
In this case we perform a single run of the simulator. It means, that all settings are fixed. For specification of this, the following entry in the initialization file is needed:

```
scan = {
  scan_mode = 0
}
```

2.2.1 Time series

Firstly we produce simple time series of the Rössler system. The appropriate settings in the initialization file can be the following:

```
investigation_methods =
{
  general_trajectory_evaluations =
  {    is_active = "yes",
       transient = 100000,
       saving =
       {
         is_active = "yes",
         type = "time_oriented",

         trajectory = "yes",
         velocity = "no",
         phase_portrait = "no",
         initial_states = "no",

         points_step = 25,
         save_only_specific_area = "no"
       }
  }
}
```

The number of the transient steps 100000 combined with the integration step size 10^{-3}, which is specified in the description of the dynamical system, implies, that the orbit will be saved for time $t > 100$ time units. The end time of the saving is 250 time units, because the complete number of iteration steps, diven in the description of the dynamical system, is 250000. The saving in equidistant with the step 0.25 time units. Figures 2.1, 2.1 and 2.1 show the corresponding time series for the state variables x, y and z.

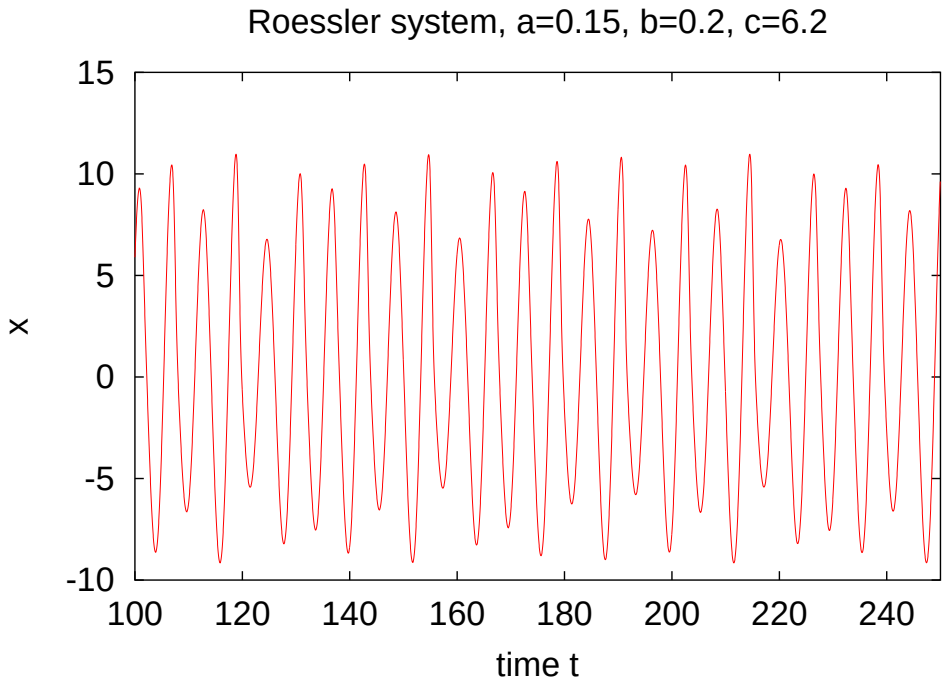
Roessler system, a=0.15, b=0.2, c=6.2
15
10
5
0
-5
-10
x
100
120
140
160
180
200
220
240
time t

Figure 2.1:

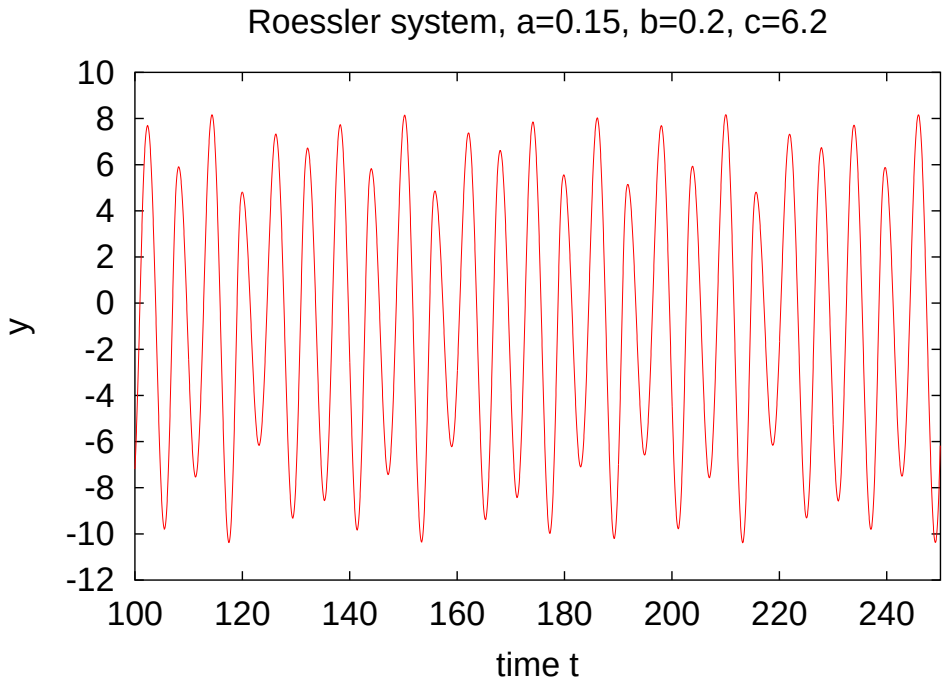
Roessler system, a=0.15, b=0.2, c=6.2
y
time t
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
100
120
140
160
180
200
220
240

Figure 2.2:

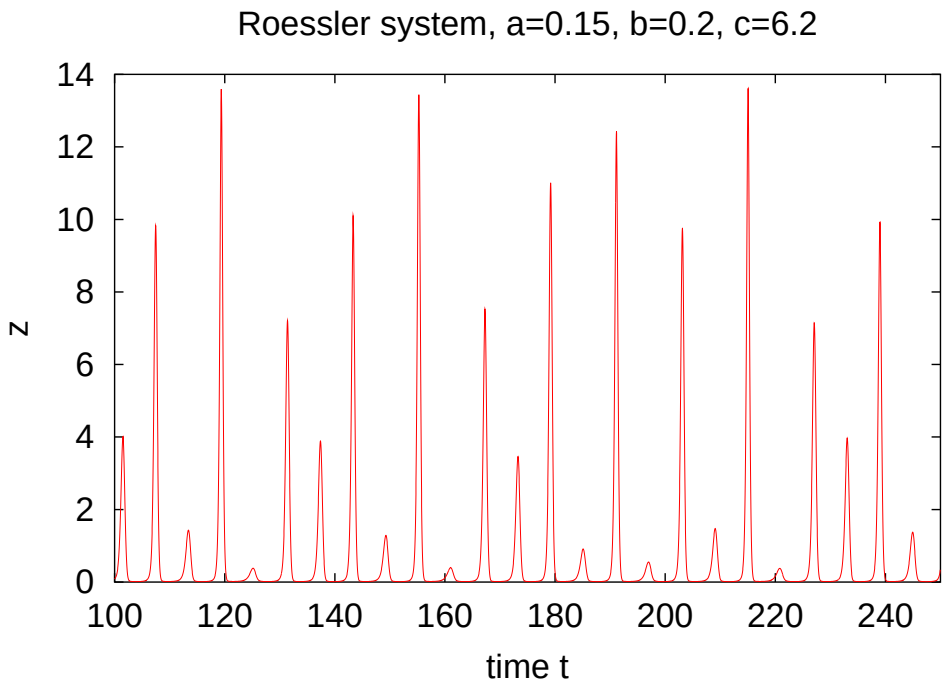
Roessler system, a=0.15, b=0.2, c=6.2
z
time t
0
2
4
6
8
10
12
14
100
120
140
160
180
200
220
240

Figure 2.3:

2.2.2 Phase portraits (I)

There are two different types of plots, which are usually denoted as phase portraits. The first and the simplest possibility is to plot the state variables against each other. Figure 2.4 presents the Rössler attractor in the state space $[x \times y \times z]$ at the parameter setting $a = 0.2$, $b = 0.15$, $c = 6.2$. Figures 2.5, 2.6 and 2.7 show the projections of the attractor on the $[x \times y]$, $[x \times z]$ and $[y \times z]$ planes. For producing these plots either the file `orbit.gra` or the file `phase_portrait.gra` can be used. Note, that it is more suitable to use the space oriented saving type, because the produced files are less in this case. Doing this, one has to specify the minimal distances between two subsequent points which are saved. The appropriate settings in the initialization file can be the following:

```
investigation_methods =
{
  general_trajectory_evaluations =
  {    is_active = "yes",
       transient = 100000,
       saving =
       {
          is_active = "yes",
          type = "space_oriented",

          trajectory = "no",
          velocity = "no",
          phase_portrait = "yes",
          initial_states = "no",

          state_space_distance = 0.5,
          velocity_space_distance = 1.0,

          save_only_specific_area = "no"
       }
  }
};
```

The Figures 2.4, 2.5, 2.6 and 2.7 are produced using 2500000 iteration steps.

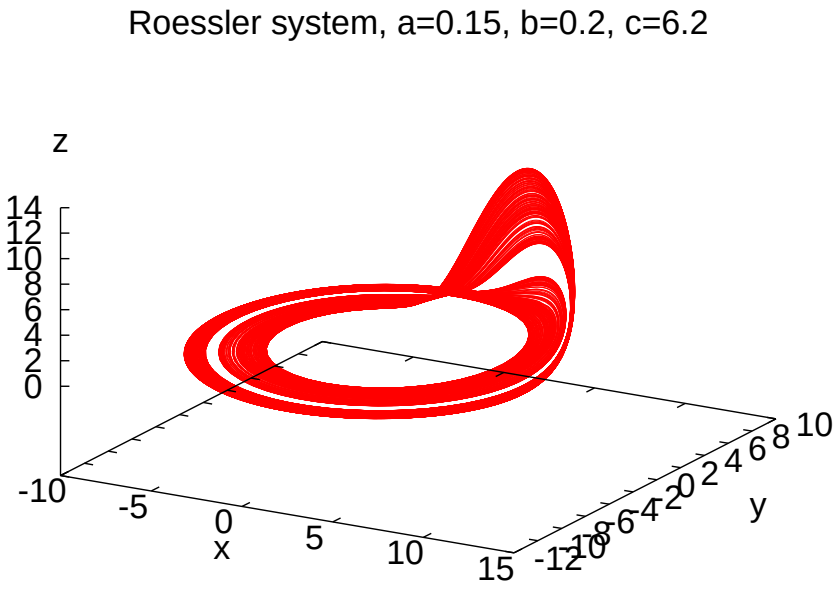
Roessler system, a=0.15, b=0.2, c=6.2
z
14
12
10
8
6
4
2
0
-10
-5
0
5
10
15
x
-12
-10
-8
-6
-4
-2
0
2
4
6
8
10
y

Figure 2.4:

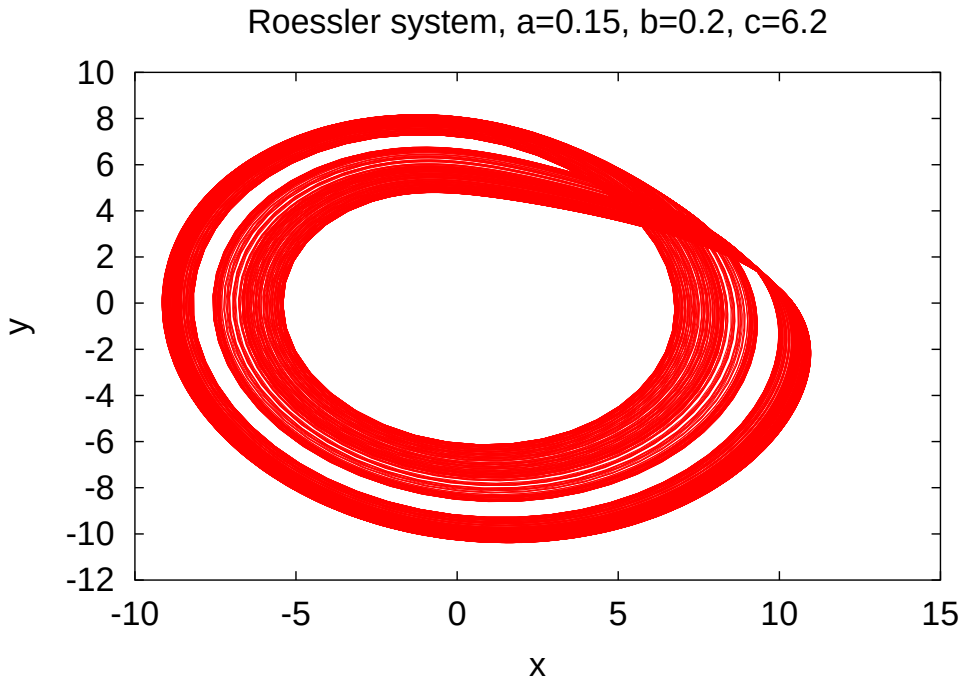
Roessler system, a=0.15, b=0.2, c=6.2
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
y
-10
-5
0
5
10
15
x

Figure 2.5:

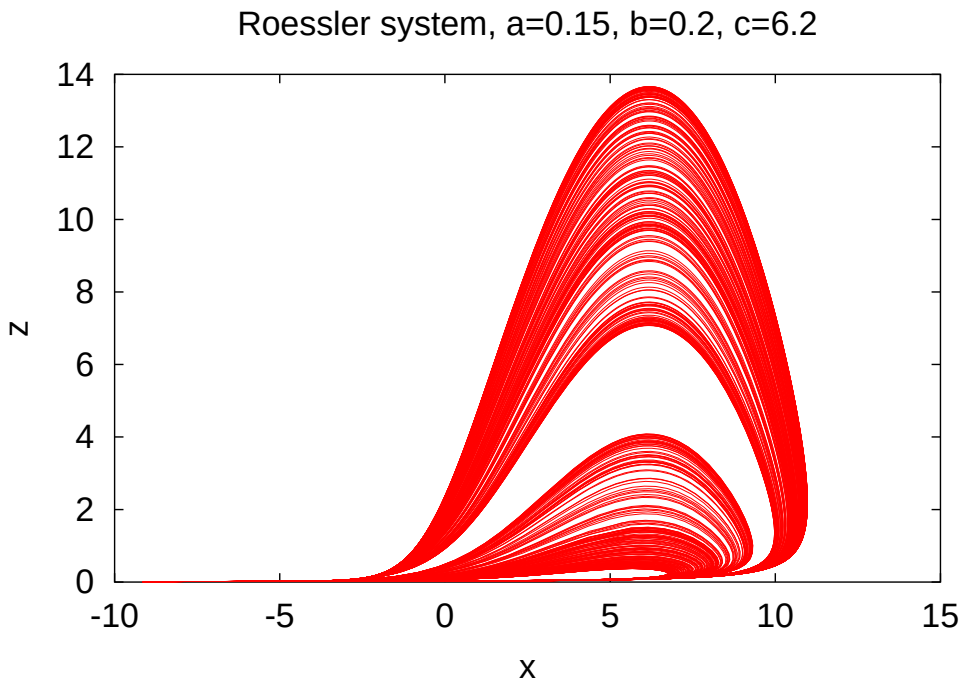

Figure 2.6:

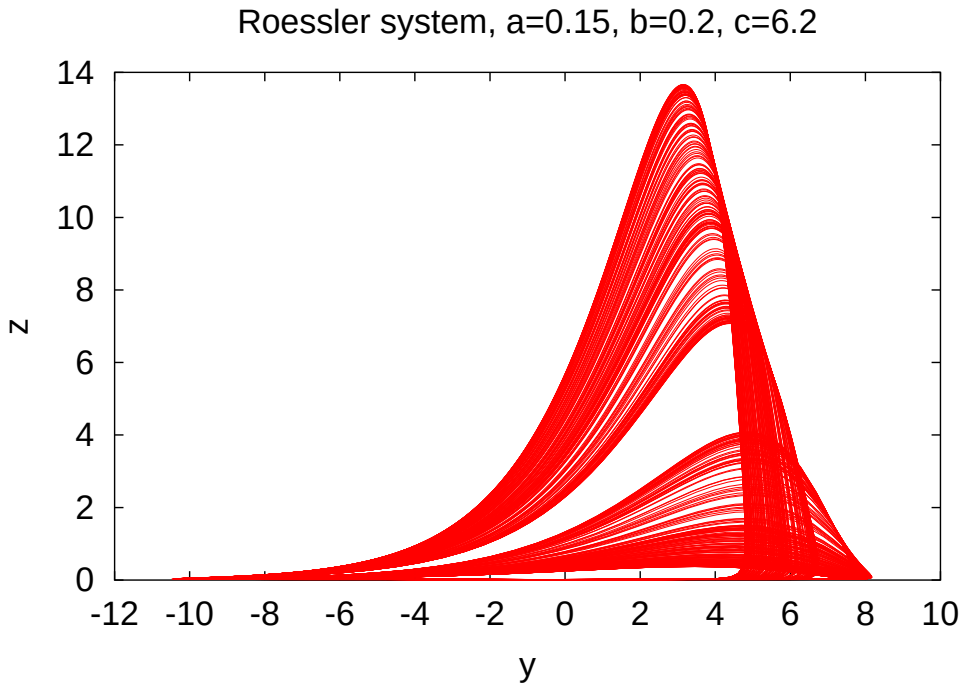
Roessler system, a=0.15, b=0.2, c=6.2
z
y
14
12
10
8
6
4
2
0
-12
-10
-8
-6
-4
-2
0
2
4
6
8
10

Figure 2.7:

2.2.3 Phase portraits (II)

Another kind of plots usually denoted as phase portraits are plots showing the state variables against their first derivatives (velocities). Using the settings described above and the file `phase_portrait.gra` one is able to produce these plots also. Figures 2.8, 2.9 and 2.10 show the Rössler attractor in the planes $[x \times \dot{x}]$, $[y \times \dot{y}]$ and $[z \times \dot{z}]$. Additionally the space $[\dot{x} \times \dot{y} \times \dot{z}]$ is shown in Figures 2.11.

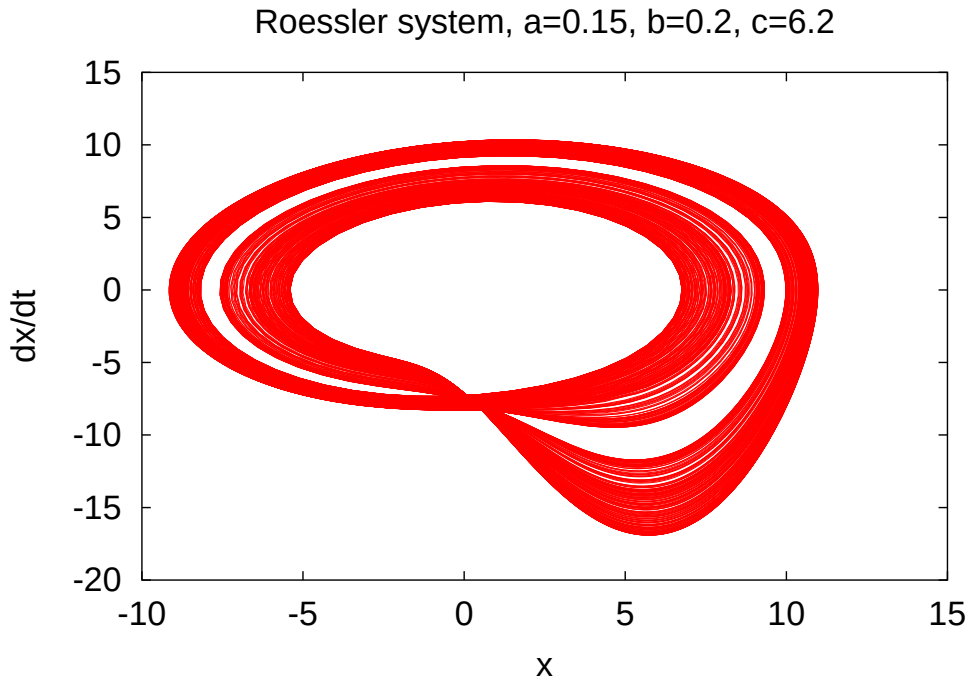
Roessler system, a=0.15, b=0.2, c=6.2
dx/dt
x
15
10
5
0
-5
-10
-15
-20
-10
-5
0
5
10
15

Figure 2.8:

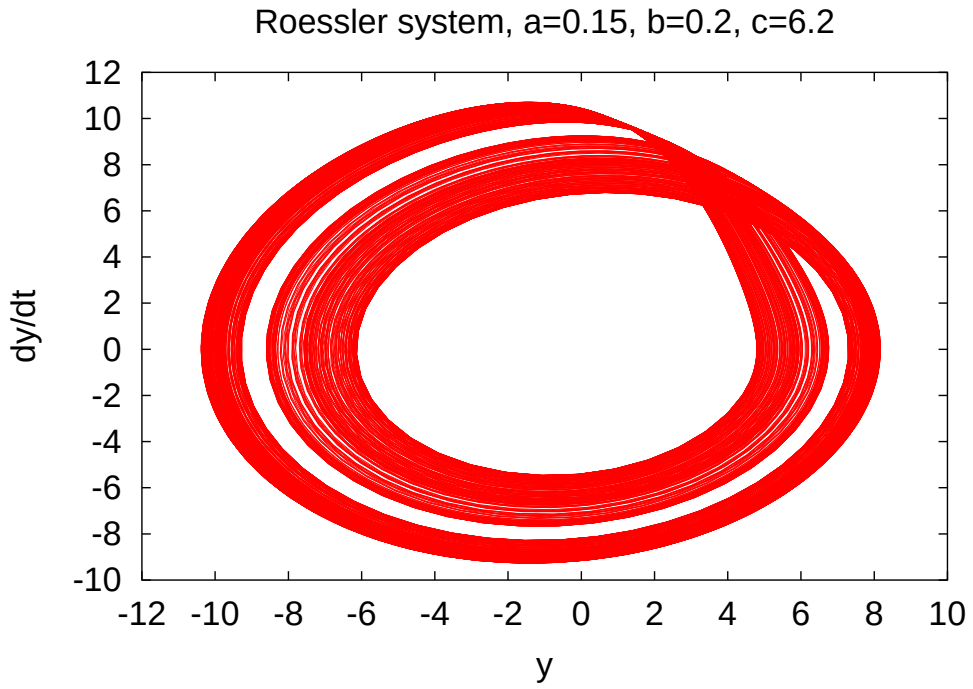
Roessler system, a=0.15, b=0.2, c=6.2
dy/dt
y
12
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
-10
-8
-6
-4
-2
0
2
4
6
8
10

Figure 2.9:

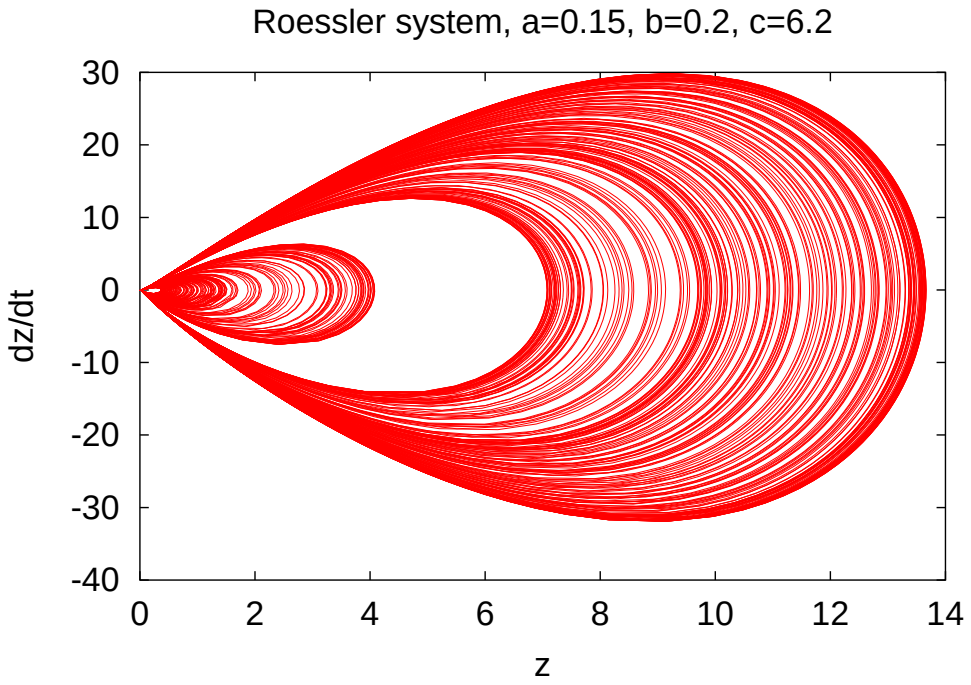
Roessler system, a=0.15, b=0.2, c=6.2
dz/dt
z
30
20
10
0
-10
-20
-30
-40
0
2
4
6
8
10
12
14

Figure 2.10:

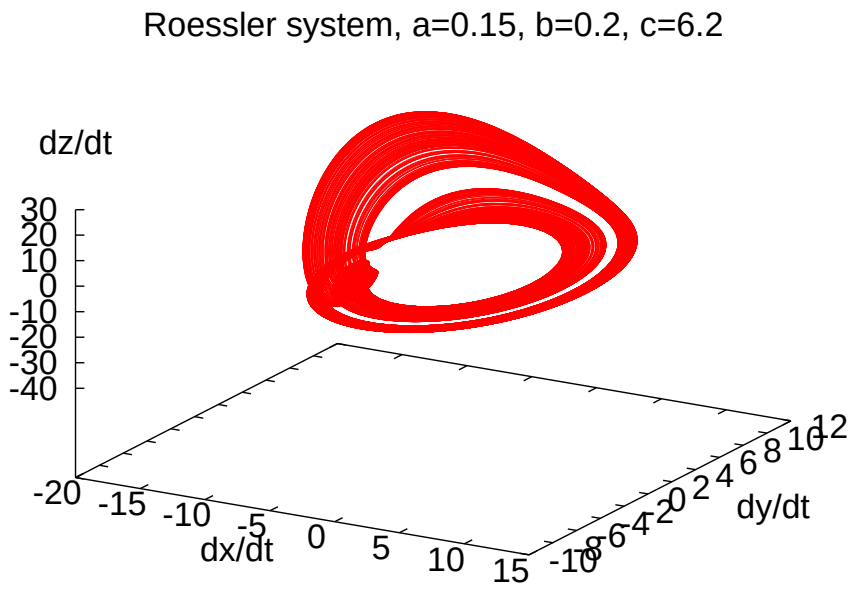
Roessler system, a=0.15, b=0.2, c=6.2
dz/dt
30
20
10
0
-10
-20
-30
-40
-20
-15
-10
-5
0
5
10
15
dx/dt
-10
-8
-6
-4
-2
0
2
4
6
8
10
12
dy/dt

Figure 2.11:

2.2.4 Power spectra

```
investigation_methods =
{
    frequency_analysis =
    {   is_active = "yes",
        points_step = 1,
        transient = 10000,
        using_variables = (0,1,2),
        frequency_output_range = (0.0, 5.0),
        frequency_step_is_given = "yes",

        frequency_step = 0.0005,
        subtract_mean_value = "on",

        neighborhood_width = 5,
        frequency_weighting = (1.0, 2.0, 3.0, 2.0, 1.0),

        real_part = "off",
        imaginary_part = "off",
        real_and_imaginary_parts = "off",
        autocorrelation = "off",
        total_power = "off",

        power_spectrum = "on",
        spectrum_max_points = "off",

        total_power = "off",
        period = "off",
        spectrum_waving = "off",
        spectrum_oscillation = "off",

        fourier_coefficients = "off"
    }
}
```

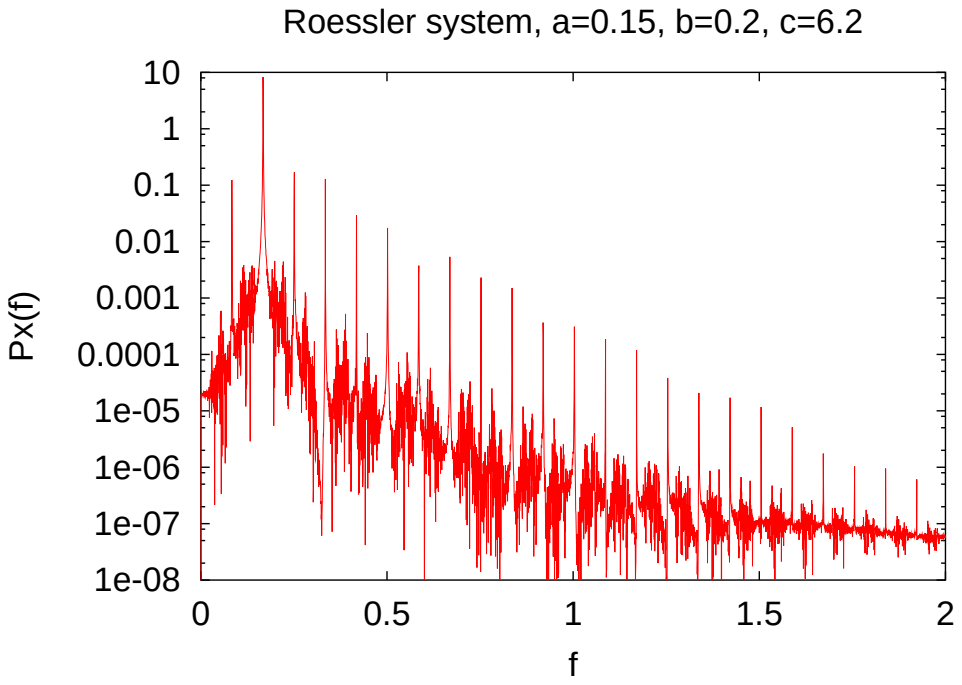
Roessler system, a=0.15, b=0.2, c=6.2
Px(f)
10
1
0.1
0.01
0.001
0.0001
1e-05
1e-06
1e-07
1e-08
0
0.5
1
1.5
2
f

Figure 2.12:

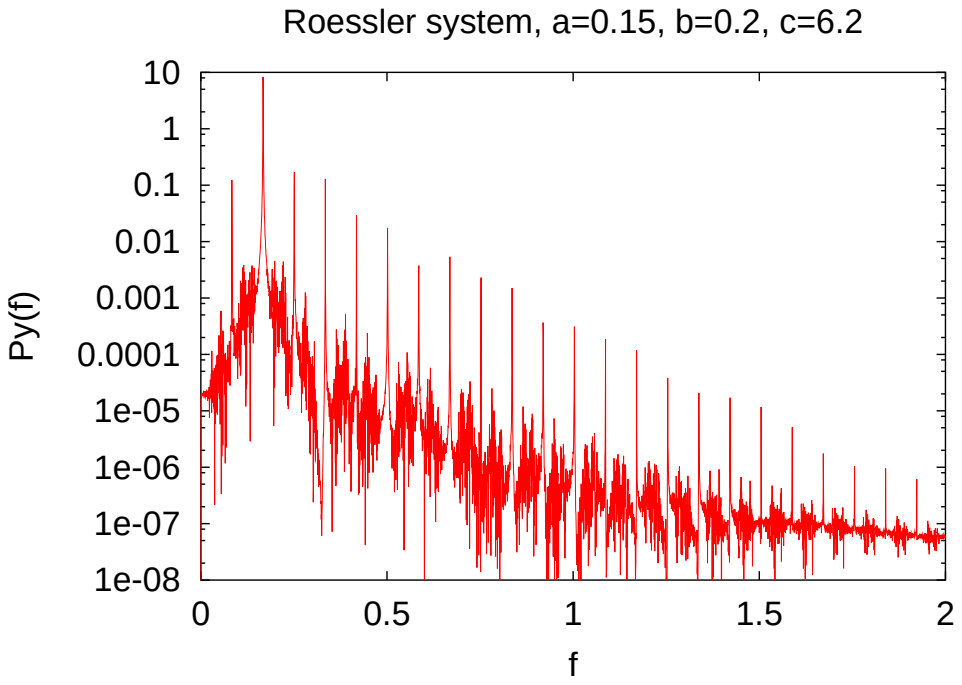
Roessler system, a=0.15, b=0.2, c=6.2
Py(f)
10
1
0.1
0.01
0.001
0.0001
1e-05
1e-06
1e-07
1e-08
0
0.5
1
1.5
2
f

Figure 2.13:

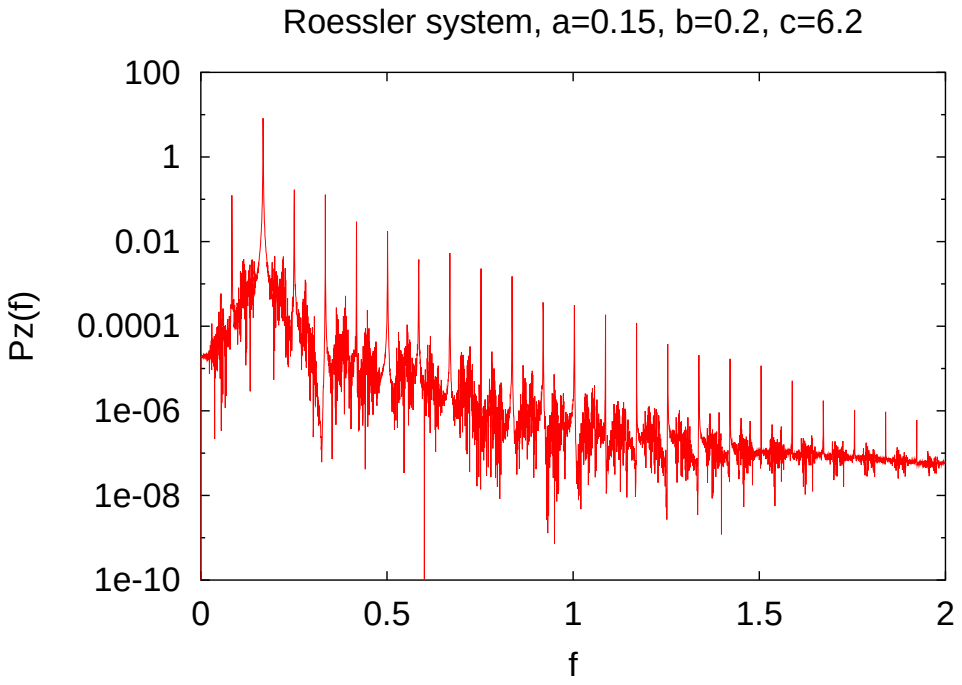
Roessler system, a=0.15, b=0.2, c=6.2
100
1
0.01
0.0001
1e-06
1e-08
1e-10
Pz(f)
0
0.5
1
1.5
2
f

Figure 2.14:

2.2.5 Invariant measure

```
investigation_methods =
{
  dimensions_analysis = {
        is_active = "yes",
        transient = 10000,
        ranges = ((-10,-12,0),(12,10,14)),
        max_layer = 10,
        partition_factor = 2,
        invariant_measure = "on",
        measure_deviation = "off",
        metric_entropy = "off",
        capacity_dimension = "off",
        information_dimension = "off",
        correlation_dimension = "off"
  }
}
```

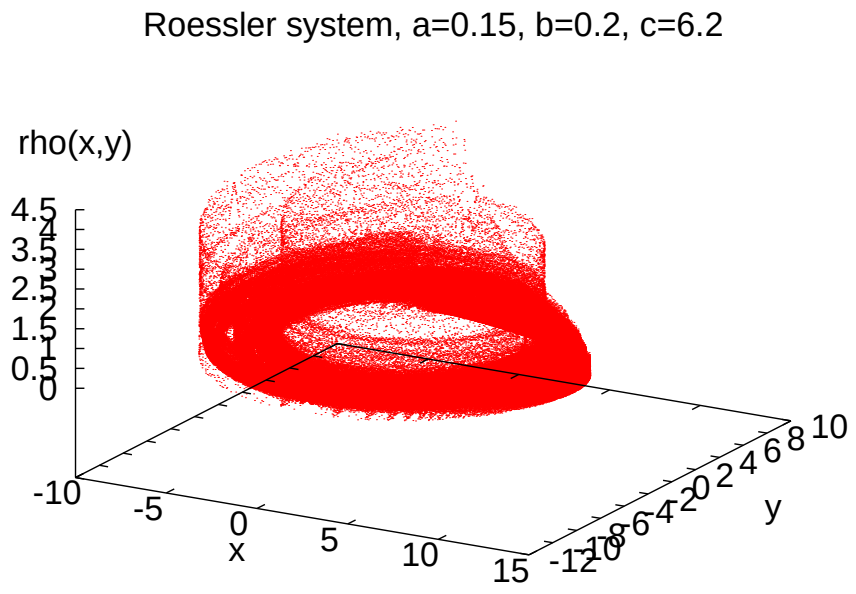
Roessler system, a=0.15, b=0.2, c=6.2
rho(x,y)
4.5
3.5
2.5
1.5
0.5
0
-10
-5
0
5
10
15
x
-12
-10
-8
-6
-4
-2
0
2
4
6
8
10
y

Figure 2.15:

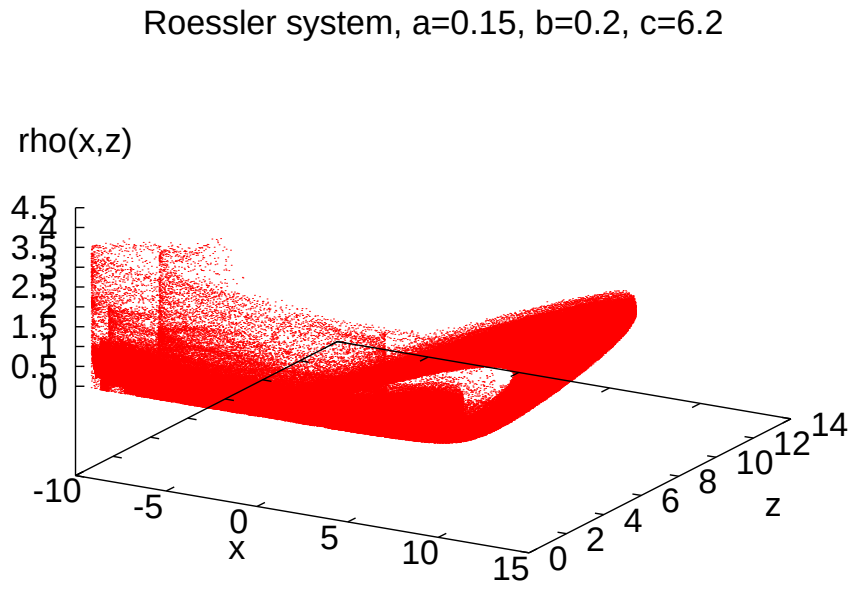
Roessler system, a=0.15, b=0.2, c=6.2
rho(x,z)
4.5
3.5
2.5
1.5
0.5
0
-10
-5
0
x
5
10
15
0
2
4
6
8
10
12
14
z

Figure 2.16:

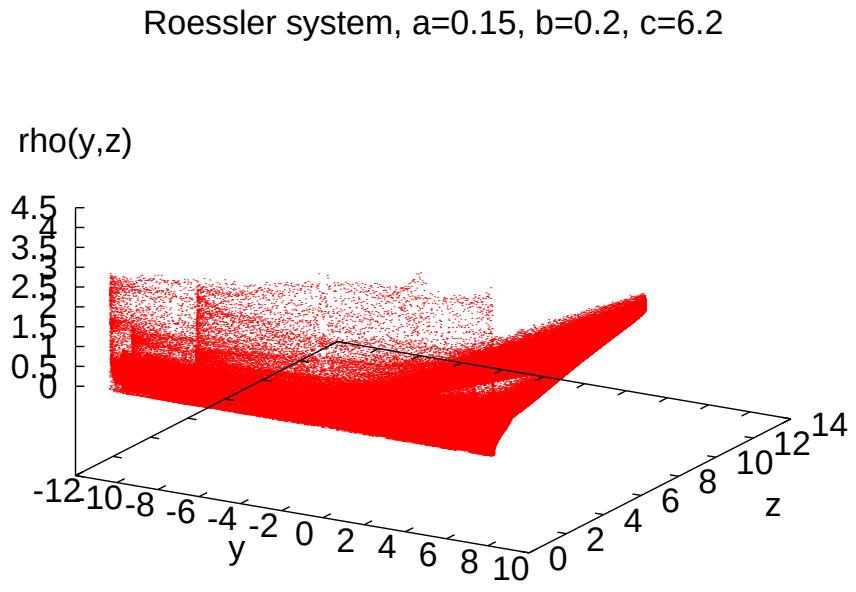
Roessler system, a=0.15, b=0.2, c=6.2
rho(y,z)
4.5
3.5
2.5
1.5
0.5
0
-12
-10
-8
-6
-4
-2
0
2
4
6
8
10
y
0
2
4
6
8
10
12
14
z

Figure 2.17:

2.3 Lyapunov exponents

Chapter 3

Phase locked loop

3.1 Definition of the system

The phase locked loop is a dynamical system continuous in time with time delay, defined by:

$$\dot{x}(t) \quad = \quad -R\sin(x(t-\tau)) \tag{3.1}$$

For the investigation within the `AnT` package the system function 3.1 has to be implemented according to the interface for delay differential equations[1] and connected to the simulator using the `DDE_Proxy`, as follows:

[1] See Reference Manual, Part II

```
#include "AnT.hpp"

#define R     parameters[0]
#define Xt    delayState[0]

bool delayed_pll (const Array<real_t>& currentState,
          const Array<real_t>& delayState,
          const Array<real_t>& parameters,
          Array<real_t>& rhs)
{
  rhs[0] = -R * sin (Xt);
  return true;
}


#undef R
#undef Xt

extern "C" {

void connectSystem ()
{
  DDE_Proxy::systemFunction = delayed_pll;
}

}
```

The part of the initialization file, needed for the simulation of the phase locked loop, concerning the dynamical system itself and its simulation, can look like the following:

```
dynamical_system =
{
  type = "dde",
  name = "pll",
  delay = 1

  state_space_dimension = 1,
  temporal_initial_function[0] = { type = "const",
                                   const_value = 1.0
                                 },
  parameter_space_dimension = 1,
  parameters = { parameter[0] = { value = 4.157,
                                  name = "R" }
               },
  integration =  { step_size = 0.001,
                   method = "rk44"
                 },
  number_of_iterations = 10000,
}
```

These settings specify, that the system is a delay differential equation with one state variable and one parameter. The delay time ($\tau = 1$), the temporal initial function for the single state variable and the value of the parameter are given. Additionally the number of the integration steps, integration method (Runge–Kutta method of the order 4 without step size adaption) and the integration step size (10^{-3} time units) are set.

3.2 Initial functions

In the examples presented in this section several temporal initial functions will be tested.

3.2.1 Constant initial function

```
...
temporal_initial_function[0] = { type = "const",
                                 const_value = 1.0
                               }
...
```

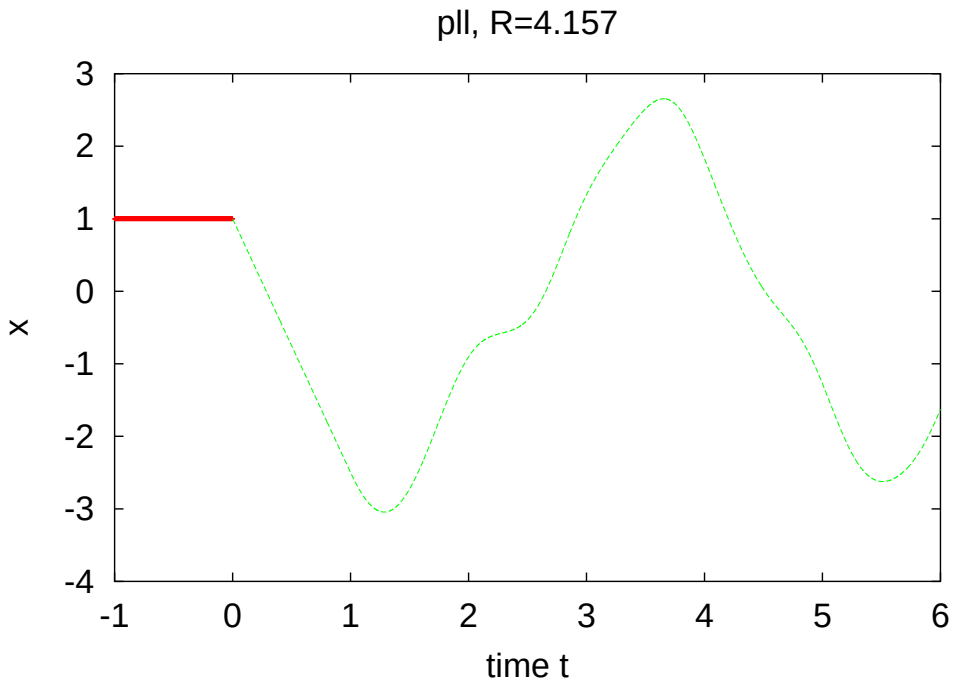

Figure 3.1:

3.2.2 Fermi initial function

```
...
temporal_initial_function[0] = {type = "fermi",
                                edge = -0.5,
                                offset = -1.0,
                                mu = 2.0,
                                sigma = 0.025
                               }
...
```

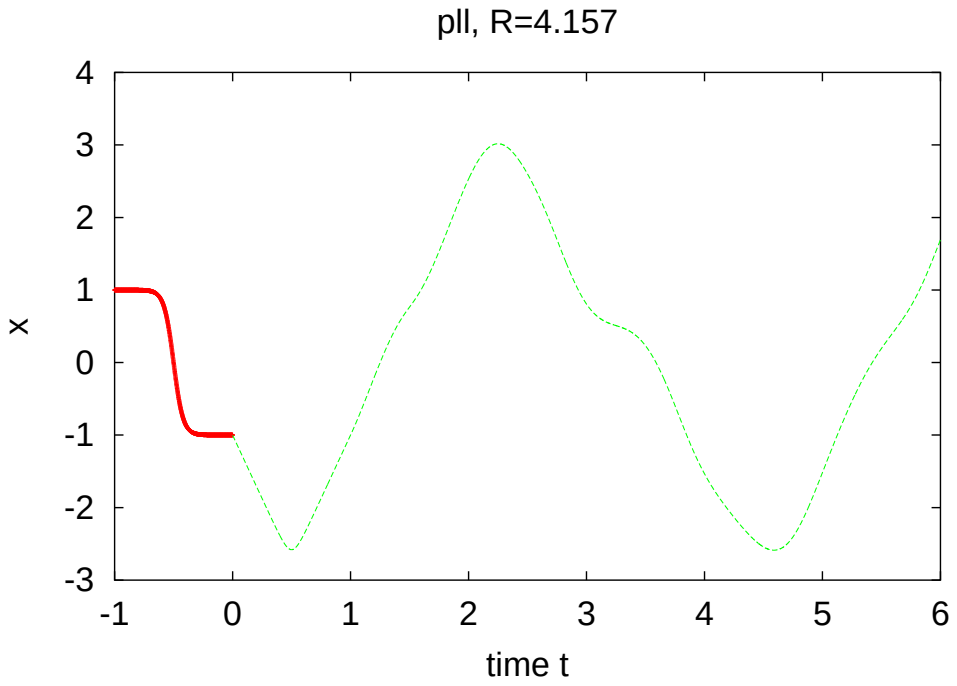

Figure 3.2:

3.2.3 Gauss initial function

```
...
temporal_initial_function[0] = {type = "gauss",
                                amplitude = 2.0,
                                offset = 0.5,
                                mu = -0.5,
                                sigma = 0.15
                               }
...
```

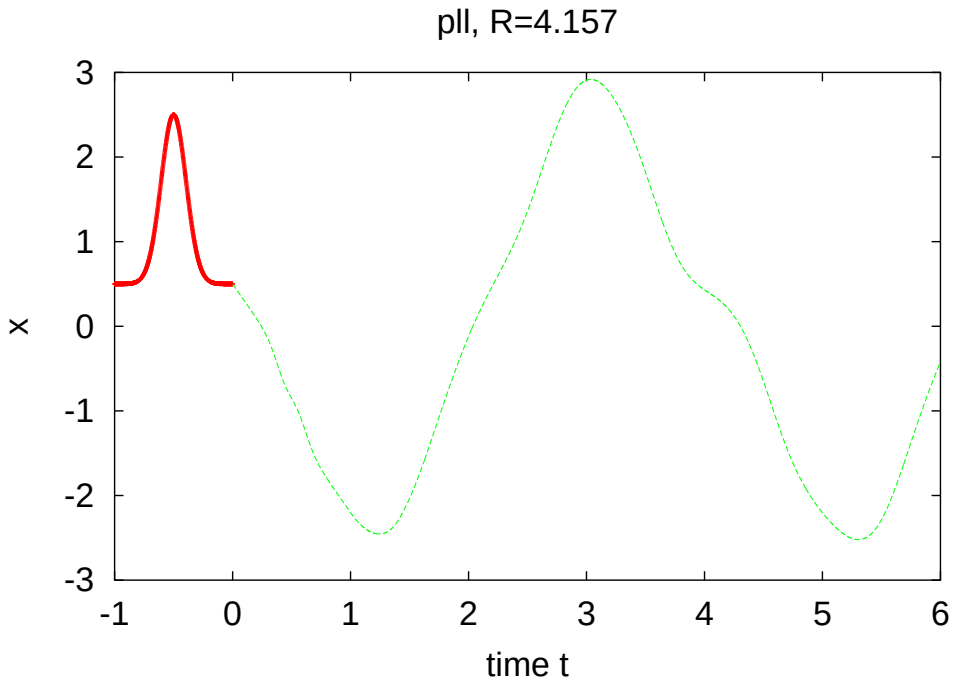

Figure 3.3:

3.2.4 Linear initial function

```
...
temporal_initial_function[0] = {type = "linear",
                                slope = 0.5,
                                offset = 1.0
                               }
...
```

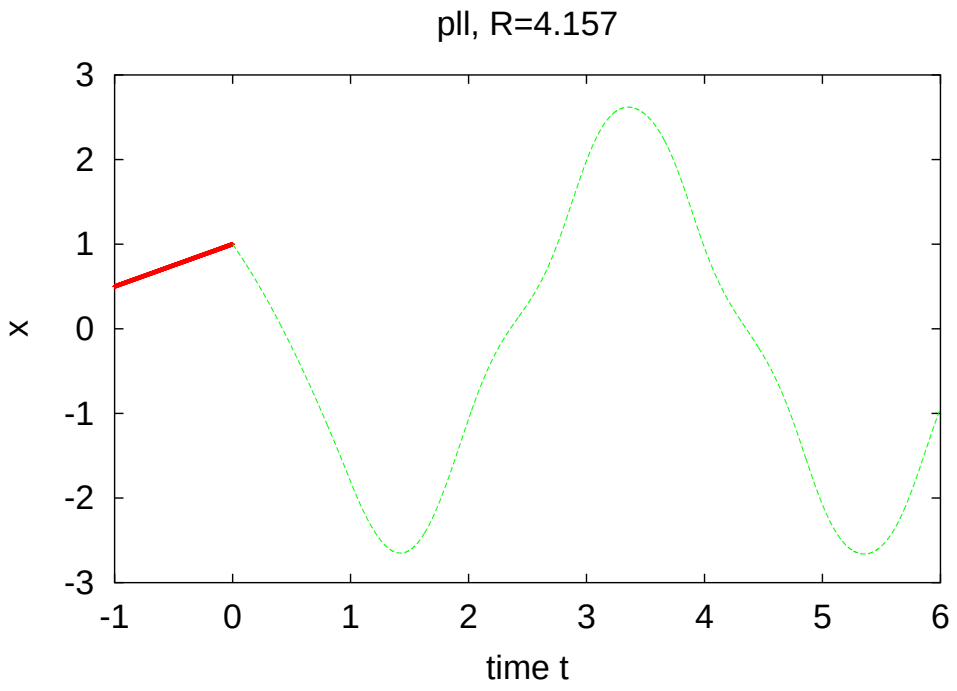

Figure 3.4:

3.2.5 Sinus initial function

```
...
temporal_initial_function[0] = {type = "sin",
                                amplitude = 1.0,
                                frequency = 2,
                                offset = 0.5,
                                phase = 0.0
                               }
...
```

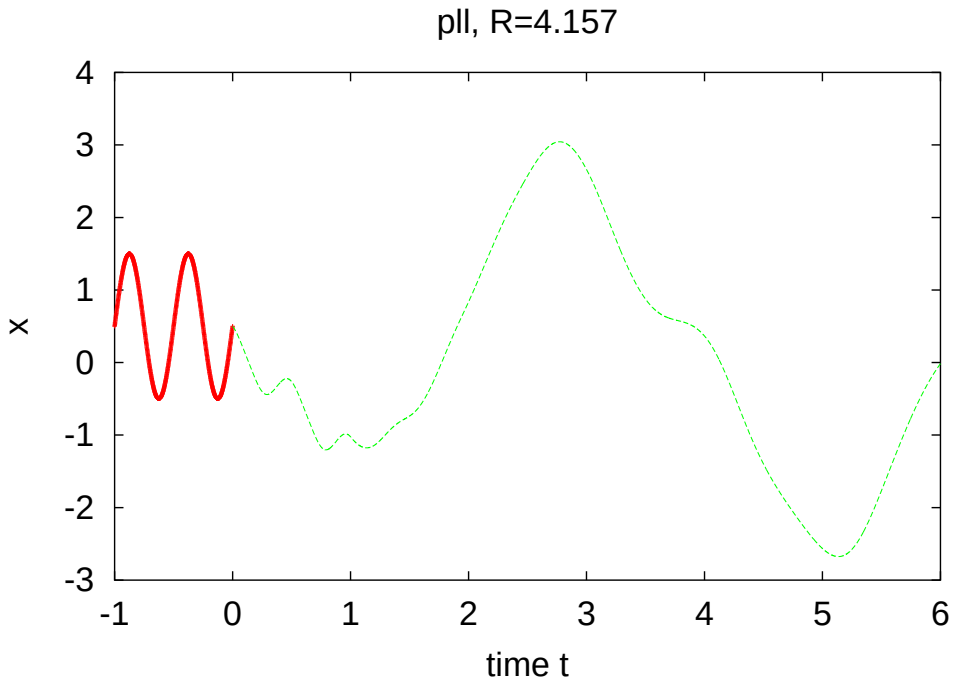

Figure 3.5:

3.2.6 Sinc initial function

```
...
temporal_initial_function[0] = {type = "sinc",
                                amplitude = 2.0,
                                frequency = 4,
                                offset = 0.5,
                                mu = -0.5,
                                phase = 0.0
                               }
...
```

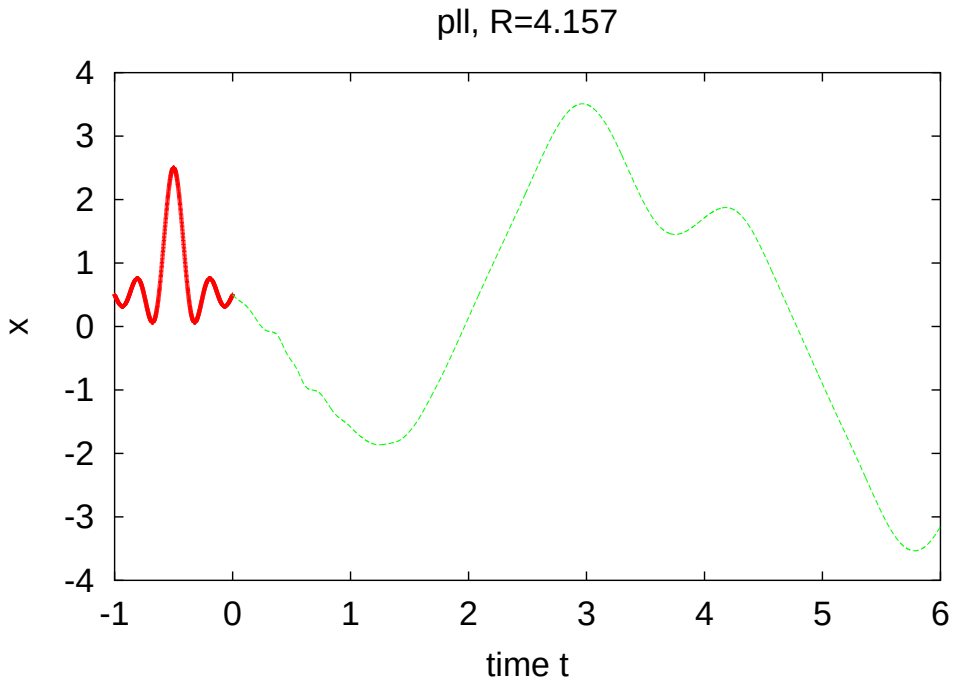

Figure 3.6:

3.2.7 Singular initial function

```
...
temporal_initial_function[0] = {type = "singular",
                                singular_value = 1.5,
                                residual_value = -0.5,
                                singular_time = -0.5
                               }
...
```

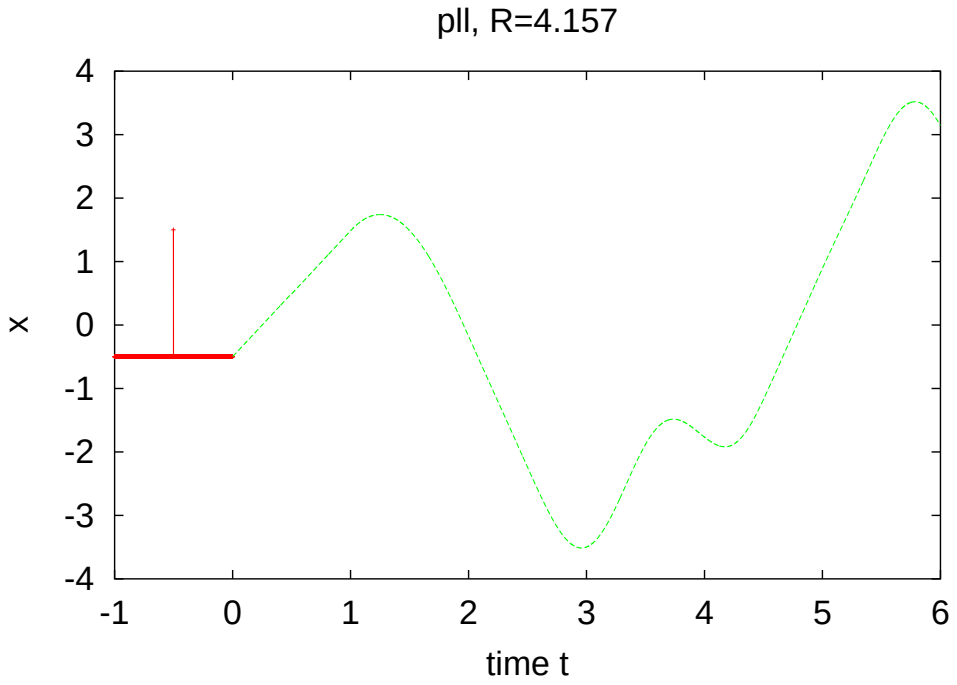

Figure 3.7:

3.2.8 Step initial function

```
...
temporal_initial_function[0] = {type = "step",
                                step_value = 1.0,
                                residual_value = 0.0,
                                first_time = -0.9,
                                second_time = -0.7
                               }
...
```

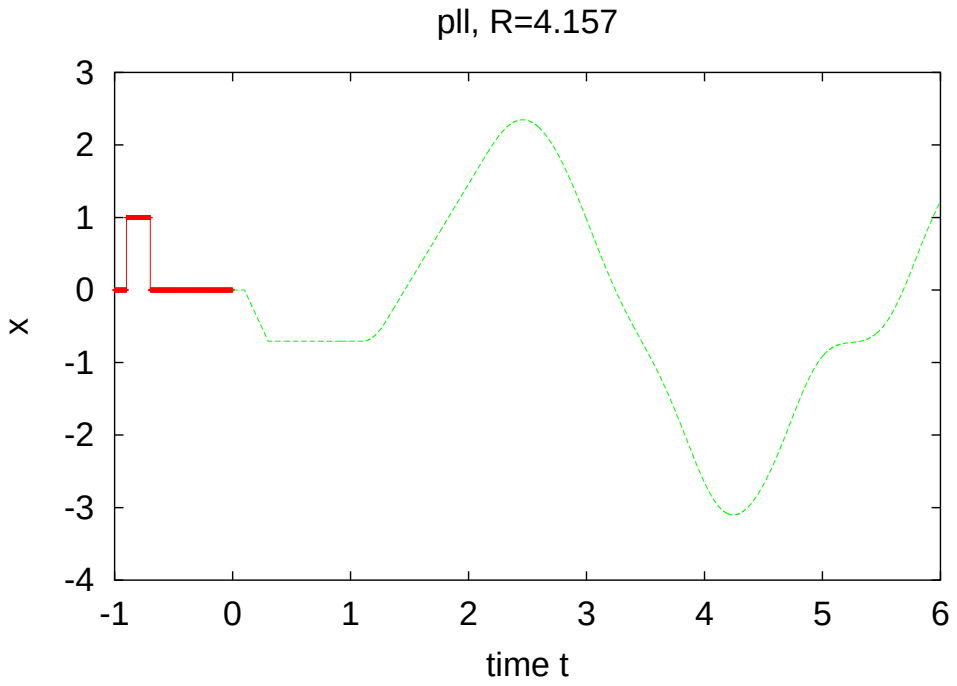

Figure 3.8: