Theoretical and Numerical Aspects of Delay Equations

Herbert Arndt

We discuss three aspects:

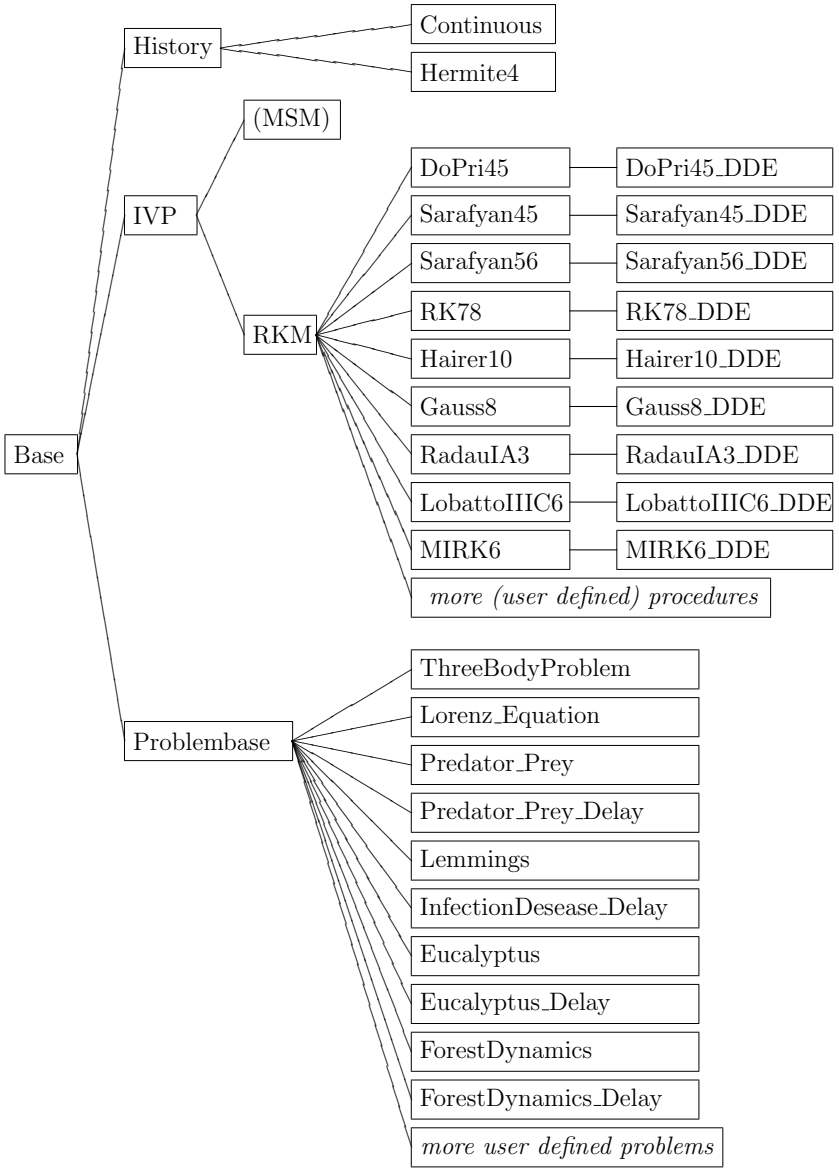
1. We consider the development of a delay equation solver using the object oriented language C++. We discuss the basic structure of the program, its main advantages over existing solvers (using a prey-predator equation) (this is mathematically more interesting and by far unsolved) and the benefits of using C++ or Java compared to a language for example like FORTRAN or C.

2. We consider a retarded predator-prey equation (with two delays) and study solution properties as a function of varying delay sizes.

3. We discuss existence theorems and local solutions to some types of initial value problems of delay equations. Surprisingly, an often cited existence theorem for neutral delay differential equations turns out to be not adequate for local solutions and must be substituted by a weaker version in Sobolev spaces.

C++ Program

Flowchart (basic structure of the C++-program dde)



Basic structure of the C++-program dde

Implemented Runge-Kutta Methods:

for nonstiff problems	for stiff problems
Euler (order 1)	EulerImp (order 1)
RK2 (order 2)	Gauss2,4,6,8 (orders 2,4,6,8)
RK4 (order 4)	RadauIA3,5 (orders 3,5)
DoPri45 (orders 4,5)	RadauIIA3,5 (orders 3,5)
Sarafyan45 (orders 4,5)	LobattoIIIC2,4,6 (orders 2,4,6)
Sarafyan56 (orders 5,6)	MIRK3,4,6 (orders 3,4,6)
RK78 (orders 7,8)	Rosenbrock methods
Hairer10 (order 10)	

Consider a simple example: Let $\tau = 1$.

$$\begin{aligned} y'(x) &= y(x-1)\ , \quad x \in [0, 20] \\ y(x) &= 1\ , \quad x \in [-1, 0] \end{aligned}$$

Implementation: (in problem.h)

```
class SimpleDelay : public ProblemBase
{
 public:
   SimpleDelay(void);
   ~SimpleDelay(void);
   int Exact(double *exact, double x); // exact solution
   int g(double * g, double x);        // initial function
   int tau(double *tau, double x);     // delays
   // Right hand side of differential equation
   int f( double * Dy, double x, double * y, double * z = 0);
   void SetInitialState();             // initial state
   void Config();                      // configuration
};
```

```
void SimpleDelay::Config() {
   // Initialisations
   // short problem description
   SetProblemDescription("Delay equation: y'(x)=y(x-1),
                          0<=x<=20; y(x)=1, -1<=x<=0");
   SetDimension(1);         // dimension of the system
   SetNumberofDelays(1);    // number of delays
   SetStartingPoint(0.0)    // initial point  x0
   SetEndPoint(20.0);       // endpoint  xend
   SetOutputFile("sd.txt");// file name of output file
   SetStepsize(0.1);        // initial step size
   SetFixedStep(0);         // 1: constant step size  0: not
```

```
SetAbsTolerance(1.0e-4);              // local error tolerance
SetRelTolerance(GetAbsTolerance()); // rel. error tolerance
// 1: print each step, 0: don't print each step
SetPrintAllSteps(1);
// 1: print rejected steps, 0: don't
SetPrintRejected(1);
```

```
   // SetProcedure("X","Y");
   // with method  X = Euler, RK2, RK4, RK78,
   //         DoPri45, Sarafyan45, Sarafyan56,
   //         EulerImp, Gauss2, Gauss4, Gauss6
   //         RadauIA3, RadauIA5,
   //         LobattoIIIC2, LobattoIIIC4, LobattoIIIC6,
   //         MIRK3, MIRK4, MIRK6
   // with history  Y = Hermite_4, Continuous
   // (for ordinary differential equations
   //      no history is needed)
   SetProcedure("Sarafyan56","Continuous");
}
```

```
// Retarded differential equations
// Dy = f(x,y,z), z = y(x-tau(x))
// are here written as  f(Dy,x,y,z),
// in this example  y'(x) = y(x-1) ( hence  f(x,y,z) = z )
int SimpleDelay::f( double * Dy, double x, double * y,
   double * z) {

   Dy[0] = z[0];  // z0 = y(x-tau1)

   return 0; // ok
}
```

```
// Initial function
int SimpleDelay::g(double * g, double x) {
   g[0] = 1.0;

   return 1; // ok
}

// Delays
int SimpleDelay::tau(double *tau, double x) {
   tau[0] = 1.0;

   return 1; // ok
}
```

```
// Initial value in  m_x
void SimpleDelay::SetInitialState() {
   // Use the value of  g  at  m_x  as initial value
   g(m_initState, m_x);
}
```

```
// Exact solution
int SimpleDelay::Exact(double *exact, double x) {
   int i, j=(int)floor(x-m_x0);
   double fac=1.0, s=1.0;

   if(j>=0)
      for(i=1; i<j+2; i++){
         fac *= (double)i;
         s += pow(x - m_x0 - (double)i + 1.0, (double)i)/fac;
      }
   exact[0] = s;

   return 1; // ok, Exact() exists
 }
```

Another example is the **Lorenz-equation** using OpenGl:

$$\begin{aligned} y_1' &= -s\,y_1 + s\,y_2 \\ y_2' &= -y_1\,y_3 + r\,y_1 - y_2 \\ y_3' &= y_1\,y_2 - b\,y_3 \end{aligned}$$

with $s = 10,\ \ r = 28,\ \ b = 8/3$.

Q.: What kind of problems can dde solve other than conventional programs?

A.: Solve a delay equation where the initial function g is the solution of an ordinary differential equation.

Initial value problem (predator-prey for example)
$y =$ prey, $z =$ predator (after Lotka-Volterra):

$$
\begin{aligned}
& \text{For } \ x \in [0, 100] \ \text{ solve} \\
y'(x) &= y(x)\big(\alpha - \beta z(x) - \lambda y(x - \tau_1)\big), \\
z'(x) &= -\gamma z(x) + z(x - \tau_2)\big(\delta y(x - \tau_2) - \mu z(x)\big) \\
& \text{with initial function} \\
y(x) &= g_1(x), \\
z(x) &= g_2(x) \qquad \big((g_1, g_2) \ \text{continuous}\big) \\
& \text{for } \ x \in [-L, 0] \ \text{ with } L := \max\{\tau_1, \tau_2\}\,.
\end{aligned}
$$

Where can we get the **initial function** from?

In all examples I know people take constant initial functions or artificial initial functions.

In any case we have the **initial value** at the initial point $x_0 = 0$.

The problem of finding a function $g \in C^0([-\tau, 0], \mathbb{R}^n)$ with $g(0) = g_0$ for given $g_0 \in \mathbb{R}^n$ has "infinitely many" solutions. We want to find a solution that has to do with the solution of the initial value problem.

Possible solutions:

1. We guess an initial function from previous experiments.

2. We take measurements for the last time unit of length L and find an appropriate initial function by interpolation (e.g. by splines).

3. We consider an ordinary differential equation that we get from the delay equation by setting $\tau_1 = \tau_2 = 0$. We get the following initial value problem (a predator-prey equation for ordinary differential equations)

$$\begin{aligned} y'(x) &= y(x)\big(\alpha - \beta z(x) - \lambda y(x)\big)\,, \\ z'(x) &= z(x)\big(-\gamma + \delta z(x) - \mu z(x)\big)\,, \\ y(0) &= g_1(0)\,, \\ z(0) &= g_2(0)\,, \end{aligned}$$

and solve it on the interval $[-L, 0]$ with a continuous Runge-Kutta method (to get a solution that is defined on the whole interval $[-L, 0]$). We use **this solution** as **initial function** for the retarded equation.

Assume that the ordinary and retarded problems are called

`PredatorPrey`

`PredatorPrey_DDE`

respectively and are configured properly. Then the problems are solved by this

```
PredatorPrey pp; // construction of the object
pp.Solve();

PredatorPrey_DDE ppd(&pp); // hand over the initial fcn
ppd.Solve();
```

The just presented solution is not the best possible. For large $L = \max\{\tau_1, \tau_2\}$ or when the right side f depends heavily on τ_1 and τ_2 the results are not always as expected. But note: The above mentioned predator-prey initial value problem is most often not the problem one wants to solve. When we observe predator and prey in nature that live there since 100 years with $L = 1$ month, then the solution is at least 1200 times differentiable at $x = 0$ (assuming g is smooth enough) whereas the solution of the initial value is in general only continuous in $x = 0$. Therefore one could impose additional smoothness properties in $x = 0$.

Interesting Numerical Experiments with a Retarded Predator-Prey Equation

Consider the retarded predator-prey equation
y = prey, z = predator (after Lotka-Volterra):

$$\begin{aligned} y'(x) &= y(x)\big(\alpha - \beta z(x) - \lambda y(x-\tau_1)\big), \\ z'(x) &= -\gamma z(x) + z(x-\tau_2)\big(\delta y(x-\tau_2) - \mu z(x)\big)\,. \end{aligned}$$

We want to study properties of the solution when we
vary the delays τ_1 and τ_2.

Using parameters

$$\begin{aligned} \alpha &= 0.08\,, & \beta &= 0.002\,, & \gamma &= 0.2\,, \\ \delta &= 0.0004\,, & \lambda &= 0.00004\,, & \mu &= 0.0005\,, \end{aligned}$$

we get the following diagram showing

$$\begin{aligned} G_i &: \text{equilibrium point } (536.6\,,\, 29.3) \\ P_i &: \text{"simple" periodic solutions} \\ MP_i &: \text{"multiple" periodic solutions} \\ C_i &: \text{chaotic solutions} \\ A &: \text{dying down of the predators} \end{aligned}$$

(The following pictures were produced by my student *Christine Dreesen* using the program dde.)

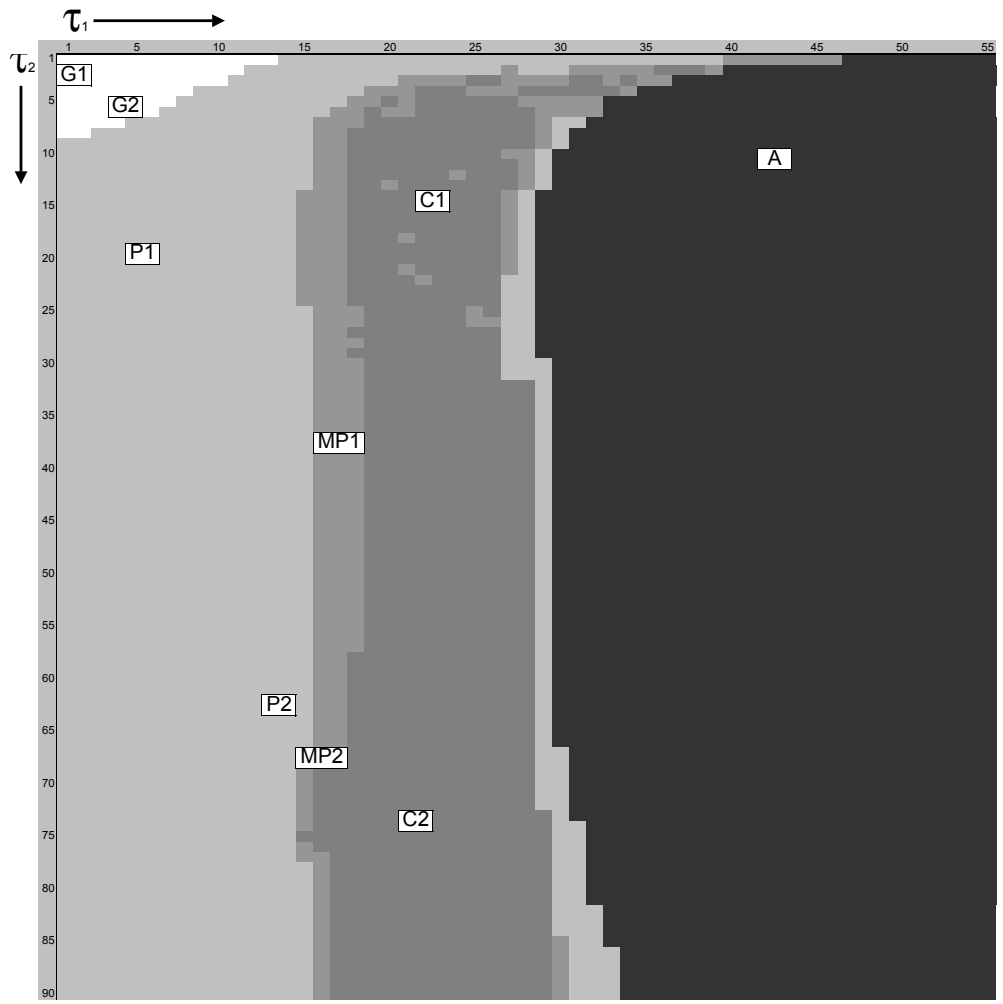


Figure 1: Solution kinds of the predator-prey equation with two delays

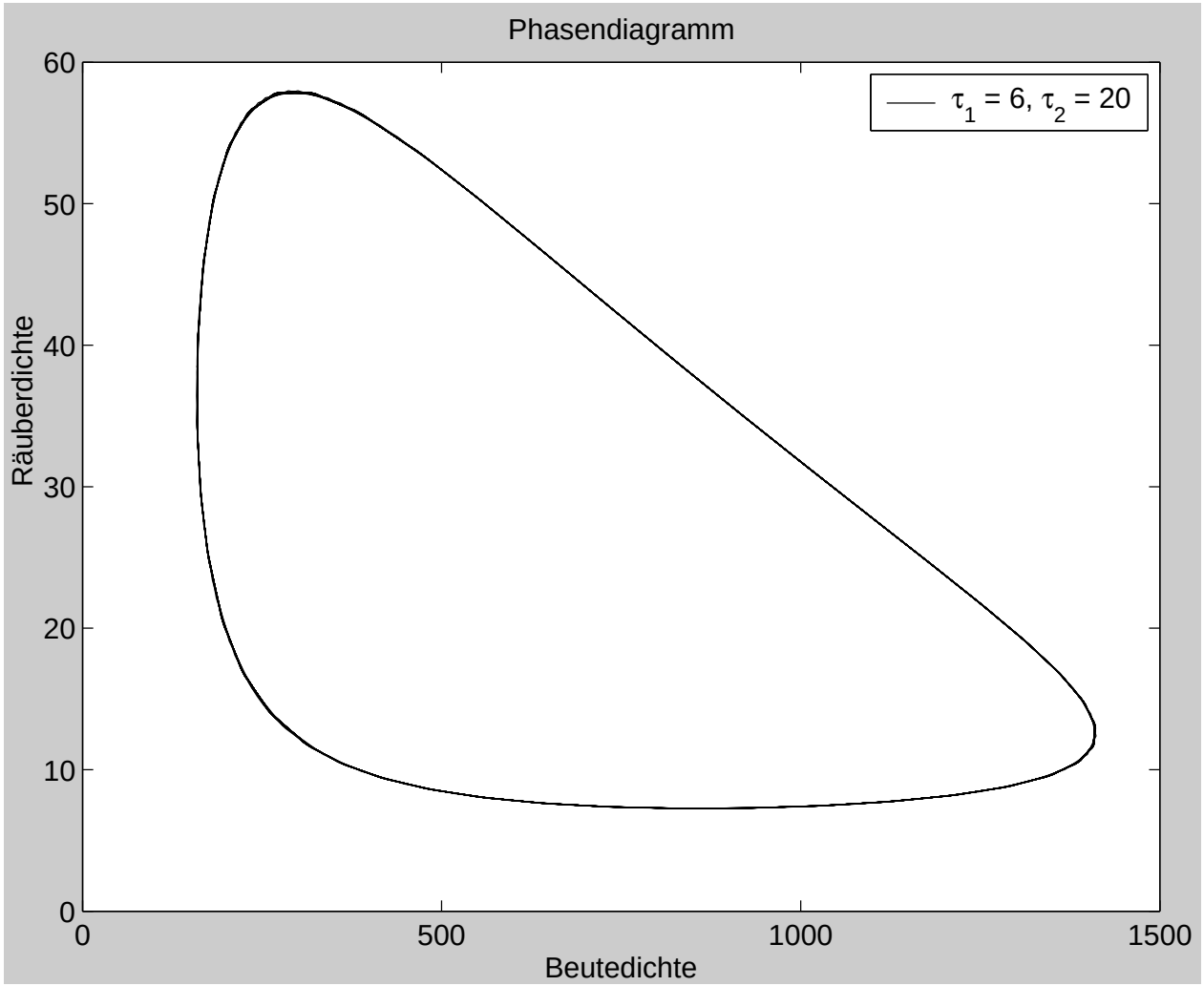


Figure 2: "Simple" periodic solution

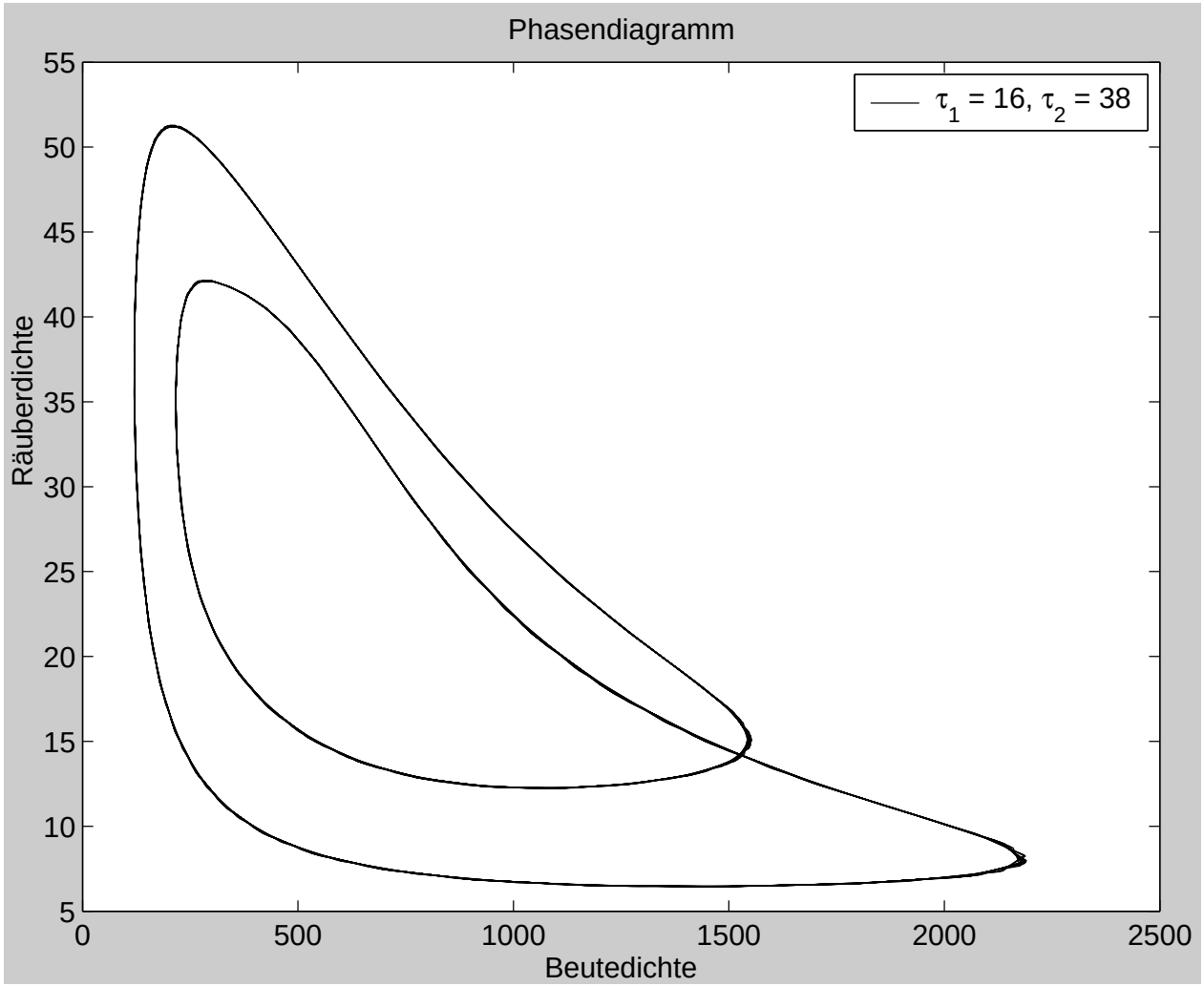


Figure 3: "Multiple" periodic solution

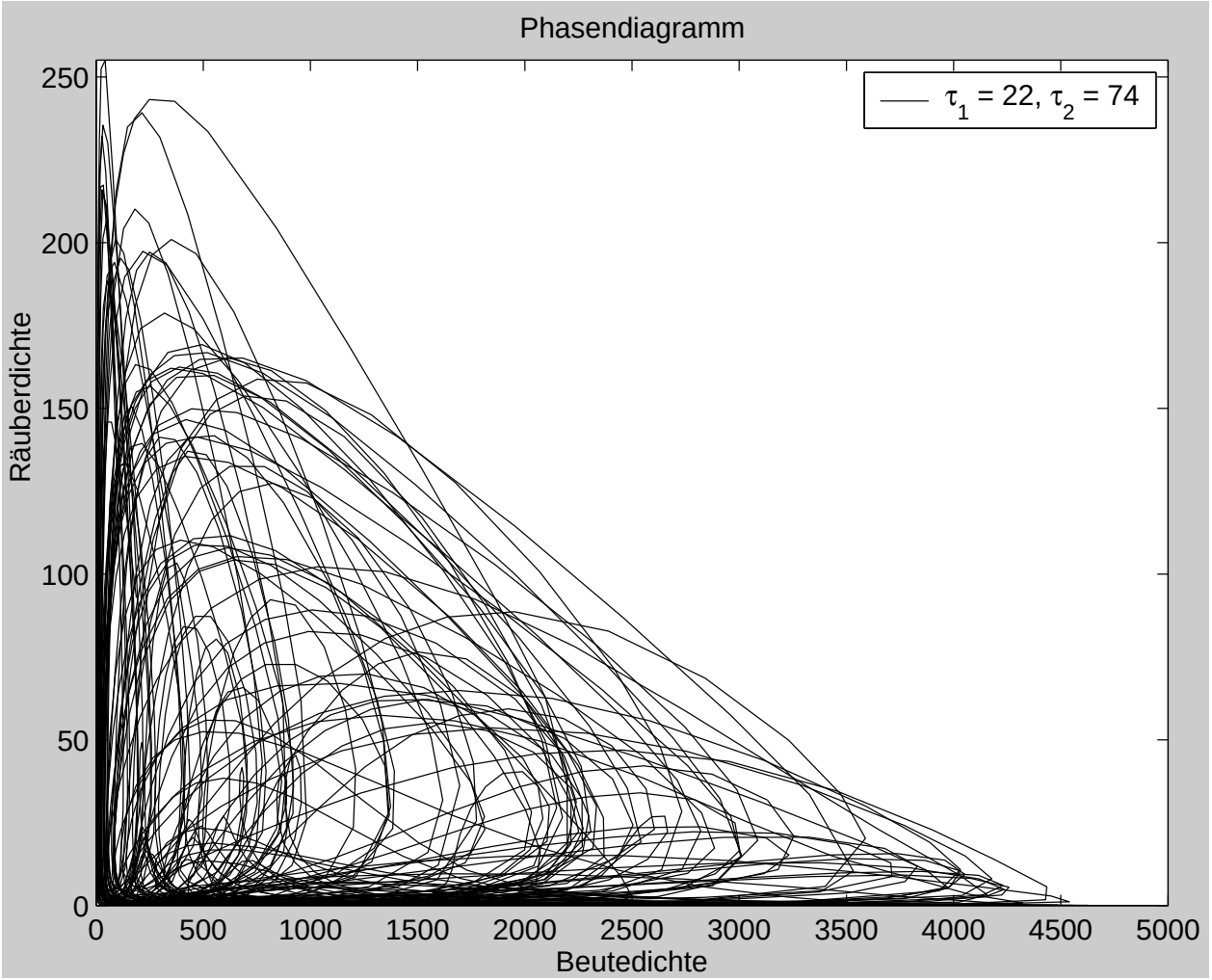


Figure 4: "Chaotic" solution

Existence Theorems in ordinary and retarded differential equations

We consider **three different types** of initial value problems:

1. Ordinary differential equations

$$\begin{aligned} y'(x) &= f(x, y(x)) \ , \quad x \in (a, b) \\ y(x_0) &= y_0 \end{aligned}$$

2. Delay differential equations

Example: Let $\tau > 0$

$$\begin{aligned} y'(x) &= y(x)y(x-\tau)\,, \quad x \geq 0 \\ y(x) &= 1\,, \quad x \in [-\tau, 0] \end{aligned}$$

or

$$\begin{aligned} y'(x) &= y(x)\int_0^x y(s)^2\,ds\,, \quad x \geq 0 \\ y(x) &= 1\,, \quad x \leq 0 \end{aligned}$$

3. Neutral delay differential equations

Example: Let $\tau > 0$

$$\begin{aligned} y'(x) &= y(x)y(x-\tau) + y'(x-\tau)\ , \quad x \geq 0 \\ y(x) &= x\ , \quad x \in [-\tau, 0] \end{aligned}$$

In the sequel we are interested in **local solutions** to these problems which naturally occur for instance during numerical computations.

Ordinary differential equations

Let $f \in C^{0,1-}((a,b) \times \mathbb{R}^n, \mathbb{R}^n), (x_0, y_0) \in (a,b) \times \mathbb{R}^n$.

Initial value problem: We seek
$y \in C^1((c,d), \mathbb{R}^n)$, $x_0 \in (c,d) \subset (a,b)$, with

$$\begin{aligned} y'(x) &= f(x, y(x)) \,, \quad x \in (a,b) \\ y(x_0) &= y_0 \end{aligned}$$

Theorem: There exists a uniquely determined solution.

Local solution $y(\cdot, \xi, \eta)$:

$$\begin{aligned} y'(x) &= f(x, y(x)) \,, \quad x \in (a, b) \\ y(\xi) &= \eta \end{aligned}$$

In numerical analysis:

$$\begin{aligned} \xi &= x_j, && x_j \text{ a meshpoint,} \\ \eta &= y^h(x_j), && y^h \text{ the approximate solution.} \end{aligned}$$

Both problems (original initial value problem and local solution problem) are essentially the same!

You can apply the same existence theorem.

Delay differential equations

Let $\alpha < a < b$, $f \in C^{0,1-}([a,b] \times C^0([a,b], \mathbb{R}^n)$,
$g \in C^0([\alpha, a], \mathbb{R}^n)$ (initial function). $(x_0 = a)$

Initial value problem: We seek
$d \in (a,b]$, $y \in C^0([\alpha, d], \mathbb{R}^n)$ with $y \in C^1([a,d], \mathbb{R}^n)$, and with

$$\begin{aligned} y'(x) &= f(x, y_{[\alpha,x]}) \,, \quad x \in [a,d] \\ y_{[\alpha,a]} &= g \end{aligned}$$

Theorem: (E.g. Hale 1971, Driver 1977) There exists a unique solution to this initial value problem.

Local solution $y(\cdot, \xi, \eta_{[\alpha,\xi]})$ for $\xi \in (a, b)$,
initial function $\eta \in C^0([\alpha, \xi], \mathbb{R}^n)$:

$$\begin{aligned} y'(x) &= f(x, y_{[\alpha,x]}) \,, \quad x \in [\xi, b] \\ y_{[\alpha,\xi]} &= \eta_{[\alpha,\xi]} \end{aligned}$$

In numerical analysis:

$$\begin{aligned} \xi &= x_j, \qquad x_j \text{ a meshpoint,} \\ \eta_{[\alpha,\xi]} &= y^h_{[\alpha,\xi]}, \qquad y^h \text{ the approximate solution.} \end{aligned}$$

Because η is assumed to be continuous and y^h is continuous, both problems are essentially the same!
You can apply the same existence theorem.

Neutral delay differential equations

Let $\alpha < a < b$,
$f : [a,b] \times C^1([\alpha,b],\mathbb{R}^n) \times C^0([\alpha,b],\mathbb{R}^n) \longrightarrow \mathbb{R}^n$,
$g \in C^1([\alpha,a],\mathbb{R}^n)$ (initial function).

Initial value problem: We seek
$d \in (a,b]$ and $y \in C^1([\alpha,d],\mathbb{R}^n)$ with

$$\begin{aligned} y'(x) &= f(x, y_{[\alpha,x]}, y'_{[\alpha,x]}) \,, \quad x \in [a,d] \\ y_{[\alpha,a]} &= g \end{aligned}$$

Theorem: (Kamont, Kwapisz 1976) Under the assumptions

1. the mapping

$$x \longrightarrow f(x, y_{[\alpha,x]}, y'_{[\alpha,x]})$$

is continuous on $[a,b]$ for any $y \in C^1([\alpha,b],\mathbb{R}^n)$,

2. we have the global Lipschitz condition

$$\begin{aligned} &|f(x, y^1_{[\alpha,x]}, z^1_{[\alpha,x]}) - f(x, y^2_{[\alpha,x]}, z^2_{[\alpha,x]})| \\ &\quad \leq \ L_1\big(\|y^1 - y^2\|^{\infty}_{[\alpha,x]} + \|z^1 - z^2\|^{\infty}_{[\alpha,x-\delta]}\big) + L_2\|z^1 - z^2\|^{\infty}_{[\alpha,x]} \end{aligned}$$

with $L_1 \geq 0$, $0 \leq L_2 < 1$, $\delta > 0$, $x \in [a,b]$,
$y^1, y^2 \in C^1([\alpha,b],\mathbb{R}^n)$, $z^1, z^2 \in C([\alpha,b],\mathbb{R}^n)$,

there exists a uniquely determined solution to the initial value problem.

Local solution $y(\cdot, \xi, \eta_{[\alpha,\xi]})$ for $\xi \in (a, b)$,
$\eta \in C^1([\alpha, \xi], \mathbb{R}^n)$:

$$\begin{aligned} y'(x) &= f(x, y_{[\alpha,x]}, y'_{[\alpha,x]}) \ , \quad x \in [\xi, b] \\ y_{[\alpha,\xi]} &= \eta_{[\alpha,\xi]} \end{aligned}$$

In numerical analysis:

$$\begin{aligned} \xi &= x_j, \qquad x_j \text{ a meshpoint,} \\ \eta_{[\alpha,\xi]} &= y^h_{[\alpha,\xi]}, \qquad y^h \text{ the approximate solution.} \end{aligned}$$

Notice: Because we seek $y \in C^1([\alpha, d], \mathbb{R}^n)$ there is the following **compatibility condition** implicitly included

$$g'(a-) \;=\; f(a, y_{[\alpha,a]}, y'_{[\alpha,a]})$$

This time both problems are in general **not** the same!

Main differences:

1. The approximate solution y^h is in general **not continuously differentiable** on $[\alpha, x_j]$ for all numerical methods.
2. **The compatibility condition is in general not fulfilled.**

Consequently the **existence** of an exact solution and the **existence** of a numerical solution for the local problem with initial function y^h is **not** guaranteed.

Consider the Sobolev space $H^{1,\infty} \subset L^\infty$ of weakly differentiable functions whose derivatives are in L^∞. We have the following theorem.

Theorem: If

1. $g \in H^{1,\infty}([\alpha, a], \mathbb{R}^n)$
2. $f : [a, b] \times H^{1,\infty}([\alpha, b], \mathbb{R}^n) \times L^\infty([\alpha, b], \mathbb{R}^n) \longrightarrow \mathbb{R}^n$
3. The mapping $x \longrightarrow f(x, y, y')$ is in $L^\infty([\alpha, b], \mathbb{R}^n)$ for all $y \in H^{1,\infty}([\alpha, b], \mathbb{R}^n)$ and all $x \in [a, b]$
4. We have

$$\begin{aligned} &|f(x, y^1, z^1) - f(x, y^2, z^2)| \\ &\quad \leq \; L_1\big(\|y^1 - y^2\|^\infty_{[\alpha,x]} + \|z^1 - z^2\|^{L^\infty}_{[\alpha,x-\delta]}\big) + L_2\|z^1 - z^2\|^{L^\infty}_{[\alpha,x]} \end{aligned}$$

with $L_1 \geq 0$, $0 \leq L_2 < 1$, $\delta > 0$, $x \in [a, b]$,
$y^1, y^2 \in H^{1,\infty}([\alpha, b], \mathbb{R}^n)$, $z^1, z^2 \in L^\infty([\alpha, b], \mathbb{R}^n)$

then there exists a unique solution $y \in H^{1,\infty}([\alpha, b], \mathbb{R}^n)$ to the initial value problem for neutral delay equations.

Among others the compatibility condition must **not** be fulfilled.

Consequently because of

$$H^{1,\infty}([\alpha, b], \mathbb{R}^n) \subset C^0([\alpha, b], \mathbb{R}^n)\,,$$

all local problems have **both** an exact and a numerical solution.