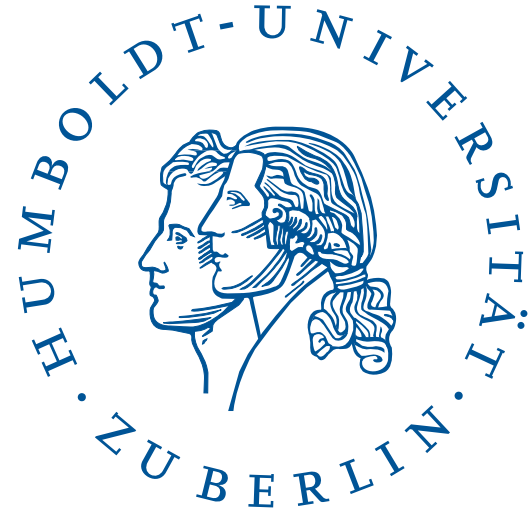
Solution of Differential Equations

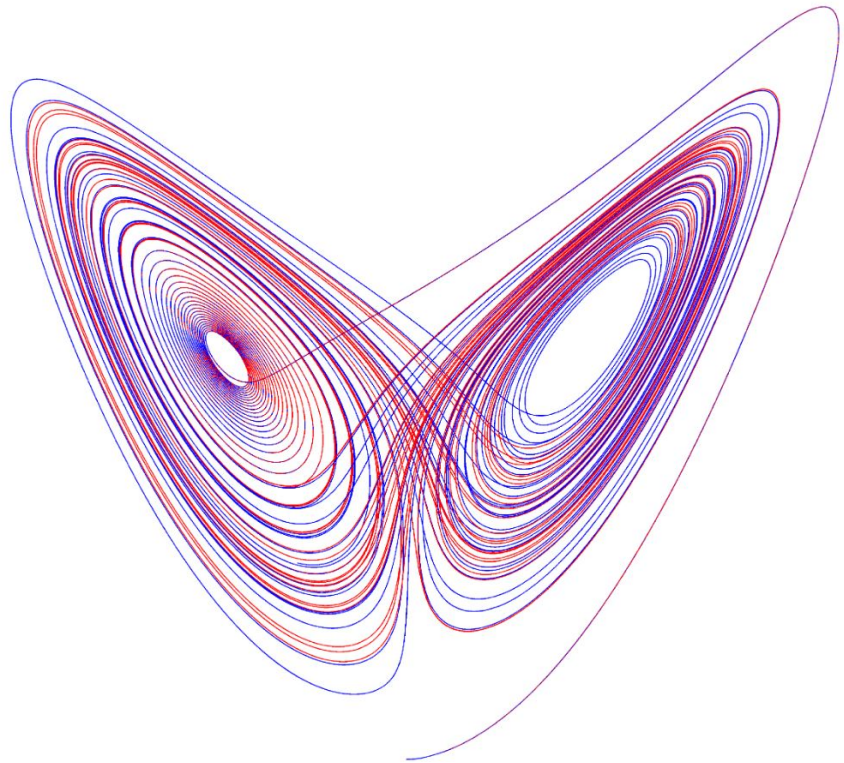
Laureen Lake

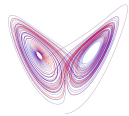
Numerical Introductory Seminar
Humboldt–Universität zu Berlin



Agenda

- Definitions
- Motivation
- Numerical Methods
  - one-step methods
  - multi-step methods
- Application





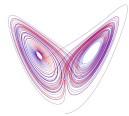
Definitions

Differential Equation (DE): Mathematical equation that relates a specific function with its derivatives

$$\frac{df(t)}{dt} = a \cdot f(t)$$

- **Ordinary Differential Equation** (ODE): DE with a **single** independent variable, e.g. t (time)
- **Partial Differential Equation** (PDE): DE that contains unknown multivariable functions and their partial derivatives, i.e **two or more** independent variables

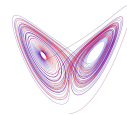
$$\frac{\partial f(x,t)}{\partial t} = c \cdot \frac{\partial^2 f(x,t)}{\partial x^2}$$



Definitions

- **Stochastic Differential Equation** (SDE): DE where one or more of the terms is a **stochastic process**, resulting in a solution, which is itself a stochastic process

$$dX_t = \mu X_t \, dt + \sigma X_t dB_t$$



Example: Continuous Compounding

When the interest of a loan is compounded continuously, then at any time the interest gets added in proportion to the current value of the loan (or investment).

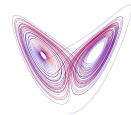
The bigger the loan the more interest it earns.

Using t for time, r for the interest rate and V(t) for the value of the loan at time t, we get:

$$\frac{dV(t)}{dt} = r \cdot V(t) \tag{1}$$

Solution:

$$V(t) = V(0) \cdot e^{rt} \tag{2}$$

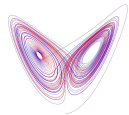


Motivation

- many differential equations arising in applications are so complicated that it is impractical to have solution formulas
- Even available solution formula may involve integrals that can be calculated only by using a numerical quadrature formula

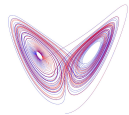
  Powerful alternative tool for solving the differential equation: Numerical methods

  - able to approximate the solution of a differential equation to any desired accuracy



Numerical Methods

- Several different methods for different types of Differential Equations
- One differentiates between
  - One-step methods vs. multi-step methods
  - Explicit methods vs. implicit methods
- Methods for Initial Value Problems (IVP) and Boundary Value Problems (BVP)



Euler Method

Consider the Initial Value Problem (IVP):

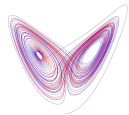
$$\frac{dY(t)}{dt} = f(t, Y(t))$$

$$Y(t_0) = Y_0$$

- find approximate solution at discrete set of nodes:

$$t_0 < t_1 < \ldots < t_N$$

where $t_k = t_0 + k \cdot h$ (equally spaced nodes)



Euler Method

- Consider standard derivative approximation:

$$\frac{dY(t)}{dt} \approx \frac{1}{h}[Y(t+h) - Y(t)]$$

and apply this to the IWP at t_n:

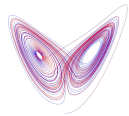
$$\frac{1}{h}[Y(t_{n+1}) - Y(t_n)] \approx f(t_n, Y(t_n))$$

$$\iff Y(t_{n+1}) \approx Y(t_n) + h \cdot f(t_n, Y(t_n))$$

  - **Euler's method**:

$$y_{n+1} = y_n + h \cdot f(t_n, y_n) \tag{3}$$

where $y_0 = Y_0$.



Backward Euler Method (implicit Euler)

- Consider *backward difference approximation*:

$$\frac{dY(t)}{dt} \approx \frac{1}{h}[Y(t) - Y(t-h)]$$

and apply this to the IWP at t_n:

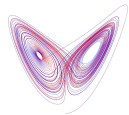
$$\frac{1}{h}[Y(t_n) - Y(t_{n-1})] \approx f(t_n, Y(t_n))$$

$$\iff \quad Y(t_n) \approx Y(t_{n-1}) + h \cdot f(t_n, Y(t_n))$$

  - **Backward Euler method**:

$$y_{n+1} = y_n + h \cdot f(t_{n+1}, y_{n+1}) \quad , \; 0 \leq n \leq N-1 \tag{4}$$

where $y_0 = Y_0$.



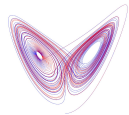
Backward Euler Method (implicit Euler)

- Main difference to Euler Method: nonlinear algebraic equation needs to be solved for y_{n+1} at each timestep
- If h is small enough, (4) has a unique solution
- Traditonal rootfinding methods are time-consuming, e.g. Newton Raphson
  - **Instead**: Use of simple iteration technique

$$y_{n+1}^{j+1} = y_n + h \cdot f(t_{n+1}, y_{n+1}^{j})$$

with initial guess $y_{n+1}^{0} \approx y_{n+1}$

$$\Longrightarrow \; y_{n+1} = y_n + h \cdot f(t_{n+1}, \underbrace{y_n + h \cdot f(t_{n+1}, y_n)}_{\textit{explicitEuler}})$$



Trapezoidal Method

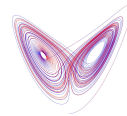
- absolutely stable and of higher order accuracy
- Use of trapezoidal rule for numerical integration:

$$\int_a^b g(s)ds \approx \frac{1}{2}(b-a)[g(a)+g(b)] \tag{5}$$

- Integrate DE $Y'(t) = f(t, Y(t))$ from t_n to t_{n+1} and use (5) to approximate the integral

$$y_{n+1} = y_n + \frac{1}{2}h \cdot [f(t_n, y_n) + f(t_{n+1}, y_{n+1})] \tag{6}$$

with $y_0 = Y_0$



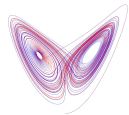
Runge-Kutta Methods

- Desirable: Higher order methods achieving higher accuracy with less computational effort
  - Solution: explicit **Runge-Kutta Methods**

$$y_{n+1} = y_n + h \cdot (b_1 K_1 + b_2 K_2 + ... + b_s K_s)$$

$$\begin{aligned} K_1 &= f(t_n, y_n) \\ K_2 &= f(t_n + c_2 \cdot h, y_n + h \cdot a_{2,1} K_1) \\ &... \\ K_s &= f(t_n + c_s \cdot h, y_n + h \cdot (a_{s,1} K_1 + ... + a_{s,s-1} K_{s-1})) \end{aligned}$$

where s denotes the number of stages. The coefficients $c_i, a_{i,j}, b_j$ are given and define the numerical method.

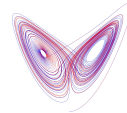


Runge-Kutta Methods

- Coefficients of Runge-Kutta methods are often displayed in so called Butcher Tableaus:

$$\begin{array}{r|cccccc} 0 = c_1 & & & & & & \\ c_2 & a_{2,1} & & & & & \\ c_3 & a_{3,1} & a_{3,2} & & & & \\ \vdots & \vdots & & \ddots & & & \\ c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s-1} & & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array}$$

- The coefficients satisfy the following conditions:
  - $a_{i,j} = 0$ for $j \geq i$
  - $\sum_{j=1}^{i-1} a_{i,j} = c_i$ for $i = 2, ..., s$



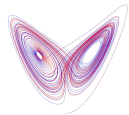
Runge-Kutta Methods

- Classical Runge-Kutta method of order 4

$$y_{n+1} = y_n + \frac{1}{6} h \cdot (K_1 + 2K_2 + 2K_3 + K_4)$$

$$\begin{aligned} K_1 &= f(t_n, y_n) \\ K_2 &= f(t_n + \frac{1}{2} \cdot h, y_n + \frac{1}{2} h \cdot K_1) \\ K_3 &= f(t_n + \frac{1}{2} \cdot h, y_n + \frac{1}{2} h \cdot K_2) \\ K_4 &= f(\underbrace{t_n + h}_{t_{n+1}}, y_n + h \cdot K_3) \end{aligned}$$

$\Longrightarrow$ extended Simpson quadrature rule from integrals to differential equations

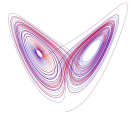


Embedded Runge-Kutta methods

- Constant step size h can be inefficient. Better, if
  - large variations of solution $\Rightarrow$ small step sizes
  - slowly varying solution $\Rightarrow$ large step sizes

Embedded Runge-Kutta Methods:

- Two Runge-Kutta methods of different orders
- Methods share most of function evaluations at each step from t_n to t_{n+1}.
- Adaptive selection of the step size keeps local error close to a given error tolerance in each time step



Embedded Runge-Kutta methods

Methods:

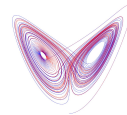
$$y_{n+1} = y_n + h \cdot \sum_{i=1}^{s} b_i K_i$$

$$\hat{y}_{n+1} = y_n + h \cdot \sum_{i=1}^{\hat{s}} \hat{b}_i K_i$$

The local error in the lower-order formula is estimated by:

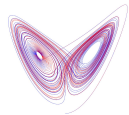
$$LE_{n+1} \approx \hat{y}_{n+1} - y_{n+1} \tag{7}$$

- Methods use common indermediate slopes $K_1 - K_s$ for $s < \hat{s}$ $\Longrightarrow$ in each step: evaluate only $\hat{s} + 1$ slopes
- Error controlled on lower order solution, but higher order solution is numerical solution used



Multistep Methods

- using information of several previously computed solution approximations
- more efficient than RK methods, especially if
  - we want to find the solution with a high degree of accuracy
  - the derivative function $f(t, y)$ is expensive to evaluate
- Methods:
  - Adam-Bashforth
  - Adam-Moulton
  - BDF
  - ...

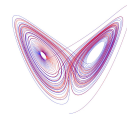


Multistep Methods - Adam Methods

- Integrate the initial IVP $Y'(t) = f(t, Y(t))$ from t_{n-1} to t_n:

$$Y(t_{n+1}) - Y(t_n) = \int_{t_n}^{t_{n+1}} f(s, Y(s))ds$$

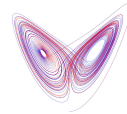
- *Adam-Bashforth*: Replace $f(t, Y(t))$ by a **polynomial of degree k-1** that interpolates f at time points t_{n-j} for $j = 0, 1, ..., k-1$.
- *Adam-Moulton*: Replace $f(t, Y(t))$ by a **polynomial of degree k** that interpolates f at time points $t_{n+1}, t_n, ..., t_{n-k+1}$.
- Integrate the interpolating poynomial



Multistep Methods - Adam Methods

- Adam-Bashforth method: Order k
- Adam-Moulton method: Order $k + 1$

- In contrast to Runge-Kutta methods, this integrator requires only one function evaluation per step
- **BUT**: Smaller time steps are necessary to achieve a comparable accuracy



Application

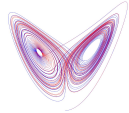
- Image Impainting
- **Calculation of satellite orbits**

Differential Equation for the orbit of a Satellite around earth an moon:

$$dx = \begin{pmatrix} x_3 \\ x_4 \\ x_1 + 2 * x_4 - (1-\mu) * \frac{(x_1+\mu)}{((x_1+\mu)^2+x_2^2)^{3/2}} - \mu * \frac{(x_1-1+\mu)}{((x_1-1+\mu)^2+x_2^2)^{3/2}} \\ x_2 - 2 * x_3 - (1-\mu) * \frac{x_2}{((x_1+\mu)^2+x_2^2)^{3/2}} - \mu * \frac{x_2}{((x_1-1+\mu)^2+x_2^2)^{3/)}} \end{pmatrix}$$

where $\mu = \frac{1}{82,45}$ denotes the relative lunar mass.

- High accuracy is extremely important $\Longrightarrow$ Which method is most suitable?



Application

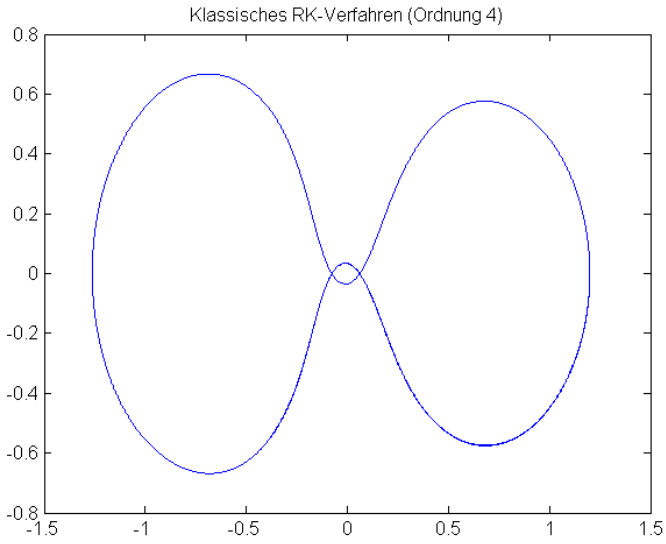


Figure 1: classical RK Method (order 4)

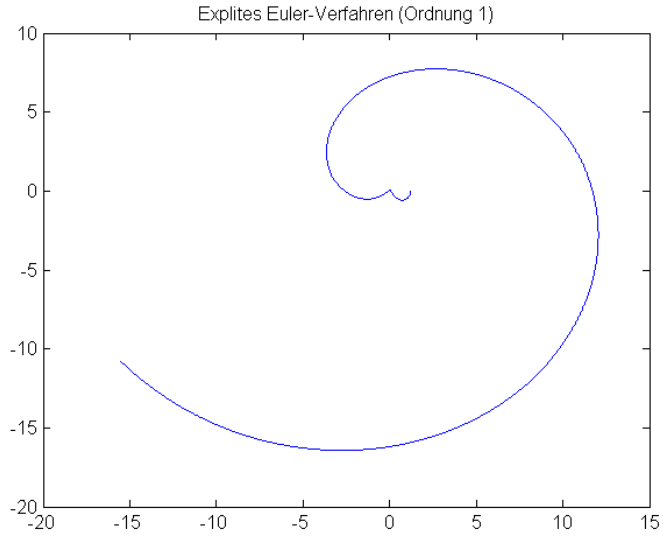
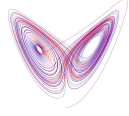
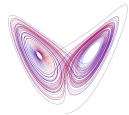


Figure 2: explicit Euler

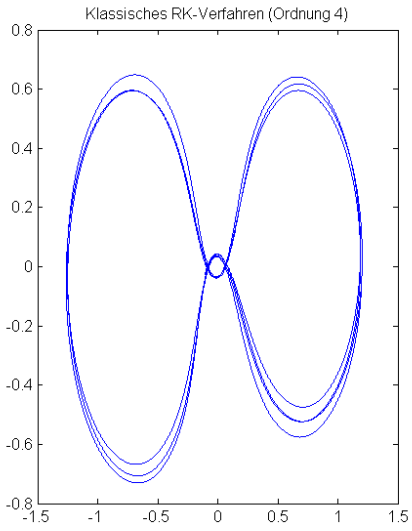


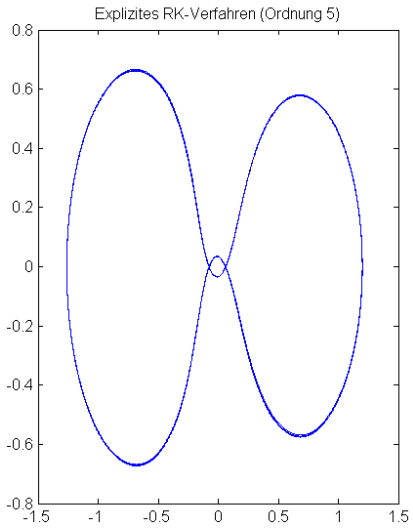
Application

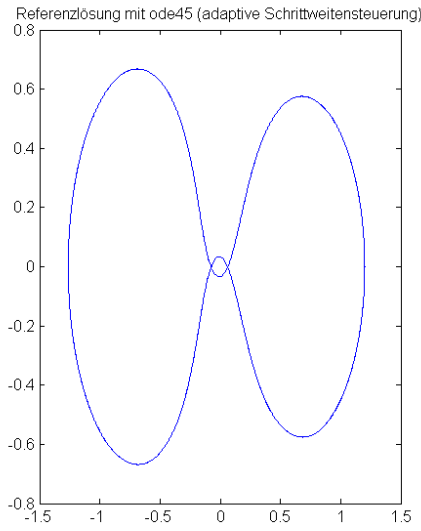
- The explicit Euler method (Fig.2) is not suitable, as the global error $e_n(h) = \| y(t_n) - y_n \|$ grows quickly $\Longrightarrow$ Order 1 of the explicit Euler-Method is not sufficient
- decreasing the step size, the explicit Euler would compute with better accuracy and the approximate solution would be better
  - **BUT**: Computational costs would strongly increase $\Longrightarrow$ Computation of the solution is not efficient anymore

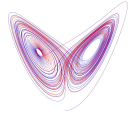


Application



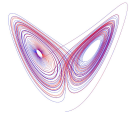






Application

- Stepsize twice as large as in the example before
- Classical RK of order 4 deviates from the exact solution at some time points t_n more than actually desired $\Longrightarrow$ accuracy is too low
- Instead use RK method of order 5: Approximate solution is closer to the reference solution calculated by the ode45-Solver of Matlab



Bibliography

Atkinson, K, Han, W. & Stewart, D., (2009). Numerical Solutions of Ordinary Differential Equations. New Jersey: John Wiley & Sons.

Hairer, E. & Lubich, C., (2011). Numerical analysis of ordinary differential equations. Univerisité de Genève and Universität Tübingen.