Scientific Computing I

Module 4: Numerical Methods for ODE

Michael Bader
Lehrstuhl Informatik V

Winter 2016/2017
Part I: Basic Numerical Methods

Motivation: Direction Fields

Euler’s Method

Discretized Model vs. Discrete Model

Implicit Euler

Analysis of Numerical Schemes for ODE
  Local Discretisation Error
  Global Discretisation Error
  Order of Consistency/Convergence
Motivation: Direction Fields

- given: initial value problem:

\[ \dot{y}(t) = f(t, y(t)), \quad y(t_0) = y_0 \]

- easily computable: direction field

- idea: “follow the arrows”
“Following the Arrows”

- direction field illustrates slope for given time $t_n$ and value $y_n$:
  \[ \dot{y}_n = f(t_n, y_n) \]

- “follow arrows” = make a small step in the given direction:
  \[ y_{n+1} := y_n + \tau \dot{y}_n = y_n + \tau f(t_n, y_n) \]

- motivates numerical scheme:
  \[
  \begin{align*}
  y_0 & := y_0 \\
  y_{n+1} & := y_n + \tau f(t_n, y_n) \quad \text{for} \ n = 0, 1, 2, \ldots
  \end{align*}
  \]
Euler’s Method

- numerical scheme is called **Euler’s method**:
  \[ y_{n+1} := y_n + \tau f(t_n, y_n) \]
- results from **finite difference** approximation:
  \[ \frac{y_{n+1} - y_n}{\tau} \approx \dot{y}_n = f(t_n, y_n) \]
  (difference quotient instead of derivative)
- or from truncation of Taylor expansion:
  \[ y(t_{n+1}) = y(t_n) + \tau \dot{y}(t_n) + \mathcal{O}(\tau^2) \]
Euler’s Method and Direction Fields

use direction at the beginning of the timestep
Euler’s Method – 1D examples

- model of Malthus, $\dot{p}(t) = \alpha p(t)$:
  
  $$p_{n+1} := p_n + \tau \alpha p_n$$

- Logistic Growth, $\dot{p}(t) = \alpha (1 - p(t)/\beta) p(t)$:
  
  $$p_{n+1} := p_n + \tau \alpha \left( 1 - \frac{p_n}{\beta} \right) p_n$$

- Logistic growth with threshold:
  
  $$p_{n+1} := p_n + \tau \alpha \left( 1 - \frac{p_n}{\beta} \right) \left( 1 - \frac{p_n}{\delta} \right) p_n$$
Euler’s Method in 2D

- Euler’s method is easily extended to systems of ODE
  → use vector notation:

  \[ \mathbf{y}_{n+1} := \mathbf{y}_n + \tau \mathbf{f}(t_n, \mathbf{y}_n) \]

- example: nonlinear extinction model

  \[
  \begin{align*}
  \dot{p}(t) &= \left( \frac{71}{8} - \frac{23}{12} p(t) - \frac{25}{12} q(t) \right) p(t) \\
  \dot{q}(t) &= \left( \frac{73}{8} - \frac{25}{12} p(t) - \frac{23}{12} q(t) \right) q(t)
  \end{align*}
  \]

- Euler’s method:

  \[
  \begin{align*}
  p_{n+1} &= p_n + \tau \left( \frac{71}{8} - \frac{23}{12} p_n - \frac{25}{12} q_n \right) p_n \\
  q_{n+1} &= q_n + \tau \left( \frac{73}{8} - \frac{25}{12} p_n - \frac{23}{12} q_n \right) q_n
  \end{align*}
  \]
Discretized Model vs. Discrete Model

- example: model of Malthus/radioactive decay ($\dot{p} = -\alpha p$):
  \[ p_{n+1} := p_n - \tau \alpha p_n, \quad \alpha > 0 \]

- compare to discrete model:
  \[ p_{n+1} := p_n - \delta p_n, \quad \delta > 0 \]

  with decay rate $\delta$ ("percentage")
  - obvious restriction in the discrete model: $\delta < 1$
  - obvious restriction for $\tau$ in the discretized model?
    \[ \tau \alpha < 1 \Rightarrow \tau < \alpha^{-1} \]

  - not that simple in non-linear models or systems of ODE!
Implicit Euler

- Euler’s method (“explicit Euler”):

\[ y_{n+1} := y_n + \tau f(t_n, y_n) \]

- implicit Euler:

\[ y_{n+1} := y_n + \tau f(t_{n+1}, y_{n+1}) \]

- results from finite difference approximation:

\[ \frac{y_{n+1} - y_n}{\tau} \approx \dot{y}_{n+1} = f(t_{n+1}, y_{n+1}) \]

- explicit formula for \( y_{n+1} \) not immediately available

- to do: solve equation for \( y_{n+1} \)
Implicit Euler and Direction Fields

use direction at **end** of the timestep
Implicit Euler – Examples

• example: Model of Malthus/radioactive decay

\[ p_{n+1} := p_n - \tau \alpha p_{n+1} \Rightarrow p_{n+1} = \frac{1}{1 + \tau \alpha} p_n \]

• correct (discrete) model?

\[ \alpha > 0 : \text{ then } 0 < (1 + \tau \alpha)^{-1} < 1 \text{ for any } \tau \]
\[ \alpha < 0 : \text{ then } \tau < -\alpha^{-1} \text{ required!} \]

• in physics \( \alpha > 0 \) is more frequent!
  (damped systems, friction, \ldots)

• implicit schemes often preferred/necessary when explicit schemes require very small \( \tau \)
Implicit Euler – 2D Example

• example: arms race

\[ p_{n+1} = p_n + \tau \left( b_1 + a_{11} p_{n+1} + a_{12} q_{n+1} \right) \]
\[ q_{n+1} = q_n + \tau \left( b_2 + a_{21} p_{n+1} + a_{22} q_{n+1} \right) \]

• solve linear system of equations:

\[ (1 - \tau a_{11}) p_{n+1} - \tau a_{12} q_{n+1} = p_n + \tau b_1 \]
\[ -\tau a_{21} p_{n+1} + (1 - \tau a_{22}) q_{n+1} = q_n + \tau b_2 \]

(for each time step \( n \to n + 1 \))

• in vector notation: \((I - \tau A) y_{n+1} = y_n + \tau b\)
Local Discretisation Error

- local influence of using differences instead of derivatives
- example: Euler’s method

\[
I(\tau) = \max_{[a,b]} \left\{ \left\| \frac{y(t + \tau) - y(t)}{\tau} - f(t, y(t)) \right\| \right\} = \frac{\tau}{2} \| \dot{y}_n \| + O(\tau^2)
\]

- memory hook: insert exact solution \( y(t) \) into

\[
\frac{y_{n+1} - y_n}{\tau} - \dot{y}_n
\]

A numerical scheme is called **consistent**, if

\[
I(\tau) \to 0 \quad \text{for} \quad \tau \to 0
\]
Global Discretisation Error

- compare numerical solution with exact solution
- example: Euler’s method

\[ e(\tau) = \max_{k=0,\ldots,k_{\text{max}}} \{ \| y_k - y(t_k) \| \} \]

\( y(t) \) exact solution;
\( y_k \) solution of the discretized equations (depends on \( \tau \))

A numerical scheme is called **convergent**, if

\[ e(\tau) \to 0 \quad \text{for} \quad \tau \to 0 \]
Global Discretisation Error – Explicit Euler

\[ e_{n+1} = y_{n+1} - y(t_{n+1}), \]
\[ y_{n+1} = y_n + \tau f(t_n, y_n), \]
\[ y(t_{n+1}) = y(t_n) + \tau f(t_n, y(t_n)) + \tau^2 \frac{\ddot{y}(t_n)}{2} + O(\tau^3), \]
\[ \Rightarrow |e_{n+1}| \leq |e_n| + \tau M |e_n| + N \tau^2 \]
\[ = (1 + M \tau) |e_n| + N \tau^2 \]
\[ \leq (1 + \tau M)^2 |e_{n-1}| + (1 + \tau M) N \tau^2 + N \tau^2 \]
\[ \leq \ldots \leq e^{(n+1) \cdot \tau M} |e_0| + N \tau^2 \frac{e^{(n+1) \cdot \tau M} - 1}{\tau M} \]
\[ = N \tau \frac{e^{(n+1) \cdot \tau M} - 1}{M} = N \tau \frac{e^{(t_{n+1} - t_0) \tau} - 1}{M} = O(\tau). \]
Order of Consistency/Convergence

A numerical scheme is called **consistent** of order \( p \) (\( p \)-th order consistent), if

\[
l(\tau) = \mathcal{O}(\tau^p)
\]

A numerical scheme is called **convergent** of order \( p \) (\( p \)-th order convergent), if

\[
e(\tau) = \mathcal{O}(\tau^p)
\]

We have shown that the explicit Euler method is consistent and convergent of order 1.

For more complicated ODE or numerical schemes → see lecture(s) in Numerical Programming
Part II: Advanced Numerical Methods

Runge-Kutta-Methods
- 2nd-order Runge-Kutta
- 4th-order Runge-Kutta

Multistep Methods
- Adams-Bashforth
- Adams-Moulton
- Explicit Midpoint Rule

Problems for Numerical Methods for ODE
- Ill-Conditioned Problems
- Stability
- Stiff Equations
- Summary
Runge-Kutta-Methods

- 1st idea: use additional evaluations of $f$:

$$y_{n+1} = y_n + \tau \sum_{i=1}^{p} \beta_i f(t^i_n, y^i_n) \text{ with } t^i_n \in [t_n; t_{n+1}]$$

open questions: Where to obtain $y^i_n, i = 1, \ldots, p$? How to choose $\beta_i, i = 1, \ldots, p$?

- 2nd idea: numerical approximations for missing values of $y$:

$$y^i_n := y_n + \tau \sum_{j=1}^{p} \alpha_{i,j} f(t^j_n, y^j_n)$$

explicit Runge-Kutta: $\alpha_{i,j} = 0$ if $j \geq i$.

- 3rd idea: choose $\beta_i$ and $\alpha_{i,j}$ such that order of consistency is maximal (use quadrature rules)
Runge-Kutta-Methods of 2nd Order

- example: 2nd-order Runge-Kutta ("method of Heun")

\[
\begin{align*}
    y_n^I &= y_n, \\
    y_n^II &= y_n + \tau f(t_n, y_n^I), \\
    y_{n+1} &= y_n + \tau \frac{1}{2}(f(t_n, y_n^I) + f(t_{n+1}, y_n^{II})).
\end{align*}
\]

- interpretation of the steps:
  1. \( y_n^{II} = y_n + \tau f(t_n, y_n^I) \)  
     \( \rightarrow \) Euler step to obtain estimate \( y_n^{II} \) for solution at time \( t_{n+1} \)
  2. \( y_{n+1} = y_n + \tau \frac{1}{2}(f(t_n, y_n^I) + f(t_{n+1}, y_n^{II})) \)  
     \( \rightarrow \) Euler step using an averaged direction
       (averages directions obtained from \( t_n, y_n^I \) and \( t_{n+1}, y_n^{II} \))

- exercise: draw these steps into direction field!
Runge-Kutta-Methods of 2nd Order (2)

- further example: “modified Euler” (also 2nd order)

\[
\begin{align*}
y_n & = y_n, \\
y_n^\| & = y_n + \frac{\tau}{2} f(t_n, y_n), \\
y_{n+1} & = y_n + \tau f(t_n + \frac{\tau}{2}, y_n^\|)
\end{align*}
\]

- interpretation of the steps:
  1. \(y_n^\| = y_n + \frac{\tau}{2} f(t_n, y_n)\)

     → Euler step with step size \(\tau/2\)
     to obtain estimate \(y_n^\|\) for solution at time \(t_{n+1/2}\)
  2. \(y_{n+1} = y_n + \tau f\left(t_n + \frac{\tau}{2}, y_n^\|\right)\)

     → Euler step using direction obtained from \(t_{n+1/2}, y_n^\|\)

- exercise: draw these steps into direction field!
Runge-Kutta-Method of 4th order

classical 4th-order Runge-Kutta:

- intermediate steps:

\[ t_I^n = t_n, \quad y_I^n = y_n, \]  \hspace{2cm}  (1)

\[ t_{II}^n = t_n + \frac{\tau}{2}, \quad y_{II}^n = y_n + \frac{\tau}{2} f(t_I^n, y_I^n), \]  \hspace{2cm}  (2)

\[ t_{III}^n = t_n + \frac{\tau}{2}, \quad y_{III}^n = y_n + \frac{\tau}{2} f(t_{II}^n, y_{II}^n), \]  \hspace{2cm}  (3)

\[ t_{IV}^n = t_{n+1}, \quad y_{IV}^n = y_n + \tau f(t_{III}^n, y_{III}^n). \]  \hspace{2cm}  (4)

- explicit scheme:

\[ y_{n+1} = y_n + \frac{\tau}{6} \left( f(t_I^n, y_I^n) + 2f(t_{II}^n, y_{II}^n) + 2f(t_{III}^n, y_{III}^n) + f(t_{IV}^n, y_{IV}^n) \right) \]

- How would you translate this method into efficient pseudo-code? Remember that unnecessary evaluations of \( f \) can be very costly!
Pseudo-Code Runge-Kutta-Method of 4th order

\[
\begin{align*}
    f[1] &= f(t, y[n]); \\
    dt2 &= dt/2; \\
    t2 &= t + dt2; \\
    f[2] &= f(t2, y[n] + dt2*f[1]); \\
    t &= t + dt; \\
    f[4] &= f(t, y[n] + dt*f[3]); \\
\end{align*}
\]

Note: each \( f[\ldots] \) will be a vector (1D array) for systems of ODE
Runge-Kutta Methods – Butcher Tableau

Generic notation for \( p \)-step Runge-Kutta schemes (\( i = 1, \ldots, p \)):

- use intermediate time steps \( t^i_n := t_n + \gamma_i \tau \), with \( \gamma_i \in [0, 1] \)

- intermediate directions \( k^i := f(t^i_n, y_n + \tau \sum_{j=1}^{i-1} \alpha_{ij} k^j) \) with weights \( \alpha_{ij} \)

- “final” step \( y_{n+1} = y_n + \tau \sum_{i=1}^{p} \beta_i k^i \) with suitable \( \beta_i \)

Leads to “Butcher” tableau of parameters \( \gamma_i, \beta_i \) and \( \alpha_{ij} \):

\[
\begin{array}{c|cccc}
\gamma_1 & \alpha_{11} & \alpha_{12} & \ldots & \alpha_{1p} \\
\gamma_2 & \alpha_{21} & \alpha_{22} & \ldots & \alpha_{2p} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\gamma_p & \alpha_{p1} & \alpha_{p2} & \ldots & \alpha_{pp} \\
\hline
\beta_1 & \beta_2 & \ldots & \beta_p \\
\end{array}
\]

with \( \alpha_{ij} = 0 \) if \( j \geq i \) (for explicit schemes)
Multistep Methods

- 1st idea: use previous steps for computation:

\[ y_{n+s} = g(y_n, y_{n+1}, \ldots, y_{n+s-1}) \]

- 2nd idea: use integral form of ODE

\[
\begin{align*}
\dot{y}(t) &= f(t, y(t)) \\
\int_{t_{n+s-1}}^{t_{n+s}} \dot{y}(\theta) d\theta &= \int_{t_{n+s-1}}^{t_{n+s}} f(\theta, y(\theta)) d\theta \\
y(t_{n+s}) - y(t_{n+s-1}) &= \int_{t_{n+s-1}}^{t_{n+s}} f(\theta, y(\theta)) d\theta = ?
\end{align*}
\]
Multistep and Numerical Quadrature

• 3rd idea: use numerical method for integration
  → interpolate $f$ using a polynomial $p$:

\[ y(t_{n+s}) - y(t_{n+s-1}) = \int_{t_{n+s-1}}^{t_{n+s}} f(\theta, y(\theta)) d\theta \approx \int_{t_{n+s-1}}^{t_{n+s}} p(\theta) d\theta \]

where

\[ p(t_{n+j}) = f(t_{n+j}, y(t_{n+j})) \quad \text{for } j = 0, \ldots, s - 1. \]

• compute integral and obtain quadrature rule:

\[ y_{n+s} = y_{n+s-1} + \tau \sum_{j=0}^{s-1} \beta_j f(t_{n+j}, y(t_{n+j})) \]
Adams-Bashforth

- $s = 1 \Rightarrow$ use $y_n$ only to compute $y_{n+1}$ (Euler’s method):
  \[ p(t) = f(t_n, y_n), \quad y_{n+1} = y_n + \tau f(t_n, y_n) \]

- $s = 2 \Rightarrow$ use $y_n$ and $y_{n+1}$ to compute $y_{n+2}$:
  \[ p(t) = \frac{t_{n+1} - t}{\tau} f(t_n, y_n) + \frac{t - t_n}{\tau} f(t_{n+1}, y_{n+1}), \]
  \[ y_{n+2} = y_{n+1} + \frac{\tau}{2} (3f(t_{n+1}, y_{n+1}) - f(t_n, y_n)) \]

- usually consistent of $s$-th order
- modified at start (no previous values available)
Adams-Moulton

- use idea of Adams-Bashforth, but:
  include value $y_{n+s}$ in polynomial $p \Rightarrow$ implicit scheme
- first order: implicit Euler
  \[ p(t) = f(t_{n+1}, y_{n+1}), \quad y_{n+1} = y_n + \tau f(t_{n+1}, y_{n+1}) \]
- second order: Crank-Nicolson method (trapezoidal rule)
  \[ y_{n+1} = y_n + \frac{\tau}{2} (f(t_n, y_n) + f(t_{n+1}, y_{n+1})) \]

- how to obtain $y_{n+s}$?
  - solve (nonlinear) equation $\Rightarrow$ difficult!
  - easier and more common: predictor-corrector approach
    (e.g.: explicit predictor plus fixpoint iteration on implicit scheme)
Explicit Midpoint Rule

- formally a multi-step scheme: \( y_{n+2} = g(y_n, y_{n+1}) \)
- derive from symmetric difference quotient:

\[
\begin{align*}
\dot{y}(t) &= f(t, y(t)) \\
y(t + \tau) - y(t - \tau) &= \frac{f(t, y(t))}{2\tau}
\end{align*}
\]

- leads to the following time-stepping scheme:

\[ y_{n+2} = y_n + 2\tau f(t_{n+1}, y_{n+1}) \]

- 2nd-order accurate with low computational effort, but tends to be unstable (see below)
Problems for Numerical Methods for ODE

Short overview on possible problems:

- **Ill-Conditioned Problems:**
  small changes in the input ⇒ big changes in the exact solution of the ODE

- **Instability:**
  big errors in the numerical solution compared to the exact solution (for arbitrarily small time steps although the method is consistent)

- **Stiffness:**
  small time steps for explicit single-step methods required for non-oscillatory behaviour of the approximate solution (although the exact solution is smooth)
Ill-Conditioned Problems

- small changes in input entail completely different results
- numerical treatment of such problems is always difficult!
- discriminate:
  - only at critical points?
  - everywhere?
- possible risks:
  - non-precise input
  - round-off errors,…
- question: what are you interested in?
  - really the solution for specific initial condition?
  - statistical info on the solution?
  - general behaviour (patterns)?
Stability: Example explicitLMM2.py

Example:

\[ \dot{y}(t) = y(t), \quad y(0) = 1 \]

- exact solution: \( y(t) = (e^t) \)
- well-conditioned: \( y_\varepsilon(0) = 1 + \varepsilon \Rightarrow y_\varepsilon(t) - y(t) = \varepsilon e^t \)
- use following explicit multistep scheme (\( s = 2 \)):

\[
y_{n+2} = \underbrace{\vphantom{\frac{1}{t}} \frac{-4}{4} y_{n+1}}_{-4} + \frac{5}{5} y_n + \tau \cdot \left[ 4 \cdot f(t_{n+1}, y_{n+1}) + 2 \cdot f(t_n, y_n) \right] \]

- leads to numerical scheme:

\[
y_{n+2} = -4y_{n+1} + 5y_n + \tau \cdot [4y_{n+1} + 2y_n] \]

- consistency of the method: order 3
Stability: Example explicitLMM2.py (2)

Observation:
for start with exact initial values: \( y_0 = y(0) \) and \( y_1 = y(\tau) \)

\[ \Rightarrow \text{multistep method explLMM2 is 3rd-order consistent, but does not converge here: instable behaviour} \]
Stability

• reason: difference equation generates spurious solutions
• analysis: roots $\mu_i$ of characteristic polynomial – are all $|\mu_i| < 1$?

Stability of ODE schemes:
• single step methods: always stable
• multistep methods: additional stability conditions
• in general: 
  
  consistency + zero-stability = convergence
Stability: Example Explicit Midpoint Rule

Example ODE (Dahlquist):
\[
\dot{y}(t) = -10y(t), \quad y(0) = 1
\]

- exact solution: \( y(t) = (e^{-10t}) \)
- use explicit midpoint rule (\( s = 2 \)):
  \[
y_{n+2} = y_n + 2\tau f(t_{n+1}, y_{n+1})
\]
- starting steps: \( y_0 = y(0) = 1, \quad y_1 = y(\tau) = e^{-10\tau} \)
- method is zero-stable
Stability: Example Explicit Midpoint Rule (2)

python demo: different $\tau = 0.01, 0.001 \Rightarrow$ Martini glass effect

![Graphs showing the behavior of different time steps](image1.png)

![Graph showing the scaled error for different time steps](image2.png)
Stability: Martini Glass Effect

slightly different problem: different \( \tau = 0.1, 0.01, 0.001 \)
Stiff Equations

Example:

\[ \dot{y}(t) = -1000y(t) + 1000, \quad y(0) = y_0 = 2 \]

- exact solution: \( y(t) = e^{-1000t} + 1 \)
- explicit Euler (i.e., a stable scheme):

\[
\begin{align*}
y_{k+1} &= y_k + \tau(-1000y_k + 1000) \\
&= (1 - 1000\tau)y_k + 1000\tau \\
&= (1 - 1000\tau)^k + 1
\end{align*}
\]

- oscillations and divergence for \( \delta t > 0.002 \)
- Why that? Consistency and stability are **asymptotic** terms!
Stiff Equations – Summary

Typical situation:
- one term in the ODE demands very small explicit time step
- but does not contribute much to the accuracy of the solution

Remedy: use implicit (or semi-implicit) methods
Summary

Runge-Kutta-methods:
- multiple evaluations of \( f \) (expensive, if \( f \) is expensive to compute)
- stable, well-behaved, easy to implement

Multistep methods:
- higher order, but only few evaluations of \( f \) (interesting, if \( f \) is expensive to compute)
- stability problems; behave “like wild horses”
- in practice: attention with non-smooth r.h.s.
  do not use uniform \( \tau \) and \( s \)

Implicit methods:
- for stiff equations
- most often used as corrector scheme

(Extrapolation methods:)
- for very long runtimes
- for very high accuracy