

Scientific Computing I

Module 4: Numerical Methods for ODE

Tobias Neckel

Winter 2015/2016



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

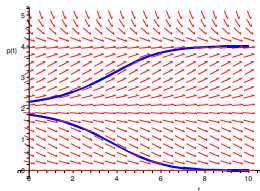
Order of Consistency/Convergence

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:

$$\begin{aligned} y_0 &:= y_0 \\ y_{n+1} &:= y_n + \tau f(t_n, y_n) \quad \text{for } n = 0, 1, 2, \dots \end{aligned}$$

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:

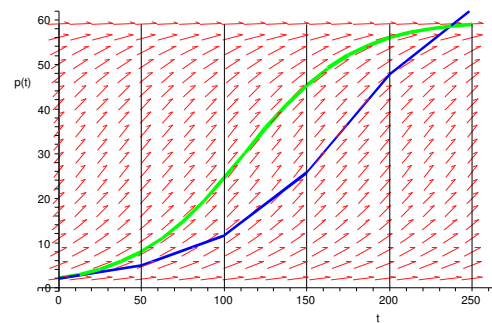
$$\frac{y_{n+1} - y_n}{\tau} \approx \dot{y}_n = f(t_n, y_n)$$

(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, $\dot{p}(t) = \alpha \left(1 - \frac{p(t)}{\beta}\right) p(t)$:

$$p_{n+1} := p_n + \tau \alpha \left(1 - \frac{p_n}{\beta}\right) p_n$$

- 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

$$\begin{aligned} \dot{p}(t) &= \left(\frac{71}{8} - \frac{23}{12}p(t) - \frac{25}{12}q(t)\right) p(t) \\ \dot{q}(t) &= \left(\frac{73}{8} - \frac{25}{12}p(t) - \frac{23}{12}q(t)\right) q(t) \end{aligned}$$

- Euler's method:

$$\begin{aligned} p_{n+1} &= p_n + \tau \left(\frac{71}{8} - \frac{23}{12}p_n - \frac{25}{12}q_n\right) p_n \\ q_{n+1} &= q_n + \tau \left(\frac{73}{8} - \frac{25}{12}p_n - \frac{23}{12}q_n\right) q_n \end{aligned}$$

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:

$$p_{n+1} := p_n - \delta p_n, \quad \delta > 0$$

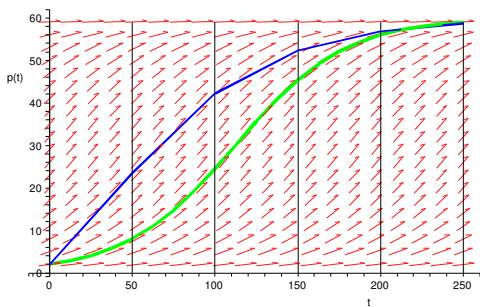
with decay rate δ ("percentage")

- obvious restriction in the discrete model: $\delta < 1$
- obvious restriction for τ in the discretized model?

$$\tau \alpha < 1 \Rightarrow \tau < \alpha^{-1}$$

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

Implicit Euler and Direction Fields



use direction at **end** of the timestep

Implicit Euler – 2D Example

- example: arms race

$$\begin{aligned} p_{n+1} &= p_n + \tau (b_1 + a_{11}p_{n+1} + a_{12}q_{n+1}) \\ q_{n+1} &= q_n + \tau (b_2 + a_{21}p_{n+1} + a_{22}q_{n+1}) \end{aligned}$$

- solve linear system of equations:

$$\begin{aligned} (1 - \tau a_{11})p_{n+1} - \tau a_{12}q_{n+1} &= p_n + \tau b_1 \\ -\tau a_{21}p_{n+1} + (1 - \tau a_{22})q_{n+1} &= q_n + \tau b_2 \end{aligned}$$

(for each time step $n \rightarrow n+1$)

- in vector notation: $(I - \tau A)\mathbf{y}_{n+1} = \mathbf{y}_n + \tau \mathbf{b}$

Global Discretisation Error

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

$$e(\tau) = \max_{k=0, \dots, k_{max}} \{ \|y_k - y(t_k)\| \}$$

$y(t)$ exact solution;

y_k solution of the discretized equations (depends on τ)

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

$$e(\tau) \rightarrow 0 \quad \text{for } \tau \rightarrow 0$$

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 – 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?

$$\begin{aligned} \alpha > 0: & \text{ then } 0 < (1 + \tau \alpha)^{-1} < 1 \text{ for any } \tau \\ \alpha < 0: & \text{ then } \tau < -\alpha^{-1} \text{ required!} \end{aligned}$$

- in physics $\alpha > 0$ is more frequent! (damped systems, friction, ...)
- implicit schemes preferred when explicit schemes require very small τ

Implicit Euler – 2D Example

Local Discretisation Error

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

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

- 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

$$l(\tau) \rightarrow 0 \quad \text{for } \tau \rightarrow 0$$

Global Discretisation Error – Explicit Euler

$$e_{n+1} = y_{n+1} - y(t_{n+1}),$$

$$y_{n+1} = y_n + \tau f(t_n, y_n),$$

$$y(t_{n+1}) = y(t_n) + \tau f(t_n, y(t_n)) + \tau^2 \frac{\ddot{y}(t_n)}{2} + O(\tau^3),$$

$$\begin{aligned} \Rightarrow |e_{n+1}| &\leq |e_n| + \tau M |e_n| + N \tau^2 \\ &= (1 + M \tau) |e_n| + N \tau^2 \\ &\leq (1 + \tau M)^2 |e_{n-1}| + (1 + \tau M) N \tau^2 + N \tau^2 \\ &\leq \dots \leq e^{(n+1) \cdot \tau M} \underbrace{|e_0|}_{=0} + N \tau^2 \frac{e^{(n+1) \cdot \tau M} - 1}{\tau M} \\ &= N \tau \frac{e^{(n+1) \cdot \tau M} - 1}{M} = N \tau \frac{e^{(t_{n+1} - t_0) M} - 1}{M} = O(\tau). \end{aligned}$$

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
Leapfrog Scheme

Problems for Numerical Methods for ODE

Ill-Conditioned Problems
Stability
Stiff Equations
Summary

Runge-Kutta-Methods

- 1st idea: use additional evaluations of f :

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

open questions: Where to obtain $y_n^i, i = 1, \dots, p$? How to choose $\beta_i, i = 1, \dots, p$?

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

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

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

- 3rd idea: choose β_i and $\alpha_{i,j}$ such that order of consistency is maximal (use quadrature rules)

Runge-Kutta-Methods of 2nd Order (2)

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

$$\begin{aligned} y_n^I &= y_n, \\ y_n^{II} &= y_n + \frac{\tau}{2} f(t_n, y_n^I), \\ y_{n+1} &= y_n + \tau f\left(t_n + \frac{\tau}{2}, y_n^{II}\right) \end{aligned}$$

- interpretation of the steps:

- $y_n^{II} = y_n + \frac{\tau}{2} f(t_n, y_n^I)$
→ Euler step with step size $\tau/2$
to obtain estimate y_n^{II} for solution at time $t_{n+\frac{1}{2}}$
 - $y_{n+1} = y_n + \tau f(t_n + \frac{\tau}{2}, y_n^{II})$
→ Euler step using direction obtained from $t_{n+\frac{1}{2}}, y_n^{II}$
- exercise: draw these steps into direction field!

Pseudo-Code Runge-Kutta-Method of 4th order

```
f[1] = f(t, y[n]);
dt2 = dt/2;
t2 = t + dt2;
f[2] = f(t2, y[n] + dt2*f[1]);
f[3] = f(t2, y[n] + dt2*f[2]);
t = t + dt;
f[4] = f(t, y[n] + dt*f[3]);
y[n+1] = y[n] + dt * (f[1] + 2*f[2] + 2*f[3] + f[4]);
```

Note: each $f[\dots]$ will be a vector (1D array) for systems of ODE

Runge-Kutta-Methods of 2nd Order

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

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

- interpretation of the steps:

- $y_n^{II} = y_n + \tau f(t_n, y_n^I)$
→ Euler step to obtain estimate y_n^{II} for solution at time t_{n+1}
 - $y_{n+1} = y_n + \tau \frac{1}{2} (f(t_n, y_n^I) + f(t_{n+1}, y_n^{II}))$
→ 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-Method of 4th order

classical 4th-order Runge-Kutta:

- intermediate steps:

$$t_n^I = t_n, \quad y_n^I = y_n, \quad (1)$$

$$t_n^{II} = t_n + \frac{\tau}{2}, \quad y_n^{II} = y_n + \frac{\tau}{2} f(t_n^I, y_n^I), \quad (2)$$

$$t_n^{III} = t_n + \frac{\tau}{2}, \quad y_n^{III} = y_n + \frac{\tau}{2} f(t_n^{II}, y_n^{II}), \quad (3)$$

$$t_n^{IV} = t_{n+1}, \quad y_n^{IV} = y_n + \tau f(t_n^{III}, y_n^{III}). \quad (4)$$

- explicit scheme:

$$y_{n+1} = y_n + \frac{\tau}{6} \left(f(t_n^I, y_n^I) + 2f(t_n^{II}, y_n^{II}) + 2f(t_n^{III}, y_n^{III}) + f(t_n^{IV}, y_n^{IV}) \right)$$

- How would you translate this method into efficient pseudo-code?
Remember that unnecessary evaluations of f can be very costly!

Multistep Methods

- 1st idea: use previous steps for computation:

$$y_{n+1} = g(y_n, y_{n-1}, \dots, y_{n-q+1})$$

- 2nd idea: use integral form of ODE

$$\begin{aligned} \dot{y}(t) &= f(t, y(t)) \\ \int_{t_n}^{t_{n+1}} \dot{y}(t) dt &= \int_{t_n}^{t_{n+1}} f(t, y(t)) dt \\ y(t_{n+1}) - y(t_n) &= \int_{t_n}^{t_{n+1}} f(t, y(t)) dt =? \end{aligned}$$

Multistep and Numerical Quadrature

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

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t)) dt \approx \int_{t_n}^{t_{n+1}} p(t) dt$$

where

$$p(t_j) = f(t_j, y(t_j)) \quad \text{for } j = n - s + 1, \dots, n.$$

- compute integral and obtain quadrature rule:

$$y_{n+1} = y_n + \sum_{j=n-s+1}^n \beta_j f(t_j, y_j)$$

Adams-Moulton

- use idea of Adams-Bashforth, but:
include value $y_{n+1} \Rightarrow$ **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

$$y_{n+1} = y_n + \frac{\tau}{2} (f(t_n, y_n) + f(t_{n+1}, y_{n+1}))$$

- how to obtain y_{n+1} ?
 - solve (nonlinear) equation \Rightarrow difficult!
 - easier and more common: *predictor-corrector* approach

Problems for Numerical Methods for ODE

Short overview on possible problems:

- **Ill-Conditioned Problems:**
small changes in the input \Rightarrow 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 required for acceptable errors in the approximate solution (although the exact solution is smooth)

Stability

Example:

$$\dot{y}(t) = -2y(t) + 1, \quad y(0) = 1$$

- exact solution: $y(t) = \frac{1}{2}(e^{-2t} + 1)$
- well-conditioned: $y_\varepsilon(0) = 1 + \varepsilon \Rightarrow y_\varepsilon(t) - y(t) = \varepsilon e^{-2t}$
- use **midpoint** rule (multistep scheme):

$$y_{n+1} = y_{n-1} + 2\tau \cdot f(t_n, y_n)$$

- leads to numerical scheme:

$$y_{n+1} = y_{n-1} + 2\tau(1 - 2y_n)$$

Adams-Bashforth

- $s = 1 \Rightarrow$ use y_n only (leads to 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-1} and y_n :

$$p(t) = \frac{t_n - t}{\tau} f(t_{n-1}, y_{n-1}) + \frac{t - t_{n-1}}{\tau} f(t_n, y_n),$$

$$y_{n+1} = y_n + \frac{\tau}{2} (3f(t_n, y_n) - f(t_{n-1}, y_{n-1}))$$

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

Mitpoint Rule

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

$$\dot{y}(t) = f(t, y(t))$$

$$\frac{y(t+\tau) - y(t-\tau)}{2\tau} = f(t, y(t))$$

- leads to the following time-stepping scheme:

$$y_{n+1} = y_{n-1} + 2\tau f(t_n, y_n)$$

- 2nd-order accurate with low computational effort, but tends to be unstable

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 (2)

Observation:

- 2-step rule:

$$y_{n+1} = y_{n-1} + 2\tau(1 - 2y_n)$$

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

- numerical results for different sizes of τ :
 - $\tau = 1.0 \Rightarrow y_9 = -4945.5, y_{10} = 20953.9$
 - $\tau = 0.1 \Rightarrow y_{79} = -1725.3, y_{80} = 2105.7$
 - $\tau = 0.01 \Rightarrow y_{999} = -154.6, y_{1000} = 158.7$
- midpoint rule is 2nd-order consistent, but does not converge here: oscillations or instable behaviour

Stability (3)

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

Stability of ODE schemes:

- single step schemes: always stable
- multistep schemes: additional stability conditions
- in general:
 - consistency + stability = convergence**

Stiff Equations

Example:

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

- exact solution: $y(t) = e^{-1000t} + 1$
- explicit Euler (stable):

$$\begin{aligned} y_{k+1} &= y_k + \tau(-1000y_k + 1000) \\ &= (1 - 1000\tau)y_k + 1000\tau \\ &= (1 - 1000\tau)^{k+1} + 1 \end{aligned}$$

- 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 time step
- but does not contribute much to 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: do not use uniform τ and s

Implicit methods:

- for stiff equations
- most often used as corrector scheme

