

Outlines

Part I: Basic Numerical
Methods

Part II: Advanced Numerical
Methods

Scientific Computing I

Module 4: Numerical Methods for ODE

Michael Bader

Lehrstuhl Informatik V

Winter 2008/2009

Outlines

Part I: Basic Numerical
Methods

Part II: Advanced Numerical
Methods

Part I: Basic Numerical Methods

- 1 Direction Fields
- 2 Euler's Method
- 3 Discretized Model vs. Discrete Model
- 4 Implicit Euler
- 5 Analysis of Numerical Schemes for ODE
 - Local Discretisation Error
 - Global Discretisation Error
 - Order of Consistency/Convergence

Outlines

Part I: Basic Numerical
Methods

Part II: Advanced Numerical
Methods

Part II: Advanced Numerical Methods

6 Runge-Kutta-Methods

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

7 Multistep Methods

- Adams-Bashforth
- Adams-Moulton

8 Problems for Numerical Methods for ODE

- Ill-Conditioned Problems
- Stability
- Stiff Equations
- Summary

Part I

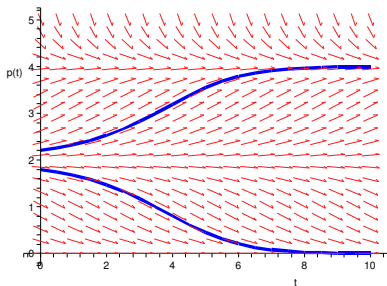
Basic Numerical Methods for ODE

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:

$$y_0 := y_0$$

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

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

Scientific
Computing I

Michael Bader

Direction Fields

Euler's Method

Discretized Model
vs. Discrete Model

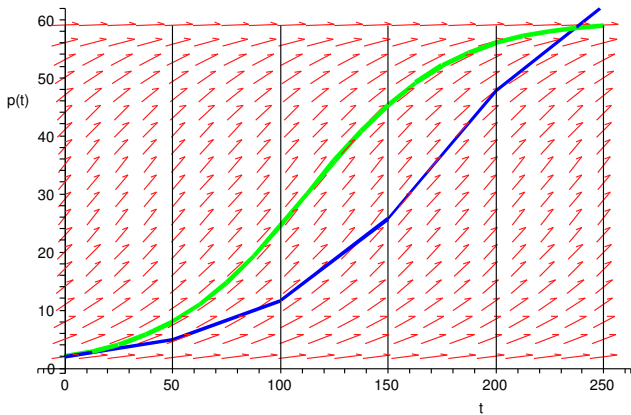
Implicit Euler

Analysis of
Numerical
Schemes for ODE

Local Discretisation Error

Global Discretisation Error

Order of
Consistency/Convergence



use direction at **begin** 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 \rightarrow use vector notation:

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

- example: nonlinear extinction model

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

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

- Euler's method:

$$p_{n+1} = p_n + \tau \left(\frac{71}{8} - \frac{23}{12}p_n - \frac{25}{12}q_n \right) p_n$$

$$q_{n+1} = q_n + \tau \left(\frac{73}{8} - \frac{25}{12}p_n - \frac{23}{12}q_n \right) q_n$$

Discretized Model vs. Discrete Model

- simplest example: model of Malthus

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

- 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

- 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:

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

- explicit formula for y_{n+1} not immediately available
- to do: solve equation for y_{n+1}

Implicit Euler and Direction Fields

Scientific
Computing I

Michael Bader

Direction Fields

Euler's Method

Discretized Model
vs. Discrete Model

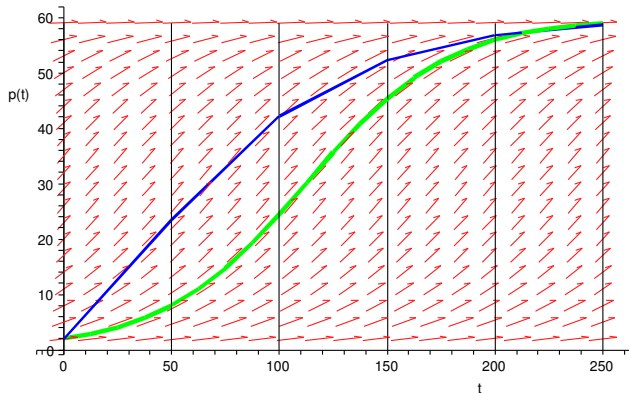
Implicit Euler

Analysis of
Numerical
Schemes for ODE

Local Discretisation Error

Global Discretisation Error

Order of
Consistency/Convergence



use direction at **end** of the timestep

Implicit Euler – Examples

- example: Model of Malthus

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

- correct (discrete) model?

$\alpha < 0$: then $0 < (1 - \tau\alpha)^{-1} < 1$ for any τ

$\alpha > 0$: then $\tau < \alpha^{-1}$ required!

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

Implicit Euler – 2D Example

- example: arms race

$$p_{n+1} = p_n + \tau(b_1 + a_{11}p_{n+1} + a_{12}q_{n+1})$$

$$q_{n+1} = q_n + \tau(b_2 + a_{21}p_{n+1} + a_{22}q_{n+1})$$

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

Local Discretisation Error

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

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

- 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

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

$$e(\tau) = \max_{[a,b]} \{ \|y_k - y(t_k)\| \}$$

$y(t)$ the exact solution;

y_k the solution of the discretized equation
(depends on τ)

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

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

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

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

Runge-Kutta-
Methods

2nd-order Runge-Kutta

4th-order Runge-Kutta

Multistep
Methods

Adams-Bashforth

Adams-Moulton

Problems for
Numerical
Methods for ODE

Ill-Conditioned Problems

Stability

Stiff Equations

Summary

Part II

Advanced Numerical Methods for ODE

Runge-Kutta-Methods

- 1st idea: use additional evaluations of f , e.g.:

$$y_{n+1} = g(y_n, f(t_n, y_n), f(t_{n+1}, y_{n+1}))$$

open question: where to obtain y_{n+1} , how to choose g

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

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

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

Runge-Kutta-Methods of 2nd order

- 3rd idea: choose g such that order of consistency is maximal
- example: 2nd-order Runge-Kutta:

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

("method of Heun")

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

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

Runge-Kutta-Method of 4th order

classical 4th-order Runge-Kutta:

- intermediate steps:

$$k_1 = f(t_n, y_n)$$

$$k_2 = f\left(t_n + \frac{\tau}{2}, y_n + \frac{\tau}{2}k_1\right)$$

$$k_3 = f\left(t_n + \frac{\tau}{2}, y_n + \frac{\tau}{2}k_2\right)$$

$$k_4 = f(t_n + \tau, y_n + \tau k_3)$$

- explicit scheme:

$$y_{n+1} = y_n + \frac{\tau}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

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

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

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

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

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:

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

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)

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:

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

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

Problems for Numerical Methods for ODE

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)

III-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:

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

- 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):

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

- leads to numerical scheme:

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

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

Runge-Kutta-
Methods

2nd-order Runge-Kutta

4th-order Runge-Kutta

Multistep
Methods

Adams-Bashforth

Adams-Moulton

Problems for
Numerical
Methods for ODE

Ill-Conditioned Problems

Stability

Stiff Equations

Summary

Stability (3)

- reason: difference equation generates spurious solutions
- analysis: roots μ_i of characteristic polynomial $y^2 = y^0 + 4\tau(1 - y)$; 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 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

Runge-Kutta-
Methods

2nd-order Runge-Kutta

4th-order Runge-Kutta

Multistep
Methods

Adams-Bashforth

Adams-Moulton

Problems for
Numerical
Methods for ODE

Ill-Conditioned Problems

Stability

Stiff Equations

Summary