CS520: NUMERICAL ODEs (CH.2)

Uri Ascher

Department of Computer Science
University of British Columbia
ascher@cs.ubc.ca
people.cs.ubc.ca/∼ascher/520.html

Outline

- ODEs
- Forward and backward Euler
- Linear multistep methods
- Runge-Kutta methods
- Stiffness
- Adaptive step size selection
- (In Chapter 6: geometric integration)

Ordinary differential equations

e.g. pendulum.

$$\frac{d^2\theta}{dt^2} \equiv \theta'' = -g\sin(\theta),$$

where g is the scaled constant of gravity, e.g., $g = 9.81$, and t is time.

- Write as first order ODE system: $y_1(t) = \theta(t)$, $y_2(t) = \theta'(t)$. Then $y_1' = y_2$, $y_2' = -g\sin(y_1)$.
- ODE in standard form:

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}), \quad a < t < b.$$

For the pendulum

$$\mathbf{f}(t, \mathbf{y}) = \begin{pmatrix} y_2 \\ -g\sin(y_1) \end{pmatrix}.$$

Side conditions

e.g.

$$y' = -y \quad \Rightarrow y(t) = c \cdot e^{-t}.$$

- Initial value problem: $\mathbf{y}(a)$ given. (In the pendulum example: $\theta(0)$ and $\theta'(0)$ given.)
- Boundary value problem: relations involving $\mathbf{y}$ at more than one point given. (In the pendulum example: $\theta(0)$ and $\theta(\pi)$ given.)

We stick to initial value ODEs!

Outline

- ODEs
- Forward and backward Euler
- Linear multistep methods
- Runge-Kutta methods
- Stiffness
- Adaptive step size selection
- (In Chapter 6: geometric integration)

Forward and backward Euler

- Simplest method for the problem

$$y' = f(t, y), \quad y(a) = c.$$

- Use to demonstrate general concepts:
  - Method derivation
  - Explicit vs. implicit methods
  - Local truncation error and global error
  - Order of accuracy
  - Convergence
  - Absolute stability and stiffness.

Forward Euler: derivation

- Mesh points $t_0 < t_1 < \cdots < t_N$ with step size $k = t_{n+1} - t_n$. Approximate solution $y_n \approx y(t_n)$.
- Proceed to march from one mesh point to the next (step by step).
- By Taylor expansion

$$f(t_n, y(t_n)) = y'(t_n) = \frac{y(t_{n+1}) - y(t_n)}{k} - \frac{k}{2} y''(\xi_n).$$

This is a forward difference. Obtain

$$y(t_{n+1}) = y(t_n) + k f(t_n, y(t_n)) + \frac{k^2}{2} y''(\xi_n).$$

- So, set

$$\begin{aligned} y_0 &= c, \\ y_{n+1} &= y_n + k f(t_n, y_n), \quad n = 0, 1, \ldots, N-1. \end{aligned}$$

Example: advection equation

- Recall advection equation $u_t + a u_x = 0$.
- Discretize in space using the one-sided scheme:

$$\frac{dv_j}{dt} = -\frac{a}{h}\left(v_{j+1} - v_j\right).$$

  Obtain ODE system for $\mathbf{v} = (v_0, v_1, v_2, \ldots, v_J)^T$.
- Alternatively, discretize in space using the centred scheme:

$$\frac{dv_j}{dt} = -\frac{a}{2h}\left(v_{j+1} - v_{j-1}\right).$$

  Obtain ODE system for $\mathbf{v} = (v_1, v_2, \ldots, v_J)^T$.
- Applying forward Euler to the first semi-discretization, obtain one-sided method from Chapter 1.
- Applying forward Euler to the second semi-discretization, obtain the (unstable) centred method from Chapter 1.

Example: advection equation cont.

- Recall advection equation $u_t + a u_x = 0$.
- Discretize in space using the one-sided scheme:

$$\frac{dv_j}{dt} = -\frac{a}{h}\left(v_{j+1} - v_j\right).$$

  Obtain ODE system for $\mathbf{v} = (v_0, v_1, v_2, \ldots, v_J)^T$.
- Alternatively, discretize in space using the centred scheme:

$$\frac{dv_j}{dt} = -\frac{a}{2h}\left(v_{j+1} - v_{j-1}\right).$$

  Obtain ODE system for $\mathbf{v} = (v_1, v_2, \ldots, v_J)^T$.
- **Caution:** Note that these are very large ODE systems... In the limit as $h \to 0$, system size $J \to \infty$.
- **Although numerical ODE analysis is relevant to PDEs, too, results do not extend automatically. Surprises do happen.**

Backward Euler: implicit vs. explicit

- Could use instead backward difference

$$y'(t_{n+1}) = \frac{y(t_{n+1}) - y(t_n)}{k} + \frac{k}{2} y''(\xi_n).$$

  Therefore,

$$y(t_{n+1}) = y(t_n) + kf(t_{n+1}, y(t_{n+1})) - \frac{k^2}{2} y''(\xi_n).$$

- So, set

$$\begin{aligned} y_0 &= c, \\ y_{n+1} &= y_n + kf(t_{n+1}, y_{n+1}), \quad n = 0, 1, \ldots, N-1. \end{aligned}$$

- But now, unknown y_{n+1} appears implicitly!
  More complicated and costly to carry out the stepping procedure.
- Forward Euler is an explicit method, backward Euler is an implicit method.

Forward and backward Euler

- Simplest method for the problem

$$y' = f(t, y), \quad y(a) = c.$$

- Use to demonstrate general concepts:
  - Method derivation
  - Explicit vs. implicit methods
  - Local truncation error and global error
  - Order of accuracy
  - Convergence
  - Absolute stability and stiffness.

Local truncation error and global error

- Local truncation error, d_i = the amount by which the exact solution fails to satisfy the difference equation, written in divided difference form.
- For forward Euler, $d_i = \frac{k}{2} y''(\xi_i)$.
- The order of accuracy is q if $\max_i |d_i| = \mathcal{O}(k^q)$.
  (So, the Euler methods are 1st order accurate.)
- Global error

$$e_n = y(t_n) - y_n, \quad n = 0, 1, \ldots, N.$$

- The method converges if $\max_{0 \le n \le N} |e_n| \to 0$ as $k \to 0$.
- For all the methods and problems we consider, there is a constant K s.t.

$$|e_n| \le K \max_i |d_i|, \quad n = 0, 1, \ldots, N.$$

So, order of accuracy is also order of convergence.

Euler convergence theorem

Let $f(t, y)$ have bounded partial derivatives in a region $\mathcal{D} = \{a \le t \le b, |y| < \infty\}$.
Note that this implies Lipschitz continuity in y: there exists a constant L such that for all (t, y) and $(t, \hat{y})$ in $\mathcal{D}$ we have

$$|f(t, y) - f(t, \hat{y})| \le L|y - \hat{y}|.$$

Then Euler's method converges and its global error decreases linearly in k.
Moreover, assuming further that

$$|y''(t)| \le M, \quad a \le t \le b,$$

the global error satisfies

$$|e_n| \le \frac{Mk}{2L}[e^{L(t_n - a)} - 1], \quad n = 0, 1, \ldots, N.$$

Forward and backward Euler

- Simplest method for the problem

$$y' = f(t, y), \quad y(a) = c.$$

- Use to demonstrate general concepts:
  - Method derivation
  - Explicit vs. implicit methods
  - Local truncation error and global error
  - Order of accuracy
  - Convergence
  - Absolute stability and stiffness

Absolute stability

- Convergence is for $k \to 0$, but in computation the step size is finite and fixed. How is the method expected to perform?
- Consider simplest, test equation

$$y' = \lambda y.$$

Solution: $y(t) = y(0)e^{\lambda t}$. Assuming λ real, solution increases for $\lambda > 0$, decreases for $\lambda < 0$.

- Forward Euler:

$$y_{n+1} = y_n + k\lambda y_n = (1 + k\lambda)y_n = \ldots = (1 + k\lambda)^{n+1} y(0).$$

- So, approximate solution does not grow only if $|1 + k\lambda| \leq 1$. Important when $\lambda \leq 0$.
- For $\lambda < 0$ must require

$$k \leq \frac{2}{-\lambda}.$$

Absolute stability and stiffness

- The restriction on the step size is an absolute stability requirement.
- Note: it's a *stability, not accuracy*, requirement.
- If absolute stability requirement is much more restrictive than accuracy requirement, the problem is stiff.
- **Example**

$$y' = -1000(y - \cos(t)) - \sin(t), \quad y(0) = 1,$$

The exact solution is $y(t) = \cos(t)$: varies slowly and smoothly.

- Here $\lambda = -1000$, so applying forward Euler, must require

$$k \leq \frac{2}{1000} = .002$$

Backward Euler and implicit methods

- Apply *backward Euler* to the *test equation*:

$$y_{n+1} = y_n + k\lambda y_{n+1}.$$

Hence

$$y_{n+1} = \frac{1}{1 - k\lambda} y_n.$$

- Here $|y_{n+1}| \leq |y_n|$ for any $k > 0$ and $\lambda < 0$: no annoying absolute stability restriction.
- For the example, integrating from 0 to 1 using forward Euler with $k = .0021$ obtain solution blowup (error 5.8e+13).
  Using backward Euler with $k = .01$ obtain error 2.7e-6.
  Using backward Euler with $k = .1$ obtain error 2.4e-5.
  Using forward or backward Euler with $k = .001$ obtain comparable accuracy, with error $\approx$?

More generally

- Stiff systems do arise a lot in practice.
- If problem is very stiff, explicit methods are not effective.
- Simplest case of a system: $\mathbf{y}' = A\mathbf{y}$, A a constant $m \times m$ diagonalizable matrix.
- There is a similarity transformation T so that

$$T^{-1}AT = \mathrm{diag}(\lambda_1, \ldots, \lambda_m).$$

Then for $\mathbf{x} = T^{-1}\mathbf{y}$ obtain m test equations

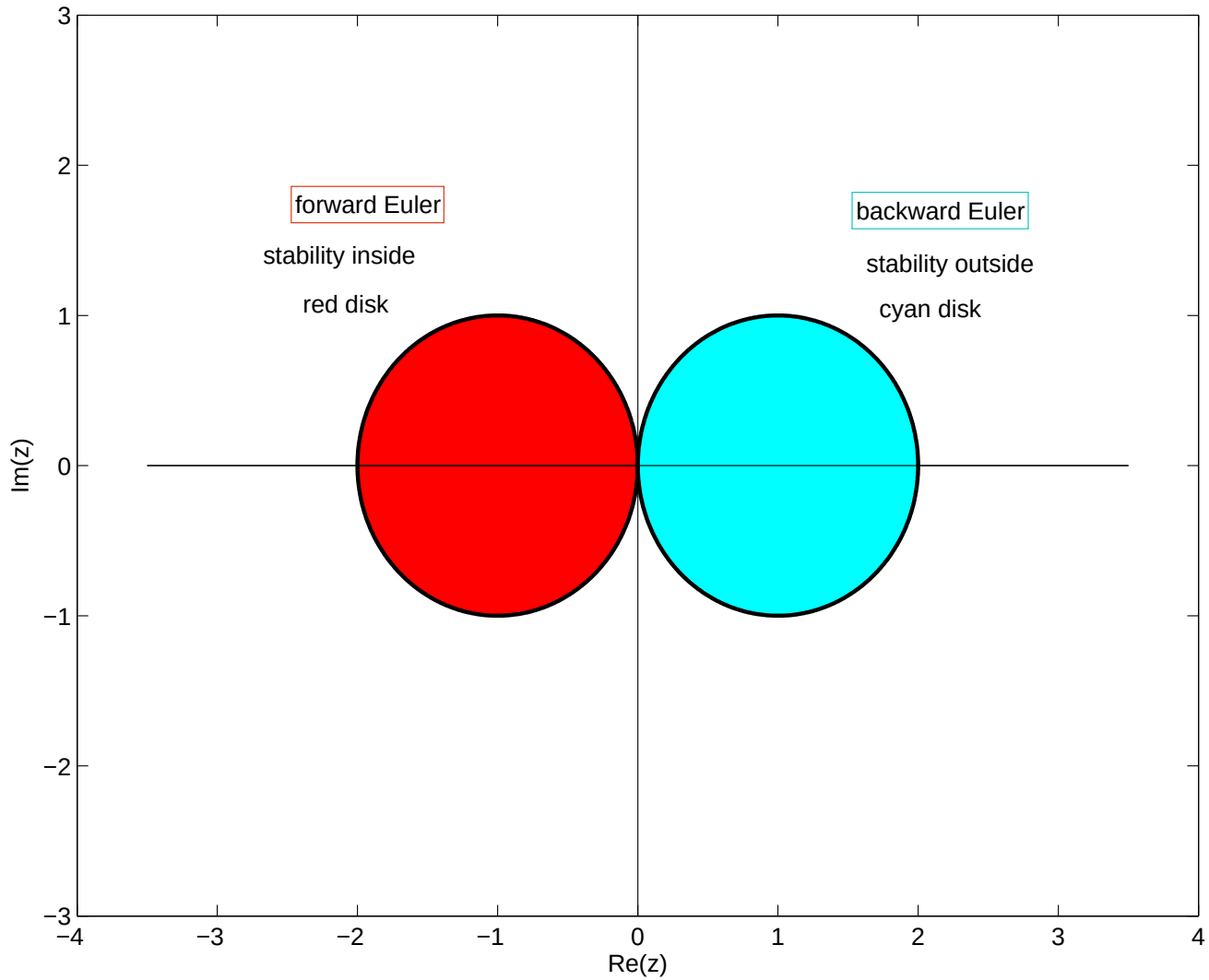
$$x_j' = \lambda_j x_j, \quad j = 1, \ldots, m.$$

- For forward Euler must require

$$|1 + k\lambda_j| \leq 1, \quad j = 1, 2, \ldots, m.$$

- **Big complication**: the eigenvalues λ_j may be complex!

Absolute stability in the complex plane



Simple example

- Consider

$$\mathbf{y}' = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{y}.$$

  Note that the matrix A is skew-symmetric.
- Eigenvalues $\lambda_1 = \imath, \ \lambda_2 = -\imath$.
- For forward Euler stability need

$$|1 \pm k\imath| \leq 1.$$

- But $1^2 + k^2 > 1$ for any $k > 0$! So forward Euler is unconditionally unstable and cannot be used.
- For backward Euler need

$$|1 \pm k\imath|^{-1} \leq 1.$$

  This occurs for all k, so method is unconditionally stable.
  However, note decay introduced by method, which is unmatched by exact solution.

Even more generally

- For a nonlinear ODE system $\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$ let $\mathbf{y}$, $\hat{\mathbf{y}}$ be two nearby trajectories and consider the progress of the perturbation $\mathbf{w}(t) = \mathbf{y}(t) - \hat{\mathbf{y}}(t)$.
- By Taylor $\mathbf{f}(\mathbf{y}) = \mathbf{f}(\hat{\mathbf{y}}) + J(\hat{\mathbf{y}})\mathbf{w} + \mathcal{O}\left(\|\mathbf{w}\|^2\right)$, with the Jacobian matrix

$$J = \frac{\partial \mathbf{f}}{\partial \mathbf{y}} = \begin{pmatrix} \frac{\partial f_1}{\partial y_1} & \cdots & \cdots & \frac{\partial f_1}{\partial y_m} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_m}{\partial y_1} & \cdots & \cdots & \frac{\partial f_m}{\partial y_m} \end{pmatrix}.$$

So

$$\mathbf{w}' = J(\hat{\mathbf{y}})\mathbf{w} + \mathcal{O}\left(\|\mathbf{w}\|^2\right) \approx J(\mathbf{y})\mathbf{w}.$$

- Hence, must consider the eigenvalues of the Jacobian matrix along the solution trajectory!

Example: see Ascher & Greif book (2011)

- Problem from plant physiology

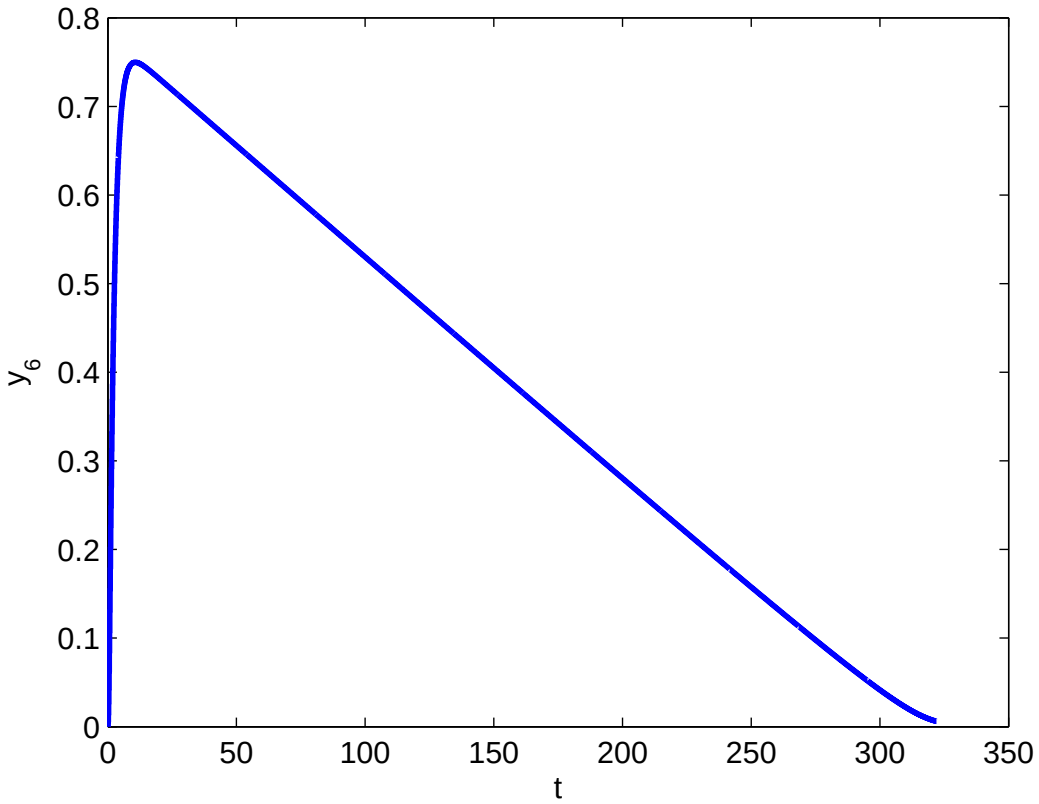
$$\mathbf{f}(t, \mathbf{y}) = (bla)^T, \ \mathbf{y}(0) = (1, 0, 0, 0, 0, 0, 0, .0057)^T, \ 0 \le t \le 322.$$

Then the Jacobian matrix is $J(\mathbf{y}) =$

$$\begin{pmatrix}
-1.71 & .43 & 8.32 & 0 & 0 & 0 & 0 & 0 \\
1.71 & -8.75 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -10.03 & .43 & .035 & 0 & 0 & 0 \\
0 & 8.32 & 1.71 & -1.12 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1.745 & .43 & .43 & 0 \\
0 & 0 & 0 & .69 & 1.71 & 0 & .69 & 0 \\
0 & 0 & 0 & 0 & 0 & -280y_8 - .43 & 0 & -280y_6 \\
0 & 0 & 0 & 0 & 0 & 280y_8 & -1.81 & 280y_6 \\
0 & 0 & 0 & 0 & 0 & -280y_8 & 1.81 & -280y_6
\end{pmatrix}.$$

Example cont.

The 6th solution component:



Example cont.

- Eigenvalues of $J(\mathbf{y}_0)$ are

$$0, -10.48, -8.28, -0.26, -0.51, -2.67 \pm 0.15\imath, -2.31.$$

- So, for forward Euler, $k = .1$ appears to be a safe step size choice.
- ...But it's not: a huge error results.
- Indeed, at $t \approx 10.7$, the eigenvalues equal

$$-211.77, -10.48, -8.28, -2.39, -2.14, -0.49, -3 \times 10^{-5}, -3 \times 10^{-12}.$$

- So, $k = .005$ appears to be a safe step size choice for forward Euler.
- ... And it works! However, many steps are now required to reach $t = 322$. To improve efficiency, use either a variable, adaptive step size, or backward Euler.

Solving nonlinear systems

- Upon discretizing $\mathbf{y}' = \mathbf{f}(\mathbf{y})$ using (an implicit method such as) backward Euler, obtain at each time step n

$$\mathbf{y}_{n+1} - k\mathbf{f}(\mathbf{y}_{n+1}) = \mathbf{y}_n,$$

  which is a nonlinear algebraic system for $\mathbf{y}_{n+1}$.
- Use an iterative method.
  Good news: usually $\mathbf{y}_n$ (which is known at this stage) is a good initial guess for $\mathbf{y}_{n+1}$.
  Bad news: for a stiff problem, $k\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ is not small, so a simple fixed point iteration will not work.
- So, use some variant of Newton's method: starting, e.g., with $\mathbf{y}_{n+1}^0 = \mathbf{y}_n$, for $\nu = 0, 1, \ldots,$
  1. Solve the linear system

$$(I - k\frac{\partial \mathbf{f}}{\partial \mathbf{y}})\delta\mathbf{y} = -(\mathbf{y} - k\mathbf{f}(\mathbf{y}) - \mathbf{y}_n)$$

     for the correction $\delta\mathbf{y}$, where $\mathbf{y} = \mathbf{y}_{n+1}^\nu$.
  2. Update $\mathbf{y}_{n+1}^{\nu+1} = \mathbf{y}_{n+1}^\nu + \delta\mathbf{y}$.

Newton's method variants

1. Solve the linear system

$$\left(I - k\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)\delta\mathbf{y} = -\left(\mathbf{y} - k\mathbf{f}(\mathbf{y}) - \mathbf{y}_n\right)$$

for the correction $\delta\mathbf{y}$, where $\mathbf{y} = \mathbf{y}_{n+1}^{\nu}$.

- Newton's method converges quadratically, but iteration may be expensive and cumbersome to carry out. Possible simplifications:
  - Calculate and decompose $\left(I - k\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)$ only once every few time steps.
  - Take only one Newton step per time step:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \delta\mathbf{y},$$

where the correction $\delta\mathbf{y}$ solves the above linear system at $\mathbf{y} = \mathbf{y}_n$.
  - Use a mix of backward and forward Euler so that only stiff components of problem are discretized implicitly, thereby simplifying $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$, making it sparser, or symmetric positive definite, etc.
- All of these simplifications are useful, but none is a cure-all!

Outline

- ODEs
- Forward and backward Euler
- Linear multistep methods
- Runge-Kutta methods
- Stiffness
- Adaptive step size selection
- (In Chapter 6: geometric integration)

Higher order methods

- The Euler methods are only first order accurate: want higher accuracy.
- **Example**: Implicit trapezoidal method

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

- Easy to show that the local truncation error is $d_i = \mathcal{O}(k^2)$: Use

$$y(t_{n+1/2\pm 1/2}) = y(t_{n+1/2}) \pm \frac{k}{2} y'(t_{n+1/2}) + \frac{k^2}{8} y''(t_{n+1/2}) + \mathcal{O}(k^3).$$

- **But** resulting method is implicit!

Linear multistep methods

- Use not only current solution but also $s-1$ previous solution values to approximate next one.
- **Example**: $f(y_{n+1/2}) = f_{n+1/2} \approx \frac{1}{2}(3f_n - f_{n-1})$, so expect 2nd order from the two-step Adams-Bashforth method

$$y_{n+1} = y_n + \frac{k}{2}(3f_n - f_{n-1}).$$

- General form

$$\sum_{j=0}^{s} \alpha_j y_{n+1-j} = k \sum_{j=0}^{s} \beta_j f_{n+1-j}.$$

  Here, $f_{n+1-j} = f(t_{n+1-j}, y_{n+1-j})$ and α_j, β_j are coefficients, with $\alpha_0 = 1$ for definiteness.
- The method is *explicit* if $\beta_0 = 0$ and *implicit* otherwise.

Order of accuracy

- Define local truncation error as

$$d_n = k^{-1} \sum_{j=0}^{s} \alpha_j y(t_{n+1-j}) - \sum_{j=0}^{s} \beta_j y'(t_{n+1-j}).$$

  This is the amount by which the exact solution fails to satisfy the difference equations divided by k.

- The method has order of accuracy (or *order* for short) p if for all problems with sufficiently smooth exact solutions $y(t)$,

$$d_n = O(k^p).$$

e.g. the two-step Adams-Bashforth has order 2.

Adams-Bashforth and Adams-Moulton

- Derived by considering integrating ODE

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

  and approximating the integrand $f(t, y)$ by an interpolating polynomial through previously computed values of $f(t_l, y_l)$. Thus, $\alpha_1 = -1$, $\alpha_l = 0$, $l > 1$.

- s-step Adams-Bashforth: explicit. Interpolate

$$(t_n, f_n), (t_{n-1}, f_{n-1}), \ldots, (t_{n-s+1}, f_{n-s+1})$$

  (and extrapolate to $[t_n, t_{n+1}]$). The method has order s.

- s-step Adams-Moulton: implicit. Interpolate

$$(t_{n+1}, f_{n+1}), (t_n, f_n), \ldots, (t_{n-s+1}, f_{n-s+1})$$

  The method has order $s + 1$.

- Often used as predictor-corrector (PECE)

Backward differentiation formulae (BDF)

- Evaluate f only at right end of the current step, (t_{n+1}, y_{n+1}), and differentiate an interpolating polynomial of y through $t = t_{n+1}, t_n, t_{n-1}, \ldots, t_{n+1-s}$. The method has order s and is good for *stiff* problems.
- e.g., $s = 1$ yields Backward Euler

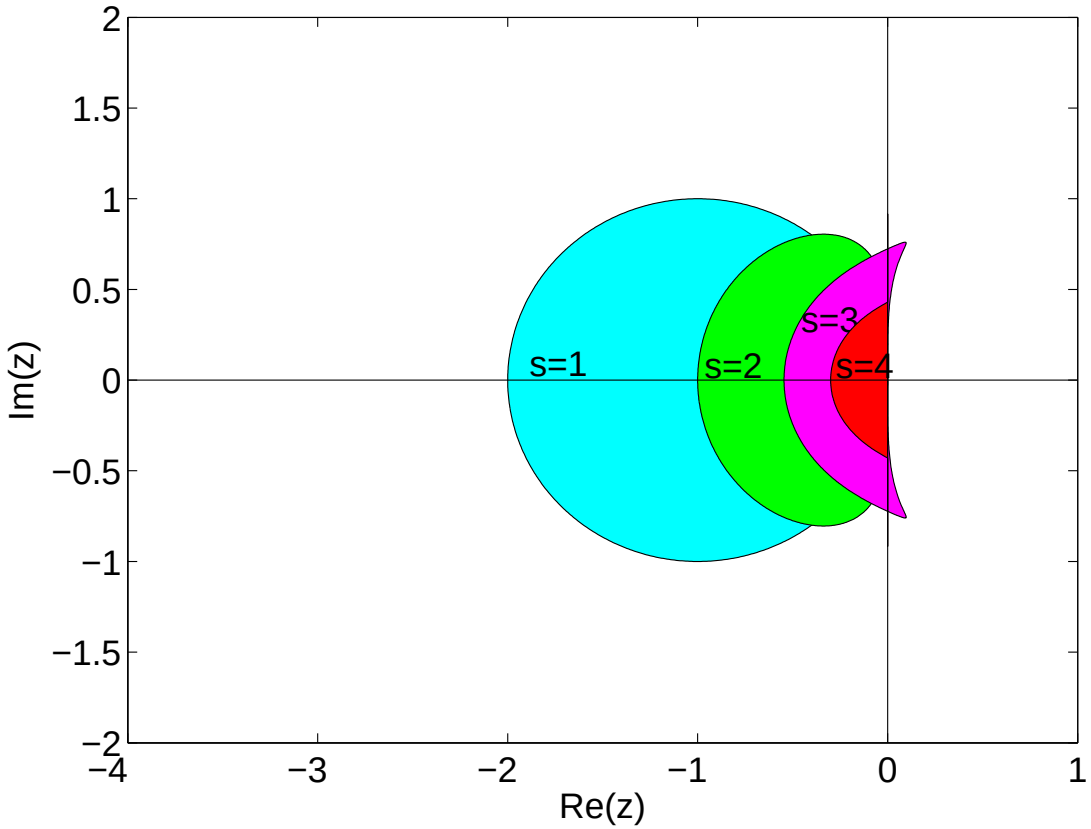
$$y_{n+1} = y_n + kf_{n+1}$$

- e.g., $s = 2$ yields

$$y_{n+1} = \frac{1}{3}[4y_n - y_{n-1} + 2kf_{n+1}]$$

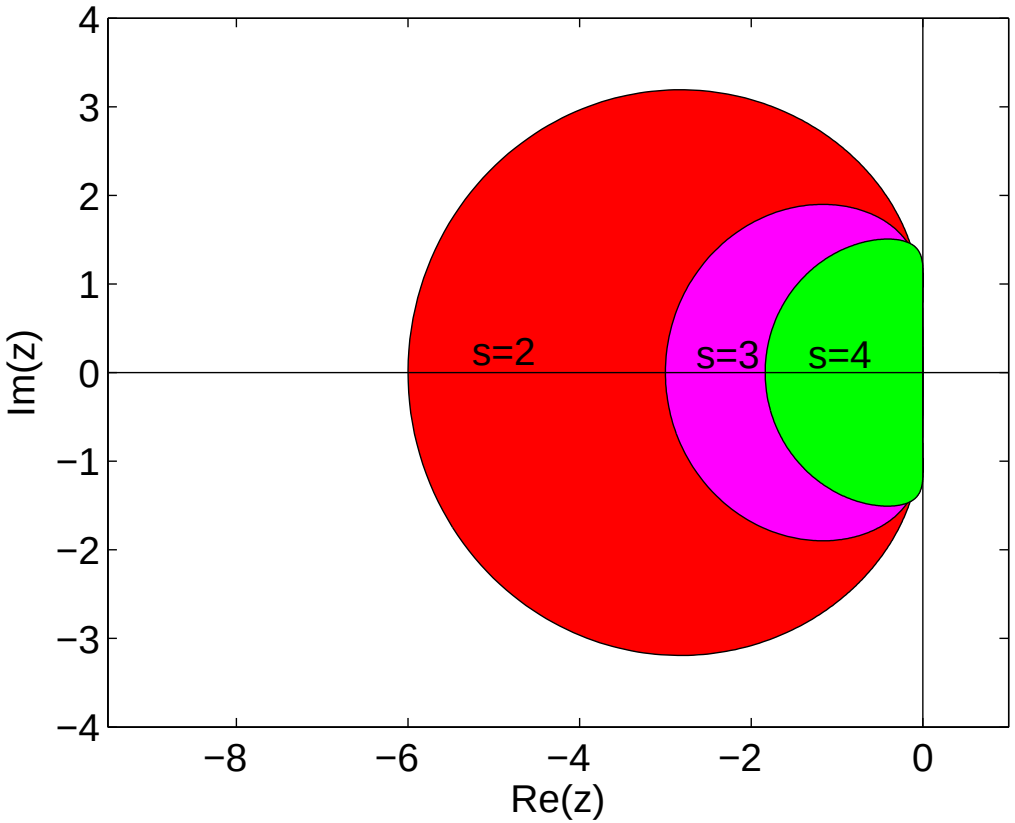
Absolute stability regions of Adams-Bashforth methods

$s = 1, 2, 3, 4$



Absolute stability regions of Adams-Moulton methods

$s = 2, 3, 4$

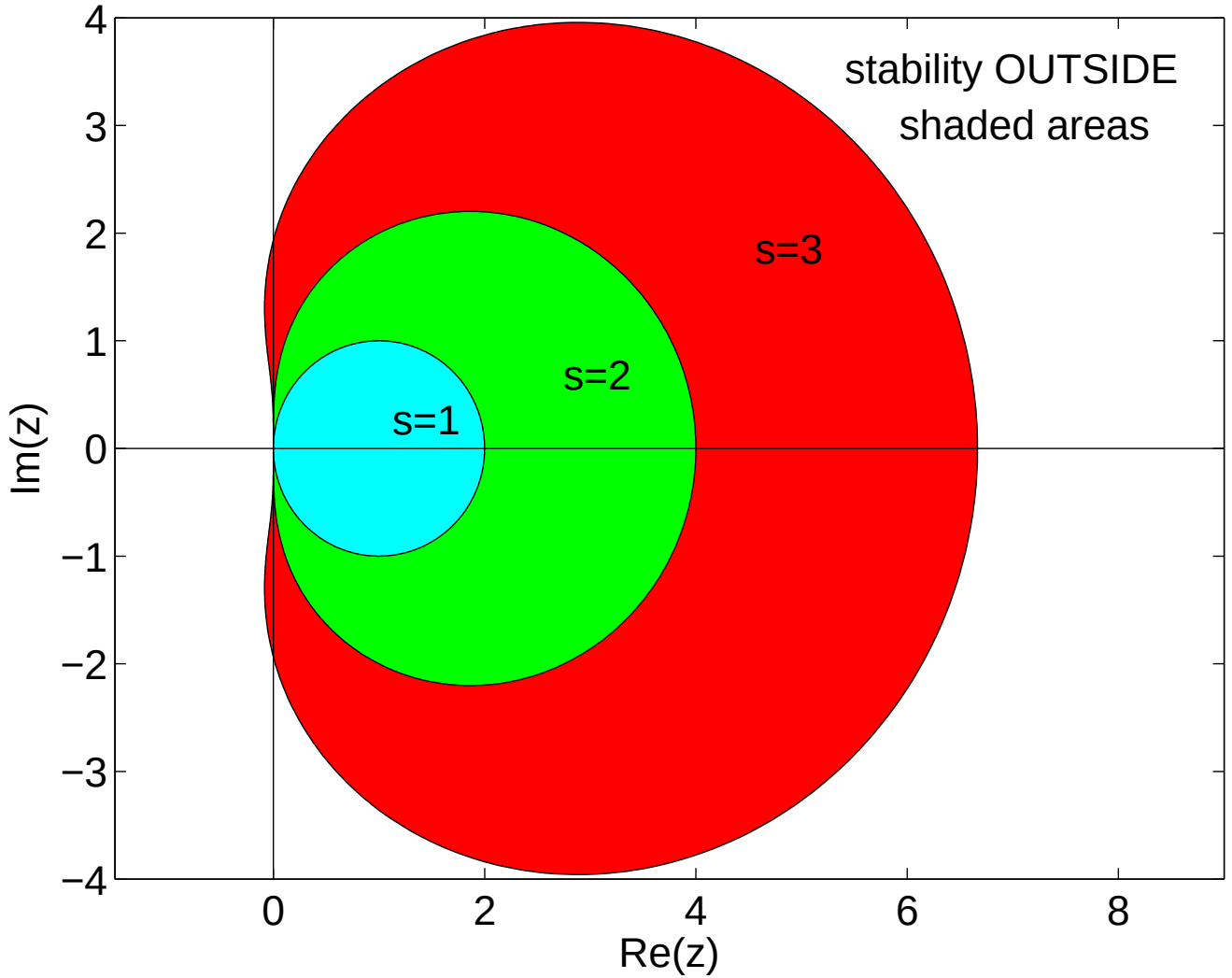


Adams methods

- Good only for nonstiff problems.
- Even then, Adams-Bashforth regions too restrictive for higher order methods.
- So, want to use Adams-Moulton: for non-stiff problems can apply fixed-point iteration.
- (i) Predict y_{n+1} using Adams-Bashforth, (ii) Evaluate $f(y_{n+1})$, (iii) Correct for better y_{n+1} using Adams-Moulton of same order or one order higher, (iv) Evaluate $f(y_{n+1})$.
- For this PECE obtain order and stability like Moulton, plus estimation of local error: good for adaptive step size selection.
- Highly effective for smooth problems.
- But incorporating events and restarts are cumbersome: for nonstiff problems, Runge-Kutta methods are currently more in vogue.

Absolute stability regions of BDF methods

$s = 1, 2, 3$



Absolute stability regions of BDF methods

- Very stable methods for $s = 1, 2, \ldots, 6$.
- Highly damping for large $k|\lambda|$. For $\Re(\lambda) \ll -1$ this is L-stability: highly desirable.
- Leading candidates for stiff problems because in addition, the nonlinear system is "only" of the size of the given ODE system.

Outline

- ODEs
- Forward and backward Euler
- Linear multistep methods
- Runge-Kutta methods
- Stiffness
- Adaptive step size selection
- (In Chapter 6: geometric integration)

Simple explicit Runge-Kutta methods

- "Bootstrap" implicit trapezoidal: use forward Euler to approximate $f(t_{n+1}, y_{n+1})$ first.

$$\begin{aligned} Y &= y_n + kf(t_n, y_n), \\ y_{n+1} &= y_n + \frac{k}{2}\Big(f(t_n, y_n) + f(t_{n+1}, Y)\Big). \end{aligned}$$

- Obtain explicit trapezoidal, a special case of an explicit Runge-Kutta method.
- Can do the same based on midpoint method:

$$\begin{aligned} Y &= y_n + \frac{k}{2} f(t_n, y_n), \\ y_{n+1} &= y_n + kf(t_n + h/2, Y). \end{aligned}$$

Explicit trapezoidal & midpoint methods

- Can write explicit trap as

$$\begin{aligned} K_1 &= f(t_n, y_n), \\ K_2 &= f(t_{n+1}, y_n + kK_1), \\ y_{n+1} &= y_n + \frac{k}{2}(K_1 + K_2). \end{aligned}$$

- Can write explicit midpoint as

$$\begin{aligned} K_1 &= f(t_n, y_n), \\ K_2 &= f(t_n + .5k, y_n + .5kK_1), \\ y_{n+1} &= y_n + kK_2. \end{aligned}$$

- Both are explicit 2-stage methods of order 2.

Explicit s-stage RK method

Method is defined by coefficients $a_{i,j},\ b_i,\quad 1 \le i \le s,\ 1 \le j \le i-1$:

$$\begin{aligned}
K_1 &= f(t_n, y_n), \\
K_2 &= f(t_n + kc_2, y_n + ka_{2,1}K_1), \\
\vdots &= \vdots \\
K_i &= f(t_n + kc_i, y_n + k\sum_{j=1}^{i-1} a_{i,j}K_j), \quad 1 \le i \le s \\
y_{n+1} &= y_n + k\sum_{i=1}^{s} b_i K_i.
\end{aligned}$$

where $c_i = \sum_{j=1}^{i-1} a_{i,j},\ 1 = \sum_{j=1}^{s} b_j$.

General s-stage RK method

- Method is defined by coefficients $a_{i,j},\ b_i,\quad 1 \le i \le s,\ 1 \le j \le s$:

$$K_i = f\left(t_n + kc_i, y_n + k\sum_{j=1}^{s} a_{i,j}K_j\right), \quad 1 \le i \le s$$

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

  where $c_i = \sum_{j=1}^{s} a_{i,j},\ 1 = \sum_{j=1}^{s} b_j$.

- Alternatively,

$$t_i = t_n + kc_i,$$

$$Y_i = y_n + k\sum_{j=1}^{s} a_{i,j} f(t_j, Y_j), \quad 1 \le i \le s$$

$$y_{n+1} = y_n + k\sum_{i=1}^{s} b_i f(t_i, Y_i).$$

General s-stage RK method

- Observe notational convention: the indices i and j in

$$\begin{aligned} t_i &= t_n + kc_i, \\ Y_i &= y_n + k\sum_{j=1}^{s} a_{i,j} f(t_j, Y_j), \quad 1 \le i \le s \\ y_{n+1} &= y_n + k\sum_{i=1}^{s} b_i f(t_i, Y_i) \end{aligned}$$

  are **internal** to current subinterval $[t_n, t_{n+1}]$. Only the end result y_{n+1} (and perhaps $f(t_{n+1}, y_{n+1})$ as well) gets reported when moving on to the next step.

- Thus, Y_i are internal stages in current subinterval $[t_n, t_{n+1}]$. Generally, Y_i approximates $y(t_i)$ to a lower order than y_{n+1} approximates $y(t_{n+1})$.

RK in tableau form

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

where $c_i = \sum_{j=1}^{s} a_{i,j}$ for $i = 1, 2, \ldots, s$.

Explicit if the matrix A is strictly lower triangular: $a_{i,j} = 0$ if $i \leq j$.

Implicit otherwise.

Necessary conditions for order p (sufficient if $p \leq 3$):

$$\mathbf{b}^T A^i C^{l-1} \mathbf{1} = \frac{(l-1)!}{(l+i)!} = \frac{1}{l(l+1)\cdots(l+i)}, \qquad 1 \leq l+i \leq p\,.$$

(For each l, $1 \leq l \leq p$, we have order conditions for $i = 0, 1, \ldots, p-l$.)

Classical 4th order RK

- The original 4-stage Runge method:

$$\begin{aligned} K_1 &= f(t_n, y_n), \\ K_2 &= f(t_n + k/2, y_n + \frac{k}{2} K_1), \\ K_3 &= f(t_i + k/2, y_n + \frac{k}{2} K_2), \\ K_4 &= f(t_{n+1}, y_n + k K_3), \\ y_{n+1} &= y_n + \frac{k}{6} (K_1 + 2K_2 + 2K_3 + K_4) . \end{aligned}$$

- Tableau form

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

RK4 compared to multistep

... for nonstiff problems... i.e., compare to a PECE method of order 4.
RK4:

- Requires no starting procedure
- Easy to change step size or accommodate special event (e.g. solution jump).
- Better absolute stability region, especially near imaginary axis.
- Requires more function evaluations.
- Showing that it is 4th order accurate is surprisingly painful.
- Harder to extend to higher order methods.

Example with known solution

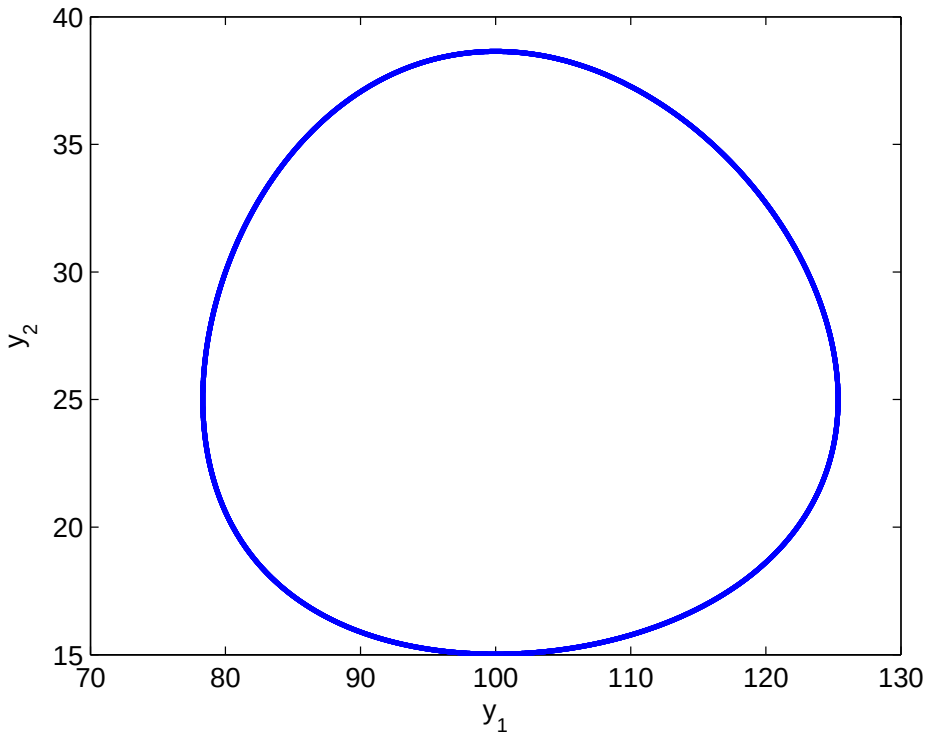
$y' = -y^2,\ y(1) = 1 \implies y(t) = 1/t.$

k	Euler	rate	RK2	rate	RK4	rate
0.2	4.7e-3		3.3e-4		2.0e-7	
0.1	2.3e-3	1.01	7.4e-5	2.15	1.4e-8	3.90
0.05	1.2e-3	1.01	1.8e-5	2.07	8.6e-10	3.98
0.02	4.6e-3	1.00	2.8e-6	2.03	2.2-11	4.00
0.01	2.3e-4	1.00	6.8e-7	2.01	1.4e-12	4.00
0.005	1.2e-4	1.00	1.7e-7	2.01	8.7e-14	4.00
0.002	4.6e-5	1.00	2.7e-8	2.00	1.9e-15	4.19

Lotka-Volterra predator-prey model

$$
\begin{aligned}
y_1' &= .25y_1 - .01y_1y_2, \quad y_1(0) = 80, \\
y_2' &= -y_2 + .01y_1y_2, \quad y_2(0) = 30.
\end{aligned}
$$

Integrating from $a = 0$ to $b = 100$ using RK4 with step size $h = 0.01$:



Lorenz equations: pocket-size chaos

$$\begin{aligned} y_1' &= \sigma(y_2 - y_1), \\ y_2' &= r y_1 - y_2 - y_1 y_3, \\ y_3' &= y_1 y_2 - b y_3, \end{aligned}$$

Parameters: $\sigma = 10,\ b = 8/3,\ r = 28$.
Initial values: $\mathbf{y}(0) = (0, 1, 0)^T$.

Run **fig2_3**

Outline

- ODEs
- Forward and backward Euler
- Linear multistep methods
- Runge-Kutta methods
- Stiffness
- Adaptive step size selection
- (In Chapter 6: geometric integration)

Absolute stability

- Convergence is for $k \to 0$, but in computation the step size is finite and fixed. How is the method expected to perform?
- Consider simple test equation for complex scalar λ (representing an eigenvalue of a system matrix)

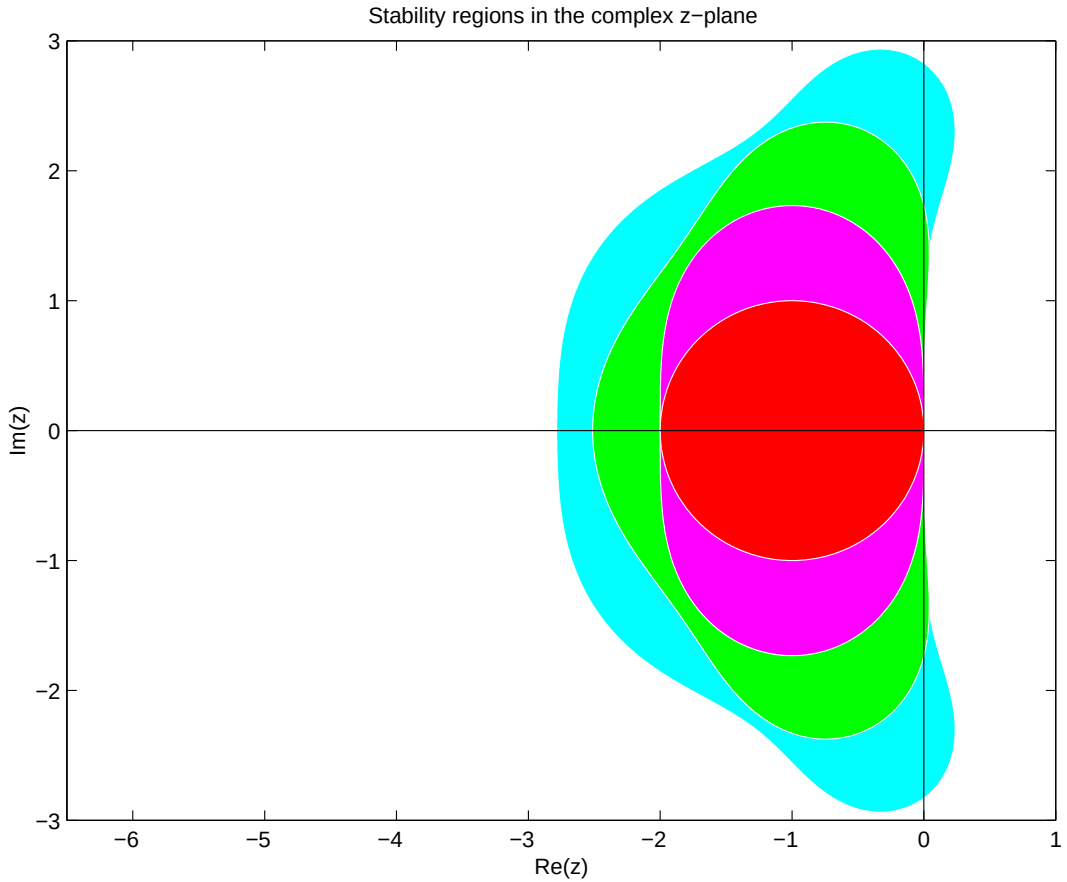
$$y' = \lambda y.$$

Solution: $y(t) = y(0)e^{\lambda t}$. So $|y(t)| = |y(0)|e^{\Re(\lambda)t}$. Magnitude increases for $\Re(\lambda) > 0$, decreases for $\Re(\lambda) < 0$.

- Let $z = k\lambda$. For one-step method (such as RK) write $y_{n+1} = R(z)y_n$. Require $|R(z)| \leq 1$ when $\Re(z) \leq 0$.
- **Example**: for Forward Euler, $R(z) = 1 + k\lambda = 1 + z$.
- So, approximate solution does not grow only if $|1 + z| \leq 1$. Important when $\Re(\lambda) \leq 0$.
- For $\lambda < 0$ must require $k \leq 2/(-\lambda)$.

Absolute stability regions: explicit RK

Four p-stage methods of order p. Red circle: forward Euler. Cyan kidney: RK4.



Simple example

- Recall

$$\mathbf{y}' = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{y}.$$

  Eigenvalues $\lambda_1 = \imath, \ \lambda_2 = -\imath$.

- For forward Euler method was shown to be unconditionally unstable.
- However, RK4 is conditionally stable! For this reason it is popular in CFD.

Stiff problem

- The ODE is stiff if an unreasonably small step size k must be used for forward Euler.
- In this case explicit methods are inadequate – resort to implicit methods.
- Method is A-stable if absolute stability region contains entire left half plane.
  Both backward Euler and implicit trapezoidal are A-stable
- Method is L-stable if $|R(z)| \to 0$ as $\Re(z) \to -\infty$.
  Backward Euler is L-stable, implicit trapezoidal is not.

Summary of 4 methods

$$y' = f(y)$$

Forward Euler

$$y_{n+1} = y_n + kf(y_n).$$

- Explicit; simple
- 1st order accurate; one-sided
- Limited utility for stiff equations; unstable for imaginary eigenvalues.

Summary of 4 methods

$$y' = f(y)$$

Backward Euler

$$y_{n+1} = y_n + kf(y_{n+1}).$$

- Implicit; simple but requires solution of algebraic equations each time step
- 1st order accurate; one-sided
- A-stable, L-stable; highly damping for imaginary eigenvalues.

Summary of 4 methods

RK4

$$
\begin{aligned}
y_{n+1} &= y_n + \frac{k}{6}[f(Y_1) + 2f(Y_2) + 2f(Y_3) + f(Y_4)], \quad \text{where} \\
Y_1 &= y_n \\
Y_2 &= y_n + \frac{k}{2} f(Y_1) \\
Y_3 &= y_n + \frac{k}{2} f(Y_2) \\
Y_4 &= y_n + k f(Y_3)
\end{aligned}
$$

- Explicit; straightforward but more expensive
- 4th order accurate; non-symmetric
- Limited utility for stiff eqns; conditionally stable for imaginary eigs.

Summary of 4 methods

$$y' = f(y)$$

Trapezoidal

$$y_{n+1} = y_n + \frac{k}{2}[f(y_n) + f(y_{n+1})].$$

- Implicit; simple but requires solution of algebraic equations each time step
- 2nd order accurate; symmetric
- A-stable; good for imaginary eigenvalues.

Adaptive step size selection

- Intuitively, want small time steps (for accuracy) where solution varies rapidly, but large steps (for efficiency) where solution varies slowly. So, $k = k_n$.
- Indeed, local truncation error d_n of a method of order p behaves like $|d_n| \sim (k_n)^p |\frac{d^{p+1}y}{dt^{p+1}}(t_n)|$. Typically, want these quantities to be roughly the same for all n.
- Ideally, we want to control the global error $e_n = y(t_n) - y_n$. However, this implies inflexibility for interactive time step estimation as the integration proceeds.
- So, estimate local error instead. At (t_n, y_n) and current step size $k = k_n$, integrate next step twice: once with method of order p, obtaining y_{n+1}, and once with method of order $p+1$, obtaining $\tilde{y}_{n+1}$. The difference $l_{n+1} = |\tilde{y}_{n+1} - y_{n+1}|$ estimates local error in y_{n+1}.
- Try to make l_{n+1}/k fall below a given tolerance by adjusting step size k (knowing p).

Pairs of methods of orders p and $p+1$

- The Adams methods provide natural pairs using the PECE arrangement.
- For BDF or Runge-Kutta, require two separate methods.
- In the RK context, seek a pair of methods that share internal stages!
- MATLAB's ode45 uses the Dormand-Prince pair of methods with $p = 4$ which requires only 6 function evaluations.
- Once step is accepted, cheat by setting $y_{n+1} = \tilde{y}_{n+1}$.
- This procedure usually works very well; however, if applied carelessly it may occasionally fail miserably in producing a qualitatively correct solution.