SF2520 — Applied numerical methods

Lecture 3

Absolute stability
Stiff problems

Olof Runborg
Numerical analysis
Department of Mathematics, KTH

2023-09-06

Today's lecture

- Computer Exercise Help Sessions
- Summary last lecture
- Absolute stability
- Stiff ODEs

Help sessions for computer exercises

- First help session on Thursday 6 Sep, kl 10-12.
- Room: "Spelhallen" (5O1Spe) in the D-building.
- Help sessions listed on Canvas (Computer exercises page). Marked "Laboration" in the KTH schedule.
- We use the electronic queueing system Stay a While.
  To ask a questions, connect to Stay a While (see link in Canvas, queue = SF2520) and type in your location and press "Join queue".
- Teachers will take your questions in turn.
- Discussion forum also open for questions.

Summary Lecture 2

- Global error defined as $e_n = u_n - y(t_n)$.
- Local truncation error (LTE) defined as residual when exact solution entered into numerical scheme.
  Example: (Explicit Euler for $y' = f(t, y)$)

$$y(t_{n+1}) = \underbrace{y(t_n) + hf(t_n, y(t_n))}_{\text{Explicit Euler with exact solution}} + \underbrace{\ell_{n+1}}_{\text{LTE}}$$

- Local truncation error analzyed/estimated by Taylor expansion
- **Theorem for one-step methods:** If local error is $\ell_n = O(h^{p+1})$ the method has order of accuracy p and the global error $e_n = O(h^p)$, i.e. one order lower. (Under mild conditions on method.)
- Intuition:
  $O(1/h)$ steps where $O(h^{p+1})$ accumulates give global error $O(h^p)$.
- Adaptivity: Choose step length h_n such that $\ell_n/h_n \sim$TOL, constant in n.

Absolute stability

Example

Want to approximate solution to

$$y' = -25y, \qquad y(0) = 1.$$

Exact solution is $y(t) = e^{-25t}$.

- Computer tests: Explicit and Implicit Euler.
- Results:
  - Explicit Euler useless for fixed $h = 0.1$. (Error $\to \infty$ when $n \to \infty$.)
  - Implicit Euler gives an ok solution for the same $h = 0.1$.
  - Both methods are convergent as $h \to 0$.
  - Explicit Euler $\approx$ Implicit Euler for $h = 0.01$.
- Need to distinguish this "good" and "bad" behaviour of convergent methods $\Rightarrow$ absolute stability concept.

Absolute stability and explicit/implicit methods

- **Explicit methods** (Expl. Euler, Heun, ...)
  Stability limit for time step $h \leq h_{\text{stab}}$. Unstable for $h > h_{\text{stab}}$, where h_{stab} depends on both method and problem.
- Computer example.
- **Implicit Methods** (Impl. Euler, Trapezoidal method, ...)
  - No stability limit. Stable for all time steps $h > 0$.
  - More expensive time stepping. In every step an equation must be solved in general, often numerically.
- **Stiff problems**
  - Stability limit $\ll$ accuracy requirement, i.e. $h_{\text{stab}} \ll h_{\text{acc}}$.
  - Explicit methods require excessively small h. Implicit methods accurate enough also for $h \gg h_{\text{stab}}$.
  - Implicit methods better: more expensive per time step, but can use fewer steps.

Absolute stability, definitions.

- An absolutely stable numerical solution should satisfy

  if $|\tilde{u}_0 - u_0|$ small enough then $|\tilde{u}_n - u_n|$ is bounded for all n,

  whenever exact solution y behaves like this. (I.e. for stable solutions.)
- This means: *The effect of a small perturbation of initial data (or later) remains small as $n \to \infty$.*
- A numerical solution that is *not* absolutely stable, typically *blows up* as $n \to \infty$. (Even if the exact solution y is bounded.)
- In practice, the numerical solution must be absolutely stable if the exact solution y is stable. Otherwise we don't get a qualitatively correct and useful approximation.
- For linear equations the absolute stability depends on the eigenvalues of the system matrix, the time step and the method. For nonlinear equations, it also depends on the solution itself.

Absolute stability – linear scalar case

Consider the "test problem"

$$y' = \lambda y, \qquad y(0) = y_0, \qquad \text{Real}(\lambda) \leq 0.$$

Exact solution is $y(t) = e^{\lambda t}$ is stable since $\text{Real}(\lambda) \leq 0$. Numerical solution must then be absolutely stable.

Example: Explicit Euler

- Let u_n and $\tilde{u}_n$ be the exact and peturbed numerical solution

$$\begin{aligned} u_{n+1} &= u_n + h\lambda u_n, & u_0 &= y_0, \\ \tilde{u}_{n+1} &= \tilde{u}_n + h\lambda \tilde{u}_n, & \tilde{u}_0 &= y_0 + \delta. \end{aligned}$$

- This gives for the difference $w_n := \tilde{u}_n - u_n$,

$$w_{n+1} = w_n + h\lambda w_n, \qquad w_0 = \delta.$$

- Iterating on this gives us

$$w_n = (1 + h\lambda) w_{n-1} = (1 + h\lambda)^2 w_{n-2} = \cdots = (1 + h\lambda)^n \delta.$$

- Hence, u_n is absolutely stable if $|1 + h\lambda| \leq 1$ since then $|\tilde{u}_n - u_n| = |w_n| \leq \delta$ is bounded as $n \to \infty$. (Small effect of δ.)

Absolute stability – Explicit Euler

For the "test problem"

$$y' = \lambda y, \qquad y(0) = y_0, \qquad \text{Real}(\lambda) \leq 0,$$

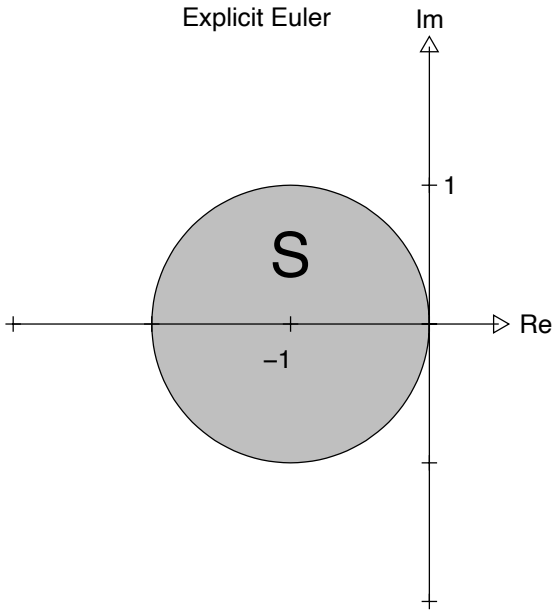
- Explicit Euler is absolutely stable iff

$$|1 + h\lambda| \leq 1.$$

- Alternatively: iff

$$h\lambda \in \mathcal{S},$$

where $\mathcal{S}$ is the region of absolute stability for Explicit Euler.



Recall earlier example ($\lambda = -25$):

- With $h = 0.1$ we have $|1 + h\lambda| = |1 - 2.5| = 1.5 > 1$, i.e. unstable.
- With $h = 0.01$ we have $|1 + h\lambda| = |1 - 0.25| = 0.75 \leq 1$, i.e. stable.
- Stability limit h_{stab} given by $|1 - 25h| \leq 1$ iff $h \leq \frac{2}{25} = 0.08 =: h_{\text{stab}}$.

Absolute stability – Implicit Euler

For the same "test problem"

$$y' = \lambda y, \qquad y(0) = y_0, \qquad \text{Real}(\lambda) \leq 0,$$

and notation $w_n := \tilde{u}_n - u_n$ we get for **Implicit Euler:**

- $w_{n+1} = w_n + h\lambda w_{n+1}$, with $w_0 = \delta$.
- Iterating on this gives us

$$w_n = \frac{w_{n-1}}{1 - h\lambda} = \frac{w_{n-2}}{(1 - h\lambda)^2} = \cdots = \frac{\delta.}{(1 - h\lambda)^n}$$

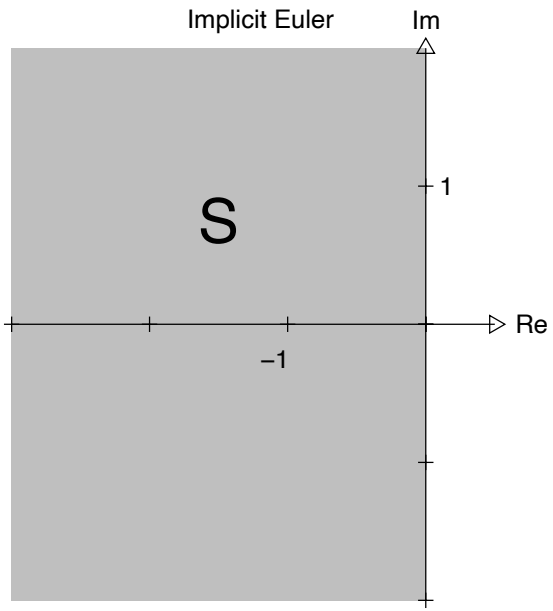
- Implicit Euler absolutely stable iff

$$|1 - h\lambda| \geq 1.$$

  which always holds when $\text{Real}(\lambda) \leq 0$!

  Recall result of example before!

- Alternatively: iff $h\lambda \in \mathcal{S}$ where $\mathcal{S}$ = the whole left half plane is the region of absolute stability for Implicit Euler.



Absolute stability – general methods

Consider again the "test problem"

$$y' = \lambda y, \qquad y(0) = y_0, \qquad \text{Real}(\lambda) \leq 0.$$

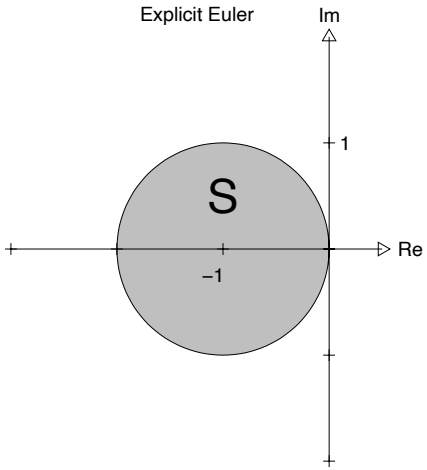
and let w_n be a numerical solution using a general method.

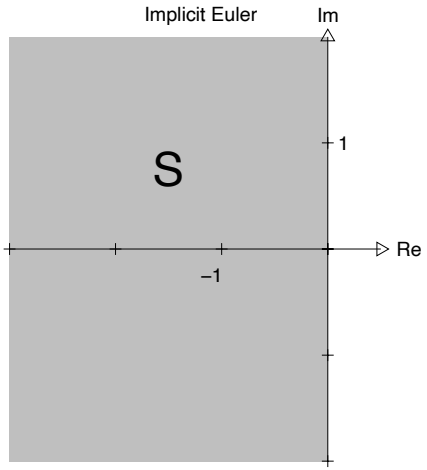
- By the same arguments as for the Euler methods, the method is absolutely stable if w_n stays bounded when $n \to \infty$.
  (Assuming also that the method is linear in u_n and u_{n+1} for linear equations.)
- We therefore define the region of absolute stability $\mathcal{S}$ for the method as those values $h\lambda \in \mathbb{C}$ for which the numerical solution of the test problem w_n is bounded as $n \to \infty$.
- Remarks:
  - The stability region depends only on the method.
    (Defined in terms of test problem, but has larger importance.)
  - The stability will always depend on the product $h\lambda$, not on h and λ individually.

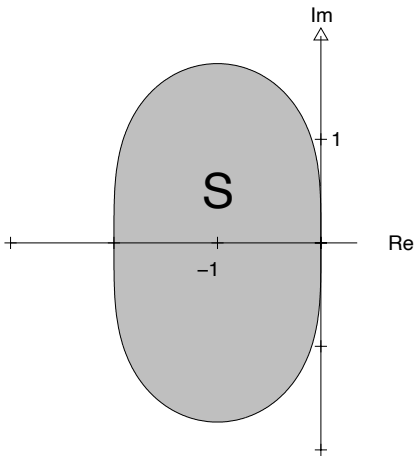
Stability regions – examples

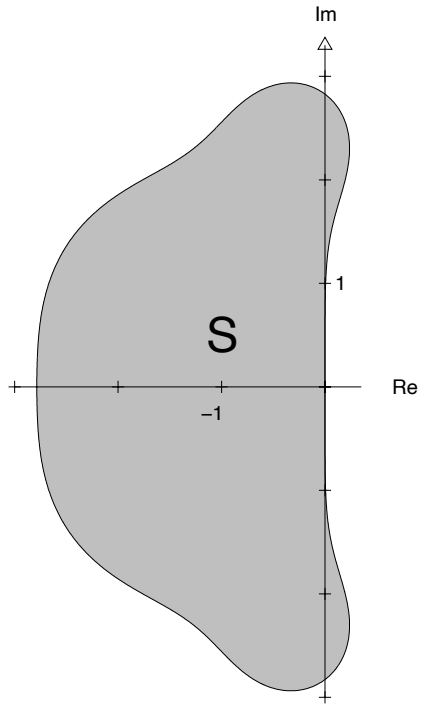
- Every ODE method has its own stability region $\mathcal{S}$.
- Explicit methods typically have a *bounded* $\mathcal{S}$ (with the origin on its boundary). This gives a stability limit for h. Since λ is fixed, we must have $h \leq h_{\text{stab}}$.
- Implicit methods typically have an *unbounded* $\mathcal{S}$. Then there is no stability limit. The method is "unconditionally stable".

  (There are examples of useful implicit methods with bounded $\mathcal{S}$.)









Absolute stability – linear systems

Consider a linear system of ODEs

$$\boldsymbol{y}' = A\boldsymbol{y} + \boldsymbol{g}(t), \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0, \qquad A \in \mathbb{R}^{d\times d}.$$

- Suppose the eigenvalues λ_k of A all satisfy $\text{Real}(\lambda_k) < 0$, so that the system is stable. ($\text{Real}(\lambda_k) = 0$ is also ok, if λ_k is simple eigenvalue.)
- Then a numerical method with stability region $\mathcal{S}$ is absolutely stable if

$$h\lambda_k \in \mathcal{S}, \qquad \text{for all eigenvalues } \lambda_k \text{ of } A.$$

  (See notes for derivations for Explicit Euler when A can be diagonalized.)
- Remarks:
  - The function $\boldsymbol{g}$ does not affect stability.
  - λ_k usually complex even if A real.

Absolute stability – variable coefficients

Consider a scalar linear ODE with variable coefficient and Explicit Euler

$$y' = a(t)y + g(t), \qquad y(0) = y_0.$$

- As before we have for $w_n = \tilde{u}_n - u_n$

$$w_{n+1} = w_n + ha(t_n)w_n, \qquad w_0 = \delta.$$

- Iterating on this

$$w_{n+1} = [1 + ha(t_n)]w_n = [1 + ha(t_n)][1 + ha(t_{n-1})]w_{n-1} = \cdots = \delta \prod_{k=0}^{n} [1 + ha(t_k)].$$

- A sufficient condition for w_n to be bounded is therefore that all factors are ≤ 1, i.e. that

$$ha(t) \in \mathcal{S} \qquad \forall t.$$

- In practice one can often allow $ha(t)$ to be outside $\mathcal{S}$ for short times.
- Similar conclusion holds for systems $\boldsymbol{y}' = A(t)\boldsymbol{y} + \boldsymbol{g}(t)$,

$$\boldsymbol{w}_{n+1} = \prod_{k=0}^{n} [I + hA(t_k)]\delta.$$

For slowly varying $A(t)$ absolute stability if $h\lambda_k(t) \in \mathcal{S}$ for all t, k.
(Fast variations can change the stability properties.)

Absolute stability – nonlinear systems

Consider a nonlinear autonomous system of ODEs

$$\boldsymbol{y}' = \boldsymbol{F}(\boldsymbol{y}), \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0.$$

- Stability theory based on linearization and small perturbations.
- Consider Explicit Euler, with $\delta \ll 1$,

$$\begin{aligned} \boldsymbol{u}_{n+1} &= \boldsymbol{u}_n + h\boldsymbol{F}(\boldsymbol{u}_n), & \boldsymbol{u}_0 &= \boldsymbol{y}_0, \\ \tilde{\boldsymbol{u}}_{n+1} &= \tilde{\boldsymbol{u}}_n + h\boldsymbol{F}(\tilde{\boldsymbol{u}}_n), & \tilde{\boldsymbol{u}}_0 &= \boldsymbol{y}_0 + \delta. \end{aligned}$$

  Then $\boldsymbol{w}_n := \tilde{\boldsymbol{u}}_n - \boldsymbol{u}_n$ satsifies (J is Jacobian of $\boldsymbol{F}$)

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + h\Big(\boldsymbol{F}(\tilde{\boldsymbol{u}}_n) - \boldsymbol{F}(\boldsymbol{u}_n)\Big) \approx \boldsymbol{w}_n + hJ(\boldsymbol{u}_n)\boldsymbol{w}_n, \qquad \boldsymbol{w}_0 = \delta.$$

- Perturbations $\boldsymbol{w}_n$ approximately governed by Explicit Euler for a linear system with variable coefficients $\boldsymbol{y}' = A(t)\boldsymbol{y}$ where $A(t) = J(u(t))$.
- Absolute stability if

$$\boldsymbol{w}_{n+1} \approx \prod_{n=0}^{N} (I + hJ(\boldsymbol{u}_n))\delta \text{ is bounded}$$

- As for systems with variable coefficients, we need $h\lambda_k(\boldsymbol{u}_n) \in \mathcal{S}$ for all k, n. (Fast variations and strong nonlinearities can change this.)

Absolute stability – summary of conditions

- Stability region $\mathcal{S}$ is defined in terms of simple test problem, but characterizes the absolute stability also for general ODEs:

ODE	Stability condition
$y' = \lambda y + g$	$h\lambda \in \mathcal{S}$
$\boldsymbol{y}' = A\boldsymbol{y} + \boldsymbol{g}(t)$	$h\lambda_k \in \mathcal{S}$ for all eigenvalues λ_k of A
$\boldsymbol{y}' = A(t)\boldsymbol{y} + \boldsymbol{g}(t)$	$h\lambda_k(t) \in \mathcal{S}$ for all t, k
$\boldsymbol{y}' = \boldsymbol{F}(\boldsymbol{y})$	$h\lambda_k(\boldsymbol{u}_n) \in \mathcal{S}$ for all eigenvalues $\lambda_k(\boldsymbol{u}_n)$ of $J(\boldsymbol{u}_n)$, the Jacobian matrix of $\boldsymbol{F}$, for all n

- Remarks
  - Stability conditions for last two cases based on approximations. Still gives good indication of when the method and solution are stable.
  - Transition between unstable and stable solutions often not as distinct and clear-cut as for the first cases. Stability properties vary with time and state of solution.

Absolute stability – conclusions

Final remarks

- Absolute stability necessary in practice. Ensures that the effect of small perturbations remains small, and that numerical solution behaves qualitatively correct.
- When exact solution is stable, absolutely stability of numerical solution means that global error is bounded by sum of local errors,

$$|e_n| \leq \sum_{k=1}^{n} |\ell_k(h)|.$$

  Accuracy indicated by how well exact solution is resolved.
- Absolute stability depends on the method and $h\lambda_k$ for linear systems, and additionally on the solution for nonlinear systems.
- For explicit methods there is typically a clear stability limit h_{stab} for the time step h, giving the requirement $h < h_{\text{stab}}$.
- The fact that a method is absolutely stable for a certain time step does not mean that the solution is accurate. The time step must be chosen such that the method is both stable and accurate.

Stiff ODEs – example

- Scalar ODE

$$y' = -y, \qquad y(0) = 1, \qquad y(t) = e^{-t}.$$

- Stiff system of ODEs

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}' = \begin{pmatrix} -1000 & 999 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad y_1(0) = y_2(0) = 1,$$

  with same solution componentwise: $y_1(t) = y_2(t) = e^{-t}$.

- Same exact solutions. Same local truncation errors (LTE) and h_{acc}.
- Computer example: System requires much smaller h than scalar ODE.
- Very different stability limits (Expl. Euler):
  Scalar ODE: $h_{\text{stab}} = 2$, System: $h_{\text{stab}} = 0.002$.
- General solution of system is

$$y(t) = \alpha_1 \begin{pmatrix} 1 \\ 1 \end{pmatrix} e^{-t} + \underbrace{\alpha_2 \begin{pmatrix} 1 \\ 0 \end{pmatrix} e^{-1000t}}_{\text{Very small after short time.}} .$$

  Second term irrelevant for LTE, but important for stability.

Note: In exact arithmetic we have $\alpha_2 = 0$. Still, term important due to perturbations.

Stiff ODE

- In stiff problems explicit methods require much smaller timestep h than expected based on exact solution and LTE, because of stability requirement: $h \leq h_{\text{stab}} \ll h_{\text{acc}}$.
- Explicit methods become expensive as very many steps are needed, and "unnecessarily" accurate.
- Implicit methods win:
  - Every step more expensive, ...
  - ... but fewer steps needed since no stability limit.
- Common situation in e.g. diffusion and heat transfer (parabolic PDEs) and chemical reactions (Robertson problem).
- Underlying reason: ODE contains very different time scales $\Rightarrow$ eigenvalues of very different sizes (in linear problems):

$$\boldsymbol{y}(t) = \underbrace{\alpha_1 \boldsymbol{v}_1 e^{\lambda_1 t} + \alpha_2 \boldsymbol{v}_2 e^{\lambda_2 t} +}_{\text{Slow time scales determine solution shape and LTE}} \cdots \underbrace{+\alpha_{d-1} \boldsymbol{v}_{d-1} e^{\lambda_{d-1} t} + \alpha_d \boldsymbol{v}_d e^{\lambda_d t}}_{\text{Fast time scales set stability limit}}$$

  (Here we assume $\lambda_d \leq \lambda_{d-1} \leq \cdots \ll \lambda_2 \leq \lambda_1 < 0$.)
- Stiffness sometimes measured by "stiffness ratio": $|\lambda_{\max}|/|\lambda_{\min}|$.

Matlab

Matlab for stiff systems $\boldsymbol{y}' = \boldsymbol{F}(t, \boldsymbol{y})$

```
>> [t,Y] = ode23s(F, [0 T], Y0);
```

- Adaptive ("almost") implicit method. Arguments and output as `ode45`.
- Uses only one fixed Newton iteration to solve equation in each step. (Rosenbrock method.)
- Computes an approximation of the Jacobian by numerical differentiation of `F` – expensive if systems is large!
- By setting various options this cost can be reduced considerably:
  - `Jacobian` – Handle to a function returning the Jacobian, or just a constant matrix if independent of time (as in LCC systems). Best option if possible.
  - `Jpattern` – Give Jacobian sparsity pattern. Good if Jacobian is sparse.
- **Example:** To solve $\boldsymbol{y}' = A\boldsymbol{y}$ one could do (here the Jacobian is A):

```
>> options = odeset('Jacobian',A);
>> [t, Y] = ode23s(@(t,y) A*y, [0 T], Y0, options);
```