# CHAPTER 3: *Basic methods, basic concepts*

Concentrate on 3 methods

- Forward Euler, (or just *Euler's method*)
- Backward Euler, (a.k.a. implicit Euler)
- Trapezoidal, (a.k.a. implicit mid-point)

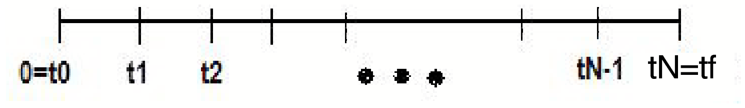
for solving IVPs

$$\begin{aligned} \dot{\mathbf{y}} &= \mathbf{f}(t, \mathbf{y}), \qquad 0 \le t \le t_f, \\ \mathbf{y}(0) &= \mathbf{y}_0, \end{aligned}$$

- Assume unique solution and as many bounded derivatives as needed.
- Can think in terms of scalar ODE, but vector interpretation often possible.

## 3.1 Forward Euler

Imagine discretizing $[0, t_f]$ by a mesh



Define $\quad \Delta t_n = t_n - t_{n-1}, \quad n^{th}$ step size,
(size of interval $n$)
$n = 1, 2, \ldots, N,$

we then compute

$$\text{IC} \ = \mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_{N-1}, \mathbf{y}_N,$$

where

$$\mathbf{y}_n \approx \mathbf{y}(t_n).$$

$\rightarrow$ Given only $\mathbf{y}_0$, generate $\mathbf{y}_1, \mathbf{y}_2, \cdots$.

## Review: Order notation

We often describe computational errors as a function of $\Delta t$ as $\Delta t \to 0 \quad (\Delta t > 0)$.

**Definition 1.**

$$\mathbf{d} = \mathcal{O}((\Delta t)^p)$$

*if* $\exists\ p, C > 0$ *such that* $\forall \Delta t > 0$ *sufficiently small,*

$$\|\mathbf{d}\| \leq C(\Delta t)^p.$$

*Typically, we are interested in the largest* $p$ *for which this is true; i.e.,*

$$\rightarrow \frac{\|\mathbf{d}\|}{(\Delta t)^p} \approx C \leftrightarrow \|\mathbf{d}\| \text{ decreases like } (\Delta t)^p \text{ as } \Delta t \to 0^+.$$

*In estimating computational complexity, we assume* $N = \mathcal{O}(\frac{1}{\Delta t}),\ N \to \infty$*.*
*e.g.,*

$$w = \mathcal{O}(N \log N)$$

$\Rightarrow \exists\, C$ *such that*

$$w \leq C N \log N \quad \textit{as} \quad N \to \infty.$$

Consider Taylor's expansion:

$$\begin{aligned}\mathbf{y}(t_n) &= \mathbf{y}(t_{n-1}) + \Delta t_n \dot{\mathbf{y}}(t_{n-1}) + \frac{1}{2!}(\Delta t_n)^2 \ddot{\mathbf{y}}(t_{n-1}) + \cdots \\ &= \mathbf{y}(t_{n-1}) + \Delta t_n \dot{\mathbf{y}}(t_{n-1}) + \mathcal{O}((\Delta t_n)^2).\end{aligned}$$

Assuming $\mathcal{O}((\Delta t_n)^2)$ can be neglected ...

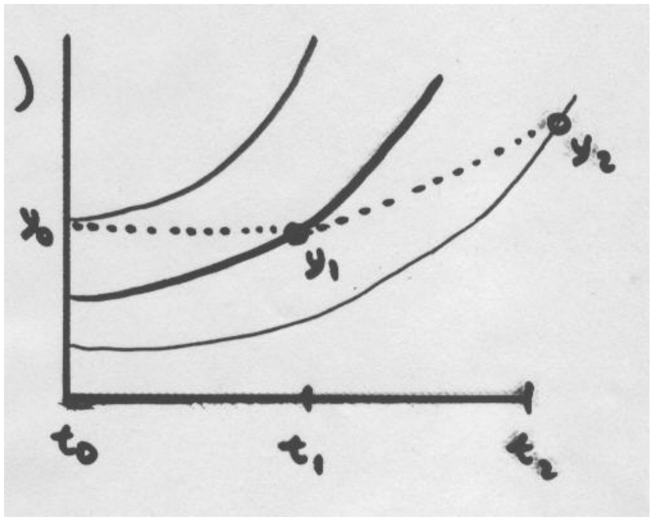
$$\mathbf{y}_n = \mathbf{y}_{n-1} + \Delta t_n \mathbf{f}(t_{n-1}, \mathbf{y}_{n-1}).$$

This is forward Euler !

**Note 1.** • *This is a explicit method:*
*$\mathbf{y}_n$ is given as an explicit function of past $\mathbf{y}$ values.*

- *This is a one-step method:*
  *The only quantities that appear are $\mathbf{y}_{n-1}, \mathbf{y}_n$.*

- *It has a nice geometric interpretation:*



*Follow tangent line at $(t_{n-1}, \mathbf{y}_{n-1})$ for a horizontal distance $\Delta t_n$.*
*Repeat as desired.*

## 3.2 Convergence, Accuracy, Consistency, 0-stability

Rewrite forward Euler

$$\frac{\mathbf{y}_n - \mathbf{y}_{n-1}}{\Delta t_n} - \mathbf{f}(t_{n-1}, \mathbf{y}_{n-1}) = \mathbf{0}.$$

Define

$$\mathcal{N}_h \mathbf{u}(t_n) \equiv \frac{\mathbf{u}(t_n) - \mathbf{u}(t_{n-1})}{\Delta t_n} - \mathbf{f}(t_{n-1}, \mathbf{u}(t_{n-1}))$$

for any function $\mathbf{u}(t)$ defined at the mesh points $\{t_n\}_{n=0}^{N}$ with $\mathbf{u}(t_0)$ given.

Consider a function $\mathbf{y}_h(t)$ (mesh function)
such that $\mathbf{y}_h(t_n) = \mathbf{y}_n.$

Clearly, $\mathcal{N}_h \mathbf{y}_h(t_n) = \mathbf{0}.$

Define local truncation error

$$\mathbf{d}_n = \mathcal{N}_h \mathbf{y}(t_n).$$

This is the amount by which the true solution fails to satisfy the difference equation.

$\leftrightarrow$ Measures how closely the difference operator approximates the differential operator.

A difference method is consistent (or accurate) of order $p$ if

$$\mathbf{d}_n = \mathcal{O}((\Delta t_n)^p)$$

for a positive integer $p$.

For forward Euler,

$$\mathbf{d}_n = \frac{\Delta t_n}{2} \ddot{\mathbf{y}}(t_n) + \mathcal{O}((\Delta t_n)^2). \qquad \text{(verify !)}$$

$\rightarrow$ Forward Euler is accurate of order 1.

It is easy to design difference approximations to be consistent. But the property we really want is convergence $\leftrightarrow$ consistency over many steps.

Let

$$\Delta t = \max_{1 \le n \le N} \Delta t_n$$

and assume $N\Delta t$ is bounded independent of $N$.

A difference method is convergent of order $p$ if the global error $\mathbf{e}_n = \mathbf{y}(t_n) - \mathbf{y}_n$, $\mathbf{e}_0 = \mathbf{0}$ satisfies $\mathbf{e}_n = \mathcal{O}((\Delta t)^p)$ for $n = 1, 2, \ldots, N$.

**Note 2.** *The order of consistency and convergence do not have to be equal.*

We would like to assume they are.

For that, we need the concept of 0-stability.

$\uparrow$

zero

**Definition 2.** *A difference method is 0-stable if* $\exists\ \Delta t_0, K > \mathbf{0}$ *such that for any mesh functions* $\mathbf{x}_h, \mathbf{z}_h$ *with* $\Delta t \leq \Delta t_0$

$$\|\mathbf{x}_n - \mathbf{z}_n\| \leq K\{\|\mathbf{x}_0 - \mathbf{z}_0\| + \max_{1\leq j\leq n} \|\mathcal{N}_h\mathbf{x}_h(t_j) - \mathcal{N}_h\mathbf{z}_h(t_j)\|\}, \quad 1 \leq n \leq N.$$

$\mathbf{x}_n \leftrightarrow$ *Method in question to produce* $\mathbf{y}_n$.

$\mathbf{z}_n \leftrightarrow$ *Method in question with perturbed initial condition.*

$\rightarrow$ Analogous to stability of differential equation.

0-stability $\leftrightarrow$ Stability of difference equation.

$\rightarrow$ Concept has limited use in proofs.
$\rightarrow$ Tricky to directly prove forward Euler is 0-stable.

**Theorem 1.** $\underset{\textit{order } p}{\textit{Consistency}} + \textit{0-stability} \Rightarrow \underset{\textit{order } p}{\textit{Convergence}}$

$$\|\mathbf{e}_n\| \leq K \max_{1\leq j\leq n} \|\mathbf{d}_j\| = \mathcal{O}((\Delta t)^p).$$

$\rightarrow$ As an error bound, this is very pessimistic and cannot be used for practical error estimation.

- Another related error measure is the local error

  $$\uparrow$$

  the error made at each step

  Let

  $$\begin{aligned} \dot{\bar{\mathbf{y}}}(t) &= \mathbf{f}(t, \bar{\mathbf{y}}(t)), \\ \bar{\mathbf{y}}(t_{n-1}) &= \mathbf{y}_{n-1}. \end{aligned}$$

  Then the local error is

  $$\mathbf{l}_n = \bar{\mathbf{y}}(t_n) - \mathbf{y}_n.$$

  Usually, $\|\mathbf{d}_n\| = \|\mathcal{N}_h \bar{\mathbf{y}}(t_n)\| + \mathcal{O}((\Delta t)^{p+1}),$

  and $\Delta t_n \|\mathcal{N}_h \bar{\mathbf{y}}(t_n)\| = \|\mathbf{l}_n\| \big(1 + \mathcal{O}(\Delta t_n)\big).$

  $\rightarrow \Delta t_n \mathbf{d}_n, \mathbf{l}_n$ are closely related !

## 3.3 Absolute stability

Recall the test equation:

$$\begin{aligned} \dot{y} = \lambda y \qquad & \text{(scalar)} \\ \lambda - & \text{complex} \\ y(0) = y_0 > 0 \qquad & \text{(for convenience)} \end{aligned}$$

Exact solution: $y(t_n) = e^{\lambda t_n} y_0$.

Consider forward Euler with fixed step size $\Delta t_n = \Delta t$:

$$\begin{aligned} y_n &= y_{n-1} + \Delta t \lambda y_{n-1} \\ &= (1 + \Delta t \lambda) y_{n-1} \\ &\vdots \\ &= (1 + \Delta t \lambda)^n y_0 \end{aligned}$$

Three cases :

- $\mathcal{R}e\lambda > 0$ : $\|y(t)\| = y_0 e^{(\mathcal{R}e\lambda)t} \to \infty$ as $t \to \infty$.
  $\to$ Problem is unstable.
  If $e^{(\mathcal{R}e\lambda)t_f}$ is not too large, we can compute meaningful solutions in a relative sense.

- $\mathcal{R}e\lambda = 0$: distance between solution curves is constant.

- $\mathcal{R}e\lambda < 0$ : $\|y(t)\| = y_0 e^{(\mathcal{R}e\lambda)t} \to 0$ as $t \to \infty$.
  Solution is asymptotically stable.
  $\to$ Absolute stability requirement:

$$\|y_n\| \leq \|y_{n-1}\|, \quad n = 1, 2, \cdots .$$

**Definition 3.** *The region of absolute stability of a numerical method is the region in the complex z-plane where*

$$\|y_n\| \leq \|y_{n-1}\|$$

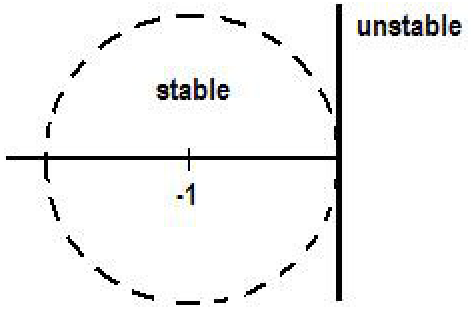
*for the test equation $\dot{y} = \lambda y$ and $z = \lambda \Delta t$.*

**Example 1.** *For forward Euler,*

$$\begin{aligned}\|y_n\| \le \|y_{n-1}\| \Rightarrow |1+\Delta t\lambda| &\le 1\\ \Rightarrow |1+z| &\le 1.\end{aligned}$$

$$\downarrow$$

*circle centred* $(-1,0)$ *and radius* $1$



*Suppose* $\lambda$ *is a real negative number.*
*Then for stability, we must restrict* $\Delta t$ *such that*

$$\Delta t \le \frac{2}{-\lambda}. \qquad \textit{(verify)}$$

Exercise: For $\lambda = -200$, use forward Euler to solve $\dot{y} = \lambda y,\ y(0) = 1$ with $\Delta t = 0.011,\ 0.0099,\ 0.0049$ for 100 steps each compare with the exact answer. Comment on the difference between the last 2 solutions.

The absolute stability restriction is a stability restriction NOT an accuracy restriction !

e.g. If $y_0 = 10^{-15}$, then the approximation $y_n \equiv 0$ never has error larger than $10^{-15}$.
Because roundoff errors inevitably occur, if you use a stepsize $\Delta t$ outside of the stability region, the numerical solution will blow up !

- For systems of linear, constant-coefficient equations, the stability restriction is given by the eigenvalue with the most negative real part.

## 3.4 Stiffness and Backward Euler

Important rule of thumb:

We want to choose $\Delta t_n$ based on accuracy requirements NOT stability requirements.

When we cannot do this the problem is called stiff.

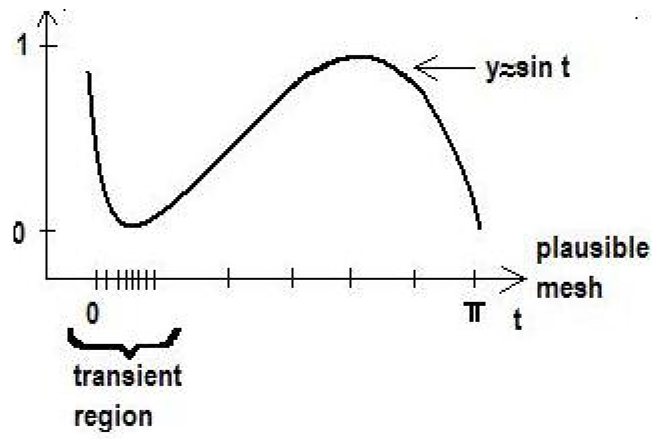
What does this mean ?

- A given tolerance (accuracy) requires a certain $\Delta t_n^{(1)}$.
- Stability restriction also imposes a certain $\Delta t_n^{(2)}$.
- For stiff problems, $\Delta t_n^{(2)} \ll \Delta t_n^{(1)}$;
  i.e., you get much more accuracy than you ask for.

  What's wrong with that ? It's not for free !

**Example 2.**

$$\begin{aligned} \dot{y} &= -100(y - \sin t), \qquad t \geq 0, \\ y(0) &= 1. \end{aligned}$$



*Rapid variation at beginning requires small step. But later, solution is smooth, so we would like to take a large step.*

Other scientists and engineers try to quantify stiffness in terms of multiple scales; i.e., eigenvalues (time constants) have widely differing values.

Then $\Delta t_n$ is restricted by the transients, even after they have died off !

The best way to understand stiffness is in a qualitative sense:

Stiffness is characterized in terms of the behaviour of an explicit method (like forward Euler) on a given problem.

An IVP is stiff in some interval $[0, t_f]$ if the stepsize needed to maintain stability is much smaller than that needed to meet the accuracy requirements.

**Note 3.** *Stiffness depends on*

- *the IVP (DE, ICs, $[0, t_f]$),*
- *the accuracy requirement,*
- *the absolute stability region of the method.*

$\rightarrow$ If the tolerance is small enough, no problem is stiff!

Example 2 is stiff after about $t = 0.03$.

## 3.4.1 Backward Euler

We would like a method with a nice absolute stability region so that we can take a large $\Delta t$ even when the problem is stiff.

Such a method is backward Euler.

It can be derived like forward Euler, but with Taylor expansions about $t = t_n$.

This leads to:

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

**Note 4.**

- *This is a first-order method.* *(verify)*
- *Geometrically, the tangent is drawn from the future point $(t_n, \mathbf{y}_n)$.*

- *It is an implicit method.*
  *$\rightarrow$ The unknown $\mathbf{y}_n$ is on both sides of the equation. So we need to solve a nonlinear system of equations at each step.*
  *$\rightarrow$ Each step costs more than a forward Euler step.*

- *Stability region: applying backward Euler to $\dot{y} = \lambda y$,*

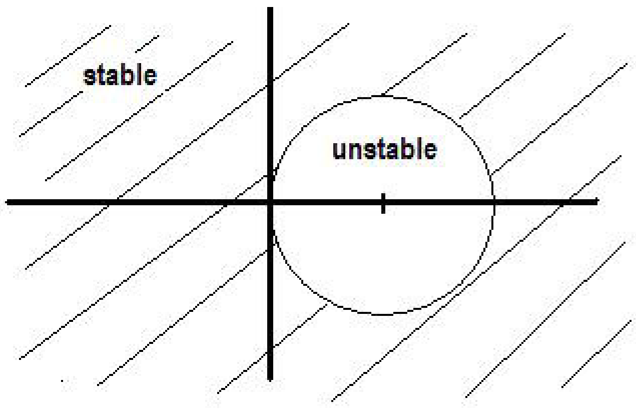
$$\begin{aligned} y_n &= y_{n-1} + \lambda \Delta t y_n, \\ \Rightarrow \quad y_n &= \frac{y_{n-1}}{1 - \lambda \Delta t}. \end{aligned}$$

  $\frac{1}{1-\lambda\Delta t}$ *is the amplification factor for backward Euler.*
  *( Recall: for forward Euler, it was $(1 + \lambda \Delta t)$. )*

  *For $\Delta t > 0$ and $\mathcal{R}e(\lambda) \leq 0$, we have*

$$\frac{1}{|1 - \lambda \Delta t|} \leq 1.$$

$\rightarrow$ This method is unconditionally stable !


stable
unstable

## 3.4.2 Solving nonlinear equations

For any implicit method, equations need to be solved at every step.

(Not a recipe anymore !)

If the equations are linear, specialized techniques may be used (e.g., Gauss elimination).

Usually the equations are nonlinear.

We will discuss two methods:

- Functional (or fixed-point) iteration
- Newton iteration

- Functional iteration
  Guess $\mathbf{y}_n^{(0)} = \mathbf{y}_{n-1}$,

  then iterate

$$\mathbf{y}_n^{(\nu+1)} = \mathbf{y}_{n-1} + \Delta t_n \mathbf{f}(t_n, \mathbf{y}_n^{(\nu)}) \quad \nu = 0, 1, 2, \ldots .$$

  <u>Advantage</u>: simple !

  <u>Disadvantage</u>: Theory tells us that for functional iteration to converge, we must have

$$\Delta t \left\| \frac{\partial \mathbf{f}}{\partial \mathbf{y}} \right\| < 1.$$

  But for stiff problems $\|\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\|$ is large.

  $\rightarrow$ You have to make $\Delta t$ small !
  (This defeats the purpose !)

  Functional iteration is used for implicit (predictor-corrector) methods applied to non-stiff problems.

**Example 3.**

$$\begin{aligned}\dot{y} &= \lambda\left(ty^2 - \frac{1}{t}\right) - \frac{1}{t^2}, \quad t > 1,\\ y(1) &= 1, \quad \lambda < 0.\end{aligned}$$

*Exact solution:* $y(t) = \frac{1}{t}$.
*Apply backward Euler:*

$$y_n = y_{n-1} + \Delta t_n \left[\lambda\left(t_n y_n^2 - \frac{1}{t_n}\right) - \frac{1}{t_n^2}\right].$$

*Solve this equation by functional iteration:*

$$y_n^{(\nu+1)} = y_{n-1} + \Delta t_n \left[\lambda\left(t_n (y_n^{(\nu)})^2 - \frac{1}{t_n}\right) - \frac{1}{t_n^2}\right],$$

$$\nu = 0, 1, \ldots.$$

*Under what conditions will this iteration converge rapidly?*

*Define the error at iteration* $(\nu+1)$ *to be*

$$\begin{aligned}
\epsilon_n^{(\nu+1)} &= y_n - y_n^{(\nu+1)}. \\
\textit{Then} \quad \epsilon_n^{(\nu+1)} &= \Delta t_n \lambda t_n \big(y_n^2 - (y_n^{(\nu)})^2\big) \qquad \textit{(verify)} \\
&= \Delta t_n \lambda t_n \big(y_n + y_n^{(\nu)}\big) \epsilon_n^{(\nu)} \\
&\approx 2\Delta t_n \lambda \epsilon_n^{(\nu)}. \quad \left( \textit{use } y_n \approx y_n^{(\nu)} \approx \frac{1}{t_n}. \right)
\end{aligned}$$

$\rightarrow$ *Iteration will converge if*

$$|2\Delta t_n \lambda| < 1 \quad \textit{or} \quad \Delta t_n < \frac{1}{2|\lambda|}.$$

*e.g., if* $\lambda = -500, \quad \Delta t_n < 0.001.$

*Stepsize will likely be restricted due to stability, not accuracy !*

$\rightarrow$ *We don't want that !*

- Newton iteration (some review !)

  For a scalar nonlinear equation

$$g(x) = 0,$$

  given an initial guess $x_0$,
  we produce a sequence of iterates

$$x^{(\nu+1)} = x^{(\nu)} - \frac{g(x^{(\nu)})}{g'(x^{(\nu)})}, \quad \nu = 1, 2, \ldots .$$

  For a system of nonlinear equations,

$$\mathbf{g}(\mathbf{x}) = \mathbf{0},$$

  this generalizes to

$$\mathbf{x}^{(\nu+1)} = \mathbf{x}^{(\nu)} - \left[ \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}^{(\nu)}} \right]^{-1} \mathbf{g}(\mathbf{x}^{(\nu)}), \ \nu = 1, 2, \ldots .$$

  **Note 5.** *It is bad practice to compute inverses!*

Instead solve the linear system for the update $\boldsymbol{\delta}^{(\nu)}$:

$$\left.\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^{(\nu)}} \boldsymbol{\delta}^{(\nu)} = -\mathbf{g}(\mathbf{x}^{(\nu)}),$$

then update: $\mathbf{x}^{(\nu+1)} = \mathbf{x}^{(\nu)} + \boldsymbol{\delta}^{(\nu)}$.

Variants of Newton's method are used in virtually all modern stiff ODE codes.

For backward Euler,

$$\mathbf{g}(\mathbf{y}_n) = \mathbf{y}_n - \mathbf{y}_{n-1} - \Delta t_n \mathbf{f}(t_n, \mathbf{y}_n) = \mathbf{0},$$

leading to the Newton iteration

$$\mathbf{y}_n^{(\nu+1)} = \mathbf{y}_n^{(\nu)} - \left[\mathbf{I} - \Delta t_n \frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right]^{-1}_{\mathbf{y}=\mathbf{y}_n^{(\nu)}} \left(\mathbf{y}_n^{(\nu)} - \mathbf{y}_{n-1} - \Delta t_n \mathbf{f}(t_n, \mathbf{y}_n^{(\nu)})\right), \quad \nu = 0, 1, 2, \ldots.$$

$\left[\mathbf{I} - \Delta t_n \frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right]_{\mathbf{y}=\mathbf{y}_n^{(\nu)}}$ : iteration matrix

The cost of forming and solving the linear systems (for $\boldsymbol{\delta}^{(\nu)}$!) is the dominant cost in an implicit solver.

We will iterate until

$$\|\mathbf{y}^{(\nu+1)} - \mathbf{y}^{(\nu)}\| \leq NTOL.$$

$NTOL$: specified by the user,
well above roundoff error.

We can take as initial guess

$$\mathbf{y}_n^{(0)} = \mathbf{y}_{n-1}.$$

$\rightarrow$ It is sometimes possible to do better.

Because this is such a good guess, convergence can occur in only a few Newton iterations.
Software can be designed so that if convergence does not occur quickly, $\Delta t_n$ can be decreased.

- Many other tricks go into practical Newton codes; e.g., damped Newton
$$\mathbf{x}^{(\nu+1)} = \mathbf{x}^{(\nu)} + \rho\, \boldsymbol{\delta}^{(\nu)}, \quad 0 < \rho \le 1.$$
Frozen Jacobian:
$\rightarrow$ do not update $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ at each iteration (or even each step !)
Then each iteration costs $\mathcal{O}(m^2)$, not $\mathcal{O}(m^3)$.
Review: Matrix decompositions !

- Approximating the Jacobian matrix
In real applications, ODE systems are often large and complicated.
This makes the computation of $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ a difficult and error-prone task.
A convenient technique is to use difference approximations to automate this process.

Given $\mathbf{y}^{(\nu)}$, perturb *one component* as follows
$$\hat{y}_j = y_j + \epsilon, \quad \tilde{y}_j = y_j - \epsilon, \quad 0 < \epsilon \ll 1.$$

Evaluate $\hat{\mathbf{f}} = \mathbf{f}(t_n, \hat{\mathbf{y}}), \quad \tilde{\mathbf{f}} = \mathbf{f}(t_n, \tilde{\mathbf{y}})$.

Then the $j^{th}$ column of $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ is approximated by

$$\frac{\partial \mathbf{f}}{\partial y_j} \approx \frac{1}{2\epsilon}(\hat{\mathbf{f}} - \tilde{\mathbf{f}}).$$

– How do you choose $\epsilon$ ?
If computer has $2d$ significant digits,
choose $\epsilon = 10^{-d}$
e.g., a good choice in double precision is
$\epsilon = 10^{-7}$

**Note 6.** – *This strategy is not foolproof !*

– *It may be very expensive*
*(especially if $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ has many zeros (sparse)).*

– *More sophisticated and reliable software exists*
*$\rightarrow$ Basic question: How to choose $\epsilon$.*
*Also automatic differentiation software.*

– *Most general-purpose software has an option for finite-difference Jacobians.*

– *Good as a check for obvious programming errors!*

## 3.5 A-Stability and Stiff Decay

The perfect world: the numerical method mimics all properties of the DE for all DEs.

The real world: methods that capture essential properties for a class of DEs.

For all stable solutions to the test equation,

$$|y(t_n)| \leq |y(t_{n-1})|.$$

$\rightarrow$ Numerical method should satisfy

$$|y_n| \leq |y_{n-1}|.$$

This leads to the concept of A-stability.

**Definition 4.** *A numerical method is A-stable if its region of absolute stability contains the entire left-half of the complex $z$-plane ($z = \lambda \Delta t$).*
*e.g., backward Euler is A-stable.*

But there are two problems with this definition:

- No distinction made between cases

  $$\mathcal{R}e(\lambda) \ll -1$$

  and

  $$-1 \ll \mathcal{R}e(\lambda) \leq 0, \quad |\mathcal{I}m(\lambda)| \gg 1.$$

  The latter cases gives rise to a highly oscillatory exact solution that does not decay much.
  $\rightarrow$ This has not mattered to us so far.

- In the stiff limit $\mathcal{R}e(\lambda) \ll -1$,

  $$|y(t_n)| \ll |y(t_{n-1})|.$$

  But absolute stability only requires

  $$|y_n| \le |y_{n-1}|.$$

  This is too weak sometimes !

  In particular, it allows $\qquad |y_n| \approx |y_{n-1}|.$

Consider a sightly generalized test problem

$$\dot{y} = \lambda(y - g(t)),$$

where $g(t)$ is bounded, but otherwise arbitrary.

Rewrite as $\epsilon\dot{y} = \hat{\lambda}(y - g(t))$, where $\epsilon = \frac{1}{|\mathcal{R}e(\lambda)|}$, $\hat{\lambda} = \epsilon\lambda$.

When $\epsilon = 0$, we get the reduced solution $y(t) = g(t)$.

A numerical method has stiff decay if for fixed $t_n > 0$,

$$|y_n - g(t_n)| \to 0 \quad \text{as} \quad \Delta t_n \mathcal{R}e(\lambda) \to -\infty.$$

This is a stronger requirement than absolute stability in the very stiff limit; it is not concerned with other parts of complex $z$-plane.

$\rightarrow$ Skips transient phase but gives good description of long-term (slowly varying !) behaviour.
Potential for efficient use, but danger for misuse !

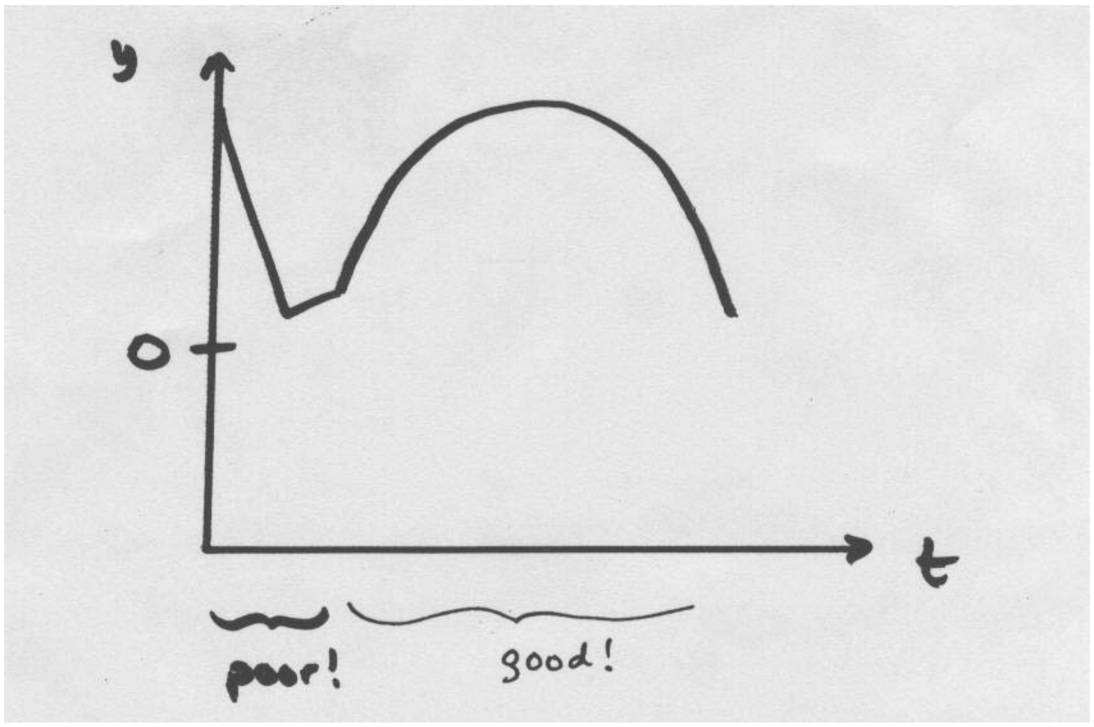
Backward Euler has stiff decay:

$$y_n - g(t_n) = \frac{y_n - g(t_n)}{1 - \lambda \Delta t_n} \to 0 \quad \text{as} \quad \Delta t_n \mathcal{R}e(\lambda) \to -\infty.$$

With $\Delta t_n \equiv 0.1$, on Example 2:

$$\begin{aligned} \dot{y} &= -100(y - \sin t), \quad t \geq 0, \\ y(0) &= 1, \end{aligned}$$

we get



## 3.6 Symmetry and Trapezoidal Method

Forward Euler is based on Taylor expansion at $t_{n-1}$.
Backward Euler is based on Taylor expansion at $t_n$.

Both are first-order accurate.
$\longrightarrow$ Generally too inefficient in practice.

Better accuracy obtained by centering expansions at

$$t_{n-\frac{1}{2}} = t_{n-1} + \frac{\Delta t_n}{2}.$$

$$y(t_n) = y\left(t_{n-\frac{1}{2}}\right) + \frac{\Delta t_n}{2}\dot{y}\left(t_{n-\frac{1}{2}}\right) + \frac{(\Delta t_n)^2}{8}\ddot{y}\left(t_{n-\frac{1}{2}}\right) + \frac{(\Delta t_n)^3}{48}\dddot{y}\left(t_{n-\frac{1}{2}}\right) + \cdots$$

$$y(t_{n-1}) = y\left(t_{n-\frac{1}{2}}\right) - \frac{\Delta t_n}{2}\dot{y}\left(t_{n-\frac{1}{2}}\right) + \frac{(\Delta t_n)^2}{8}\ddot{y}\left(t_{n-\frac{1}{2}}\right) - \frac{(\Delta t_n)^3}{48}\dddot{y}\left(t_{n-\frac{1}{2}}\right) + \cdots$$

(verify)

Subtract and divide by $\Delta t_n$:

$$\frac{y(t_n) - y(t_{n-1})}{\Delta t_n} = \dot{y}\left(t_{n-\frac{1}{2}}\right) + \frac{(\Delta t_n)^2}{24}\dddot{y}\left(t_{n-\frac{1}{2}}\right) + \mathcal{O}((\Delta t_n)^4).$$ (verify)

But

$$\dot{y}\left(t_{n-\frac{1}{2}}\right) = \frac{1}{2}\left[\dot{y}(t_n)+\dot{y}(t_{n-1})\right] - \frac{(\Delta t_n)^2}{8}\dddot{y}\left(t_{n-\frac{1}{2}}\right) + \mathcal{O}((\Delta t_n)^4). \text{ (verify)}$$

This yields the (implicit) trapezoidal method:

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \frac{\Delta t_n}{2}\Big[\mathbf{f}(t_n, \mathbf{y}_n) + \mathbf{f}(t_{n-1}, \mathbf{y}_{n-1})\Big].$$

**Note 7.** • *This is an implicit method.*

• *It is second-order accurate.*

• *It is symmetric:*

$\rightarrow$ *If you change* $t = -\tau$

$\updownarrow$

*integrate from right to left on* $[t_{n-1}, t_n]$,

*the answer does not change !*

More formally, consider a general numerical method

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \Delta t_n \psi(\mathbf{y}_{n-1}, \mathbf{y}_n; \Delta t_n).$$

e.g., for trapezoidal method,

$$\psi = \tfrac{1}{2}\Big[\mathbf{f}(t_{n-1}, \mathbf{y}_{n-1}) + \mathbf{f}(t_n, \mathbf{y}_n)\ \Big].$$

A method is symmetric if it is invariant under the transformation

$$\begin{aligned} &\mathbf{y}_n \to \mathbf{y}_{n-1},\ \mathbf{y}_{n-1} \to \mathbf{y}_n,\ \Delta t_n \to -\Delta t_n, \\ &t_n \to t_{n-1},\ t_{n-1} \to t_n. \end{aligned}$$

$\rightarrow$ Important for reversible flows.

$$\updownarrow$$

e.g., energy-conserving.

Transform trapezoidal rule:

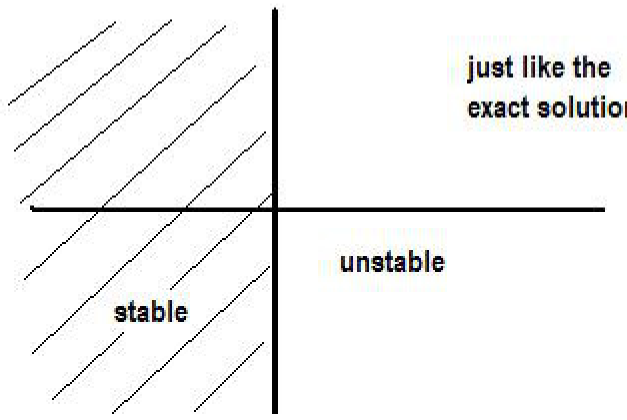
$$\mathbf{y}_{n-1} = \mathbf{y}_n - \Delta t_n \Big[\frac{1}{2}\big(\mathbf{f}(t_{n-1}, \mathbf{y}_{n-1}) + \mathbf{f}(t_n, \mathbf{y}_n)\big)\Big].$$

Rearrange to get the original rule ! (verify)

- Trapezoidal method is 0-stable.
- Check absolute stability:

$$y_n = \frac{2 + \lambda \Delta t_n}{2 - \lambda \Delta t_n} y_{n-1}. \qquad \text{(verify)}$$

If $\mathcal{R}e(\lambda) > 0$, $\left| \frac{2+\lambda\Delta t_n}{2-\lambda\Delta t_n} \right| > 1$. A-stable in exactly
If $\mathcal{R}e(\lambda) \le 0$, $\left| \frac{2+\lambda\Delta t_n}{2-\lambda\Delta t_n} \right| \le 1$. the left-hand plane.



- What about stiff decay ?

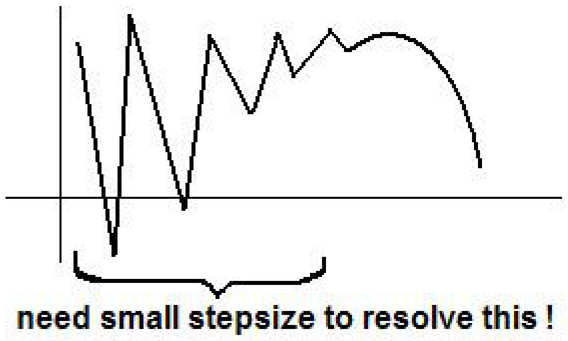
$$\lim_{\Delta t_n \mathcal{R}e(\lambda) \to -\infty} \frac{2 + \lambda \Delta t_n}{2 - \lambda \Delta t_n} = -1 \neq 0.$$

$\Rightarrow$ No stiff decay. (typical of symmetric methods)

$\rightarrow$ Solution is basically oscillatory

$$y_n \approx -y_{n-1}.$$

- **Example 5.** Solve Example 2 with trapezoidal rule.



## 3.7 Non-smooth Problems

We usually assume "sufficient smoothness" of all derivatives. $\rightarrow$ This is often the case, but not always !

In general, if $\mathbf{f}(t, \mathbf{y})$ has $k$ bounded derivatives at $\mathbf{y}(t)$, i.e.,

$$\sup_{t_0 \le t \le t_f} \left\| \frac{d^j}{dt^j} \mathbf{f}(t, \mathbf{y}(t)) \right\| \le M, \qquad j = 0, 1, \ldots, k,$$

then $\mathbf{y}(t)$ has $k+1$ bounded derivatives

$$\left\| \frac{d^j}{dt^j} \mathbf{y} \right\| \le M, \qquad j = 1, 2, \ldots, k+1.$$

So if $\mathbf{f}(t, \mathbf{y})$ is discontinuous but bounded, then $\mathbf{y}(t)$ has a discontinuous but bounded first derivative.

But the higher derivatives are not generally bounded, so the Taylor series expansion is invalid and discretization across such a point may yield inaccurate results.

Suppose there is a $\bar{t} \in [0, t_f]$ where $\mathbf{f}$ is discontinuous.

To get a (non-smooth) solution, we solve 2 problems:

$$\dot{\mathbf{y}}_1 = \mathbf{f}(t, \mathbf{y}_1), \quad 0 < t < \bar{t}, \quad \mathbf{y}_1(0) = \mathbf{y}_0,$$

and $\quad \dot{\mathbf{y}}_2 = \mathbf{f}(t, \mathbf{y}_2), \quad \bar{t} < t < t_f, \quad \mathbf{y}_2(\bar{t}) = \mathbf{y}_1(\bar{t}).$

The numerical method does not know about $\bar{t}$.
We can expect the usual accuracy if we break the problem up at $\bar{t}$ !

**Example 4.** *Let $\tau > 0$ be a parameter and*

$$f(t, y) = t - j\tau, \quad j\tau \leq t < (j+1)\tau,\ j = 0, 1, \dots, J.$$

Exercise: Find the exact solution of the IVP

$$\dot{y} = f(t, y), \qquad y(0) = 0.$$

Show that any second-order method returns the exact solution if it uses the points $t_j = j\tau$, $j = 1, 2, \dots$.

$\rightarrow$ We may not know where $\bar{t}$ is beforehand !

- What if we blindly step over it ?
  We get an $\mathcal{O}(\Delta t_{\bar{n}})$ error, regardless of the (formal) order of accuracy of the method.

The error is generally $\mathcal{O}(\tau \Delta t)$ at each step.

So if we take $\mathcal{O}(1/\Delta t)$ steps (discontinuity is jumped over many times during integration), error is $\mathcal{O}(1)$.

Similarly if $\tau = \mathcal{O}(1/\Delta t)$ (sharp teeth), error is $\mathcal{O}(1)$.

The common way to describe discontinuities in $\mathbf{f}(t, \mathbf{y})$ is in terms of *switching functions* $g(t, \mathbf{y})$,

$$\mathbf{f}(t, \mathbf{y}) = \begin{cases} \mathbf{f}_I(t, \mathbf{y}) & \text{if } g(t, \mathbf{y}) < 0, \\ \mathbf{f}_{II}(t, \mathbf{y}) & \text{if } g(t, \mathbf{y}) > 0. \end{cases}$$

e.g., simulations involving dry friction.

The standard practice is to use an *event location* algorithm that combines an interpolant of the numerical solution with a nonlinear algebraic equation solver to locate the time $t_*$ such that $g(t_*, y(t_*)) = 0$.

Note however that this becomes more complicated when $g$ is a vector (i.e., there are multiple events) because the first such $t_*$ must be detected.

Alternatively, one could simply rely on adaptive step-size control to detect discontinuities and take small steps over them.

However, this approach is generally neither as efficient nor robust as using event location.

In general, a method of order $p$ matches the first $p+1$ terms in the Taylor series of the exact solution and has a truncation error of $\mathcal{O}(\|\mathbf{y}^{(p+1)}\|\Delta t_n^p)$.

**Example 5.** *Consider the harmonic oscillator*

$$\ddot{u} + \omega^2 u = 0,\ u(0) = 1,\ \dot{u}(0) = 0, \quad 0 < t < t_f,$$

*with exact solution $u(t) = \cos(\omega t)$.*

*Noting*

$$\|u^{(p)}\| = \omega^p,$$

*we see that the higher derivatives of the solution grow in size for high frequencies $\omega \gg 1$.*

*Thus the local error is $\mathcal{O}(\Delta t^{p+1}\omega^{p+1})$.*

*So in order to resolve solutions to such* highly oscillatory *problems, we must take $\Delta t < 1/\omega$, independent of $p$!*

*In fact, increasing $p$ if $\Delta t > 1/\omega$ is pointless.*