

Checking stability for Implicit Euler

Applying **Implicit Euler** method to the model problem gives
($y_{n+1} = y_n + \Delta t f(t_{n+1}, y_{n+1})$ with $f(t_{n+1}, y_{n+1}) = ay_{n+1}$)

$$y_{n+1} = y_n + a\Delta t y_{n+1} \quad n = 0, 1, \dots \implies y_{n+1} = \frac{1}{1 - a\Delta t} y_n = G y_n.$$

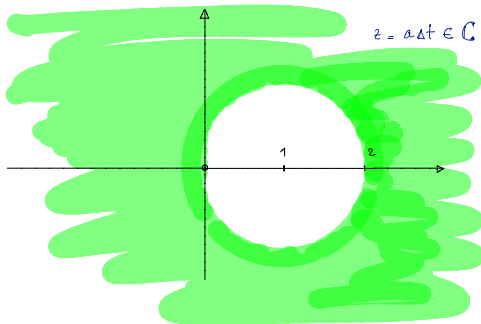
We have $|G| < 1 \iff |1 - a\Delta t| > 1$ and this happens as soon as $a\Delta t$ is outside the unit disk centered at 1.

If $\operatorname{Re}(a) < 0$, then $|G| < 1$ and we say that Implicit Euler is A-stable: the A-stability condition can be written as

$\text{If } \operatorname{Re} a < 0 \text{ then } |G| < 1 \quad =: \text{A-stability}$

However, for Implicit Euler method, the absolute stability region is too large: if $a\Delta t$ is outside the unit circle centered in 1, y_n decays even if the exact solution does not.

Stability for Implicit Euler: the complex setting



The scheme is A-stable:

$$|G| < 1 \Leftrightarrow |1 - a\Delta t| > 1 \Leftrightarrow \operatorname{Re}(a) < 0$$

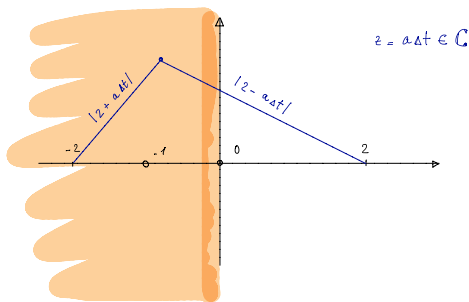
Checking stability for Crank-Nicolson

Crank-Nicolson gives

$$y_{n+1} = y_n + \frac{a\Delta t}{2}(y_n + y_{n+1}) \quad n = 0, 1, \dots \implies y_{n+1} = \frac{1 + \frac{a\Delta t}{2}}{1 - \frac{a\Delta t}{2}} y_n = G y_n.$$

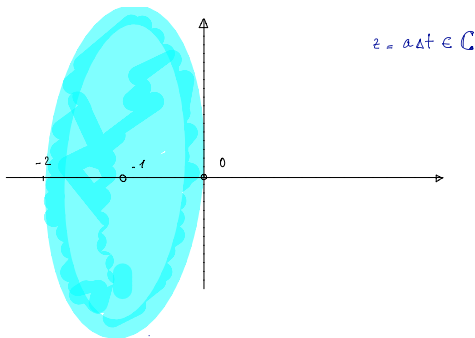
The scheme is A-stable:

$$|G| < 1 \Leftrightarrow |2 + a\Delta t| < |2 - a\Delta t| \Leftrightarrow \operatorname{Re}(a)\Delta t < 0 \Leftrightarrow \operatorname{Re}(a) < 0$$



Checking stability for Heun

Heun method is not A-stable, only **conditionally absolutely stable** like Explicit Euler. In fact, $G = 1 + a\Delta t + \frac{(a\Delta t)^2}{2}$, and for real $a < 0$, $|G| < 1$ iff $0 < \Delta t < 2/|a|$. For complex a , use $|1 + z + z^2/2| < 1$, $z = a\Delta t$.



Stability regions: comparison

For each of these methods we have defined the stability region in the complex plane as

$$A := \{a\Delta t \in \mathbb{C} : \lim_{n \rightarrow \infty} |y_n| = 0\} \equiv \{a\Delta t \in \mathbb{C} : |G| < 1\}$$

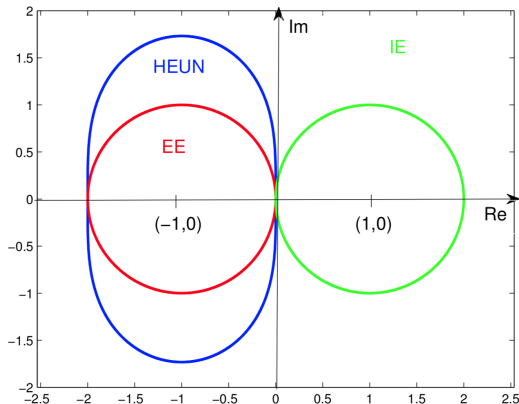
and compared it with the stability region of the continuous problem: the half plane $\operatorname{Re} a < 0$.

For Explicit Euler $A = \{a\Delta t \in \mathbb{C} : |1 + a\Delta t| < 1\}$ is an open disk with center $(-1, 0)$ and radius 1 (too small!).

For Implicit Euler $A = \{a\Delta t \in \mathbb{C} : |1 - a\Delta t| > 1\}$ is the whole plane outside the closed disk with center $(1, 0)$ and radius 1 (too big!)

For Crank-Nicolson the region is the left-half plane, exactly as for the true solution (but this alone does not guarantee strong damping of stiff modes).

Stability regions



absolute stability regions for EE (the region inside the red circle), Heun (the region bounded by the blue curve), IE (green, the region outside the circle with center $(1, 0)$ and radius 1).

Conclusions (for the four basic methods)

Explicit Runge–Kutta and standard explicit linear multistep methods are not A-stable. On a decaying test problem their absolute stability imposes restrictions on the step size. Some implicit methods are A-stable.

For the method we have considered:

Method	Consistency	Stability
Explicit Euler	yes, order 1	conditionally absolutely stable
Implicit Euler	yes, order 1	A-stable
Crank-Nicolson	yes, order 2	A-stable
Heun	yes, order 2	conditionally absolutely stable

Generalizing Heun idea: Runge-Kutta methods

The celebrated Runge-Kutta methods are compound 1-step methods.

The basic idea is very simple: choose a high precision quadrature formula for $\int f$ on each interval $[t_n, t_{n+1}]$. Then, since the values at the quadrature nodes are not known, we predict them somehow (and this is where the detailed description could become quite complicated).

The simplest explicit RK is Heun: the starting point is the trapezoidal rule for $\int_{t_n}^{t_{n+1}} f$, and since we want to go explicit, instead of the value y_{n+1} (that would be needed in the trapezoidal rule) we use $y_n + \Delta t f(t_n, y_n)$, that is, the value predicted by Explicit Euler.

Runge-Kutta methods

Heun scheme:

$$\begin{cases} y_0 \text{ given} \\ y_{n+1}^* = y_n + \Delta t f(t_n, y_n) \\ y_{n+1} = y_n + \frac{\Delta t}{2} \left(f(t_n, y_n) + f(t_{n+1}, y_{n+1}^*) \right) \end{cases} \quad n = 0, 1, \dots, N-1$$

Denoting by K_1 and K_2 the values of f at the two nodes t_n and $t_{n+1} = t_n + \Delta t$ we can rewrite Heun method as:

$$\begin{aligned} K_1 &= f(t_n, y_n), \quad K_2 = f(t_n + \Delta t, y_n + \Delta t K_1) \\ y_{n+1} &= y_n + \frac{\Delta t}{2} (K_1 + K_2) \quad n = 0, 1, \dots \end{aligned}$$

Runge-Kutta methods

The more famous version of Runge-Kutta, **RK4**, is compounded *four times*, it is based on *Simpson rule*, and achieves order $p = 4$. Let us see how it is obtained.

Simpson rule uses on the interval $[t_n, t_{n+1}]$ three nodes t_n , $t_{n+1/2} = t_n + \Delta t/2$, and $t_{n+1} = t_n + \Delta t$, and would give

$$\int_{t_n}^{t_{n+1}} f(t, y(t)) dt \simeq \frac{\Delta t}{6} (f_n + 4f_{n+1/2} + f_{n+1}).$$

$$(f_n = f(t_n, y_n), f_{n+1/2} = f(t_{n+1/2}, y_{n+1/2}), f_{n+1} = f(t_{n+1}, y_{n+1}))$$

The value f_n is known, so we set $K_1 = f(t_n, y_n)$; then write $4f_{n+1/2}$ as $2f_{n+1/2}^{(1)} + 2f_{n+1/2}^{(2)}$, and we choose two different “predictions” for $y_{n+1/2}$:

$$f_{n+1/2}^{(1)} = f(t_{n+1/2}, y_n + \frac{\Delta t}{2} K_1) =: K_2$$

$$f_{n+1/2}^{(2)} = f(t_{n+1/2}, y_n + \frac{\Delta t}{2} K_2) =: K_3$$

RK4

We had

$$K_1 = f(t_n, y_n), \quad K_2 = f_{n+1/2}^{(1)} = f(t_{n+1/2}, y_n + \frac{\Delta t}{2} K_1),$$

$$K_3 = f_{n+1/2}^{(2)} = f(t_{n+1/2}, y_n + \frac{\Delta t}{2} K_2)$$

For f_{n+1} we take here $f_{n+1} = f(t_{n+1}, y_n + \Delta t K_3)$ (although other choices could have been possible), so that the scheme is:

$$K_1 = f(t_n, y_n), \quad K_2 = f(t_{n+1/2}, y_n + \frac{\Delta t}{2} K_1)$$

$$K_3 = f(t_{n+1/2}, y_n + \frac{\Delta t}{2} K_2), \quad K_4 = f(t_{n+1}, y_n + \Delta t K_3)$$

$$y_{n+1} = y_n + \frac{\Delta t}{6} (K_1 + 2K_2 + 2K_3 + K_4) \quad n = 0, 1, \dots$$

Explicit Runge-Kutta

The family of explicit Runge-Kutta methods is a generalisation of the RK4 scheme above:

$$y_0 \text{ given} \quad y_{n+1} = y_n + \Delta t \sum_{i=1}^s b_i K_i, \quad n = 0, 1, \dots \quad (1)$$

where

$$K_i = f(t_n + c_i \Delta t, y_n + \Delta t \sum_{j=1}^{i-1} a_{ij} K_j)$$

To specify a particular method one needs to provide the integer s (the number of stages), and the coefficients a_{ij} , b_i , c_i . The matrix $[a_{ij}]$ is called *Runge-Kutta matrix*, while the b_i and c_i are called *weights* and *nodes*, respectively. These coefficients are usually arranged in the *Butcher tableau*

Butcher tableau

0					
c_2	$a_{2,1}$				
c_3	$a_{3,1}$	$a_{3,2}$			
$\vdots$		$\vdots$	$\ddots$		
c_s	$a_{s,1}$	$a_{s,2}$	$\cdots$	$a_{s,s-1}$	
	b_1	b_2	$\cdots$	b_{s-1}	b_s

Consistency conditions

For consistency the coefficients must verify algebraic conditions;

a RK scheme is consistent **iff** $\sum_{i=1}^s b_i = 1$.

(This condition is always verified since the b_i are the weights of the quadrature formula used, which has to be exact at least on constants).

Assume the standard convention $c_i = \sum_j a_{ij}$. For higher order, other relations must be satisfied. For instance, for a 2 stage explicit RK to have order 2 we need, together with $b_1 + b_2 = 1$, also $b_2 c_2 = 1/2$. Without $c = A\mathbf{1}$, order 2 requires both $\sum_i b_i c_i = 1/2$ and $\sum_{ij} b_i a_{ij} = 1/2$.

RK order and stages * NOT FOR THE EXAM *

Theorem 1

An explicit s -stages Runge-Kutta method cannot have order of accuracy p greater than s . For general ODE systems, an explicit method with $s \geq 5$ cannot attain $p = s$.

order p	1	2	3	4	5	6	7	8
s_{min}	1	2	3	4	6	7	9	11

Source: J. C. Butcher, order barriers for explicit RK methods.

RK methods are very successful and widely used in the codes for their ductility: the time step can easily be modified from one interval to another if needed, the initial value y_0 is all what is needed to start the method, and they have high accuracy.

A first multistep method

Let us start with an example. Let $t_0, t_1, \dots, t_N = T$ be a set of *equally spaced* points in $[t_0, T]$, and let $\Delta t = \frac{T - t_0}{N}$ be the time step (this time the points **must** be equally spaced).

We want to construct an explicit scheme of order 2 going back two steps:

$$y_{n+1} = y_n + \Delta t \left(\alpha f(t_n, y_n) + \beta f(t_{n-1}, y_{n-1}) \right), n = 1, 2, \dots$$

This requires values at time $t_{n-1} = t_0 + (n-1)\Delta t$ as well as at time t_n . Therefore the initial value y_0 is not enough to start the procedure and we need to compute y_1 with a 1-step method. Then we have to find α and β such that the scheme has order 2.

Choosing the parameters in a multistep scheme....

The starting point is the same as for 1-step methods:

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} y'(t) dt = \int_{t_n}^{t_{n+1}} f(t, y(t)) dt. \quad (*)$$

The function f is then approximated by its Lagrange interpolant polynomial of degree ≤ 1 with respect to the nodes t_{n-1} and t_n :

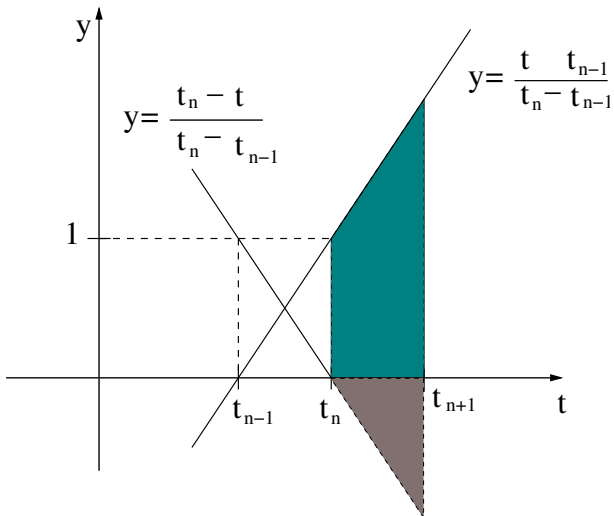
$$f(t, y(t)) \simeq \Pi_1(t) := \frac{t - t_{n-1}}{t_n - t_{n-1}} f(t_n, y_n) + \frac{t_n - t}{t_n - t_{n-1}} f(t_{n-1}, y_{n-1})$$

Consequently,

$$\int_{t_n}^{t_{n+1}} f(t, y(t)) dt \sim \int_{t_n}^{t_{n+1}} \Pi_1(t) dt = \frac{3}{2} \Delta t f(t_n, y_n) - \frac{1}{2} \Delta t f(t_{n-1}, y_{n-1})$$

The order of accuracy is 2: if $f \in \mathbb{P}_1$, then $f \equiv \Pi_1$ and $\int f$ is computed exactly. On the other hand, $f \in \mathbb{P}_1$ implies $y \in \mathbb{P}_2$.

Integral of Π_1



Generalization

With the same approach we can design schemes that use values computed at p previous steps and are p -accurate:

on each interval $[t_n, t_{n+1}]$ the function f is replaced by its Lagrange interpolant polynomial (of degree $\leq p-1$) with respect to the p points $t_n, t_{n-1}, \dots, t_{n+1-p}$:

$$f(t, y(t)) \simeq \Pi_{p-1}(t) \quad \text{with } \Pi_{p-1} \in \mathbb{P}_{p-1} \text{ verifying}$$

$$\Pi_{p-1}(t_n) = f(t_n, y_n),$$

$$\Pi_{p-1}(t_{n-1}) = f(t_{n-1}, y_{n-1}),$$

$\dots$

$$\Pi_{p-1}(t_{n+1-p}) = f(t_{n+1-p}, y_{n+1-p}).$$

The $\int_{t_n}^{t_{n+1}} \Pi_{p-1}(t) dt$ is then computed exactly; to complete the p -step scheme we will need to compute $p-1$ “initial values” $y_1, y_2, \dots, y_{p-1}$ in addition to y_0 (for instance with a 1-step method).

Adams-Bashforth schemes

The resulting scheme will be:

$$\begin{cases} y_0 \text{ given, } y_1, y_2, \dots, y_{p-1} \text{ to be computed} \\ y_{n+1} = y_n + \Delta t \left(c_1 f_n + c_2 f_{n-1} + \dots + c_p f_{n+1-p} \right), \\ n = p-1, p, p+1, \dots \end{cases} \quad (2)$$

where $\Delta t c_1, \Delta t c_2, \dots$ are the integrals of the characteristic Lagrange polynomials, and $f_n = f(t_n, y_n)$, $f_{n-1} = f(t_{n-1}, y_{n-1})$ and so on.

The multistep methods obtained in this way are “**Adams-Bashforth**” methods: they are **explicit**, p -accurate. In Table 1 below the coefficients of the first four schemes.

Adams-Bashforth schemes ** NOT FOR THE EXAM **

	c_1	c_2	c_3	c_4
$p = 1$	1			
$p = 2$	$3/2$	$-1/2$		
$p = 3$	$23/12$	$-16/12$	$5/12$	
$p = 4$	$55/24$	$-59/24$	$37/24$	$-9/24$

Table: First Adams-Bashforth schemes of order p

Adams-Moulton

Note: A similar construction gives **implicit methods**, called **Adams-Moulton**. Compared with (2) they have an extra term $c_0 f_{n+1}$ at the new time level. Properly chosen, that adds one extra order of accuracy (as it did for the Crank-Nicolson scheme).

	c_0	c_1	c_2	c_3	A-stability	order
$p = 0$	1				yes	1
$p = 1$	1/2	1/2			yes	2
$p = 2$	5/12	8/12	-1/12		no	3
$p = 3$	9/24	19/24	-5/24	1/24	no	4

Table: First four Adams-Moulton schemes: order $p + 1$ ** NOT FOR THE EXAM
**

Multistep consistency * NOT FOR THE EXAM *

For this linear multistep formula, consistency of order at least 2 means that the residual vanishes for every quadratic polynomial. By linearity, it suffices to test $y = 1, t, t^2$.

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - \left(\alpha f(t_n, y(t_n)) + \beta f(t_{n-1}, y(t_{n-1})) \right) \quad n = 1, 2, \dots$$

$$\text{When } y(t) = 1, y' = 0 = f, \implies \tau_n = \frac{1 - 1}{\Delta t} = 0;$$

$$\begin{aligned} \text{When } y(t) = t, y' = 1 = f, \implies \tau_n &= \frac{t_{n+1} - t_n}{\Delta t} - (\alpha + \beta) = 0; \\ \implies \alpha + \beta &= 1 \implies \beta = 1 - \alpha; \end{aligned}$$

$$\begin{aligned} \text{When } y(t) = t^2, y' = 2t = f, \implies \tau_n &= \frac{t_{n+1}^2 - t_n^2}{\Delta t} - (2\alpha t_n + 2\beta t_{n-1}) \\ &= t_{n+1} + t_n - 2(\alpha t_n + \beta t_{n-1}) = 0 \\ \implies \alpha t_n + \beta t_{n-1} &= \frac{t_{n+1} + t_n}{2}. \end{aligned}$$

Multistep consistency * NOT FOR THE EXAM *

$$\beta = 1 - \alpha, \quad \alpha t_n + \beta t_{n-1} = \frac{t_{n+1} + t_n}{2}$$

Inserting $\beta = 1 - \alpha$ in the second equation we have

$$\begin{aligned}\alpha(t_n - t_{n-1}) &= \frac{t_n + t_{n+1} - 2t_{n-1}}{2} \\ &= \frac{n\Delta t + (n+1)\Delta t - 2(n-1)\Delta t}{2} = \frac{3\Delta t}{2}.\end{aligned}$$

Therefore we obtain $\alpha = \frac{3}{2}$ and $\beta = -\frac{1}{2}$. The 2-step scheme is then

$$\begin{cases} y_0 \text{ given, } y_1 \text{ to be computed} \\ y_{n+1} = y_n + \Delta t \left(\frac{3}{2}f(t_n, y_n) - \frac{1}{2}f(t_{n-1}, y_{n-1}) \right), n = 1, 2, \dots \end{cases}$$

By construction the scheme is consistent of order 2. Being explicit, it will not be A-stable, only conditionally absolutely stable.

More schemes ** NOT FOR THE EXAM **

Another way of constructing explicit methods with a good accuracy is to choose an implicit scheme, and make it explicit with a very simple and successful trick:

- P: use the explicit formula to *predict* a new y_{n+1}^*
- E: use y_{n+1}^* to *evaluate* $f_{n+1}^* = f(t_{n+1}, y_{n+1}^*)$
- C: use f_{n+1}^* in the implicit formula to *correct* the new y_{n+1}

This is the **predictor-corrector** method (see Heun method). A final E step evaluates f_{n+1} at the corrected value for use in the next step. This gives the PECE sequence. Its stability depends on the predictor, corrector and number of corrections.

Repeated CE corrections converge to the implicit solution under a contraction assumption: for $y = C + hb_0 f(t, y)$, $h|b_0|L < 1$ is sufficient. Finite corrections do not inherit the full implicit stability region. The iteration count and comparison with Newton depend on the problem.

Systems of ODEs

It is much more common to have systems of differential equations than a single equation. The unknown is now a vector $\underline{Y}(t)$, and so is the right-hand side $\underline{F}(t, \underline{Y}(t))$. The problem is: find $\underline{Y}(t)$ solution of

$$\begin{cases} \underline{Y}'(t) = \underline{F}(t, \underline{Y}(t)) & t \in [t_0, T] \\ \underline{Y}(t_0) = \underline{Y}^{(0)}. \end{cases} \quad (3)$$

with

$$\underline{Y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_N(t) \end{bmatrix}, \quad \underline{F}(t, \underline{Y}(t)) = \begin{bmatrix} f_1(t, \underline{Y}(t)) \\ f_2(t, \underline{Y}(t)) \\ \vdots \\ f_N(t, \underline{Y}(t)) \end{bmatrix}, \quad \underline{Y}^{(0)} = \begin{bmatrix} y_1^{(0)} \\ y_2^{(0)} \\ \vdots \\ y_N^{(0)} \end{bmatrix}$$

Systems of ODEs

The numerical schemes used for a single equation apply directly to systems of ODEs.

For instance, the two Euler methods become:

$$(EE) \quad \begin{cases} \underline{Y}^{(0)} \text{ given} \\ \underline{Y}^{(n+1)} = \underline{Y}^{(n)} + \Delta t \underline{F}(t_n, \underline{Y}^{(n)}) \end{cases} \quad n = 0, 1, \dots$$

Ex: $N = 2$ equations, and 2 unknowns y_1, y_2 :

$$\begin{cases} y_1^{(0)}, y_2^{(0)} \text{ given} \\ y_1^{(n+1)} = y_1^{(n)} + \Delta t f_1(t_n, y_1^{(n)}, y_2^{(n)}) \\ y_2^{(n+1)} = y_2^{(n)} + \Delta t f_2(t_n, y_1^{(n)}, y_2^{(n)}) \end{cases} \quad n = 0, 1, \dots$$

Systems of ODEs

$$(IE) \quad \begin{cases} \underline{Y}^{(0)} \text{ given} \\ \underline{Y}^{(n+1)} = \underline{Y}^{(n)} + \Delta t \underline{F}(t_{n+1}, \underline{Y}^{(n+1)}) \quad n = 0, 1, \dots \end{cases}$$

Ex: $N = 2$ equations, and 2 unknowns y_1, y_2 :

$$\begin{cases} y_1^{(0)}, y_2^{(0)} \text{ given} \\ y_1^{(n+1)} = y_1^{(n)} + \Delta t f_1(t_{n+1}, y_1^{(n+1)}, y_2^{(n+1)}) \quad n = 0, 1, \dots \\ y_2^{(n+1)} = y_2^{(n)} + \Delta t f_2(t_{n+1}, y_1^{(n+1)}, y_2^{(n+1)}) \quad n = 0, 1, \dots \end{cases}$$

much more expensive now: at each step, to go from $\underline{Y}^{(n)}$ to $\underline{Y}^{(n+1)}$ requires the solution of a **non-linear system**!

** NOT FOR THE EXAM **

$$\underline{Y}^{(n+1)} = \underline{Y}^{(n)} + \Delta t \underline{F}(t_{n+1}, \underline{Y}^{(n+1)})$$

find $\underline{X}$ such that

$$\underline{G}(\underline{X}) := \underline{X} - \Delta t \underline{F}(t_{n+1}, \underline{X}) - \underline{Y}^{(n)} = 0,$$

and set $\underline{Y}^{(n+1)} := \underline{X}$

Newton would give:

$$\begin{cases} \underline{X}^{(0)} \text{ given} \\ J_{\underline{G}[\underline{X}^{(0)}]} \underline{\delta X} = -\underline{G}(\underline{X}^{(0)}) \\ \underline{X}^{(1)} = \underline{X}^{(0)} + \underline{\delta X} \end{cases}$$

$$\underline{X}^{(0)} = ?? \quad \text{for example: } \underline{X}^{(0)} = \underline{Y}^{(n)} + \Delta t \underline{F}(t_n, \underline{Y}^{(n)}) \quad (EE)$$

** NOT FOR THE EXAM **

Or you could use *PECE* with a few cycles of *CE*:

$$P : \underline{X}^{(0)} = \underline{Y}^{(n)} + \Delta t \underline{F}(t_n, \underline{Y}^{(n)}) \quad (EE)$$

$$E : \underline{F}^{(0)}(t_{n+1}, \underline{X}^{(0)})$$

$$C : \underline{X}^{(1)} = \underline{Y}^{(n)} + \Delta t \underline{F}^{(0)}(t_{n+1}, \underline{X}^{(0)})$$

$$E : \underline{F}^{(1)}(t_{n+1}, \underline{X}^{(1)})$$

$$C : \underline{X}^{(2)} = \underline{Y}^{(n)} + \Delta t \underline{F}^{(1)}(t_{n+1}, \underline{X}^{(1)})$$

$$E : \underline{F}^{(2)}(t_{n+1}, \underline{X}^{(2)})$$

Then set:

$$\underline{Y}^{(n+1)} = \underline{Y}^{(n)} + \Delta t \underline{F}^{(2)}(t_{n+1}, \underline{X}^{(2)})$$

Lack of A-stability ** NOT FOR THE EXAM **

For a single equation the lack of A-stability is not a major drawback: to have a good accuracy small Δt have to be used. Instead for systems it could be more severe when the problem has different time scales.

$$\underline{Y}'(t) = A\underline{Y}(t)$$

the eigenvalues λ_j of the square matrix A take the place of the single number a .

Explicit Euler would give

$$\underline{Y}^{(n+1)} = (I + \Delta t A)\underline{Y}^{(n)} \quad \forall n \implies \underline{Y}^{(n+1)} = (I + \Delta t A)^{n+1}\underline{Y}^{(0)}$$

The growth factor is now a matrix

$$G = (I + \Delta t A) \quad \text{with eigenvalues } g_j = 1 + \Delta t \lambda_j.$$

Lack of A-stability ** NOT FOR THE EXAM **

Assume A is diagonalizable; for the methods above, $G = R(hA)$ has the same eigenvector basis (when defined). Then,

- each component of the discrete solution grows like g_j^n ,
- each component of the continuous solution grows like $e^{\lambda_j t}$.

The continuous solution is stable if all the λ_j are negative (or $\text{Re } \lambda_j < 0$): hence $e^{\lambda_j t} \rightarrow 0$ for $t \rightarrow \infty$.

The discrete solution is stable if all the $|g_j| < 1$, so that $g_j^n \rightarrow 0$ for $n \rightarrow \infty$.

If the problem has different time scales we are in trouble...

Since Δt is the same for all the components, for real negative eigenvalues its size is controlled by the most negative eigenvalue, which corresponds to the fastest decay and dies out first in the true solution.

Stiffness * NOT FOR THE EXAM *

A problem is stiff when stability forces explicit methods to take much smaller steps than the accuracy of the solution of interest would require. Widely separated decaying time scales are a common cause. Stiffness also depends on the interval and requested accuracy; it can occur in a scalar ODE. For a diagonalizable system $Y' = AY$ with negative real eigenvalues, explicit Euler requires

$$0 < h < \min_j \frac{2}{|\lambda_j|}.$$

The fastest decaying mode can restrict the step even after it has become negligible in the solution of interest. For complex eigenvalues, instead test $|1 + h\lambda_j| < 1$ for every mode.

A stiff system: exact solution and step restriction

For $y_1(0) = c_1$, $y_2(0) = c_2$, consider

$$Y' = \begin{pmatrix} -2 & 1 \\ 0 & -100 \end{pmatrix} Y.$$

The exact solution is

$$\begin{aligned} y_2(t) &= c_2 e^{-100t}, \\ y_1(t) &= (c_1 + c_2/98)e^{-2t} - (c_2/98)e^{-100t}. \end{aligned}$$

Explicit Euler requires $h < 1$ and $h < 1/50$, hence $h < 1/50$. When the slow mode is present, it governs the late-time solution, but the fast mode still restricts explicit stability. Round-off and perturbations can excite that mode even after its exact contribution has become tiny.

Matlab functions

Most commonly used Matlab functions:

Non stiff problems:

ode23 (low order RK), ode45 (medium order RK), ode113 (variable order)

Stiff: ode15s (low to medium order), ode23s (low order RK)

Three notions: consistency and stability

For a fixed-step linear multistep formula

$$\sum_{j=0}^k \alpha_j y_{n+j} = h \sum_{j=0}^k \beta_j f(t_{n+j}, y_{n+j}), \quad \rho(\zeta) = \sum_{j=0}^k \alpha_j \zeta^j,$$

zero-stability means: every root of ρ has modulus at most 1, and roots on the unit circle are simple (the root condition). For Lipschitz f , consistency plus zero-stability gives convergence on a fixed time interval. Consistency of order p and starting errors of order h^p yield global error of order h^p for smooth solutions. Absolute stability instead concerns decay on $y' = ay$ at fixed $z = ha$. A-stability requires the whole left half-plane to be absolutely stable. These notions answer different questions.

A-stability and damping of stiff modes

Both implicit Euler and Crank–Nicolson are A-stable. Their amplification factors are

$$R_{\text{IE}}(z) = \frac{1}{1 - z}, \quad R_{\text{CN}}(z) = \frac{1 + z/2}{1 - z/2}.$$

For a rapidly decaying mode with $z = ha \rightarrow -\infty$,

$$e^z \rightarrow 0, \quad R_{\text{IE}}(z) \rightarrow 0, \quad R_{\text{CN}}(z) \rightarrow -1.$$

Implicit Euler damps the mode strongly. Crank–Nicolson can retain slowly decaying sign oscillations if the step does not resolve the fast scale. An

A-stable one-step method is L-stable if its amplification factor also tends to zero as $|z| \rightarrow \infty$ in the left half-plane. Implicit Euler is L-stable; Crank–Nicolson is not.