Ordinary Differential Equations (Ode's): Example 1

$m(t)$ = money at time t

assume you invest your money $m(t_0)$ at time t_0 and you get an interest $i(t)$, while you pay to the bank a management cost $c(m) \cdot m$ (for example: 1% for $m \leq 10k$€, 0.8% for higher amount)

then the mathematical model looks like:

$$\begin{cases} \dfrac{dm}{dt}(t) = \big(i(t) - c(m)\big)\, m(t) \\ m(t_0) = \text{given initial amount} \end{cases}$$

Ordinary Differential Equations (Ode's): Example 1

$m(t)$ = money at time t

assume you invest your money $m(t_0)$ at time t_0 and you get an interest $i(t)$, while you pay to the bank a management cost $c(m) \cdot m$ (for example: 1% for $m \leq 10k$€, 0.8% for higher amount)

then the mathematical model looks like:

$$\begin{cases} \dfrac{dm}{dt}(t) = \overbrace{\left(i(t) - c(m)\right) m(t)}^{f(t,\, m(t))} \\ m(t_0) = \text{given initial amount} \end{cases}$$

note that if $i=0$ and $c=0$ then m remains constant.

Ordinary Differential Equations (Ode's): Example 2

$\underline{x}(t)$ = position in $\mathbb{R}^3$ of a rigid particle subject to a force $\underline{f}(\underline{x}, t)$, for example

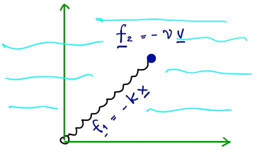
ν = constant related to viscosity

$\underline{f}_2$ = viscous force from the steady liquid to the moving particle

$\underline{v}$ = velocity of the particle = $\dfrac{d\underline{x}}{dt}$

$\underline{f}_1$ = force due to the spring

m = mass of the particle



$$\begin{cases} \dfrac{d\underline{x}}{dt}(t) = \underline{v}(t) & \leftarrow \text{definition of velocity} \\[2ex] \dfrac{d\underline{v}}{dt}(t) = \dfrac{1}{m}\left(-k\,\underline{x}(t) - \nu\,\underline{v}(t)\right) & \leftarrow \underbrace{m \cdot \dfrac{dv}{dt} = \text{force} = f_1 + f_2}_{\text{Newton's law}} \\[2ex] \underline{x}(t_0) = \text{initial position} \\ \underline{v}(t_0) = \text{initial velocity} \end{cases}$$

Ordinary Differential Equations (Ode's): Example 2

$$\begin{cases} \dfrac{d\underline{x}}{dt}(t) = \underline{v}(t) \\ \dfrac{d\underline{v}}{dt}(t) = \dfrac{1}{m}\left(-k\,\underline{x}(t) - \nu\,\underline{v}(t)\right) \\ \underline{x}(t_0) = \text{initial position} \\ \underline{v}(t_0) = \text{initial velocity} \end{cases}$$

← definition of velocity

← $\underbrace{m \cdot \dfrac{dv}{dt} = \text{force}}_{\text{Newton's law}} = f_1 + f_2$

introducing $\underline{y}(t) = \begin{bmatrix} \underline{x}(t) \\ \underline{v}(t) \end{bmatrix} \in \mathbb{R}^6$ then

the Cauchy problem above reads:

$$\begin{cases} \dfrac{d}{dt}\underline{y}(t) = \underline{f}(t, \underline{y}(t)) \\ \underline{y}(t_0) = \underline{y}_0 \end{cases}$$

← system of differential eq.

← initial condition

Ordinary Differential Equations (Ode's)

We shall consider some numerical schemes to solve initial-value problems (Cauchy's problems) written as: Find $y = y(t)$ solution of

$$\begin{cases} y'(t) = f(t, y(t)) & t \in [t_0, T] \\ y(t_0) = y_0. \end{cases} \tag{1}$$

Ordinary Differential Equations (Ode's)

We shall consider some numerical schemes to solve initial-value problems (Cauchy's problems) written as: Find $y = y(t)$ solution of

$$\begin{cases} y'(t) = f(t, y(t)) & t \in [t_0, T] \\ y(t_0) = y_0. \end{cases} \tag{1}$$

We assume $y : [t_0, T] \to \mathbb{R}$ but can be generalized to $y : [t_0, T] \to \mathbb{R}^d$

Ordinary Differential Equations (Ode's)

We shall consider some numerical schemes to solve initial-value problems (Cauchy's problems) written as: Find $y = y(t)$ solution of

$$\begin{cases} y'(t) = f(t, y(t)) & t \in [t_0, T] \\ y(t_0) = y_0. \end{cases} \tag{1}$$

In general $f(t, y(t))$ is a non-linear function describing the evolution in time of $y(t)$. The true solution $y(t)$ of (1) evolves continuously in time, and we want to follow it by a discrete approximation.

Ordinary Differential Equations (Ode's)

We shall consider some numerical schemes to solve initial-value problems (Cauchy's problems) written as: Find $y = y(t)$ solution of

$$\begin{cases} y'(t) = f(t, y(t)) & t \in [t_0, T] \\ y(t_0) = y_0. \end{cases} \tag{1}$$

In general $f(t, y(t))$ is a non-linear function describing the evolution in time of $y(t)$. The true solution $y(t)$ of (1) evolves continuously in time, and we want to follow it by a discrete approximation.

Both exact and discrete solution of (1) start from the same initial value y_0 at t_0. The discrete one takes finite steps Δt, and after n steps it reaches a value y_n. We hope and expect that y_n is close to the exact value $y(t_0 + n\Delta t)$. We shall see that this may or may not happen.

let us see some schemes to solve numerically (1). They are numerous, and a first distinction is among 1-step methods and multi-step methods. Let us see 1-step methods.

let us see some schemes to solve numerically (1). They are numerous, and a first distinction is among 1-step methods and multi-step methods. Let us see 1-step methods.

They can all be derived in the following way.
Le $t_0, t_1, \cdots, t_N = T$ be a set of points in $[t_0, T]$; as usual, for simplify notation we take them equally spaced: (N given, we define $\Delta t = \dfrac{T - t_0}{N}$ and we set $t_0, t_1 = t_0 + \Delta t, t_2 = t_1 + \Delta t, \cdots, t_N = T$).

Numerical methods for Ode's

let us see some schemes to solve numerically (1). They are numerous, and a first distinction is among 1-step methods and multi-step methods. Let us see 1-step methods.

They can all be derived in the following way.
Le $t_0, t_1, \cdots, t_N = T$ be a set of points in $[t_0, T]$; as usual, for simplify notation we take them equally spaced: (N given, we define $\Delta t = \dfrac{T - t_0}{N}$ and we set $t_0, t_1 = t_0 + \Delta t, t_2 = t_1 + \Delta t, \cdots, t_N = T$).
At each step, (on each subinterval $[t_n, t_{n+1}]$) *we integrate the differential equations...*

let us see some schemes to solve numerically (1). They are numerous, and a first distinction is among 1-step methods and multi-step methods. Let us see 1-step methods.

They can all be derived in the following way.
Le $t_0, t_1, \cdots, t_N = T$ be a set of points in $[t_0, T]$; as usual, for simplify notation we take them equally spaced: (N given, we define $\Delta t = \dfrac{T - t_0}{N}$ and we set $t_0, t_1 = t_0 + \Delta t, t_2 = t_1 + \Delta t, \cdots, t_N = T$).
At each step, (on each subinterval $[t_n, t_{n+1}]$) *we integrate the differential equations...*

$$y'(t) \quad = \quad f(t, y(t))$$

let us see some schemes to solve numerically (1). They are numerous, and a first distinction is among 1-step methods and multi-step methods. Let us see 1-step methods.

They can all be derived in the following way.
Le $t_0, t_1, \cdots, t_N = T$ be a set of points in $[t_0, T]$; as usual, for simplify notation we take them equally spaced: (N given, we define $\Delta t = \dfrac{T - t_0}{N}$ and we set $t_0, t_1 = t_0 + \Delta t, t_2 = t_1 + \Delta t, \cdots, t_N = T$).
At each step, (on each subinterval $[t_n, t_{n+1}]$) *we integrate the differential equations...*

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

Then, as $y(t)$ in the interval $[t_n, t_{n+1}]$ is unknown to us (and moreover, in general, we are unable to compute the integral exactly), we use some quadrature formula.

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

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

$$\int_{t_n}^{t_{n+1}} f(t, y(t))dt \sim \text{ quadrature formula.}$$

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

$$\int_{t_n}^{t_{n+1}} f(t, y(t))dt \sim \text{ quadrature formula.}$$

Different choices of quadrature formulas give rise to different schemes.

Example 1 We consider first the quadrature formula

$$\int_c^d g(s)ds \simeq (d-c)\,g(c) \tag{2}$$

that, indeed, is very poor (and is exact only for $g = constant$).

Examples of numerical schemes

Example 1 We consider first the quadrature formula

$$\int_c^d g(s)ds \simeq (d-c)\,g(c) \tag{2}$$

that, indeed, is very poor (and is exact only for $g = \mathit{constant}$). Then

$$\int_{t_n}^{t_{n+1}} f(t,y(t))dt \simeq (t_{n+1}-t_n)f(t_n,y(t_n)) \quad n = 0,1,2,\cdots \tag{3}$$

Example 1 We consider first the quadrature formula

$$\int_c^d g(s)ds \simeq (d-c)\,g(c) \tag{2}$$

that, indeed, is very poor (and is exact only for $g = constant$). Then

$$\int_{t_n}^{t_{n+1}} f(t, y(t))dt \simeq (t_{n+1} - t_n) f(t_n, y(t_n)) \quad n = 0, 1, 2, \cdots \tag{3}$$

Using (3) into $y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} y'(t)dt = \int_{t_n}^{t_{n+1}} f(t, y(t))dt$ we get:

Example 1 We consider first the quadrature formula

$$\int_c^d g(s)ds \simeq (d-c)\, g(c) \tag{2}$$

that, indeed, is very poor (and is exact only for $g = constant$). Then

$$\int_{t_n}^{t_{n+1}} f(t, y(t))dt \simeq (t_{n+1} - t_n) f(t_n, y(t_n)) \quad n = 0, 1, 2, \cdots \tag{3}$$

Using (3) into $y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} y'(t)dt = \int_{t_n}^{t_{n+1}} f(t, y(t))dt$ we get:

$$y(t_1) \simeq y(t_0) + \Delta t\, f(t_0, y(t_0)) = y_0 + \Delta t\, f(t_0, y_0) =: y_1$$
$$y(t_2) \simeq y(t_1) + \Delta t\, f(t_1, y(t_1)) \simeq y_1 + \Delta t\, f(t_1, y_1) =: y_2$$
$$\vdots$$
$$y(t_N) \simeq y(t_{N-1}) + \Delta t\, f(t_{N-1}, y(t_{N-1})) \simeq y_{N-1} + \Delta t\, f(t_{N-1}, y_{N-1}) =: y_N$$

Examples of numerical schemes

It is clear from this that errors accumulate at each step and might produce unexpected results. We will analyse the scheme later on. Let us write it in a compact form:

$$\begin{cases} y_0 \text{ given} \\ y_{n+1} = y_n + \Delta t\, f(t_n, y_n) \quad n = 0, 1, \cdots, N-1 \qquad (EE) \end{cases}$$

This is called EXPLICIT EULER method or FORWARD EULER method: at each step, the value y_n can be explicitly computed using values at the previous steps. It is very simple and inexpensive but, as we shall see, there is a "but"...

Examples of numerical schemes

Example 2 This time we consider the quadrature formula

$$\int_c^d g(s)ds \simeq (d-c)\,g(d) \tag{4}$$

that is also very poor and is exact only if $g = constant$, like the previous one. However the resulting scheme will be very different.

Examples of numerical schemes

Example 2 This time we consider the quadrature formula

$$\int_c^d g(s)ds \simeq (d-c)\, g(d) \tag{4}$$

that is also very poor and is exact only if $g = \mathit{constant}$, like the previous one. However the resulting scheme will be very different.
In fact, applying it to our case we get

$$\int_{t_n}^{t_{n+1}} f(t, y(t))dt \simeq (t_{n+1} - t_n) f(t_{n+1}, y(t_{n+1})) \quad n = 0, 1, 2, \cdots$$

Examples of numerical schemes

Example 2 This time we consider the quadrature formula

$$\int_c^d g(s)ds \simeq (d-c)\,g(d) \tag{4}$$

that is also very poor and is exact only if $g = constant$, like the previous one. However the resulting scheme will be very different.
In fact, applying it to our case we get

$$\int_{t_n}^{t_{n+1}} f(t,y(t))dt \simeq (t_{n+1}-t_n)f(t_{n+1},y(t_{n+1})) \quad n = 0,1,2,\cdots$$

that used into $y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} y'(t)dt = \int_{t_n}^{t_{n+1}} f(t,y(t))dt$ gives

Example 2 This time we consider the quadrature formula

$$\int_c^d g(s)ds \simeq (d-c)\,g(d) \tag{4}$$

that is also very poor and is exact only if $g = constant$, like the previous one. However the resulting scheme will be very different.
In fact, applying it to our case we get

$$\int_{t_n}^{t_{n+1}} f(t,y(t))dt \simeq (t_{n+1}-t_n)f(t_{n+1},y(t_{n+1})) \quad n = 0,1,2,\cdots$$

that used into $y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} y'(t)dt = \int_{t_n}^{t_{n+1}} f(t,y(t))dt$ gives

$$\begin{aligned}
&y(t_1) \simeq y(t_0) + \Delta t\, f(t_1, y(t_1)) \simeq y_0 + \Delta t\, f(t_1, y_1) =: y_1 \\
&y(t_2) \simeq y(t_1) + \Delta t\, f(t_2, y(t_2)) \simeq y_1 + \Delta t\, f(t_2, y_2) =: y_2 \\
&\vdots \\
&y(t_N) \simeq y(t_{N-1}) + \Delta t\, f(t_N, y(t_N)) \simeq y_{N-1} + \Delta t\, f(t_N, y_N) =: y_N
\end{aligned}$$

The scheme becomes

$$\begin{cases} y_0 \text{ given} \\ y_{n+1} = y_n + \Delta t\, f(t_{n+1}, y_{n+1}) \quad n = 0, 1, \cdots, N-1 \end{cases} \qquad (IE)$$

The scheme becomes

$$\begin{cases} y_0 \text{ given} \\ y_{n+1} = y_n + \Delta t\, f(t_{n+1}, y_{n+1}) \quad n = 0, 1, \cdots, N-1 \end{cases} \qquad (IE)$$

This is called IMPLICIT EULER method or BACKWARD EULER method. Note that, at every time step, the unknown y_{n+1} in (IE) appears both on the left-hand side *and in the right-hand side*, and in order to perform the step we must *solve an equation* in the unknown y_{n+1}. Since f is in general non-linear, at each step, to find y_n we need to solve a non-linear equation (for example, with Newton method). The method is obviously more expensive than Explicit Euler.

Example 3 As a third example we consider the quadrature formula

$$\int_c^d g(s)ds \simeq (d-c)\left(\frac{g(c)+g(d)}{2}\right) \tag{5}$$

(trapezoidal rule) that is better than the previous ones since it is exact whenever g is a polynomial of degree ≤ 1.

Example 3 As a third example we consider the quadrature formula

$$\int_c^d g(s)ds \simeq (d-c)\left(\frac{g(c)+g(d)}{2}\right) \tag{5}$$

(trapezoidal rule) that is better than the previous ones since it is exact whenever g is a polynomial of degree ≤ 1.
Applying it to our case we get

$$\int_{t_n}^{t_{n+1}} f(t,y(t))dt \simeq \frac{(t_{n+1}-t_n)}{2}\Big(f(t_n,y(t_n))+f(t_{n+1},y(t_{n+1}))\Big) \quad \forall n$$

Examples of numerical schemes

Example 3 As a third example we consider the quadrature formula

$$\int_c^d g(s)ds \simeq (d-c)\left(\frac{g(c)+g(d)}{2}\right) \tag{5}$$

(trapezoidal rule) that is better than the previous ones since it is exact whenever g is a polynomial of degree ≤ 1.
Applying it to our case we get

$$\int_{t_n}^{t_{n+1}} f(t,y(t))dt \simeq \frac{(t_{n+1}-t_n)}{2}\Big(f(t_n,y(t_n))+f(t_{n+1},y(t_{n+1}))\Big) \quad \forall n$$

The corresponding scheme becomes

$$\begin{cases} y_0 \text{ given} \\ y_{n+1} = y_n + \dfrac{\Delta t}{2}\Big(f(t_n,y_n)+f(t_{n+1},y_{n+1})\Big) \ n = 0,1,\cdots,N-1 \end{cases}$$

Examples of numerical schemes

$$\begin{cases} y_0 \text{ given} \\ y_{n+1} = y_n + \dfrac{\Delta t}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\Big) \; n = 0, 1, \cdots, N-1 \end{cases}$$

This is called CRANK-NICOLSON method. It is an implicit method (and hence, as the previous Implicit Euler, expensive) but it has a good accuracy, as we shall see.

Examples of numerical schemes

Explicit methods could be derived from an implicit scheme:

- 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}

Example 4 In the Crank-Nicolson scheme, use Explicit Euler as a predictor, we get rid of the implicit part and obtain a new explicit method, called HEUN method, which reads

$$\begin{cases} y_0 \text{ given} \\ y^*_{n+1} = y_n + \Delta t\, f(t_n, y_n) \\ y_{n+1} = y_n + \dfrac{\Delta t}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y^*_{n+1})\Big) \quad n = 0, 1, \cdots, N-1 \end{cases}$$

Systems of Ode's

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

$$\begin{cases} \underline{Y}'(t) = \underline{F}(t, \underline{Y}(t)) \quad t \in [t_0, T] \\ \underline{Y}(t_0) = \underline{Y}^{(0)}. \end{cases} \tag{6}$$

Systems of Ode's

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

$$\begin{cases} \underline{Y}'(t) = \underline{F}(t, \underline{Y}(t)) \quad t \in [t_0, T] \\ \underline{Y}(t_0) = \underline{Y}^{(0)}. \end{cases} \tag{6}$$

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 Ode's

The numerical schemes used for a single equation apply directly to systems of Ode's.
For instance, the two Euler methods become:

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

Systems of Ode's

The numerical schemes used for a single equation apply directly to systems of Ode's.
For instance, the two Euler methods become:

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

Systems of Ode's

The numerical schemes used for a single equation apply directly to systems of Ode's.
For instance, the two Euler methods become:

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

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

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

Systems of Ode's

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

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

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

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

Systems of Ode's

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

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

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

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*!

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

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

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

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

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 $Y^{(n+1)} := \underline{X}$
One Newton would give:

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

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

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 $Y^{(n+1)} := \underline{X}$
One Newton would give:

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

$$\underline{X}^{(0)} = ??$$

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

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

One Newton would give:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Then set:

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

Study of convergence

So we have two classes: explicit methods, and implicit methods.

Study of convergence

So we have two classes: explicit methods, and implicit methods.

In all cases we want the sequence $\{y_0, y_1, \cdots, y_N\}$ to converge to the sequence $\{y_0, y(t_1), \cdots, y(T)\}$.

Study of convergence

So we have two classes: explicit methods, and implicit methods.
In all cases we want the sequence $\{y_0, y_1, \cdots, y_N\}$ to converge to the sequence $\{y_0, y(t_1), \cdots, y(T)\}$.

If, given a method, we can prove that

$$\exists C > 0 \text{ such that} \qquad \max_n |y_n - y(t_n)| \leq C\Delta t^p$$

with C independent of Δt and $p > 0$, then we say that *the method is convergent*, and *the order of convergence is p* (the bigger p, the faster the convergence).

Theorem 1 (Lax)

If a scheme is **consistent** *and* **stable**, *then it is convergent, and the order of convergence is the order of consistency.*

We have however yet to define what *consistency* and *stability* are....

Intuitive explanation of *consistency* and *stability*

- *consistency* is a measure of how much the discrete scheme resembles the differential problem: the *consistency error* at a given time step measures the error which is created at that step; it is defined as *the error made when the scheme is applied to the exact solution of the problem*.

- *stability* measures how the error, created and accumulated during the previous steps, goes to the next step (is it amplified? does it decay?...)

The detailed definition is given at the end of these slides for the methods introduced, but is not a topic for the final exam.

Consistency of a scheme

Consistency is a measure of how much the discrete scheme resembles the differential problem: the *consistency error* is *the error made when the scheme is applied to the exact solution of the problem*.

.

Consistency of a scheme

Consistency is a measure of how much the discrete scheme resembles the differential problem: the *consistency error* is *the error made when the scheme is applied to the exact solution of the problem*.

Denoting by τ the consistency error of a given scheme, if we have

$$\tau \leq C\Delta t^p$$

for some positive constant C independent of Δt and p positive, we say that the scheme is consistent (i.e., $\tau \to 0$ for $\Delta t \to 0$) and the order of consistency is p.

.

Consistency of a scheme

Consistency is a measure of how much the discrete scheme resembles the differential problem: the *consistency error* is *the error made when the scheme is applied to the exact solution of the problem*.

Denoting by τ the consistency error of a given scheme, if we have

$$\tau \leq C\Delta t^p$$

for some positive constant C independent of Δt and p positive, we say that the scheme is consistent (i.e., $\tau \to 0$ for $\Delta t \to 0$) and the order of consistency is p.

Let us see how to check consistency for the simplest scheme, Explicit Euler.

Consistency of Explicit Euler

the exact solution fulfils: $y'(t_n) - f(t_n, y(t_n)) = 0, \quad n = 0, 1, 2, \ldots$
the discrete solution fulfils: $\frac{y_{n+1} - y_n}{\Delta t} - f(t_n, y_n) = 0, \quad n = 0, 1, 2, \ldots$

Consistency of Explicit Euler

the exact solution fulfils: $y'(t_n) - f(t_n, y(t_n)) = 0, \quad n = 0, 1, 2, ...$
the discrete solution fulfils: $\frac{y_{n+1} - y_n}{\Delta t} - f(t_n, y_n) = 0, \quad n = 0, 1, 2, ...$
If we apply the EE scheme to the exact solution we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_n, y(t_n)) \neq 0 \quad n = 0, 1, 2, ...$$

Consistency of Explicit Euler

the exact solution fulfils: $y'(t_n) - f(t_n, y(t_n)) = 0, \quad n = 0, 1, 2, \ldots$
the discrete solution fulfils: $\frac{y_{n+1} - y_n}{\Delta t} - f(t_n, y_n) = 0, \quad n = 0, 1, 2, \ldots$
If we apply the EE scheme to the exact solution we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_n, y(t_n)) \neq 0 \quad n = 0, 1, 2, \ldots.$$

The quantity τ_n is called *the local truncation error at the step n*, and the *consistency error* will be $\tau = \max_n |\tau_n|$.

Consistency of Explicit Euler

the exact solution fulfils: $y'(t_n) - f(t_n, y(t_n)) = 0, \quad n = 0, 1, 2, \dots$
the discrete solution fulfils: $\frac{y_{n+1}-y_n}{\Delta t} - f(t_n, y_n) = 0, \quad n = 0, 1, 2, \dots$
If we apply the EE scheme to the exact solution we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_n, y(t_n)) \neq 0 \quad n = 0, 1, 2, \dots.$$

The quantity τ_n is called *the local truncation error at the step n*, and the *consistency error* will be $\tau = \max_n |\tau_n|$.
Since $y'(t_n) = f(t_n, y(t_n))$ by Taylor expansion at t_n we have, for z such that $t_n < z < t_{n+1}$, that $y(t_{n+1}) = y(t_n) + \Delta t\, f(t_n, y(t_n)) + \frac{\Delta t^2}{2} y''(z)$

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_n, y(t_n)) = \frac{\Delta t}{2} y''(z),$$

$$\tau = \max |\tau_n| \leq \frac{\Delta t}{2} \max_{t \in [t_0, T]} |y''(t)| = C\Delta t.$$

Consistency of Explicit Euler

the exact solution fulfils: $y'(t_n) - f(t_n, y(t_n)) = 0, \quad n = 0, 1, 2, \dots$
the discrete solution fulfils: $\frac{y_{n+1} - y_n}{\Delta t} - f(t_n, y_n) = 0, \quad n = 0, 1, 2, \dots$
If we apply the EE scheme to the exact solution we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_n, y(t_n)) \neq 0 \quad n = 0, 1, 2, \dots.$$

The quantity τ_n is called *the local truncation error at the step n*, and the *consistency error* will be $\tau = \max_n |\tau_n|$.
Since $y'(t_n) = f(t_n, y(t_n))$ by Taylor expansion at t_n we have, for z such that $t_n < z < t_{n+1}$, that $y(t_{n+1}) = y(t_n) + \Delta t\, f(t_n, y(t_n)) + \frac{\Delta t^2}{2} y''(z)$

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_n, y(t_n)) = \frac{\Delta t}{2} y''(z),$$

$$\tau = \max |\tau_n| \leq \frac{\Delta t}{2} \max_{t \in [t_0, T]} |y''(t)| = C\Delta t.$$

Consistency of Explicit Euler

the exact solution fulfils: $y'(t_n) - f(t_n, y(t_n)) = 0, \quad n = 0, 1, 2, \ldots$
the discrete solution fulfils: $\frac{y_{n+1}-y_n}{\Delta t} - f(t_n, y_n) = 0, \quad n = 0, 1, 2, \ldots$
If we apply the EE scheme to the exact solution we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_n, y(t_n)) \neq 0 \quad n = 0, 1, 2, \ldots.$$

The quantity τ_n is called *the local truncation error at the step n*, and the *consistency error* will be $\tau = \max_n |\tau_n|$.
Since $y'(t_n) = f(t_n, y(t_n))$ by Taylor expansion at t_n we have, for z such that $t_n < z < t_{n+1}$, that $y(t_{n+1}) = y(t_n) + \Delta t\, f(t_n, y(t_n)) + \frac{\Delta t^2}{2} y''(z)$

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_n, y(t_n)) = \frac{\Delta t}{2} y''(z),$$

$$\tau = \max |\tau_n| \leq \frac{\Delta t}{2} \max_{t \in [t_0, T]} |y''(t)| = C \Delta t.$$

Thus, Explicit Euler scheme is consistent with order of consistency 1.

Consistency of Implicit Euler method

$$\frac{y_{n+1} - y_n}{\Delta t} - f(t_{n+1}, y_{n+1}) = 0 \quad n = 0, 1, 2, ...$$

Consistency of Implicit Euler method

$$\frac{y_{n+1} - y_n}{\Delta t} - f(t_{n+1}, y_{n+1}) = 0 \quad n = 0, 1, 2, ...$$

Applying this scheme to the exact solution $y(\cdot)$ we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_{n+1}, y(t_{n+1})) \neq 0 \quad n = 0, 1, 2, ...$$

Consistency of Implicit Euler method

$$\frac{y_{n+1} - y_n}{\Delta t} - f(t_{n+1}, y_{n+1}) = 0 \quad n = 0, 1, 2, ...$$

Applying this scheme to the exact solution $y(\cdot)$ we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_{n+1}, y(t_{n+1})) \neq 0 \quad n = 0, 1, 2, ...$$

Since $f(t_{n+1}, y(t_{n+1})) = y'(t_{n+1})$, we use Taylor expansion at t_{n+1}:

$$\frac{y(t_n) - y(t_{n+1})}{\Delta t} = -y'(t_{n+1}) + \frac{\Delta t}{2} y''(z) \quad t_n < z < t_{n+1}.$$

Consistency of Implicit Euler method

$$\frac{y_{n+1} - y_n}{\Delta t} - f(t_{n+1}, y_{n+1}) = 0 \quad n = 0, 1, 2, ...$$

Applying this scheme to the exact solution $y(\cdot)$ we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_{n+1}, y(t_{n+1})) \neq 0 \quad n = 0, 1, 2, ...$$

Since $f(t_{n+1}, y(t_{n+1})) = y'(t_{n+1})$, we use Taylor expansion at t_{n+1}:

$$\frac{y(t_n) - y(t_{n+1})}{\Delta t} = -y'(t_{n+1}) + \frac{\Delta t}{2} y''(z) \quad t_n < z < t_{n+1}.$$

Hence we obtain

$$\tau_n = -\frac{\Delta t}{2} y''(z) \implies \tau = \max |\tau_n| \leq \frac{\Delta t}{2} \max_{t \in [t_0, T]} |y''(t)| = C\Delta t.$$

Consistency of Implicit Euler method

$$\frac{y_{n+1} - y_n}{\Delta t} - f(t_{n+1}, y_{n+1}) = 0 \quad n = 0, 1, 2, ...$$

Applying this scheme to the exact solution $y(\cdot)$ we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_{n+1}, y(t_{n+1})) \neq 0 \quad n = 0, 1, 2, ...$$

Since $f(t_{n+1}, y(t_{n+1})) = y'(t_{n+1})$, we use Taylor expansion at t_{n+1}:

$$\frac{y(t_n) - y(t_{n+1})}{\Delta t} = -y'(t_{n+1}) + \frac{\Delta t}{2} y''(z) \quad t_n < z < t_{n+1}.$$

Hence we obtain

$$\tau_n = -\frac{\Delta t}{2} y''(z) \implies \tau = \max |\tau_n| \leq \frac{\Delta t}{2} \max_{t \in [t_0, T]} |y''(t)| = C\Delta t.$$

Thus, Implicit Euler scheme is consistent with order of consistency 1.

Consistency of the Crank-Nicolson method

$$\frac{y_{n+1} - y_n}{\Delta t} - \frac{1}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\Big) = 0 \quad n = 0, 1, 2, ...$$

Consistency of the Crank-Nicolson method

$$\frac{y_{n+1} - y_n}{\Delta t} - \frac{1}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\Big) = 0 \quad n = 0, 1, 2, ...$$

Applying this scheme to the exact solution $y(\cdot)$ we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - \frac{1}{2}\Big(f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1}))\Big)$$

Consistency of the Crank-Nicolson method

$$\frac{y_{n+1} - y_n}{\Delta t} - \frac{1}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\Big) = 0 \quad n = 0, 1, 2, ...$$

Applying this scheme to the exact solution $y(\cdot)$ we will have

$$\begin{aligned}\tau_n &= \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - \frac{1}{2}\Big(f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1}))\Big)\\ &= \frac{1}{2}\Big(\frac{y(t_{n+1}) - y(t_n)}{\Delta t} - y'(t_n)\Big) + \frac{1}{2}\Big(\frac{y(t_{n+1}) - y(t_n)}{\Delta t} - y'(t_{n+1})\Big).\end{aligned}$$

Consistency of the Crank-Nicolson method

$$\frac{y_{n+1}-y_n}{\Delta t}-\frac{1}{2}\Big(f(t_n,y_n)+f(t_{n+1},y_{n+1})\Big)=0 \quad n=0,1,2,.....$$

Applying this scheme to the exact solution $y(\cdot)$ we will have

$$\tau_n=\frac{y(t_{n+1})-y(t_n)}{\Delta t}-\frac{1}{2}\Big(f(t_n,y(t_n))+f(t_{n+1},y(t_{n+1}))\Big)$$
$$=\frac{1}{2}\Big(\frac{y(t_{n+1})-y(t_n)}{\Delta t}-y'(t_n)\Big)+\frac{1}{2}\Big(\frac{y(t_{n+1})-y(t_n)}{\Delta t}-y'(t_{n+1})\Big).$$

Taylor expansion centered at t_n for the first term, at t_{n+1} for the second:

$$\tau_n=\frac{1}{2}(\frac{\Delta t}{2}y''(z_1))+\frac{1}{2}(-\frac{\Delta t}{2}y''(z_2))=\frac{1}{4}\Delta t(z_1-z_2)y'''(z_3)\leq\frac{\Delta t^2}{4}y'''(z_3)$$

Consistency of the Crank-Nicolson method

$$\frac{y_{n+1}-y_n}{\Delta t}-\frac{1}{2}\Big(f(t_n,y_n)+f(t_{n+1},y_{n+1})\Big)=0 \quad n=0,1,2,.....$$

Applying this scheme to the exact solution $y(\cdot)$ we will have

$$\tau_n=\frac{y(t_{n+1})-y(t_n)}{\Delta t}-\frac{1}{2}\Big(f(t_n,y(t_n))+f(t_{n+1},y(t_{n+1}))\Big)$$
$$=\frac{1}{2}\Big(\frac{y(t_{n+1})-y(t_n)}{\Delta t}-y'(t_n)\Big)+\frac{1}{2}\Big(\frac{y(t_{n+1})-y(t_n)}{\Delta t}-y'(t_{n+1})\Big).$$

Taylor expansion centered at t_n for the first term, at t_{n+1} for the second:

$$\tau_n=\frac{1}{2}(\frac{\Delta t}{2}y''(z_1))+\frac{1}{2}(-\frac{\Delta t}{2}y''(z_2))=\frac{1}{4}\Delta t(z_1-z_2)y'''(z_3)\leq\frac{\Delta t^2}{4}y'''(z_3)$$

where above we used the mean value theorem: $\frac{y''(z_1)-y''(z_2)}{z_1-z_2}=y'''(z_3)$

Consistency of the Crank-Nicolson method

$$\frac{y_{n+1} - y_n}{\Delta t} - \frac{1}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y_{n+1}) \Big) = 0 \quad n = 0, 1, 2, ...$$

Applying this scheme to the exact solution $y(\cdot)$ we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - \frac{1}{2}\Big(f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1})) \Big)$$

$$= \frac{1}{2}\Big(\frac{y(t_{n+1}) - y(t_n)}{\Delta t} - y'(t_n) \Big) + \frac{1}{2}\Big(\frac{y(t_{n+1}) - y(t_n)}{\Delta t} - y'(t_{n+1}) \Big).$$

Taylor expansion centered at t_n for the first term, at t_{n+1} for the second:

$$\tau_n = \frac{1}{2}(\frac{\Delta t}{2} y''(z_1)) + \frac{1}{2}(-\frac{\Delta t}{2} y''(z_2)) = \frac{1}{4}\Delta t(z_1 - z_2) y'''(z_3) \leq \frac{\Delta t^2}{4} y'''(z_3)$$

$$\Longrightarrow |\tau_n| \leq \frac{\Delta t^2}{4} |y'''(z_3)| \implies \tau = \max |\tau_n| \leq \frac{\Delta t^2}{4} \max_{t \in [t_0, T]} |y'''(t)| = C \Delta t^2.$$

Thus, Crank-Nicolson scheme is consistent with order of consistency 2.

A closer look at consistency error

We found that:

- for the two Euler methods, the consistency error is zero whenever $y'' \equiv 0$, that is, whenever the solution of (1) is a polynomial of degree up to 1.
- the consistency error for Crank-Nicolson is zero whenever $y''' \equiv 0$, that is, whenever the solution of (1) is a polynomial of degree up to 2.

This suggests that *to have order of consistency p* means that *the scheme computes exactly the solution of (1) whenever this solution is a polynomial of degree up to p*.

This is another way of checking consistency of a scheme, less rigorous but practical, and this is how we will check consistency for Heun scheme.

Consistency of Heun method

$$\frac{y_{n+1} - y_n}{\Delta t} - \frac{1}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y_n + \Delta t f(t_n, y_n)) \Big) = 0 \quad \forall n$$

Consistency of Heun method

$$\frac{y_{n+1} - y_n}{\Delta t} - \frac{1}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y_n + \Delta t f(t_n, y_n))\Big) = 0 \quad \forall n$$

Applying this scheme to the exact solution of (1) we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - \frac{1}{2}\Big(y'(t_n) + f(t_{n+1}, y(t_n) + \Delta t\, y'(t_n))\Big) \neq 0.$$

Consistency of Heun method

$$\frac{y_{n+1} - y_n}{\Delta t} - \frac{1}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y_n + \Delta t f(t_n, y_n))\Big) = 0 \quad \forall n$$

Applying this scheme to the exact solution of (1) we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - \frac{1}{2}\Big(y'(t_n) + f(t_{n+1}, y(t_n) + \Delta t\, y'(t_n))\Big) \neq 0.$$

This scheme originated from Crank-Nicolson, by making explicit the implicit part. Is the order of consistency still 2? For this we should have $\tau_n \equiv 0$ when the solution of (1) is $1, t, t^2$.

Consistency of Heun method

$$\frac{y_{n+1} - y_n}{\Delta t} - \frac{1}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y_n + \Delta t f(t_n, y_n))\Big) = 0 \quad \forall n$$

Applying this scheme to the exact solution of (1) we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - \frac{1}{2}\Big(y'(t_n) + f(t_{n+1}, y(t_n) + \Delta t\, y'(t_n))\Big) \neq 0.$$

This scheme originated from Crank-Nicolson, by making explicit the implicit part. Is the order of consistency still 2? For this we should have $\tau_n \equiv 0$ when the solution of (1) is $1, t, t^2$.

When $y(t) = 1, y' = 0 = f, \Longrightarrow \tau_n = \dfrac{1-1}{\Delta t} - \dfrac{1}{2}(0+0) = 0;$

Consistency of Heun method

$$\frac{y_{n+1} - y_n}{\Delta t} - \frac{1}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y_n + \Delta t f(t_n, y_n))\Big) = 0 \quad \forall n$$

Applying this scheme to the exact solution of (1) we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - \frac{1}{2}\Big(y'(t_n) + f(t_{n+1}, y(t_n) + \Delta t\, y'(t_n))\Big) \neq 0.$$

This scheme originated from Crank-Nicolson, by making explicit the implicit part. Is the order of consistency still 2? For this we should have $\tau_n \equiv 0$ when the solution of (1) is $1, t, t^2$.

When $y(t) = 1, y' = 0 = f, \Longrightarrow \tau_n = \dfrac{1-1}{\Delta t} - \dfrac{1}{2}(0+0) = 0;$

When $y(t) = t, y' = 1 = f, \Longrightarrow \tau_n = \dfrac{t_{n+1} - t_n}{\Delta t} - \dfrac{1}{2}(1+1) = 1 - 1 = 0;$

Consistency of Heun method

$$\frac{y_{n+1}-y_n}{\Delta t}-\frac{1}{2}\Big(f(t_n,y_n)+f(t_{n+1},y_n+\Delta t f(t_n,y_n))\Big)=0 \quad \forall n$$

Applying this scheme to the exact solution of (1) we will have

$$\tau_n=\frac{y(t_{n+1})-y(t_n)}{\Delta t}-\frac{1}{2}\Big(y'(t_n)+f(t_{n+1},y(t_n)+\Delta t\, y'(t_n))\Big)\neq 0.$$

This scheme originated from Crank-Nicolson, by making explicit the implicit part. Is the order of consistency still 2? For this we should have $\tau_n \equiv 0$ when the solution of (1) is $1, t, t^2$.

When $y(t)=1, y'=0=f, \Longrightarrow \tau_n=\dfrac{1-1}{\Delta t}-\dfrac{1}{2}(0+0)=0;$

When $y(t)=t, y'=1=f, \Longrightarrow \tau_n=\dfrac{t_{n+1}-t_n}{\Delta t}-\dfrac{1}{2}(1+1)=1-1=0;$

When $y(t)=t^2, y'=2t=f, \Longrightarrow \tau_n=\dfrac{t_{n+1}^2-t_n^2}{\Delta t}-\dfrac{1}{2}(2t_n+2t_{n+1})=0$

Consistency of Heun method

$$\frac{y_{n+1} - y_n}{\Delta t} - \frac{1}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y_n + \Delta t f(t_n, y_n))\Big) = 0 \quad \forall n$$

Applying this scheme to the exact solution of (1) we will have

$$\tau_n = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - \frac{1}{2}\Big(y'(t_n) + f(t_{n+1}, y(t_n) + \Delta t\, y'(t_n))\Big) \neq 0.$$

This scheme originated from Crank-Nicolson, by making explicit the implicit part. Is the order of consistency still 2? For this we should have $\tau_n \equiv 0$ when the solution of (1) is $1, t, t^2$.

When $y(t) = 1, y' = 0 = f, \Longrightarrow \tau_n = \dfrac{1-1}{\Delta t} - \dfrac{1}{2}(0+0) = 0;$

When $y(t) = t, y' = 1 = f, \Longrightarrow \tau_n = \dfrac{t_{n+1} - t_n}{\Delta t} - \dfrac{1}{2}(1+1) = 1 - 1 = 0;$

When $y(t) = t^2, y' = 2t = f, \Longrightarrow \tau_n = \dfrac{t_{n+1}^2 - t_n^2}{\Delta t} - \dfrac{1}{2}(2t_n + 2t_{n+1}) = 0$

Hence, the order of consistency of Heun method is 2.

Stability

The concept of *stability* is a very important and useful concept whose precise definition has to be made precise at various occurrences. Roughly speaking, stability is what guarantees that the errors generated during a numerical procedure do not grow too much.

With Ode's stability is a delicate issue, especially when the phenomenon under study has to be followed for a long time. To better see what happens, let us consider a simple model problem, for which we know the exact solution:

$$\begin{cases} y'(t) = ay(t) \quad t > 0 \\ y(0) = y_0 \end{cases}$$

Stability

The concept of *stability* is a very important and useful concept whose precise definition has to be made precise at various occurrences. Roughly speaking, stability is what guarantees that the errors generated during a numerical procedure do not grow too much.

With Ode's stability is a delicate issue, especially when the phenomenon under study has to be followed for a long time. To better see what happens, let us consider a simple model problem, for which we know the exact solution:

$$\begin{cases} y'(t) = ay(t) & t > 0 \\ y(0) = y_0 \end{cases} \qquad \text{exact solution: } y(t) = y_0 e^{at}$$

If $a > 0$ (or $\operatorname{Re} a > 0$ if $a \in \mathbb{C}$) the exact solution *grows* exponentially. We cannot expect (and we do not want!) that the discrete scheme remains bounded, and it is not even the case to discuss "stability".

If $a > 0$ (or $\mathrm{Re}\, a > 0$ if $a \in \mathbb{C}$) the exact solution *grows* exponentially. We cannot expect (and we do not want!) that the discrete scheme remains bounded, and it is not even the case to discuss "stability".
Instead, if $a < 0$ the exact solution not only is bounded, but *decays* exponentially:

$$a < 0 \longrightarrow |y(t)| \leq |y_0| \quad \textbf{and} \quad \lim_{t \to \infty} |y(t)| = 0.$$

If $a > 0$ (or $\operatorname{Re} a > 0$ if $a \in \mathbb{C}$) the exact solution *grows* exponentially. We cannot expect (and we do not want!) that the discrete scheme remains bounded, and it is not even the case to discuss "stability".
Instead, if $a < 0$ the exact solution not only is bounded, but *decays* exponentially:

$$a < 0 \;\longrightarrow\; |y(t)| \leq |y_0| \quad \textbf{and} \quad \lim_{t\to\infty} |y(t)| = 0.$$

When a is a complex number, the exact solution is given by $y(t) = y_0 e^{(\operatorname{Re} a)t}(\cos((\operatorname{Im} a)t) + i \sin((\operatorname{Im} a)t))$, and has the same behaviour if $\operatorname{Re} a < 0$:

$$a \in \mathbb{C} \text{ with } \operatorname{Re} a < 0 \;\longrightarrow\; |y(t)| \leq |y_0| \quad \textbf{and} \quad \lim_{t\to\infty} |y(t)| = 0.$$

If $a > 0$ (or $\operatorname{Re} a > 0$ if $a \in \mathbb{C}$) the exact solution *grows* exponentially. We cannot expect (and we do not want!) that the discrete scheme remains bounded, and it is not even the case to discuss "stability".
Instead, if $a < 0$ the exact solution not only is bounded, but *decays* exponentially:

$$a < 0 \;\longrightarrow\; |y(t)| \le |y_0| \quad \textbf{and} \quad \lim_{t\to\infty} |y(t)| = 0.$$

When a is a complex number, the exact solution is given by $y(t) = y_0 e^{(\operatorname{Re} a)t}(\cos((\operatorname{Im} a)t) + i\sin((\operatorname{Im} a)t))$, and has the same behaviour if $\operatorname{Re} a < 0$:

$$a \in \mathbb{C} \text{ with } \operatorname{Re} a < 0 \;\longrightarrow\; |y(t)| \le |y_0| \quad \textbf{and} \quad \lim_{t\to\infty} |y(t)| = 0.$$

In this case we need to analyse the discrete schemes, and see whether the discrete solution decays too, and behaves like the exact solution. Hence, let $a < 0$ (or $\operatorname{Re} a < 0$ if $a \in \mathbb{C}$), and let $\{y_n\}$ be the sequence generated by a numerical scheme. Does $\{y_n\}$ satisfy the following relation?

$$a \in \mathbb{C} \text{ with } \operatorname{Re} a < 0 \;\longrightarrow\; |y_n| \le |y_0| \quad \textbf{and} \quad \lim_{n\to\infty} |y_n| = 0?$$

If this happens, the scheme is called *Absolutely stable*, or *A-stable*.

Checking stability for Explicit Euler

By applying **Explicit Euler** method to the model problem, we get ($y_{n+1} = y_n + \Delta t f(t_n, y_n)$ with $f(t_n, y_n) = a y_n$)

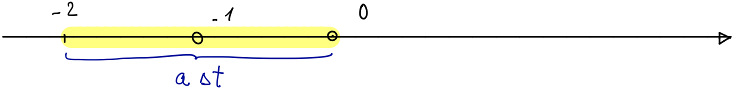
$$y_{n+1} = (1 + a\Delta t) y_n \quad n = 0, 1, \dots . \implies y_n = y_0 (1 + a\Delta t)^n.$$

The exact solution decays exponentially from the initial value y_0, while the growth-decay factor for the discrete scheme is $G = 1 + a\Delta t$.
For having $\lim_{n\to\infty} |y_n| = 0$ we need $|G| < 1$. Since a is negative, we always have $G = 1 + a\Delta t < 1$, but to have $G > -1$ we need to satisfy the condition

$$\boxed{1 + a\Delta t > -1, \text{ that is, } \Delta t < \left(2/|a|\right) =: \textit{stability condition for EE}}$$

This is the drawback of Explicit Euler scheme, and of all the explicit schemes: for small enough time steps the stability condition is satisfied, but when a is strongly negative (exactly the case of rapid decay in the true solution) we are compelled to keep Δt small.

Stability for Explicit Euler: the real setting

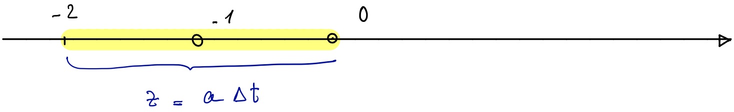


EE is stable $\iff |1 + a\Delta t| < 1$

$\iff -2 < a\Delta t < 0$

$\iff -\dfrac{2}{a} > \Delta t > 0$ (since $a<0$)

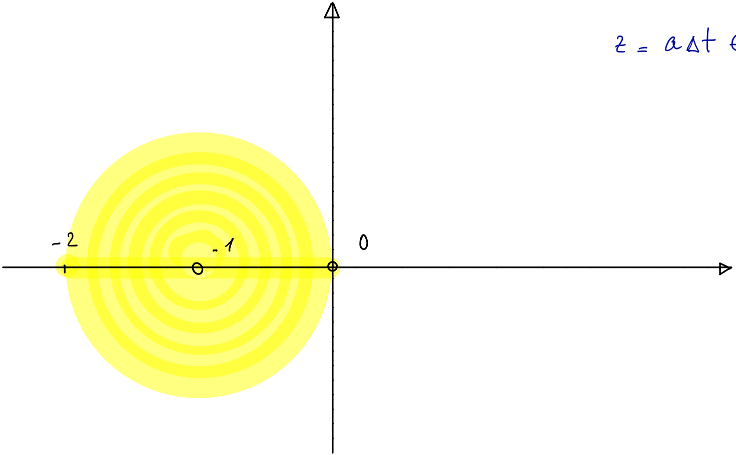
Stability for Explicit Euler: the real setting



$$EE \text{ is stable} \iff |1 + a\Delta t| < 1$$

$$\iff -2 < \underbrace{a\Delta t}_{z} < 0$$

Stability for Explicit Euler: the complex setting



$z = a\Delta t \in \mathbb{C}$

EE is stable $\iff$ $|1 + \overbrace{a\Delta t}^{z}| < 1$

$\iff$ $z \in$ circle with radius 1 and center -1

Checking stability for Implicit Euler

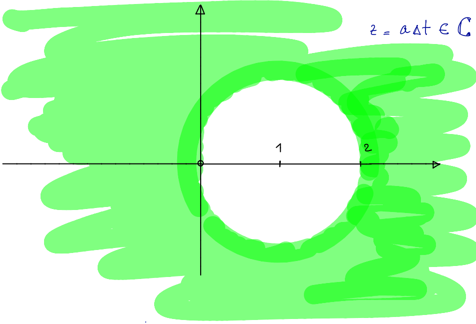
Applying **Implicit Euler** method to the model problem gives ($y_{n+1} = y_n + \Delta t f(t_{n+1}, y_{n+1})$ with $f(t_{n+1}, y_{n+1}) = a y_{n+1}$)

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

- For negative a the growth-decay factor G is positive, and the denominator is always larger than 1. Therefore $|G| < 1$ and we always have decay.
- $|G| < 1$ holds also if a is any complex number in the left-half plane

Then we say that Implicit Euler is A-stable: the A-stability condition can be written as

$$\boxed{\text{If Re } a < 0 \text{ then } |G| < 1 \qquad =: \textit{A-stability}}$$



The scheme is A-stable:

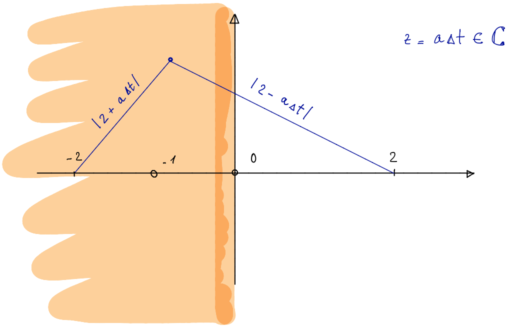
$$|G| < 1 \Leftrightarrow |1 - a\Delta t| > 1 \Leftrightarrow a\Delta t \text{ is outside the circle above} \Leftarrow \mathrm{Re}\, a < 0$$

Crank-Nicolson gives

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

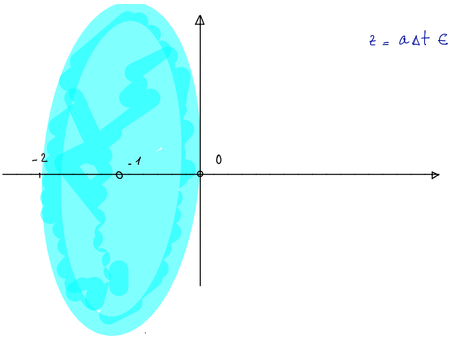
The scheme is A-stable:

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



Checking stability for Heun

Heun method is not A-stable, only conditionally A-stable like Explicit Euler. In fact, $G = 1 + a\Delta t + \frac{(a\Delta t)^2}{2}$, and the condition $|G| < 1$ is satisfied if $\Delta t < \Big(2/|a|\Big)$.



$z = a\Delta t \in \mathbb{C}$

Stability regions: comparison

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

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

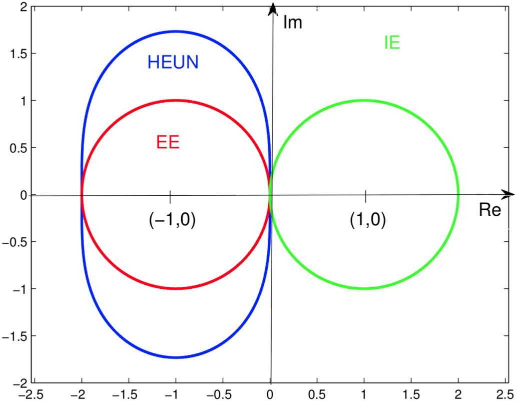
and compared it with the stability region of the continuous problem: the half plane Re $a < 0$.

For Explicit Euler $A = \{a\Delta t \in \mathbb{C} : |1 + a\Delta t| < 1\}$ is a circle 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 minus a circle 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 (the best you can have).

Stability regions



A-stability regions for EE (the region inside the red circle), Heun (the region inside the blue ellipse), IE (green, the region outside the circle with center $(1, 0)$ and radius 1

Conclusions (for the four basic methods)

In general, explicit schemes are never A-stable, only conditionally A-stable, meaning that to satisfy the A-stability property they need to proceed by small time steps. Some implicit schemes are A-stable.

For the method we have considered:

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

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.

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.

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

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

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

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

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 **

Suppose that both A and G are diagonalised. Then,

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

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

Suppose that both A and G are diagonalised. 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 $\operatorname{Re} \lambda_j < 0$): hence $e^{\lambda_j t} \to 0$ for $t \to \infty$.

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

Suppose that both A and G are diagonalised. 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 $\operatorname{Re}\lambda_j < 0$): hence $e^{\lambda_j t} \to 0$ for $t \to \infty$.

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

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

Suppose that both A and G are diagonalised. 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 $\operatorname{Re}\lambda_j < 0$): hence $e^{\lambda_j t} \to 0$ for $t \to \infty$.

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

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

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

Stiff systems ** NOT FOR THE EXAM **

When the matrix A has eigenvalues with very different magnitude, the system is called **stiff**.

Stiff systems ** NOT FOR THE EXAM **

When the matrix A has eigenvalues with very different magnitude, the system is called **stiff**.

Let us see a simple example.

$$\underline{Y}'(t) = \begin{bmatrix} -2 & 1 \\ 0 & -100 \end{bmatrix} \underline{Y}(t)$$

Stiff systems ** NOT FOR THE EXAM **

When the matrix A has eigenvalues with very different magnitude, the system is called **stiff**.

Let us see a simple example.

$$\underline{Y}'(t) = \begin{bmatrix} -2 & 1 \\ 0 & -100 \end{bmatrix} \underline{Y}(t) \longrightarrow \begin{array}{l} y_1'(t) = -2y_1(t) + y_2(t) \\ y_2'(t) = -100y_2(t) \end{array}$$

Stiff systems ** NOT FOR THE EXAM **

When the matrix A has eigenvalues with very different magnitude, the system is called **stiff**.

Let us see a simple example.

$$\underline{Y}'(t) = \begin{bmatrix} -2 & 1 \\ 0 & -100 \end{bmatrix} \underline{Y}(t) \longrightarrow \begin{array}{l} y_1'(t) = -2y_1(t) + y_2(t) \\ y_2'(t) = -100y_2(t) \end{array}$$

solution: $y_1(t) \simeq e^{-2t}, y_2(t) \simeq e^{-100t}$.

Stiff systems ** NOT FOR THE EXAM **

When the matrix A has eigenvalues with very different magnitude, the system is called **stiff**.

Let us see a simple example.

$$\underline{Y}'(t) = \begin{bmatrix} -2 & 1 \\ 0 & -100 \end{bmatrix} \underline{Y}(t) \longrightarrow \begin{array}{l} y_1'(t) = -2y_1(t) + y_2(t) \\ y_2'(t) = -100y_2(t) \end{array}$$

solution: $y_1(t) \simeq e^{-2t}, y_2(t) \simeq e^{-100t}$.

If we use Explicit Euler method we need

$$\Delta t < \frac{2}{|\lambda_1|} = 1 \quad \text{and} \quad \Delta t < \frac{2}{|\lambda_2|} = \frac{1}{50}.$$

Stability requires then $\Delta t < \frac{1}{50}$ even though it is e^{-2t} that controls the true solution: in fact, y_2 decays like e^{-100t} and dies out very fast, but its presence forces us to proceed by small time steps even when it has virtually disappeared and we are interested in following the e^{-2t} component.

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)

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 someway (and this is where the detailed description could become quite complicated).

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 someway (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:

$$\begin{cases} y_0 \text{ given} \\ y^*_{n+1} = y_n + \Delta t\, f(t_n, y_n) \\ y_{n+1} = y_n + \dfrac{\Delta t}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y^*_{n+1})\Big) \quad n = 0, 1, \cdots, N-1 \end{cases}$$

Heun scheme:

$$\begin{cases} y_0 \text{ given} \\ y^*_{n+1} = y_n + \Delta t\, f(t_n, y_n) \\ y_{n+1} = y_n + \dfrac{\Delta t}{2}\Big(f(t_n, y_n) + f(t_{n+1}, y^*_{n+1})\Big) \quad n = 0, 1, \cdots, N-1 \end{cases}$$

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:

$$K_1 = f(t_n, y_n), \quad K_2 = f(t_n + \Delta t, y_n + \Delta t\, K_1)$$

$$y_{n+1} = y_n + \frac{\Delta t}{2}\Big(K_1 + K_2\Big) \quad n = 0, 1, \cdots$$

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.

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

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

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

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

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

The 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

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

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

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

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

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

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

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

RK4

We had

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

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

RK4

We had
$K_1 = f(t_n, y_n)$, $K_2 = f^{(1)}_{n+1/2} = f(t_{n+1/2}, y_n + \frac{\Delta t}{2} K_1)$,
$K_3 = f^{(2)}_{n+1/2} = f(t_{n+1/2}, y_n + \frac{\Delta t}{2} K_2)$

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:

RK4

We had

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

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

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:

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

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

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

Explicit Runge-Kutta

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

$$y_0 \text{ given} \qquad y_{n+1} = y_n + \Delta t \sum_{i=1}^{s} b_i K_i, \quad n = 0, 1, \cdots \tag{7}$$

where

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

Explicit Runge-Kutta

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

$$y_0 \text{ given} \qquad y_{n+1} = y_n + \Delta t \sum_{i=1}^{s} b_i K_i, \quad n = 0, 1, \cdots \tag{7}$$

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

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{2,1} & & & & \\
c_3 & a_{3,1} & a_{3,2} & & & \\
\vdots & & \vdots & \ddots & & \\
c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s
\end{array}
$$

Consistency conditions

For consistency the coefficients must verify algebraic conditions;

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

Consistency conditions

For consistency the coefficients must verify algebraic conditions;

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

(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).

Consistency conditions

For consistency the coefficients must verify algebraic conditions;

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

(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).

For higher order of consistency, 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 $b_2 c_2 = 1/2$ (check!)

Accuracy and stages of RK ** NOT FOR THE EXAM **

Theorem 2

An explicit s-stages Runge-Kutta method cannot have order of accuracy p greater than s. Moreover, there are no known explicit s-stages RK methods with order $p = s$ for $s \geq 5$.

Theorem 2

An explicit s-stages Runge-Kutta method cannot have order of accuracy p greater than s. Moreover, there are no known explicit s-stages RK methods with order $p = s$ for $s \geq 5$.

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

Accuracy and stages of RK ** NOT FOR THE EXAM **

Theorem 2

An explicit s-stages Runge-Kutta method cannot have order of accuracy p greater than s. Moreover, there are no known explicit s-stages RK methods with order $p = s$ for $s \geq 5$.

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

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 $t_0, t_1, \cdots, t_N = T$ be a set of *equally spaced* points in $[t_0, T]$, and let $\Delta t = \dfrac{T - t_0}{N}$ be the time step (this time the points **must** be equally spaced).

A first multistep method

Let us start with an example. Let $t_0, t_1, \cdots, t_N = T$ be a set of *equally spaced* points in $[t_0, T]$, and let $\Delta t = \dfrac{T - t_0}{N}$ 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:

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

A first multistep method

Let us start with an example. Let $t_0, t_1, \cdots, t_N = T$ be a set of *equally spaced* points in $[t_0, T]$, and let $\Delta t = \dfrac{T - t_0}{N}$ 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:

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

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:

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

Choosing the parameters in a multistep scheme....

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

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

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:

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

Choosing the parameters in a multistep scheme....

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

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

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,

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

Choosing the parameters in a multistep scheme....

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

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

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:

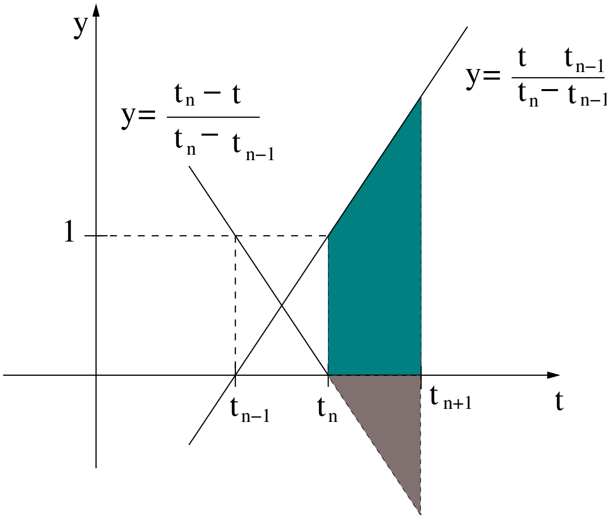
$$f(t, y(t)) \simeq \Pi_1(t) := \frac{t - t_{n-1}}{t_n - t_{n-1}} f(t_n, y_n) + \frac{t_n - t}{t_n - t_{n-1}} f(t_{n-1}, y_{n-1})$$

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 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:

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 $t_n, t_{n-1}, \cdots, t_{n+1-p}$:

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 $t_n, t_{n-1}, \cdots, t_{n+1-p}$:

$$
\begin{aligned}
& f(t, y(t)) \simeq \Pi_{p-1}(t) \quad \text{with } \Pi_{p-1} \in \mathbb{P}_{p-1} \text{ verifying} \\
& \Pi_{p-1}(t_n) = f(t_n, y_n), \\
& \Pi_{p-1}(t_{n-1}) = f(t_{n-1}, y_{n-1}), \\
& \ldots \\
& \Pi_{p-1}(t_{n+1-p}) = f(t_{n+1-p}, y_{n+1-p}).
\end{aligned}
$$

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 $t_n, t_{n-1}, \cdots, t_{n+1-p}$:

$$\begin{aligned}
&f(t, y(t)) \simeq \Pi_{p-1}(t) \quad \text{with } \Pi_{p-1} \in \mathbb{P}_{p-1} \text{ verifying} \\
&\Pi_{p-1}(t_n) = f(t_n, y_n), \\
&\Pi_{p-1}(t_{n-1}) = f(t_{n-1}, y_{n-1}), \\
&\ldots \\
&\Pi_{p-1}(t_{n+1-p}) = f(t_{n+1-p}, y_{n+1-p}).
\end{aligned}$$

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" $y_1, y_2, \cdots, y_{p-1}$ in addition to y_0 (for instance with a 1-step method).

Adams-Bashforth schemes

The resulting scheme will be:

$$\begin{cases} y_0 \text{ given}, \quad y_1, y_2, \cdots, y_{p-1} \ \text{ to be computed} \\ y_{n+1} = y_n + \Delta t \Big(c_1 f_n + c_2 f_{n-1} + \cdots + c_p f_{n+1-p} \Big), \\ \qquad\qquad\qquad\qquad\qquad n = p-1, p, p+1, \cdots \end{cases} \tag{8}$$

where $\Delta t \, c_1, \Delta t \, c_2, \cdots$ 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 resulting scheme will be:

$$\begin{cases} y_0 \text{ given}, \quad y_1, y_2, \cdots, y_{p-1} \text{ to be computed} \\ y_{n+1} = y_n + \Delta t \Big(c_1 f_n + c_2 f_{n-1} + \cdots + c_p f_{n+1-p} \Big), \\ \qquad\qquad\qquad\qquad n = p-1, p, p+1, \cdots \end{cases} \tag{8}$$

where $\Delta t\, c_1, \Delta t\, c_2, \cdots$ 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 (8) 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).

[1] see the last part of these slides for the definition

Adams-Moulton

Note: A similar construction gives **implicit methods**, called "**Adams-Moulton**. Compared with (8) 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_1	c_2	c_3	c_4	A-stability[1]	order
$p = 0$	1				yes	Δt
$p = 1$	$1/2$	$1/2$			yes	Δt^2
$p = 2$	$5/12$	$8/12$	$-1/12$		no	Δt^3
$p = 3$	$9/24$	$19/24$	$-5/24$	$1/24$	no	Δt^4

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

[1] see the last part of these slides for the definition

Consistency of multistep meth.s ** NOT FOR EXAM **

As we have already seen, consistency amounts to impose that the local truncation error is zero when the exact solution of (1) is a polynomial of degree up to 2: hence we must require that $\tau_n \equiv 0$ when the solution of (1) is $1, t, t^2$.

Consistency of multistep meth.s ** NOT FOR EXAM **

As we have already seen, consistency amounts to impose that the local truncation error is zero when the exact solution of (1) is a polynomial of degree up to 2: hence we must require that $\tau_n \equiv 0$ when the solution of (1) is $1, t, t^2$.

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

Consistency of multistep meth.s ** NOT FOR EXAM **

As we have already seen, consistency amounts to impose that the local truncation error is zero when the exact solution of (1) is a polynomial of degree up to 2: hence we must require that $\tau_n \equiv 0$ when the solution of (1) is $1, t, t^2$.

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

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

Consistency of multistep meth.s ** NOT FOR EXAM **

As we have already seen, consistency amounts to impose that the local truncation error is zero when the exact solution of (1) is a polynomial of degree up to 2: hence we must require that $\tau_n \equiv 0$ when the solution of (1) is $1, t, t^2$.

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

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

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

Consistency of multistep meth.s ** NOT FOR EXAM **

As we have already seen, consistency amounts to impose that the local truncation error is zero when the exact solution of (1) is a polynomial of degree up to 2: hence we must require that $\tau_n \equiv 0$ when the solution of (1) is $1, t, t^2$.

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

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

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

$$\Longrightarrow \alpha + \beta = 1 \Longrightarrow \beta = 1 - \alpha;$$

Consistency of multistep meth.s ** NOT FOR EXAM **

As we have already seen, consistency amounts to impose that the local truncation error is zero when the exact solution of (1) is a polynomial of degree up to 2: hence we must require that $\tau_n \equiv 0$ when the solution of (1) is $1, t, t^2$.

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

When $y(t) = 1, y' = 0 = f, \Longrightarrow \tau_n = \dfrac{1-1}{\Delta t} = 0;$

When $y(t) = t, y' = 1 = f, \Longrightarrow \tau_n = \dfrac{t_{n+1}-t_n}{\Delta t} - (\alpha+\beta) = 0;$

$$\Longrightarrow \alpha + \beta = 1 \Longrightarrow \beta = 1 - \alpha;$$

When $y(t) = t^2, y' = 2t = f, \Longrightarrow \tau_n = \dfrac{t_{n+1}^2 - t_n^2}{\Delta t} - (2\alpha t_n + 2\beta t_{n-1})$

Consistency of multistep meth.s ** NOT FOR EXAM **

As we have already seen, consistency amounts to impose that the local truncation error is zero when the exact solution of (1) is a polynomial of degree up to 2: hence we must require that $\tau_n \equiv 0$ when the solution of (1) is $1, t, t^2$.

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

When $y(t) = 1, y' = 0 = f, \Longrightarrow \tau_n = \dfrac{1-1}{\Delta t} = 0;$

When $y(t) = t, y' = 1 = f, \Longrightarrow \tau_n = \dfrac{t_{n+1}-t_n}{\Delta t} - (\alpha+\beta) = 0;$

$$\Longrightarrow \alpha + \beta = 1 \Longrightarrow \beta = 1 - \alpha;$$

When $y(t) = t^2, y' = 2t = f, \Longrightarrow \tau_n = \dfrac{t_{n+1}^2 - t_n^2}{\Delta t} - (2\alpha t_n + 2\beta t_{n-1})$

$$= t_{n+1} + t_n - 2(\alpha t_n + \beta t_{n-1}) = 0$$

Consistency of multistep meth.s ** NOT FOR EXAM **

As we have already seen, consistency amounts to impose that the local truncation error is zero when the exact solution of (1) is a polynomial of degree up to 2: hence we must require that $\tau_n \equiv 0$ when the solution of (1) is $1, t, t^2$.

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

When $y(t) = 1, y' = 0 = f, \Longrightarrow \tau_n = \dfrac{1-1}{\Delta t} = 0;$

When $y(t) = t, y' = 1 = f, \Longrightarrow \tau_n = \dfrac{t_{n+1} - t_n}{\Delta t} - (\alpha + \beta) = 0;$

$$\Longrightarrow \alpha + \beta = 1 \Longrightarrow \beta = 1 - \alpha;$$

When $y(t) = t^2, y' = 2t = f, \Longrightarrow \tau_n = \dfrac{t_{n+1}^2 - t_n^2}{\Delta t} - (2\alpha t_n + 2\beta t_{n-1})$

$$= t_{n+1} + t_n - 2(\alpha t_n + \beta t_{n-1}) = 0$$

$$\Longrightarrow \alpha t_n + \beta t_{n-1} = \frac{t_{n+1} + t_n}{2}.$$

Consistency of multistep meth.s ** NOT FOR EXAM **

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

Consistency of multistep meth.s ** NOT FOR EXAM **

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

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

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

Consistency of multistep meth.s ** NOT FOR EXAM **

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

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 $\alpha = \dfrac{3}{2}$ and $\beta = -\dfrac{1}{2}$. The 2-step scheme is then

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

Consistency of multistep meth.s ** NOT FOR EXAM **

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

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

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

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

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

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

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}

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). The stability is much improved if there is another E step to evaluate f_{n+1} with the corrected y_{n+1}. So PECE is the basic sequence

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). The stability is much improved if there is another E step to evaluate f_{n+1} with the corrected y_{n+1}. So PECE is the basic sequence
continue the correction repeating the CE steps until y_{n+1} no longer changes: it has reached its final value for the implicit formula. Often two or three corrections are enough, and this is faster than using Newton's method in solving a single step of the implicit method.