# Numerical analysis (4/7): ODEs

University of Luxembourg

Philippe Marchner

Siemens Digital Industries Software, France

November 9th, 2023

# Outline

# Outline

# Introduction

So far, we have seen methods to perform **interpolation**, **differentiation** and **integration**.
We will go a step further and study methods to solve ordinary differential equations (ODEs)

$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0, \quad t \in I = [t_0, T]$$

We will address theoretical and numerical questions

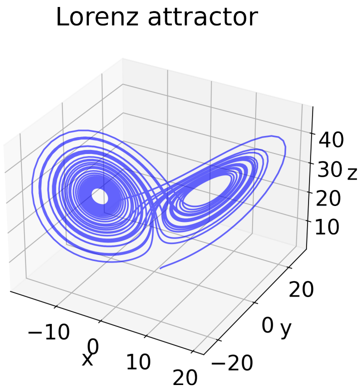
- Does the ODE have a (unique) solution ? What is the *nature* of the solution (oscillatory, stiff, divergent, etc.) ?
- Does the numerical solution *converges* to the exact solution ?
- What is the accuracy/cost of the method ?

# A famous example - The Lorenz system

Atmospheric convection, the "Butterfly effect"

$$x'(t) = \sigma(y(t) - x(t))$$
$$y'(t) = x(t)(\rho - z) - y(t)$$
$$z'(t) = x(t)y(t) - \beta z(t)$$

Numeric test: $\rho = 28$, $\sigma = 10$, $\beta = 8/3$,
$(x_0, y_0, z_0) = (1, 1, 1)$, $T = 40$, $t_0 = 0$




https://matplotlib.org/stable/gallery/mplot3d/lorenz_attractor.html

# Definitions

- A first-order ODE is an equation of the form

$$y'(t) = f\left(t, y(t)\right), \quad \forall\, t \in I$$

where $I$ is an interval of $\mathbb{R}$, $y : [0, +\infty[ \to \mathbb{R}^N$ is a vectorial function depending on the variable $t$ and $f$ is a map from $I \times \mathbb{R}^N$ onto $\mathbb{R}^N$.

- An ODE of order $p$ is an equation of the form

$$y^{(p)}(t) = f\left(t, y(t), y'(t), \ldots, y^{(p-1)}(t)\right), \quad \forall\, t \in I$$

where $I$ is an interval of $\mathbb{R}$, $y : [0, +\infty[ \to \mathbb{R}^N$ is a vectorial function with respect to $t$ and $f$ is an application from $I \times (\mathbb{R}^N)^p$ to $\mathbb{R}^N$.

# Remark

Any ODE of order $p$ can be written as a first-order ODE.

Indeed, by setting

$$x_1(t) = y(t),\ x_2(t) = y'(t),\ x_3(t) = y''(t), \dots, x_p(t) = y^{(p-1)}(t)$$

the problem writes

$$x_1'(t) = x_2(t),\ x_2'(t) = x_3(t), \dots, x_{p-1}'(t) = x_p(t)$$

and

$$x_p'(t) = f\left(t, x_1(t), x_2(t), \dots, x_p(t)\right)$$

# Remark

which can be written, by setting

$$X(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_p(t) \end{pmatrix} \quad \text{and} \quad F(t, X) = \begin{pmatrix} X_2 \\ X_3 \\ \vdots \\ f(t, x_1, x_2, \ldots, x_p) \end{pmatrix}$$

$$X'(t) = F(t, X(t))$$

## Example

Consider $y''(t) + \omega^2 y(t) = g(t)$. Define $x_1 = y, x_2 = y'$, such as

$$X'(t) = \begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ g(t) \end{pmatrix} = \begin{pmatrix} x_2 \\ -\omega^2 x_1 + g(t) \end{pmatrix} = F(t, X(t))$$

# Definition - Initial value problem

- For an interval $I$, $f : I \times \mathbb{R}^N \to \mathbb{R}^N$, $t_0 \in I$ and $y^0 \in \mathbb{R}^N$, solving the Cauchy problem

$$\begin{cases} y'(t) = f\left(t, y(t)\right), & \forall\ t \in I \\ y(t_0) = y^0 \end{cases}$$

means to determine all functions $y : I \to \mathbb{R}^N$ solutions to the ODE satisfying $y(t_0) = y_0$.

We also talk about Initial Value Problem (IVP)

## Stability of first order IVP

If $f$ is continuous in $t$ and Lipschitz continuous in $y$, i.e. $|\partial_y f(t, y)| \leq L$ for $t \in I$, then the IVP has a unique solution in $I$. Moreover for two solutions $(y_1, y_2)$ with different initial conditions we have

$$|y_1(t) - y_2(t)| \leq e^{L(t-t_0)}|y_1(t_0) - y_2(t_0)|$$

# A simple example

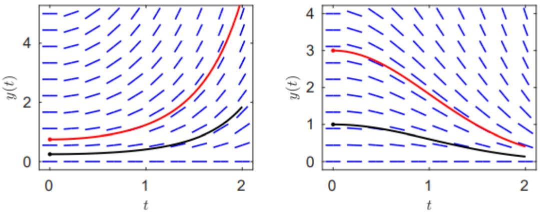
- For $a \in \mathbb{R}$, the IVP

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

has the unique solution $y(t) = y_0 e^{at}$

- more generally, if $a$ is a continuous function on $[0, +\infty[$, the IVP has the solution

$$y(t) = y_0 \exp\left( \int_0^t a(\sigma)\, d\sigma \right)$$




Difference in two solutions that start at nearby points for $y' = ty$ (left) and $y' = -ty$ (right)

# Numerical Approximation

We try to numerically solve the IVP, which means that we look for an approximate solution to

$$\begin{cases} y'(t) = f\left(t, y(t)\right), \; 0 \leq t \leq T \\ y\left(0\right) = y_0 \end{cases}$$

with $y_0 \in \mathbb{R}^N$ and $f : [0, +\infty[ \times \mathbb{R}^N \to \mathbb{R}^N$

We remark that this problem is equivalent to

$$y(t) = y_0 + \int_0^t f(s, y(s)) ds \quad \forall \; t \in [0, T]$$

Therefore, it is sufficient to obtain a numerical approximation to

$$\int_0^t f(s, y(s)) ds$$

and, to this end we can use the ideas and methods from **numerical integration**.

# Numerical Approximation

For a subdivision

$$0 = t_0 < t_1 < t_2 < \ldots < t_N = T$$

our problem implies that

$$\begin{cases} y(t_{n+1}) = y(t_n) + \displaystyle\int_{t_n}^{t_{n+1}} f(t, y(t))dt, \quad \forall\ 0 \leq n \leq N-1 \\ y(0) = y_0 \end{cases}$$

The numerical methods differ by the choice of the evaluation of the integrals

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

## Remark

The integrand depends on $y$ itself, which makes the integration more complicated

# Outline

# Outline

# The forward Euler method

- let us assume that our Cauchy problem admits one solution $y$ on $[0, T]$.
- we introduce the subdivision $0 = t_0 < t_1 < \ldots < t_N = T$ and $h_n = t_{n+1} - t_n$
- Let us recall that our problems imply

$$\begin{cases} y(t_{n+1}) = y(t_n) + \displaystyle\int_{t_n}^{t_{n+1}} f(t, y(t))dt, & \forall\ 0 \leq n \leq N-1 \\ y(0) = y_0 \end{cases}$$

- the forward Euler method (explicit) corresponds to an approximation by the left rectangle quadrature rule

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

- we then obtain, where $\tilde{y}_n$ is an approximation of $y(t_n)$,

$$\begin{cases} \tilde{y}_{n+1} = \tilde{y}_n + h_n f(t_n, \tilde{y}_n), & \forall\ 0 \leq n \leq N-1 \\ y_0 = \tilde{y}_0 \end{cases}$$

# Outline

# Convergence

It is necessary to analyze in which sense the computed value $\tilde{y}_n$ is sufficiently close to the exact value $y(t_n)$ and so we want to evaluate the discretization error

$$e_n = y(t_n) - \tilde{y}_n.$$

## Definition

we say that the method is converging if

$$\max_{0 \leq n \leq N} |e_n|$$

tends towards 0 when $h \to 0$ and $\tilde{y}_0 \to y(t_0)$.

## Remark

If the method is converging, by choosing $h$ sufficiently small, and $\tilde{y}_0$ close to $y(t_0)$, we obtain a good approximation of $y\left(t_n\right),\ n = 0, ..., N$

# Convergence

## Remarks

- it seems more natural to directly set $\tilde{y}_0 = y(t_0)$ in our scheme. However, in practice, if $y(t_0)$ is real-valued, it cannot be considered as exact (meaning in exact arithmetic) because of the round-off errors (on a computer). A correct analysis assumes $\tilde{y}_0 \neq y_0$
- like any computation, a stability problem arises:
  it is necessary to understand the consequences on the computation of small variations of $\tilde{y}_0$ and $f(t_n, \tilde{y}_n)$.

# Consistency

We first introduce a notion called consistency of a numerical scheme:
the consistency error represents the error at the $n$-th step when replacing the ODE by the discrete equation

$$\varepsilon_n = y(t_{n+1}) - y(t_n) - h_n f(t_n, y(t_n)).$$

$\varepsilon_n$ is sometimes called the local truncation error

## Definition

A method is said to be consistent if

$$\lim_{h \to 0} \sum_{n=0}^{N-1} \|\varepsilon_n\| = 0.$$

Remark: consistency is a **local** notion, it supposes that the previous data are known exactly.
On the other hand, stability relates to the propagation of local errors

# Stability

## Definition

We say that a method is stable if there exists a constant $K$ such that

$$\max_n \|\tilde{y}_n - \tilde{z}_n\| \le K \left[ \|\tilde{y}_0 - \tilde{z}_0\| + \sum_{n=0}^{N-1} \|\varepsilon_n\| \right]$$

for any $\tilde{z}_n$ solution to

$$\tilde{z}_{n+1} = \tilde{z}_n + h_n f\left(t_n, \tilde{z}_n\right) + \varepsilon_n, \;\; n = 0, \dots, N-1.$$

This notion of stability implies that small perturbations on the initial data and all the intermediate calculations leads to small perturbations on the final result

# Back to convergence

stability of the forward Euler scheme

$+$

consistency of the forward Euler scheme

$=$

convergence of the forward Euler scheme

**Remark:**

It can be shown that, more generally, for a one-step method, consistency and stability imply convergence.

**Convergence of Euler's method**

The forward Euler method is convergent. If $f$ is Lipschitz and continuous one can show

$$\max_{0 \leq n \leq N} |e_n| \leq e^{LT} \left( hMT + \|e_0\| \right)$$

# Outline

## Example

$$\begin{cases} y'(t) = 3y(t) - 3t & t \in [0,5] \\ y(0) = \frac{1}{3} \end{cases}$$

- the solution is $y(t) = \frac{1}{3} + t$
- now, if we consider the same problem but with the intial data $z(0) = \frac{1}{3} + \epsilon$, the solution is $z(t) = \frac{1}{3} + t + \epsilon e^{3t}$
- as a consequence, $z(5) = y(5) + \epsilon e^{15} \simeq y(5) + 3\epsilon 10^6$
- therefore, if one works with a computer with a round-off error equal to $10^{-6}$, it will be impossible to approximate $y(5)$, and this, independently of the numerical method
- the problem is ill-conditioned

## Example

$$\begin{cases} y'(t) = -150\, y(t) + 50 \\ y(0) = \frac{1}{3} \end{cases}$$

- the solution is $y(t) = \frac{1}{3}$
- here, the problem is well-conditioned. Indeed, if one introduces a perturbation $\epsilon$ on the initial data we have

$$|y(t) - z(t)| \leq \epsilon e^{-150t}, \quad \forall\ t \geq 0$$

- the forward Euler method leads to

$$y_{n+1} = y_n + h_n(-150 y_n + 50)$$

that is

$$y_{n+1} - \frac{1}{3} = (1 - 150 h_n)\left(y_n - \frac{1}{3}\right)$$

- for a constant step $h_n = h = \frac{1}{50}$, we have

$$y_{n+1} - \frac{1}{3} = (1 - 150h)^n \left(y_0 - \frac{1}{3}\right) = (-2)^n \left(y_0 - \frac{1}{3}\right)$$

- in particular

$$y_{50} - y(0) = (-2)^{50} \left(y_0 - \frac{1}{3}\right) \simeq 10^{15} \left(y_0 - \frac{1}{3}\right) \; !$$

- this shows that the step size is too large. On the other hand, if it is taken smaller, we will have round-off errors!
- the forward Euler scheme is a numerically unstable scheme.

## Example

$$\begin{cases} y'(t) = -\lambda y(t) & \lambda > 0 \\ y(0) = y_0 \end{cases}$$

- the solution to this problem is $y(t) = y_0 e^{-\lambda t}$
- the problem is well-conditioned. Indeed, for a small $\epsilon$ on the initial data, one gets

$$|y(t) - z(t)| \leq \epsilon e^{-\lambda t}, \quad \forall\ t \geq 0$$

- the forward Euler method applied to this problem with a constant step size $h$ gives

$$y_{n+1} = y_n - \lambda h y_n = (1 - \lambda h) y_n$$

and so

$$y_n = (1 - \lambda h)^n y_0$$

$$y_n = (1 - \lambda h)^n y_0$$

- even if the exact solution remains bounded

$$|y(t)| \leq |y_0| \quad \forall\ t \geq 0$$

  we see that if $|1 - \lambda h| > 1$ then the computed solution $y_n$ will have a growing amplitude, leading to an unstable scheme
- the absolute stability condition (CFL:=Courant-Friedrichs-Lewy) writes

$$\lambda h < 2$$

- hence, the larger $\lambda$ is, the smaller $h$ must be.
- but if $h$ is too small, then round-off errors appear !
- Stability conditions can be analyzed in the complex plane for many ODE methods

# Outline

# The backward Euler scheme (implicit)

To solve the instability problem, we often use an implicit scheme like

$$\begin{cases} y_{n+1} = y_n + h_n f\left(t_{n+1}, y_{n+1}\right) & \forall\ 0 \leq n \leq N-1 \\ y_0 = \tilde{y}_0 \end{cases}$$

It comes from the approximation of

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

by the right rectangular quadrature rule

$$\int_{t_n}^{t_{n+1}} f\left(t, y\left(t\right)\right) dt \simeq h_n f(t_{n+1}, y(t_{n+1}))$$

The relation

$$y_{n+1} = y_n + h_n f\left(t_{n+1}, y_{n+1}\right)$$

defines $y_{n+1}$ in an implicit way.

- this method is therefore more complicate to use than a forward Euler scheme
- at each iteration, this equation must be solved. Does it admit a solution? Is it unique?
- generally, the numerical solution to this equation requires the use of an iterative method (Newton, fixed point,...)
  $\rightarrow$ see Lecture on nonlinear equations
- the cost of one iteration is then higher than for the forward Euler scheme (which is explicit).
- however, the stability is greatly improved

# Back to the example

## Example

$$\begin{cases} y'(t) = -\lambda y(t) & \lambda > 0 \\ y(0) = y_0 \end{cases}$$

- we have, for a constant step $h$, $y_{n+1} = y_n - \lambda h y_{n+1}$ that is

$$y_{n+1} = \frac{y_n}{(1 + \lambda h)}$$

which also writes

$$y_n = \frac{y_0}{(1 + \lambda h)^n}$$

- in particular, we have $\lambda > 0$ and for $h > 0$, $|y_n| \leq |y_0|$.
- furthermore, we can prove that this method converges as the previous one.

# Outline

# Outline

# General study of one-step methods

A one-step method can be written in a general way as

$$\begin{cases} y_{n+1} = y_n + h_n \Phi(t_n, y_n, h_n), & \forall\ n \in [\![0, N-1]\!] \\ y_0 = \tilde{y}_0 \end{cases}$$

- the approximation $y_{n+1}$ of $y(t_{n+1})$ is therefore obtained uniquely from $t_n$, $h_n$ and $y_n$ the approximation of $y(t_n)$ obtained at the previous time step.
- this method can be implicit or explicit

# General study of one-step methods

A one-step method can be written in a general way as

$$\begin{cases} y_{n+1} = y_n + h_n \Phi(t_n, y_n, h_n), & \forall\ n \in [\![0, N-1]\!] \\ y_0 = \tilde{y}_0 \end{cases}$$

## Example

- forward Euler scheme: $y_{n+1} = y_n + h_n f(t_n, y_n)$. Here

$$\Phi(t, y, h) = f(t, y)$$

  $\Phi$ is independent of $h$.

- backward Euler scheme : $y_{n+1} = y_n + h_n f(t_{n+1}, y_{n+1})$. Here

$$\Phi(t, y, h) = f(t + h, k)$$

  with $k$ solution to $k = y + hf(t + h, k)$

# Order of accuracy

We now define the notion of order of accuracy of a one-step method

## Definition

A one-step method is said to be of order $p$ ($p > 0$), if for any solution $y$ of $y'(t) = f(t, y(t))$ such that $y \in C^{p+1}([t_0, t_0 + T])$, there exists a real-valued parameter $K$ which only depends on $y$ and $\Phi$ such that

$$\sum_{n=0}^{N-1} \|\varepsilon_n\| \leq K h^p$$

with $\varepsilon_n$ being the local truncation error

$$\varepsilon_n = y(t_{n+1}) - y(t_n) - h_n \Phi(t_n, y(t_n), h_n)$$

## Theorem

If a one-step method is stable and of order $p$ and if $f \in C^p([t_0, t_0 + T] \times \mathbb{R}^n)$, then we have

$$\|y(t_n) - \tilde{y}_n\| \leq M \left[ \|y(t_0) - \tilde{y}_0\| + Kh^p \right] \quad \forall n \in [\![0, N]\!]$$

## Examples:

- the forward and backward Euler schemes are first-order
- let us consider the more general methods

# Outline

# Runge-Kutta methods

- First-order methods require too much computational time to get a given accuracy
- It is then necessary to use a high-order method: the most known are Runge-Kutta methods that consist in using high-order numerical integration rules to approximate

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

which use **intermediate points** between $t_n$ and $t_{n+1}$

# Runge-Kutta methods

Let $(c_j, b_j)$ be an elementary quadrature formula with $s$ stages:

$$\int_0^1 g(x)dx = \sum_{j=1}^{s} b_j g(c_j)$$

Then

$$y(t_{n+1}) \simeq y(t_n) + h_n \sum_{i=1}^{s} b_i k_i$$

with $t_{n,i} = t_n + h_n c_i$, $k_i = f(t_{n,i}, y(t_{n,i}))$

**Problem**

How to evaluate $k_i = f(t_{n,i}, y(t_{n,i}))$ if $y(t_{n,i})$ is not known?

# Runge-Kutta methods

The values $y(t_{n,i})$ are also evaluated through some numerical integration formulae by using the same points $t_{n,i}$

$$y(t_{n,i}) \simeq y(t_n) + h_n \sum_{j=1}^{s} a_{i,j} f(t_{n,j}, y(t_{n,j})) \quad \forall\ i \in [\![1, s]\!]$$

The Runge-Kutta methods consists in replacing $\simeq$ by $=$
$y(t_{n,j})$ are given at other intermediates points where it can be evaluated !

# Runge-Kutta methods

## Remarks

- if the matrix $(a_{i,j})$ is strictly lower triangular, then the RK method define explicitly the values of $y_{n,j}$, otherwise implicitly.
- the method is a one-step method. Indeed, this scheme can be written as

$$y_{n+1} = y_n + h_n \Phi(t_n, y_n, h_n)$$

where $\Phi(.,.,.)$ is the function defined by the equations

$$\Phi(t_n, y_n, h_n) = \sum_{i=1}^{s} b_i k_i, \;\; k_i = f\left(t_n + c_i h_n, y_n + h_n \sum_{j=1}^{s} a_{i,j} k_j\right) \quad \forall\, i \in [\![1, s]\!]$$

# Example of Runge-Kutta method

- Explicit midpoint method: we take the midpoint formula

$$y(t_{n+1}) \simeq y(t_n) + h_n f\left(t_n + \frac{h_n}{2}, y(t_n + \frac{h_n}{2})\right)$$

and we replace the unknown value $y(t_n + \frac{h_n}{2})$ by the Euler method

$$y(t_n + \frac{h_n}{2}) \simeq y(t_n) + \frac{h_n}{2} f(t_n, y(t_n))$$

This provides

$$y_{n+1} = y_n + h_n f\left(t_n + \frac{h_n}{2}, y_n + \frac{h_n}{2} f(t_n, y_n)\right)$$

# Examples of Runge-Kutta methods

- Trapezoidal method: we take the trapezoidal quadrature formula

$$y(t_{n+1}) \simeq y(t_n) + \frac{h_n}{2}\left(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right)$$

which is the implicit trapezoidal method. If we replace the unknown value $y(t_{n+1})$ by Euler approximation, we obtain the explicit trapezoidal method

$$\begin{aligned} y_{n+1} &= y_n + \frac{h_n}{2}\left[f(t_n, y_n) + f\left(t_{n+1}, y_n + h_n f(t_n, y_n)\right)\right] \\ &= y_n + \frac{h_n}{2}\left[k_1 + k_2\right], \end{aligned}$$

$$\text{with} \quad k_1 = f(t_n, y_n), \quad k_2 = f(t_n + h_n, y_n + h_n k_1)$$

# Butcher Tableaux

A Runge-Kutta method is completely known when we have: $s$, the coefficients $a_{i,j}$, $b_j$ and $c_j$. Usually, we use the following Butcher Tableaux

$$
\begin{array}{c|cccc}
c_1 & a_{1,1} & a_{1,2} & \dots & a_{1,s} \\
c_2 & a_{2,1} & a_{2,2} & \dots & a_{2,s} \\
\vdots & \vdots & \vdots & & \vdots \\
c_s & a_{s,1} & a_{s,2} & \dots & a_{s,s} \\
\hline
 & b_1 & b_2 & \dots & b_s
\end{array}
$$

**Example**

Explicit Euler: $\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$, Explicit midpoint: $\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array}$,

# Butcher Tableau for RK4

- Example for RK4: based on Simpson's rule integration and explicit midpoint rule

$$
\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
\frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
\hline
 & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6}
\end{array}
$$

$$
\begin{array}{rclrcl}
k_{n,1} & = & f(t_n, y_n) & k_{n,2} & = & f(t_n + \frac{h_n}{2}, y_n + \frac{h_n}{2} k_{n,1}) \\
k_{n,3} & = & f(t_n + \frac{h_n}{2}, y_n + \frac{h_n}{2} k_{n,2}) & k_{n,4} & = & f(t_{n+1}, y_n + h_n k_{n,3})
\end{array}
$$

$$
y_{n+1} = y_n + \frac{h_n}{6} [k_{n,1} + 2k_{n,2} + 2k_{n,3} + k_{n,4}]
$$

# Further remarks on Runge-Kutta methods

Under some regularity assumptions on $f$, it can be proved that the Runge-Kutta methods are stable. Being stable and consistent, they are convergent.

- A RK method with $s$ stages is of order $s$
- RK methods are costly, they require many function evaluations
- varying step size RK methods can be derived, such as RK23 and RK45
- Implicit RK schemes are costly, we privilege explicit RK in practice

# Outline

# Multi-step methods

- the one-step methods only use the approximate value $y_n$ of $y(t_n)$ to compute an approximate value $y_{n+1}$ of $y(t_{n+1})$.
- the multi-step methods also involve the information obtained at the previous steps $t_{n-1}, t_{n-2}, \ldots, t_{n-r}$.
- we will describe here the Adams methods that consist in replacing $f(t, y(t))$ by an interpolation polynomial at points $t_{n-r}, t_{n-r+1}, \ldots, t_{n-1}, t_n, (t_{n+1})$, in the computation of

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

# Multi-step methods

- if $P_n$ is this polynomial, the approximate values $y_{n+1}$ will be obtained by the approximate equation
$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} P_n(t)\,dt.$$
- the formulae will be implicit (explicit, respectively) if $t_{n+1}$ is (is not, respectively) one of the interpolation points.

# Outline

# Adams-Bashforth methods with $r+1$ steps

- we assume that we know the approximate values $y_n$ of $y\left(t_n\right)$ and $f_n, f_{n-1}, \ldots, f_{n-r}$ of $f\left(t, y\left(t\right)\right)$ respectively at points $t_n, t_{n-1}, \ldots, t_{n-r}$.
- the polynomial $P_n$ is chosen as the polynomial of degree less or equal to $r$ such that

$$P_n\left(t_{n-i}\right) = f_{n-i} \qquad \forall i = 0, \ldots, r.$$

- the approximation of $y(t_{n+1})$ is then defined by

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} P_n(t)dt$$

# Adams-Bashforth methods with $r+1$ steps

- if one represents the polynomial $P_n$ by the Newton formula

$$P_n(t) = \sum_{i=0}^{r} f[t_n, t_{n-1}, \ldots, t_{n-i}] \prod_{j=0}^{i-1} (t - t_{n-j})$$

  the methods becomes

$$y_{n+1} = y_n + \sum_{i=0}^{r} f[t_n, t_{n-1}, \ldots, t_{n-i}] \left( \int_{t_n}^{t_{n+1}} \prod_{j=0}^{i-1} (t - t_{n-j}) dt \right)$$

# The Adams-Bashforth methods with $r + 1$ steps

- in the case of a constant step $h$ i.e. $t_j = t_0 + jh$, the divided differences can be written as

$$f[t_n, t_{n-1}, \dots, t_{n-i}] = \frac{\Delta^i f_n}{i! h^i}$$

where

$$\Delta^i f_k = \begin{cases} f_k & \text{if } i = 0 \\ \Delta^{i-1} f_k - \Delta^{i-1} f_{k-1} & \text{if } i \geq 1 \end{cases}$$

are the backward finite differences

- the formula then becomes

$$y_{n+1} = y_n + \sum_{i=0}^{r} \frac{\Delta^i f_n}{i! h^i} \left( \int_{t_n}^{t_{n+1}} \prod_{j=0}^{i-1} (t - t_{n-j}) dt \right)$$

# The Adams-Bashforth methods with $r+1$ steps

- now, by setting $t = t_n + sh$, $s \in [0,1]$, we have

$$\begin{aligned} \int_{t_n}^{t_{n+1}} \prod_{j=0}^{i-1} (t - t_{n-j}) dt &= h^{i+1} \int_0^1 \prod_{j=0}^{i-1} (j - s) ds \\ &= h^{i+1} i! \int_0^1 \binom{s+i-1}{i} ds \end{aligned}$$

where $\binom{s}{k}$ is the binomial coefficient generalized to non integer values

$$\binom{s}{k} = \frac{s\,(s-1)\ldots(s-k+1)}{1.2\ldots k}.$$

# The Adams-Bashforth methods with $r+1$ steps

- Hence

$$y_{n+1} = y_n + h\sum_{i=0}^{r} \gamma_i \Delta^i f_n \quad \text{with} \quad \gamma_i = \int_0^1 \binom{s+i-1}{i} ds$$

- we show that the $\gamma_i$ satisfy the relation

$$\gamma_0 = 1,\ 1 = \frac{\gamma_0}{i+1} + \frac{\gamma_1}{i} + \ldots + \frac{\gamma_{i-1}}{2} + \gamma_i$$

  which leads to their recursive computation.

- it is important to notice that they do not depend on $r$, which is useful when one wants to make the order $r$ vary in a same computation.
- one then gets

$$\gamma_0 = 1,\ \gamma_1 = \frac{1}{2},\ \gamma_2 = \frac{5}{12},\ \gamma_3 = \frac{3}{8},\ \gamma_4 = \frac{251}{720},\ \gamma_5 = \frac{95}{288}.$$

# The Adams-Bashforth methods with $r+1$ steps

- In practice, we prefer to explicitly write the relation as a function of the values of $f_{n-i}$, leading to

$$y_{n+1} = y_n + h \sum_{i=0}^{r} b_{i,r} f_{n-i}.$$

- from the finite difference formula, we can check that

$$b_{r,r} = (-1)^r \gamma_r,\ b_{i,r} = b_{i,r-1} + (-1)^i \begin{pmatrix} r \\ i \end{pmatrix} \gamma_r,\ 0 \le i \le r.$$

# The Adams-Bashforth methods with $r+1$ steps

- one gets the following tableaux:

| | $b_{0,r}$ | $b_{1,r}$ | $b_{2,r}$ | $b_{3,r}$ | $b_{4,r}$ | $b_{5,r}$ | $b_{6,r}$ | $\gamma_r$ |
|---|---|---|---|---|---|---|---|---|
| $r=0$ | $1$ | | | | | | | $1$ |
| $r=1$ | $\frac{3}{2}$ | $-\frac{1}{2}$ | | | | | | $\frac{1}{2}$ |
| $r=2$ | $\frac{23}{12}$ | $-\frac{4}{3}$ | $\frac{5}{12}$ | | | | | $\frac{5}{12}$ |
| $r=3$ | $\frac{55}{24}$ | $-\frac{59}{24}$ | $\frac{37}{24}$ | $-\frac{3}{8}$ | | | | $\frac{3}{8}$ |
| $r=4$ | $\frac{1901}{720}$ | $-\frac{1387}{360}$ | $\frac{109}{30}$ | $-\frac{637}{360}$ | $\frac{251}{720}$ | | | $\frac{251}{720}$ |
| $r=5$ | $\frac{4277}{1440}$ | $-\frac{7923}{1440}$ | $\frac{4991}{720}$ | $-\frac{3649}{720}$ | $\frac{959}{480}$ | $-\frac{95}{288}$ | | $\frac{95}{288}$ |
| $r=6$ | $\frac{199441}{60840}$ | $-\frac{18817}{2520}$ | $\frac{238783}{20160}$ | $-\frac{10979}{945}$ | $\frac{139313}{20160}$ | $-\frac{5783}{2520}$ | $\frac{19807}{60840}$ | $\frac{19807}{60840}$ |

# The Adams-Bashforth methods with $r+1$ steps

- for $r = 0$

$$y_{n+1} = y_n + hf_n \quad \text{(Euler)}$$

- for $r = 1$

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

- for $r = 2$

$$y_{n+1} = y_n + \frac{h}{12}(23f_n - 16f_{n-1} + 5f_{n-2})$$

- for $r = 3$

$$y_{n+1} = y_n + \frac{h}{124}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})$$

# The Adams-Bashforth methods with $r + 1$ steps

- this last method (Adams-Bashforth with 4 steps) is usually used.
- if one wishes to apply it to the resolution of our Cauchy problem, we have to know the four initial approximations $y_0$, $y_1$, $y_2$ and $y_3$. Next, we can use the recursive formula to compute $y_4$, $y_5$, ...
- Adams computed the Taylor series of the exact solution around the initial value to determine the initial approximations that are not known
- clearly, we can also get them by using a one-step method
- this method is of order 4 and is stable under the natural smoothness assumptions on $f$

# The Adams-Bashforth methods with $r + 1$ steps

- however, the stability constant are often very large which implies some numerical instabilities analogous to the one that have been underlined in the case of the forward Euler scheme
- to overcome this drawback, one uses some implicit methods

# Outline

# The Adams-Moulton methods with $r+1$ steps

- we interpolate the function $f\left(t, y\left(t\right)\right)$ at points $t_{n+1},\ t_n,\ \ldots,\ t_{n-r}$ by the polynomial $Q_n$ of degree less or equal to $r+1$ such that

$$\begin{cases} Q_n\left(t_{n-i}\right) = f_{n-i} \quad i = 0, 1, \ldots, r \\ Q_n\left(t_{n+1}\right) = f_{n+1} \quad \text{(the value is still unknown).} \end{cases}$$

- by a similar computation, we obtain

$$y_{n+1} = y_n + h \sum_{i=0}^{r+1} \gamma_i^* \Delta^i f_{n+1}$$

where

$$\gamma_i^* = \int_{-1}^{0} \frac{s\left(s+1\right)\ldots\left(s+i-1\right)}{i!} ds, \ i \geq 1, \ \gamma_0^* = 1.$$

- we check that

$$\gamma_i^* = \gamma_i - \gamma_{i-1}, \ i \geq 1.$$

# The Adams-Moulton method with $r+1$ steps

- as before, we prefer to write

$$y_{n+1} = y_n + h \sum_{i=-1}^{r} b^*_{i,r} f_{n-i}$$

where, as it can be easily proved, the $b^*_{i,r}$ satisfy to

$$b^*_{r,r} = (-1)^{r+1} \gamma^*_{r+1}, \; b^*_{i,r} = b^*_{i,r-1} + (-1)^{i+1} \binom{r+1}{i+1} \gamma^*_{r+1}.$$

# The Adams-Moulton method with $r+1$ steps

- one gets the tableau

| | $b^*_{-1,r}$ | $b^*_{0,r}$ | $b^*_{1,r}$ | $b^*_{2,r}$ | $b^*_{3,r}$ | $b^*_{4,r}$ | $b^*_{5,r}$ | $b^*_{6,r}$ | $b^*_{7,r}$ |
|---|---|---|---|---|---|---|---|---|---|
| $r=0$ | $\frac{1}{2}$ | $\frac{1}{2}$ | | | | | | | $1$ |
| $r=1$ | $\frac{5}{12}$ | $\frac{2}{3}$ | $-\frac{1}{12}$ | | | | | | $-\frac{1}{2}$ |
| $r=2$ | $\frac{9}{24}$ | $\frac{19}{24}$ | $-\frac{5}{24}$ | $\frac{1}{24}$ | | | | | $-\frac{1}{12}$ |
| $r=3$ | $\frac{251}{720}$ | $\frac{323}{360}$ | $-\frac{11}{30}$ | $\frac{53}{360}$ | $-\frac{19}{720}$ | | | | $-\frac{1}{24}$ |
| $r=4$ | $\frac{95}{288}$ | $\frac{1427}{1440}$ | $-\frac{133}{240}$ | $\frac{241}{720}$ | $-\frac{173}{1440}$ | $\frac{3}{160}$ | | | $-\frac{19}{720}$ |
| $r=5$ | $\frac{19087}{60480}$ | $\frac{2713}{2520}$ | $-\frac{15487}{20160}$ | $\frac{586}{945}$ | $-\frac{6737}{20160}$ | $\frac{263}{2520}$ | $-\frac{863}{60480}$ | | $-\frac{3}{160}$ |
| $r=6$ | $\frac{36799}{120960}$ | $\frac{139849}{120960}$ | $-\frac{121797}{120960}$ | $\frac{123133}{120960}$ | $-\frac{88545}{120960}$ | $\frac{41499}{120960}$ | $-\frac{11351}{120960}$ | $\frac{275}{24192}$ | $-\frac{863}{60480}$ |

# The Adams-Moulton methods with $r + 1$ steps

- for $r = 0$

$$y_{n+1} = y_n + \frac{h}{2}(f_{n+1} + f_n)$$

- for $r = 1$

$$y_{n+1} = y_n + \frac{h}{12}(5f_{n+1} + 8f_n - f_{n-1})$$

- for $r = 2$

$$y_{n+1} = y_n + \frac{h}{24}(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})$$

# The Adams-Moulton methods with $r + 1$ steps

- this last method (3-steps Adams-Moulton method) is the most commonly used method.
- we show that under some smoothness assumptions this method is of order 4 and stable
- the stability coefficients are much better (smaller) than for the explicit fourth-order Adams-Bashforth method
- of course, we must pay the price since we implicitly define $y_{n+1}$ through $f_{n+1} = f\left(t_{n+1}, y_{n+1}\right).$
- a nonlinear system must then be solved.
- to this end, we can consider the following predictor-corrector method

# Outline

# Predictor-corrector method

- To solve the equation

$$y_{n+1} = y_n + \frac{h}{24}(9f(t_{n+1}, y_{n+1}) + 19f_n - 5f_{n-1} + f_{n-2})$$

  we can use a successive approximation method (i.e. fixed-point) consisting in building the sequence $\tilde{y}_0, \tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_p$ defined by

$$\begin{cases} \tilde{y}_{p+1} = y_n + \frac{h}{24}\left(9f\left(t_{n+1}, \tilde{y}_p\right) + 19f_n - 5f_{n-1} + f_{n-2}\right) \\ \tilde{y}_0 \text{ to choose.} \end{cases}$$

- one can iterate until convergence (in general $\tilde{y}_p$ converges towards $y_{n+1}$ when $p$ tends to infinity)

# Predictor-corrector method

- most of the time, one only iterates a few times, even sometimes 1 or 2.
- in addition, the initial value $\tilde{y}_0$ is often obtained through one step of an explicit method of the same order
- then, we have a predictor-corrector method : the evaluation of $\tilde{y}_0$ corresponds to a prediction; this value is then next corrected through one or two iterations of a fixed point algorithm.

# Predictor-corrector method

- finally, the following scheme is often used

$$\begin{cases} \text{Predictor: fourth-order Adams-Bashforth method} \\ \tilde{y}_0 = y_n + \frac{h}{24}\left(55 f_n - 59 f_{n-1} + 37 f_{n-2} - 9 f_{n-3}\right) \\ \text{Corrector: one or two iterations of the Adams-Moulton method} \\ \text{of order 4} \\ \tilde{y}_{p+1} = y_n + \frac{h}{24}\left(9 f\left(t_{n+1}, \tilde{y}_p\right) + 19 f_n - 5 f_{n-1} + f_{n-2}\right), \; p = 0, 1. \end{cases}$$

- we show that this method is also of order 4
- its stability is clearly better than for the Adams-Bashforth scheme
- the solution to the nonlinear system related to the Adams-Moulton formula is finally done explicitly.

# Outline

# Backward differentiation formula

A last category of multi-step method consists of evaluating $f$ at the end of the current step $(t_{n+s}, y_{n+s})$, and driving an interpolating polynomial for $y$ with the points $(t_{n+s}, \dots, t_n)$.
We start from $y'(t_{n+s}) = f(t_{n+s}, y(t_{n+s}))$ and use the approximation

$$p'_{n,s}(t_{n+s}) = f(t_{n+s}, y_{n+s})$$

By doing so we end up with BDF schemes of order $s$

$$\sum_{k=0}^{s} a_k y_{n+k} = h\beta f(t_{n+s}, y_{n+s})$$

**Example**

BDF1: $y_{n+1} - y_n = hf(t_{n+1}, y_{n+1})$
BDF2: $y_{n+2} - \frac{4}{3}y_{n+1} + \frac{1}{3}y_n = \frac{2h}{3}f(t_{n+2}, y_{n+2})$

# Backward differentiation formula (BDF)

**Remarks**

- BDF schemes with $s$ stages are of order $s$
- BDF schemes are implicit
- they are popular for stiff problems because of their stability property
- methods with $s > 6$ cannot be used

# Summary of the contents

We have seen some numerical methods to solve IVP

1. One-step methods of different orders (Euler, RK)
2. Multi-step methods (AM, AB, BDF)
3. Predictor-corrector methods

They are various differences between the schemes

- The methods can be explicit or implicit, and of different orders of accuracy
- Implicit methods are more stable than explicit methods, but are also more costly
- Usually an adaptive step size solver is necessary
- Writing an efficient ODE solver requires a good knowledge and experience of these methods