Programming, numerics and optimization

Lecture B-2:
Numerical integration of ordinary differential equations

Łukasz Jankowski
ljank@ippt.pan.pl

Institute of Fundamental Technological Research
Room 4.32, Phone +22.8261281 ext. 428

March 30, 2021[1]

[1] Current version is available at http://info.ippt.pan.pl/~ljank

Outline

1. Basics
2. Explicit one-step methods
3. Implicit one-step methods
4. Multistep methods
5. Methods for 2$^{\text{nd}}$ order equations
6. Homework 5

Outline

1. Basics
   - Basic notions
   - Reduction to first order ODE
   - Problem classification
   - Sources of errors
2. Explicit one-step methods
3. Implicit one-step methods
4. Multistep methods
5. Methods for 2$^{\text{nd}}$ order equations
6. Homework 5

Basic notions

Ordinary differential equation (ODE)

An ordinary differential equation (ODE) of order n is an equality involving

- a function and
- its derivatives up to order n with respect to only one variable.

The highest order derivative can be given *explicitly*,

$$y^{(n)} = F(t, y, y', \ldots, y^{(n-1)}),$$

or in an *implicit* form,

$$F(t, y, y', \ldots, y^{(n)}) = 0.$$

$y^{(k)} = \frac{\partial^k y}{\partial t^k}$ denotes the kth order derivative of y with respect to t.

Basic notions

Linear ODE

An ODE of order n is called linear, if it is of the form

$$a_n(t)y^{(n)} + a_{n-1}(t)y^{(n-1)} + \ldots + a_1(t)y' + a_0(t)y = q(t),$$

or

$$y^{(n)} = a_{n-1}(t)y^{(n-1)} + \ldots + a_1(t)y' + a_0(t)y + q(t).$$

Homogenous ODE

A linear ODE is said to be homogeneous, if $q(t) = 0$. Otherwise, it is called inhomogeneous (or nonhomogeneous).

Basic notions

Autonomous ODE (time-invariant)

An ODE of the form

$$y^{(n)} = F(t, y, y', \dots, y^{(n-1)})$$

is called autonomous, if F does not depend explicitly on t, that is if it is of the form

$$y^{(n)} = F(y, y', \dots, y^{(n-1)}).$$

If t is time, an autonomous ODE is called time-invariant system.

In an autonomous (time-invariant) system, there are no external, time-dependent influences (such as an external driving force in a mechanical system). For example, most laws of nature are generally assumed to be time-invariant.

Reduction to first order ODE

An *n*th order ODE,

$$y^{(n)} = F(t, y, y', \dots, y^{(n-1)}),$$

can be reduced to a first order ODE

$$\mathbf{x}' = \mathbf{G}(t, \mathbf{x}), \qquad \text{where } \mathbf{x} = (x_1, \dots, x_n),$$

by the following substitution:

$$\begin{aligned} x_1 &= y \\ x_2 &= y' \\ \dots & \\ x_n &= y^{(n-1)}, \end{aligned}$$

Reduction to first order ODE

In other words, an nth order ODE in one variable,

$$y^{(n)} = F(t, y, y', \ldots, y^{(n-1)})$$

can be reduced to a 1st order ODE in n variables,

$$\begin{cases} x_1' = x_2 \\ x_2' = x_3 \\ \ldots \\ x_{n-1}' = x_n \\ x_n' = F(t, x_1, x_2, \ldots, x_n), \end{cases}$$

where

$$\begin{aligned} x_1 &= y \\ x_2 &= y' \\ \ldots \\ x_n &= y^{(n-1)}. \end{aligned}$$

Reduction to first order ODE — example

Linear equation of motion

Linear equation of motion,

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{f},$$

can be reduced by the substitution $\mathbf{u} = \mathbf{x}$, $\mathbf{v} = \dot{\mathbf{x}}$ to a first order equation (*state space form*)

$$\begin{cases} \dot{\mathbf{u}} = \mathbf{v} \\ \dot{\mathbf{v}} = \mathbf{M}^{-1}\left[\mathbf{f} - \mathbf{K}\mathbf{u} - \mathbf{C}\mathbf{v}\right], \end{cases}$$

that is, to

$$\begin{bmatrix} \dot{\mathbf{u}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1} \end{bmatrix} \mathbf{f}.$$

Problem classification

Problems involving ODEs can be classified into three main classes:

1. Initial value problems (IVP), in which values of the unknown function are specified in one time moment only (initial conditions):

$$\text{solve } \dot{\mathbf{x}} = \mathbf{F}(t, \mathbf{x}), \text{ given } \mathbf{x}(0).$$

2. Boundary value problems (BVP), in which values of the unknown function are specified in two or more time moments:

$$\text{solve } \dot{\mathbf{x}} = \mathbf{F}(t, \mathbf{x}), \text{ given } x_1(t_1), x_2(t_2), \ldots, x_n(t_n).$$

3. Eigenproblems, which are similar to eigenproblems in linear algebra:

$$\text{solve } \boldsymbol{\mathcal{L}} \boldsymbol{x} = \boldsymbol{\lambda} \mathbf{x},$$

where $\boldsymbol{\mathcal{L}}$ is a given linear differential operator.

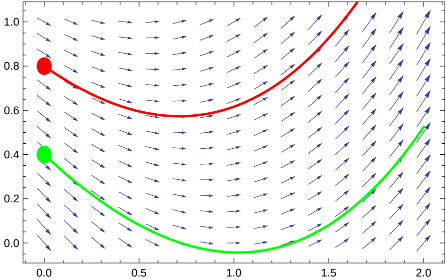
Problem classification

Initial value problems

Given an ODE $\dot{\mathbf{x}} = \mathbf{F}(t, \mathbf{x})$ and the initial conditions $\mathbf{x}(0) = \mathbf{x}_0$.

There is a *unique solution* to the system within a neighborhood D of the initial point $\mathbf{x}_0$, if for $\mathbf{x} \in D$ and $t \in [0, T]$ the function $\mathbf{F}$ is differentiable with respect to t and $x_1, \ldots, x_n$ (in fact, Lipschitz continuity suffices).

- The function **F** defines a vector velocity field.
- If $\mathbf{F} = \mathbf{F}(\mathbf{x})$, the system is autonomous (the vector velocity field does not change in time).



Problem classification

Initial value problems — example

Non-linearized pendulum

Let the motion of a non-linearized pendulum be governed by

$$\ddot{\alpha} + \sin\alpha = 0.$$

The *phase space* is formed by α and $\dot{\alpha}$,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \alpha \\ \dot{\alpha} \end{bmatrix},$$

so that

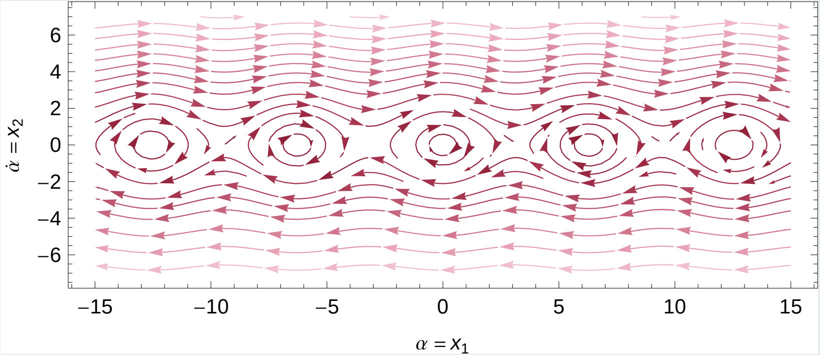
$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \dot{\alpha} \\ \ddot{\alpha} \end{bmatrix} = \begin{bmatrix} x_2 \\ -\sin x_1 \end{bmatrix}.$$

Problem classification

Initial value problems — example

Non-linearized pendulum

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \dot{\alpha} \\ \ddot{\alpha} \end{bmatrix} = \begin{bmatrix} x_2 \\ -\sin x_1 \end{bmatrix}.$$

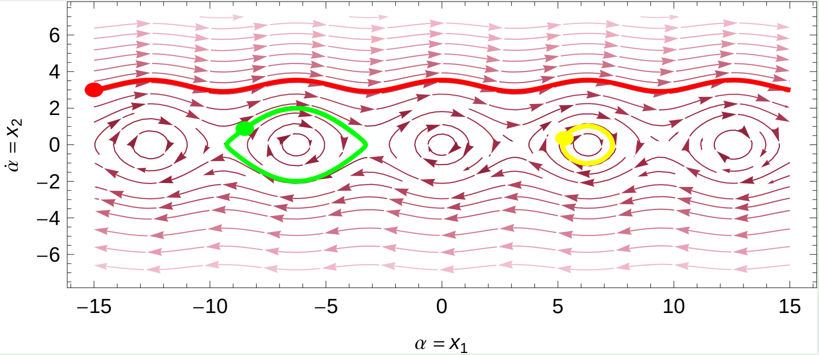


Problem classification

Initial value problems — example

Non-linearized pendulum

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \dot{\alpha} \\ \ddot{\alpha} \end{bmatrix} = \begin{bmatrix} x_2 \\ -\sin x_1 \end{bmatrix}.$$



Sources of errors

Two general sources of errors in numerical integration of ODEs

1. Approximate data

   In practice, all the data (initial conditions, equation parameters) are known approximately. Ill-conditioned problems will magnify these errors.

2. Numerical algorithms

   - *Stability*. A method is unconditionally stable, if the computed solution to $\dot{x}(t) = \lambda x(t)$ converges to zero whenever $\operatorname{Re}\lambda < 0$ (irrespective of the step size). If the convergence holds only for Δt small enough, the method is called conditionally stable.
   - *Local truncation error (LTE)* is the error introduced in each time step. A method is consistent, if LTE vanishes as $\Delta t \to 0$.
   - *Global truncation error* is the current LTE plus all the errors accumulated from the previous time steps. A method is convergent if the global truncation error vanishes as $\Delta t \to 0$.

In general, convergence = consistency + stability.

Outline

1. Basics
2. Explicit one-step methods
   - Euler method
   - Runge–Kutta methods
   - Convergence, order and stability of RK2 methods
   - Richardson extrapolation
   - Adaptive step size control
3. Implicit one-step methods
4. Multistep methods
5. Methods for 2^{nd} order equations

Basic idea

The time is discretized and the solution is constructed stepwise, *using the information from the last step only*.

The value $\mathbf{x}(t + \Delta t)$ is computed based on

1. the value $\mathbf{x}(t)$ and
2. the velocity vector $\dot{\mathbf{x}}(t)$.

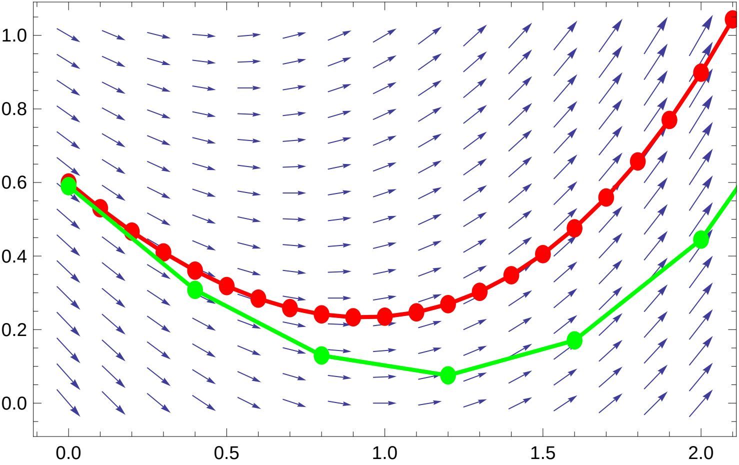
Sometimes intermediate time points from the last step $[t, t + \Delta t]$ are also used.

Explicit methods are

- numerically cheap (all formulas are explicit),
- can have an arbitrarily high global accuracy $O(\Delta t^n)$,
- but are only *conditionally stable*.

Basic idea

The time is discretized and the solution is constructed stepwise, *using the information from the last step only*.



Euler method

Euler method is probably the simplest method:

$$\frac{\mathbf{x}(t+\Delta t)-\mathbf{x}(t)}{\Delta t} \approx \dot{\mathbf{x}}(t) = \mathbf{F}(t,\mathbf{x}),$$

hence

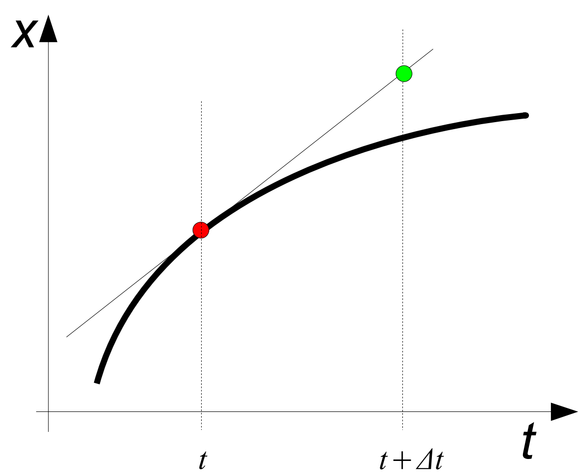
$$\mathbf{x}(t+\Delta t) \approx \mathbf{x}(t) + \Delta t\mathbf{F}(t,\mathbf{x}).$$

The first point $\mathbf{x}(0)$ is defined by the initial conditions.

1. The computed solution is very sensitive to the length of the time step Δt:
   - small Δt: long computing time, better accuracy (up to the accuracy of the floating point arithmetics)
   - large Δt: shorter computing time, worse accuracy
2. Global accuracy is $O(\Delta t)$ (a first order method).
3. The method uses local linear approximation to $\mathbf{x}(t)$.

Euler method

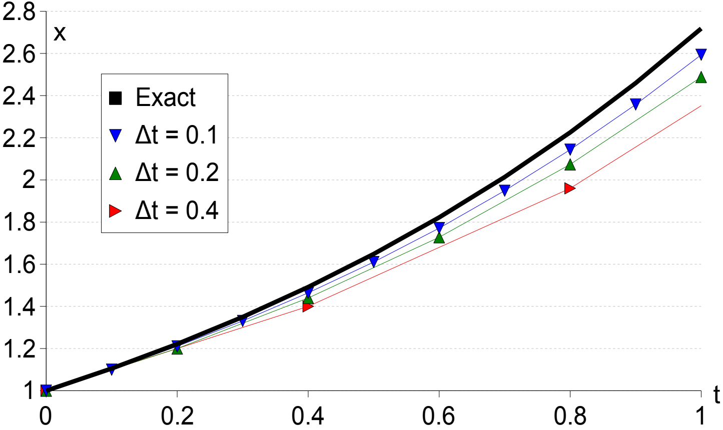
Euler method



Euler method — example

Equation: $\dot{x} = x, \quad x(0) = 1$
Exact solution: $x(t) = e^t$

Step: $x(t + \Delta t) \approx x(t)(1 + \Delta t)$
Accuracy: $O(\Delta t)$



Runge–Kutta methods

- In each step, the Euler method takes into account only the most recent point from the solution.
- Runge–Kutta methods are a family of methods, which (unlike the Euler method) take into account also intermediate time steps within the current time interval $[t, t + \Delta t]$.
- Methods that use
  - two time points (for example t and $t + \frac{1}{2}\Delta t$) are said to be of the 2^{nd} *order*,
  - four time points are said to be of the 4^{th} *order*, etc.
- The most commonly used method of this family is so popular that it is often called *the Runge–Kutta method* or RK4 (4^{th} order Runge–Kutta).

2^{nd} order Runge–Kutta methods

In the Euler method, the slope $\mathbf{D}$ that is used to compute the next point by

$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \Delta t\, \mathbf{D}$$

is approximated as $\mathbf{D} \approx \dot{\mathbf{x}}(t)$. The two most popular 2^{nd} order Runge–Kutta methods improve the approximation accuracy by using either the

1. *derivative in the midpoint* $t + \frac{1}{2}\Delta t$ (this is usually called the 2^{nd} order Runge–Kutta method) or
2. the *mean derivative* at t and $t + \Delta t$ (called the Heun's method or the modified 2^{nd} order Runge–Kutta method).

2nd order Runge–Kutta methods

$$\mathbf{x}(t+\Delta t) \approx \mathbf{x}(t) + \Delta t\, \mathbf{D},$$

1. The 2nd order Runge–Kutta method approximates **D** with the derivative in the midpoint $t + \frac{1}{2}\Delta t$,

$$\mathbf{D} \approx \dot{\mathbf{x}}\left(t + \frac{1}{2}\Delta t\right) = \mathbf{F}\left(t + \frac{1}{2}\Delta t,\ \mathbf{x}\left(t + \frac{1}{2}\Delta t\right)\right).$$

The value in the midpoint is approximated using the simple Euler formula,

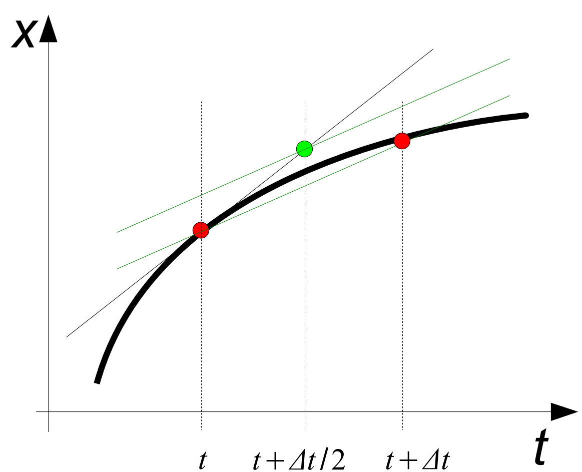
$$\mathbf{x}\left(t + \frac{1}{2}\Delta t\right) \approx \mathbf{x}(t) + \frac{1}{2}\Delta t\, \mathbf{F}(t, \mathbf{x}(t)),$$

so that

$$\mathbf{D} = \mathbf{F}\left(t + \frac{1}{2}\Delta t,\ \mathbf{x}(t) + \frac{1}{2}\Delta t\, \mathbf{F}(t, \mathbf{x}(t))\right).$$

2^{nd} order Runge–Kutta methods

2^{nd} order Runge–Kutta



2^{nd} order Runge–Kutta methods

$$\mathbf{x}(t+\Delta t) \approx \mathbf{x}(t) + \Delta t\, \mathbf{D},$$

2. The Heun's method (aka modified 2^{nd} order Runge–Kutta) approximates $\mathbf{D}$ with the mean derivative at t and $t+\Delta t$. The latter is approximated via the Euler formula. Thus,

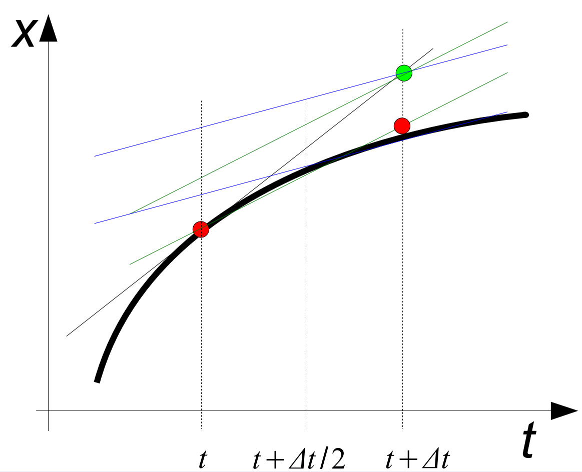
$$\mathbf{D} \approx \frac{\dot{\mathbf{x}}(t) + \dot{\mathbf{x}}(t+\Delta t)}{2},$$

where

$$\begin{aligned}
\dot{\mathbf{x}}(t) &= \mathbf{F}(t, \mathbf{x}(t)), \\
\dot{\mathbf{x}}(t+\Delta t) &= \mathbf{F}(t+\Delta t, \mathbf{x}(t+\Delta t)) \\
&\approx \mathbf{F}(t+\Delta t, \mathbf{x}(t) + \Delta t\, \dot{\mathbf{x}}(t)) \\
&\approx \mathbf{F}(t+\Delta t, \mathbf{x}(t) + \Delta t\, \mathbf{F}(t, \mathbf{x}(t))).
\end{aligned}$$

2^{nd} order Runge–Kutta methods

Heun's method (modified 2^{nd} order Runge–Kutta)



2^{nd} order Runge–Kutta methods

Using a more algorithmic notation:

2^{nd} order Runge–Kutta method

$$\mathbf{d}_1 = \mathbf{F}\left(t, \mathbf{x}(t)\right)$$

$$\mathbf{d}_2 = \mathbf{F}\left(t + \frac{1}{2}\Delta t,\ \mathbf{x}(t) + \frac{1}{2}\Delta t\,\mathbf{d}_1\right)$$

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t\,\mathbf{d}_2$$

Heun's method (modified RK2)

$$\mathbf{d}_1 = \mathbf{F}\left(t, \mathbf{x}(t)\right)$$

$$\mathbf{d}_2 = \mathbf{F}\left(t + \Delta t,\ \mathbf{x}(t) + \Delta t\,\mathbf{d}_1\right)$$

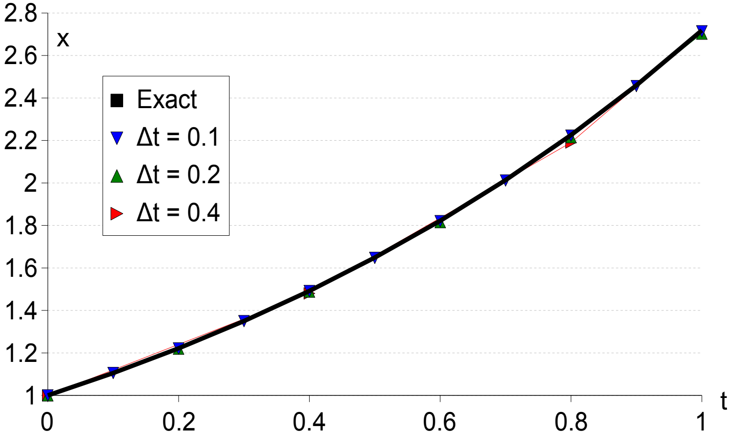
$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t\frac{\mathbf{d}_1 + \mathbf{d}_2}{2}$$

- Both methods require two computations of **F** per step.
- Global accuracy is $O(\Delta t^2)$ (a second order method).
- Local quadratic approximation to $\mathbf{x}(t)$ is used.

2^{nd} order Runge–Kutta method — example

Equation: $\dot{x} = x, \quad x(0) = 1$
Exact solution: $x(t) = \mathrm{e}^t$

2^{nd} order Runge–Kutta method
Accuracy: $O(\Delta t^2)$

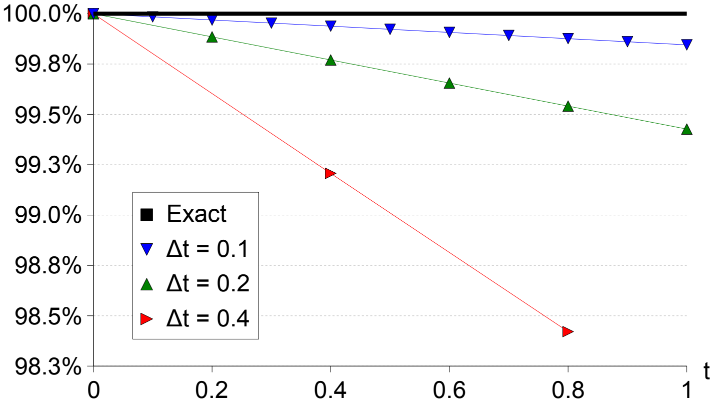


2^{nd} order Runge–Kutta method — example

Equation: $\dot{x} = x, \quad x(0) = 1$
Exact solution: $x(t) = \mathrm{e}^t$

2^{nd} order Runge–Kutta method
Accuracy: $O(\Delta t^2)$

per cent of the exact solution



4th order Runge–Kutta method

As in the Euler and 2nd order Runge–Kutta methods, the next point is computed using an approximation of the slope $\mathbf{D}$ as

$$\mathbf{x}(t+\Delta t) \approx \mathbf{x}(t) + \Delta t\, \mathbf{D}.$$

The most popular 4th order Runge–Kutta method approximates $\mathbf{D}$ using a weighted average of derivatives at *four* time points,

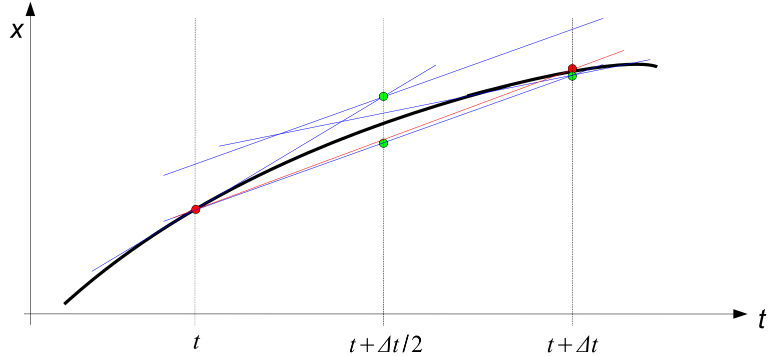
$$\begin{aligned}
\mathbf{d}_1 &= \mathbf{F}\left(t, \mathbf{x}(t)\right)\\
\mathbf{d}_2 &= \mathbf{F}\left(t + \frac{1}{2}\Delta t,\ \mathbf{x}(t) + \frac{1}{2}\Delta t\, \mathbf{d}_1\right)\\
\mathbf{d}_3 &= \mathbf{F}\left(t + \frac{1}{2}\Delta t,\ \mathbf{x}(t) + \frac{1}{2}\Delta t\, \mathbf{d}_2\right)\\
\mathbf{d}_4 &= \mathbf{F}\left(t + \Delta t,\ \mathbf{x}(t) + \Delta t\, \mathbf{d}_3\right),
\end{aligned}$$

to obtain

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t\, \frac{\mathbf{d}_1 + 2\mathbf{d}_2 + 2\mathbf{d}_3 + \mathbf{d}_4}{6}.$$

4th order Runge–Kutta method

4th order Runge–Kutta method



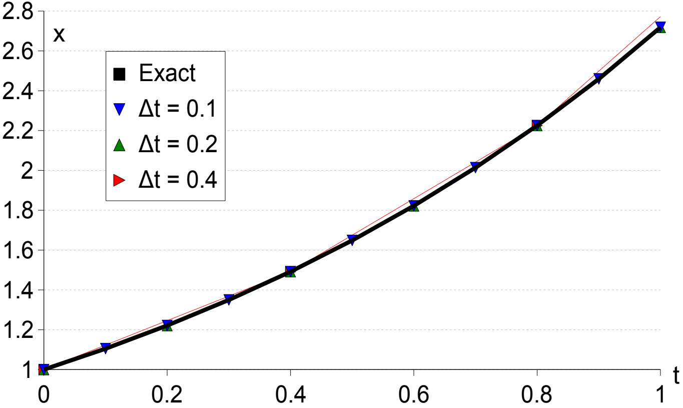
- The method requires four computations of **F** per step.
- Global accuracy is $O(\Delta t^4)$ (a fourth order method).
- A fourth-order local approximation to $\mathbf{x}(t)$ is used.

4^{th} order Runge–Kutta method — example

Equation: $\dot{x} = x, \quad x(0) = 1$
Exact solution: $x(t) = e^t$

4^{th} order Runge–Kutta method
Accuracy: $O(\Delta t^4)$

Notice the need for a smooth interpolation at large step sizes

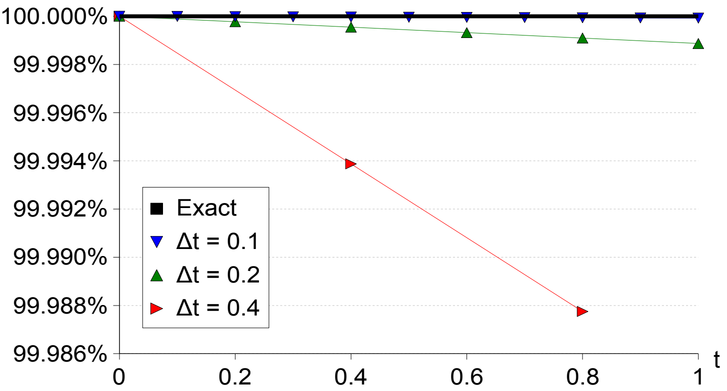


4^{th} order Runge–Kutta method — example

Equation: $\dot{x} = x, \quad x(0) = 1$
Exact solution: $x(t) = e^t$

4^{th} order Runge–Kutta method
Accuracy: $O(\Delta t^4)$

per cent of the exact solution



Convergence, order and stability of RK2 methods

Assumption of smoothness

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t, \mathbf{x}(t))$$

Assume that both $\mathbf{x}$ and $\mathbf{F}(t, \mathbf{x})$ are *sufficiently smooth*, so that

1. The second derivative of $\mathbf{x}(t)$ can be computed as

$$\ddot{\mathbf{x}}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\dot{\mathbf{x}}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{F}(t, \mathbf{x}(t)) = \mathbf{F}_t(t, \mathbf{x}(t)) + \mathbf{F}_{\mathbf{x}}(t, \mathbf{x}(t))\,\dot{\mathbf{x}}(t).$$

2. $\mathbf{F}$ can be expanded into its Taylor series around $(t, \mathbf{x})$ as

$$\begin{aligned}\mathbf{F}(t + \Delta t, \mathbf{x} + \Delta\mathbf{x}) = {} & \mathbf{F}(t, \mathbf{x}) + \mathbf{F}_t(t, \mathbf{x})\,\Delta t + \mathbf{F}_{\mathbf{x}}(t, \mathbf{x})\,\Delta\mathbf{x} \\ & + O(\Delta t^2) + O(\Delta t\|\Delta\mathbf{x}\|) + O(\|\Delta\mathbf{x}\|^2),\end{aligned}$$

where the rest simplifies to $O(\Delta t^2)$.

Convergence, order and stability of RK2 methods

General scheme of RK2 methods

RK2 methods for $\dot{\mathbf{x}}(t) = \mathbf{F}(t, \mathbf{x}(t))$

$$\mathbf{x}_{\mathrm{RK2}}(t + \Delta t) = \mathbf{x}(t) + \Delta t \left(a_1 \mathbf{d}_1 + a_2 \mathbf{d}_2\right),$$

where

$$\begin{aligned} \mathbf{d}_1 &= \mathbf{F}(t, \mathbf{x}(t)) = \dot{\mathbf{x}}(t), \\ \mathbf{d}_2 &= \mathbf{F}(t + b\Delta t,\, \mathbf{x}(t) + c\Delta t\, \dot{\mathbf{x}}(t)) \end{aligned}$$

and a_1, a_2, b, c are certain coefficients.

	a_1	a_2	b	c
"the RK2 method"	0	1	0.5	0.5
Heun's method	0.5	0.5	1	1

Convergence, order and stability of RK2 methods

Taylor series expansion of $\mathbf{x}(t)$:

$$\begin{aligned}\mathbf{x}(t+\Delta t) &= \mathbf{x}(t) + \Delta t\,\dot{\mathbf{x}}(t) + \frac{1}{2}\Delta t^2 \ddot{\mathbf{x}}(t) + O(\Delta t^3)\\ &= \mathbf{x}(t) + \Delta t\,\dot{\mathbf{x}}(t)\\ &\quad + \frac{1}{2}\Delta t^2\left[\mathbf{F}_t(t,\mathbf{x}(t)) + \mathbf{F}_{\mathbf{x}}(t,\mathbf{x}(t))\,\dot{\mathbf{x}}(t)\right] + O(\Delta t^3)\end{aligned}$$

Taylor series expansion of $\mathbf{F}$ in the RK2 formula:

$$\begin{aligned}\mathbf{x}_{\mathrm{RK2}}(t+\Delta t) &= \mathbf{x}(t) + a_1\Delta t\,\dot{\mathbf{x}}(t) + a_2\Delta t\,\mathbf{F}(t+b\Delta t, \mathbf{x}(t) + c\Delta t\dot{\mathbf{x}}(t))\\ &= \mathbf{x}(t) + a_1\Delta t\,\dot{\mathbf{x}}(t) + a_2\Delta t\,\mathbf{F}(t,\mathbf{x}(t))\\ &\quad + a_2 b\Delta t^2\mathbf{F}_t(t,\mathbf{x}(t)) + a_2 c\Delta t^2\mathbf{F}_{\mathbf{x}}(t,\mathbf{x}(t))\,\dot{\mathbf{x}}(t) + O(\Delta t^3)\\ &= \mathbf{x}(t) + (a_1+a_2)\Delta t\,\dot{\mathbf{x}}(t)\\ &\quad + a_2\Delta t^2\left[b\mathbf{F}_t(t,\mathbf{x}(t)) + c\mathbf{F}_{\mathbf{x}}(t,\mathbf{x}(t))\,\dot{\mathbf{x}}(t)\right] + O(\Delta t^3).\end{aligned}$$

Convergence, order and stability of RK2 methods

Therefore, by equating the coefficients in the two expansions:

- The method is *consistent*, if

$$a_1 + a_2 = 1$$

- If additionally

$$a_2 b = 0.5,$$
$$a_2 c = 0.5,$$

then the local (one-step) truncation error (LTE) is $O(\Delta t^3)$ and so the global error is $O(\Delta t^2)$.

Convergence, order and stability of RK2 methods

The RK2 family

There are three equations with four unknowns, so they give a one-parameter family of Runge–Kutta second order methods:

- If $a_2 \neq 0$, then

$$\begin{aligned}\mathbf{x}_{\mathrm{RK2}}(t+\Delta t) &= \mathbf{x}(t) + \Delta t\left[(1-a_2)\mathbf{d}_1 + a_2\mathbf{d}_2\right],\\ \mathbf{d}_1 &= \mathbf{F}(t,\mathbf{x}(t)) = \dot{\mathbf{x}}(t),\\ \mathbf{d}_2 &= \mathbf{F}\left(t + \frac{1}{2a_2}\Delta t,\, \mathbf{x}(t) + \frac{1}{2a_2}\Delta t\,\dot{\mathbf{x}}(t)\right).\end{aligned}$$

- If $a_2 = 0$, then the Euler method is obtained:

$$\begin{aligned}\mathbf{x}_{\mathrm{RK2}}(t+\Delta t) &= \mathbf{x}(t) + \Delta t\,\mathbf{d}_1,\\ \mathbf{d}_1 &= \mathbf{F}(t,\mathbf{x}(t)) = \dot{\mathbf{x}}(t).\end{aligned}$$

Convergence, order and stability of RK2 methods

The RK2 family — stability

The test case for stability is

$$\dot{\mathbf{x}}(t) = \lambda \mathbf{x}(t), \qquad \mathrm{Re}\,\lambda < 0.$$

Substitution of the RK2 family formulas ($a_2 \neq 0$) yields

$$\mathbf{x}_{\mathrm{RK2}}(t+\Delta t) = \mathbf{x}_{\mathrm{RK2}}(t)\left[1 + \Delta t\lambda + \frac{1}{2}\Delta t^2\lambda^2\right],$$

which defines a convergent solution if and only if

$$|1 + \Delta t\lambda + \frac{1}{2}\Delta t^2\lambda^2| < 1.$$

Similarly, for the Euler method it must hold that

$$|1 + \Delta t\lambda| < 1,$$

and for the RK4 method

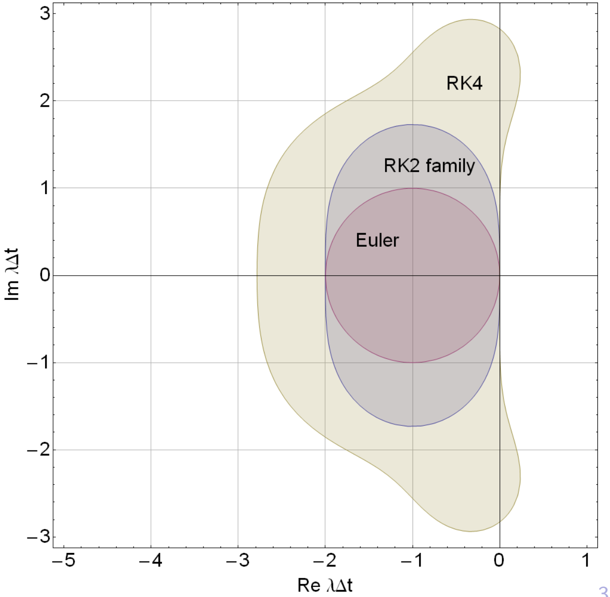
$$|1 + \Delta t\lambda + \frac{1}{2}\Delta t^2\lambda^2 + \frac{1}{6}\Delta t^3\lambda^3 + \frac{1}{24}\Delta t^4\lambda^4| < 1.$$

Convergence, order and stability of RK methods

Domains of absolute stability on the complex plane: convergence to 0 of the computed solution to

$$\dot{\mathbf{x}}(t) = \lambda \mathbf{x}(t).$$

All explicit RK methods are conditionally stable only.



Richardson extrapolation

Assume the order of the global accuracy is known, e.g. for the Euler method

$$\mathbf{x}_{\Delta t}(t) = \mathbf{x}_{\text{exact}}(t) + \Delta t\, \mathbf{c}(t) + O(\Delta t^2).$$

In each time step, the next point can be computed using two time steps, e.g. a single Δt and two $\frac{1}{2}\Delta t$, to obtain two approximations, $\mathbf{x}_{\Delta t}(t)$ and $\mathbf{x}_{\frac{1}{2}\Delta t}(t)$. If the higher-order terms are neglected, a simple linear system is obtained,

$$\begin{aligned} \mathbf{x}_{\Delta t}(t) &\approx \mathbf{x}_{\text{exact}}(t) + \Delta t\, \mathbf{c}(t) \\ \mathbf{x}_{\frac{1}{2}\Delta t}(t) &\approx \mathbf{x}_{\text{exact}}(t) + \frac{1}{2}\Delta t\, \mathbf{c}(t), \end{aligned}$$

which yields

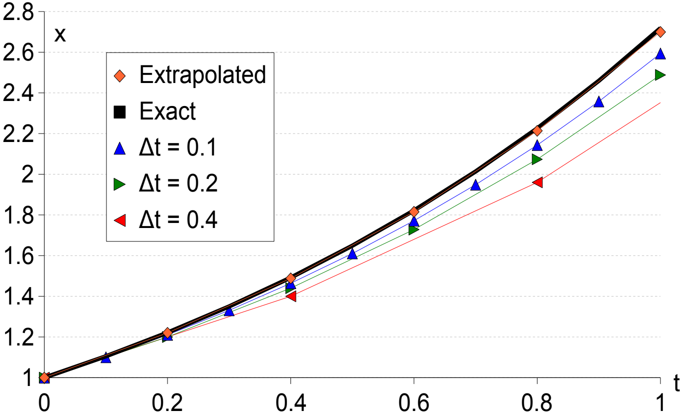
$$\mathbf{x}_{\text{exact}}(t) \approx 2\mathbf{x}_{\frac{1}{2}\Delta t}(t) - \mathbf{x}_{\Delta t}(t).$$

Richardson extrapolation

- If more higher-order terms are accounted for, computations at three or more time steps have to be done.
- In “Numerical recipes” Richardson extrapolation is described *“for alchemist readers as turning lead into gold”*.
- The Richardson extrapolation improves the accuracy of the computed solution. However, the estimate of the accuracy is lost.

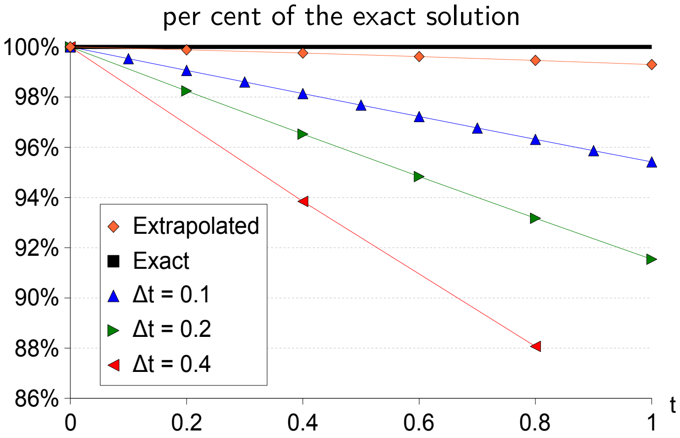
Richardson extrapolation — example

- Euler method example, accuracy $O(\Delta t)$
- Extrapolation based on time steps $\Delta t = 0.1$ and $\Delta t = 0.2$



Richardson extrapolation — example

- Euler method example, accuracy $O(\Delta t)$
- Extrapolation based on time steps $\Delta t = 0.1$ and $\Delta t = 0.2$



Richardson extrapolation — example

- Standard Richardson extrapolation uses a polynomial expansion in Δt, e.g. for fourth-order Runge–Kutta methods:

$$\mathbf{x}_{\Delta t}(t) = \mathbf{x}_{\text{exact}}(t) + \Delta t^4 \mathbf{c}_4(t) + \Delta t^5 \mathbf{c}_5(t) + \ldots$$

- There are many variations to the standard approach. E.g. a rational expansion might be used,

$$\mathbf{x}_{\Delta t}(t) \approx \mathbf{x}_{\text{exact}}(t) + \frac{\mathbf{P}(\Delta t)}{\mathbf{Q}(\Delta t)},$$

where **P** and **Q** are polynomials.

Adaptive step size control

The methods described so far use constant step sizes. However, to minimize the numerical costs, an adaptive step size can be used.

> *Many small steps should tiptoe through treacherous terrain, while a few great strides should speed through smooth uninteresting countryside. /Numerical recipes/*

In each time step, the accuracy can be monitored by comparing $\mathbf{x}_{\Delta t}(t)$ and $\mathbf{x}_{\frac{1}{2}\Delta t}(t)$ (*step doubling procedure*):

- If the difference is small enough, retain the half-step solution and double the time step.
- Otherwise, halve the time step and repeat the (halved) step.

Additionally, since $\mathbf{x}(t + \Delta t)$ is computed twice, the Richardson extrapolation can be used.

Outline

1. Basics
2. Explicit one-step methods
3. Implicit one-step methods
4. Multistep methods
5. Methods for 2$^{\text{nd}}$ order equations
6. Homework 5

Implicit methods

Explicit methods rely basically on the formula

$$\mathbf{x}(t+\Delta t) \approx \mathbf{x}(t) + \Delta t\, \mathbf{D}$$

with the slope $\mathbf{D}$ is approximated using the *explicitly* given derivative $\mathbf{F}(t, \mathbf{x}(t))$ at the beginning of the current time step interval $[t, \ldots, t+\Delta t]$. The derivatives at the end or in the intermediate points are only *estimated* using this initial value.

Implicit methods make use of the (initially unknown) derivative at the end of the time step, $\dot{\mathbf{x}}(t+\Delta t) = \mathbf{F}(t+\Delta t, \mathbf{x}(t+\Delta t))$.

- The value $\mathbf{x}(t+\Delta t))$ is given only implicitly, so it has to be found by solving a (possibly nonlinear) equation, which might be time-consuming.
- Many popular implicit methods are unconditionally stable.

Implicit methods

- The simplest implicit method (*backward Euler*) is symmetric to the explicit Euler method:

$$\mathbf{x}(t+\Delta t) \approx \mathbf{x}(t) + \Delta t\, \mathbf{F}(t+\Delta t, \mathbf{x}(t+\Delta t)).$$

- Slightly more advanced is the *trapezoidal method*:

$$\mathbf{x}(t+\Delta t) \approx \mathbf{x}(t) + \frac{1}{2}\Delta t \left[\mathbf{F}(t, \mathbf{x}(t)) + \mathbf{F}(t+\Delta t, \mathbf{x}(t+\Delta t))\right].$$

  It is similar to the Heun's method, but uses the *exact* derivative at $t + \Delta t$ instead of an estimate.

In such formulas, the unknown $\mathbf{x}(t+\Delta t)$ is given only implicitly. It has to be found by solving a (possibly nonlinear) equation.

Implicit methods — example

Equation: $\dot{x} = x$, $x(0) = 1$. Exact solution: $x(t) = \mathrm{e}^t$.

- (Explicit) Euler method:

$$\begin{aligned} \mathbf{x}(t+\Delta t) &\approx \mathbf{x}(t) + \Delta t\, \mathbf{x}(t) \\ &= (1+\Delta t)\mathbf{x}(t). \end{aligned}$$

- Backward Euler method (simple implicit):

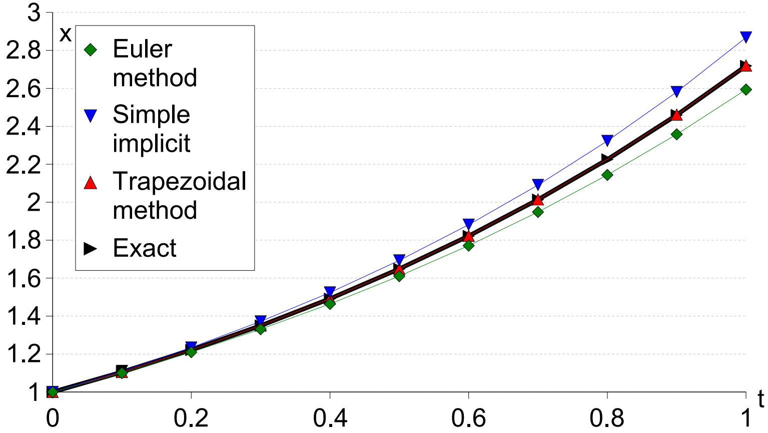
$$\begin{aligned} \mathbf{x}(t+\Delta t) &\approx \mathbf{x}(t) + \Delta t\, \mathbf{x}(t+\Delta t) \\ &= \frac{\mathbf{x}(t)}{1-\Delta t}. \end{aligned}$$

- Trapezoidal method:

$$\begin{aligned} \mathbf{x}(t+\Delta t) &\approx \mathbf{x}(t) + \frac{1}{2}\Delta t\, [\mathbf{x}(t) + \mathbf{x}(t+\Delta t)] \\ &= \mathbf{x}(t) \frac{1+\frac{1}{2}\Delta t}{1-\frac{1}{2}\Delta t}. \end{aligned}$$

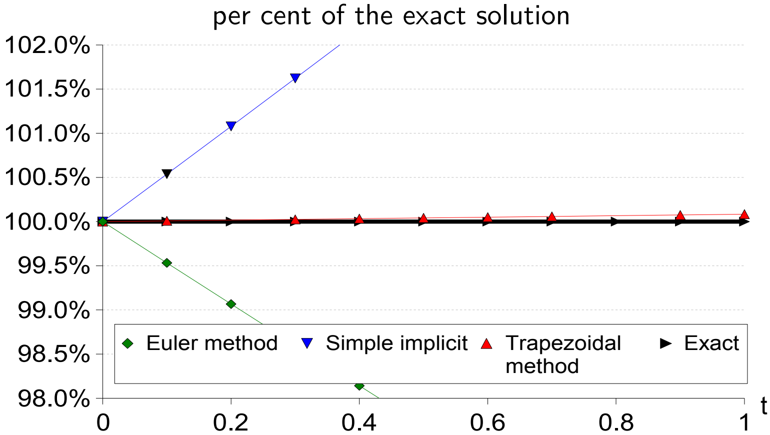
Implicit methods — example

Equation: $\dot{x} = x$, $x(0) = 1$. Exact solution: $x(t) = e^t$.



Implicit methods — example

Equation: $\dot{x} = x$, $x(0) = 1$. Exact solution: $x(t) = e^t$.



Outline

1. Basics
2. Explicit one-step methods
3. Implicit one-step methods
4. Multistep methods
    - Modified midpoint method
    - Predictor-corrector methods
5. Methods for 2$^{\text{nd}}$ order equations
6. Homework 5

Multistep methods

All methods described so far are one-step methods, i.e. the solution is advanced based on the values and derivatives computed for the last time step only, often at fractional time instances.

Multistep methods take into account results obtained in several previous time steps:

- On one hand, they are usually quicker than one-step methods, because they do not need calculations at intermediate, fractional time steps.
- On the other hand, the accuracy can be lower, because the solution is computed based on the results that are more distant in time than in one-step methods. Thus, multistep methods should be used with smooth functions.

Modified midpoint method

The probably simplest multistep method is based on the formula

$$\frac{\mathbf{x}(t+\Delta t)-\mathbf{x}(t-\Delta t)}{2\Delta t} \approx \dot{\mathbf{x}}(\mathbf{t}) = \mathbf{F}(t, \mathbf{x}(t)),$$

which yields

$$\mathbf{x}(t+\Delta t) \approx \mathbf{x}(t-\Delta t) + 2\Delta t\, \mathbf{F}(t, \mathbf{x}(t)).$$

It is called the *modified midpoint method*. Two points are necessary to start the procedure:

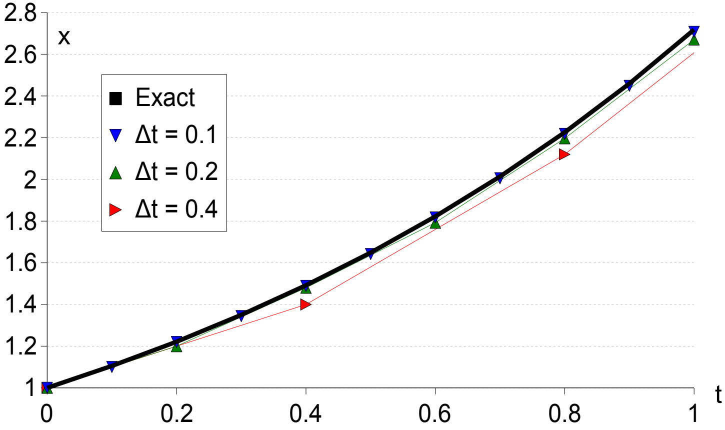
1. $\mathbf{x}(0)$ is defined by the initial conditions,
2. $\mathbf{x}(\Delta t)$ has to be computed, e.g. using the Euler formula.

- Global accuracy is $O(\Delta t^2)$.
- Two points are necessary to start.

Modified midpoint method — example

Equation: $\dot{x} = x, \quad x(0) = 1$
Exact solution: $x(t) = \mathrm{e}^{t}$

Modified midpoint method
Accuracy: $O(\Delta t^2)$



Predictor-corrector methods

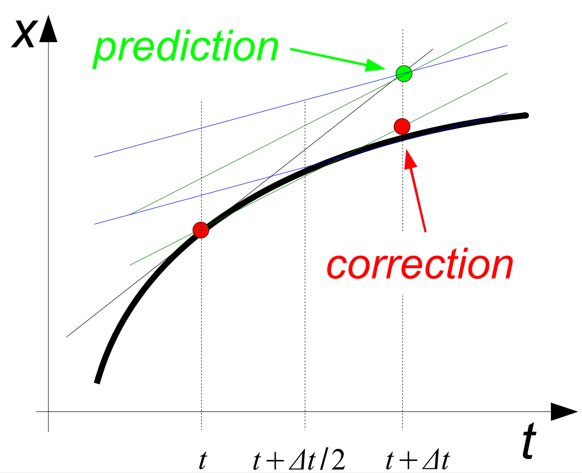
Most of the multipoint methods used in practice belong to the class of predictor-corrector methods, which basically

- compute initial approximations to $\mathbf{x}(t+\Delta t)$ and $\dot{\mathbf{x}}(t+\Delta t)$ based on the values and derivatives computed in several previous time steps (*prediction step*);
- correct the result based on the predicted derivative (*correction step*).

One of the simplest predictor-corrector methods is the Heun's method (here the "multi" of multistep is reduced to one), since it first computes the new value by the Euler method and then uses the corresponding derivative to improve the solution.

Predictor-corrector methods

Heun's method



Predictor-corrector methods — ABM-PC

The probably most known predictor-corrector method is called ABM-PC (Adams-Bashforth-Moulton Predictor-Corrector) and uses

- Adams-Bashforth method in the prediction step,
- Adams-Moulton method in the correction step.

The Lagrange interpolating polynomial is the polynomial of the lowest possible degree (degree $\leq n-1$) that passes through the n points (t_i, x_i), $i = 0, 1, \ldots, n-1$. It is explicitly given by

$$x_{\mathrm{L}}(t) = \sum_{i=0}^{n-1} x_i \prod_{\substack{j=0 \\ j \neq i}}^{n-1} \frac{t - t_j}{t_i - t_j}.$$

Predictor-corrector methods — ABM-PC

Prediction step (Adams-Bashforth method)

1. Given the derivatives in the n previously computed points $\dot{\mathbf{x}}(t - i\Delta t)$, $i = 0, 1, \ldots, n-1$, the derivative $\dot{\mathbf{x}}$ is interpolated using the Lagrange interpolating polynomial as

$$\dot{\mathbf{x}}_{\mathrm{L}_1}(t) \approx \sum_{i=0}^{n-1} \dot{\mathbf{x}}(t - i\Delta t) \prod_{\substack{j=0\\ j\neq i}}^{n-1} \frac{t - t_j}{t_i - t_j}.$$

2. The interpolating polynomial is used to predict the value of the function and its derivative in the next time step $t + \Delta t$ by

$$\mathbf{x}_{\mathrm{pred}}(t+\Delta t) \approx \mathbf{x}(t) + \int_0^{\Delta t} \dot{\mathbf{x}}_{\mathrm{L}_1}(t+\tau)\mathrm{d}\tau,$$

$$\dot{\mathbf{x}}_{\mathrm{pred}}(t+\Delta t) = \mathbf{F}\left(t+\Delta t,\, \mathbf{x}_{\mathrm{pred}}(t+\Delta t)\right).$$

Predictor-corrector methods — ABM-PC

Correction step (Bashforth-Moulton method)

1. The "time window" of n steps is shifted one step ahead to include the step $t + \Delta t$, and the Lagrange interpolating formula is again used to interpolate the derivatives in the points $\dot{\mathbf{x}}(t - i\Delta t)$, $i = -1, 0, 1, \ldots, n-2$,

$$\dot{\mathbf{x}}_{\mathrm{L}_2}(t) \approx \sum_{i=-1}^{n-2} \dot{\mathbf{x}}(t - i\Delta t) \prod_{\substack{j=-1 \\ j\neq i}}^{n-2} \frac{t - t_j}{t_i - t_j}.$$

2. Then this new interpolating polynomial is used in the same way as before to compute the (corrected) value of the function and its derivative in the next time step $t + \Delta t$,

$$\mathbf{x}(t+\Delta t) \approx \mathbf{x}(t) + \int_0^{\Delta t} \dot{\mathbf{x}}_{\mathrm{L}_2}(t+\tau)\mathrm{d}\tau,$$
$$\dot{\mathbf{x}}(t+\Delta t) = \mathbf{F}\left(t+\Delta t,\, \mathbf{x}(t+\Delta t)\right).$$

Predictor-corrector methods — ABM-PC

The most popular is probably the four-point ABM-PC method:

1. The value in the new time step is predicted as

$$\mathbf{x}_{\text{pred}}(t+\Delta t) = \mathbf{x}(t)+\Delta t\left[-\frac{9}{24}\dot{\mathbf{x}}(t-3\Delta t)+\frac{37}{24}\dot{\mathbf{x}}(t-2\Delta t)\right.$$
$$\left.-\frac{59}{24}\dot{\mathbf{x}}(t-\Delta t)+\frac{55}{24}\dot{\mathbf{x}}(t)\right]$$

   and used to compute the predicted derivative,

$$\dot{\mathbf{x}}_{\text{pred}}(t+\Delta t) = \mathbf{F}(t+\Delta t,\, \mathbf{x}_{\text{pred}}(t+\Delta t)),$$

2. which is used in the correction step to obtain

$$\mathbf{x}(t+\Delta t) \approx \mathbf{x}(t)+\Delta t\left[\frac{1}{24}\dot{\mathbf{x}}(t-2\Delta t)-\frac{5}{24}\dot{\mathbf{x}}(t-\Delta t)\right.$$
$$\left.+\frac{19}{24}\dot{\mathbf{x}}(t)+\frac{9}{24}\dot{\mathbf{x}}_{\text{pred}}(t+\Delta t)\right],$$

$$\dot{\mathbf{x}}(t+\Delta t) = \mathbf{F}(t+\Delta t,\, \mathbf{x}(t+\Delta t)).$$

Outline

1. Basics
2. Explicit one-step methods
3. Implicit one-step methods
4. Multistep methods
5. Methods for 2$^{\text{nd}}$ order equations
   - Newmark method
   - Other methods
6. Homework 5

Methods for 2$^{\text{nd}}$ order equations

- Typical methods for numerical integration of ODEs require the ODE to be 1$^{\text{st}}$ order.
- However, a typical equation of motion is of the 2$^{\text{nd}}$ order.

There is a large class of methods specialized for equations of motion

- the classical Newmark method and its variations (structural mechanics)
- symplectic methods (energy conservation in systems of classical mechanics: orbital dynamics, molecular dynamics, discrete element method, etc.)

Methods for 2$^{\text{nd}}$ order equations

Newmark method

Taylor series with Lagrange remainder:

$$\begin{aligned} \mathbf{x}(t+\Delta t) &= \mathbf{x}(t) + \Delta t\,\dot{\mathbf{x}}(t) + \frac{\Delta t^2}{2}\ddot{\mathbf{x}}(t+2\beta\Delta t), & \beta \in \left[0, \frac{1}{2}\right] \\ \dot{\mathbf{x}}(t+\Delta t) &= \dot{\mathbf{x}}(t) + \Delta t\,\ddot{\mathbf{x}}(t+\gamma\Delta t), & \gamma \in [0, 1] \end{aligned}$$

If linear interpolation of acceleration is used

$$\begin{aligned} \ddot{\mathbf{x}}(t+2\beta\Delta t) &\approx (1-2\beta)\ddot{\mathbf{x}}(t) + 2\beta\,\ddot{\mathbf{x}}(t+\Delta t) \\ \ddot{\mathbf{x}}(t+\gamma\Delta t) &\approx (1-\gamma)\ddot{\mathbf{x}}(t) + \gamma\,\ddot{\mathbf{x}}(t+\Delta t) \end{aligned}$$

So that, together with the standard equation of motion,

$$\begin{cases} \mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t\,\dot{\mathbf{x}}(t) + \dfrac{\Delta t^2}{2}\left[(1-2\beta)\ddot{\mathbf{x}}(t) + 2\beta\,\ddot{\mathbf{x}}(t+\Delta t)\right] \\ \dot{\mathbf{x}}(t+\Delta t) = \dot{\mathbf{x}}(t) + \Delta t\left[(1-\gamma)\ddot{\mathbf{x}}(t) + \gamma\,\ddot{\mathbf{x}}(t+\Delta t)\right] \\ \mathbf{f}(t+\Delta t) = \mathbf{M}\ddot{\mathbf{x}}(t+\Delta t) + \mathbf{C}\dot{\mathbf{x}}(t+\Delta t) + \mathbf{K}\mathbf{x}(t+\Delta t) \end{cases}$$

Methods for 2$^{\text{nd}}$ order equations

Newmark method

A rearrangement leads to the matrix form

$$\mathbf{A}\begin{bmatrix}\mathbf{x}(t+\Delta t)\\ \dot{\mathbf{x}}(t+\Delta t)\\ \ddot{\mathbf{x}}(t+\Delta t)\end{bmatrix} = \mathbf{B}\begin{bmatrix}\mathbf{x}(t)\\ \dot{\mathbf{x}}(t)\\ \ddot{\mathbf{x}}(t)\end{bmatrix} + \mathbf{C}\,\mathbf{f}(t+\Delta t),$$

where

$$\mathbf{A} = \begin{bmatrix}\mathbf{I} & \mathbf{0} & -\beta\Delta t^2\mathbf{I}\\ \mathbf{0} & \mathbf{I} & -\gamma\Delta t\,\mathbf{I}\\ \mathbf{K} & \mathbf{C} & \mathbf{M}\end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix}\mathbf{I} & \Delta t\,\mathbf{I} & (0.5-\beta)\Delta t^2\mathbf{I}\\ \mathbf{0} & \mathbf{I} & (1-\gamma)\Delta t\,\mathbf{I}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix} \quad \mathbf{C} = \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{I}\end{bmatrix},$$

which yields the Newmark update rule

$$\begin{bmatrix}\mathbf{x}(t+\Delta t)\\ \dot{\mathbf{x}}(t+\Delta t)\\ \ddot{\mathbf{x}}(t+\Delta t)\end{bmatrix} = \mathbf{A}^{-1}\mathbf{B}\begin{bmatrix}\mathbf{x}(t)\\ \dot{\mathbf{x}}(t)\\ \ddot{\mathbf{x}}(t)\end{bmatrix} + \mathbf{A}^{-1}\mathbf{C}\,\mathbf{f}(t+\Delta t).$$

Methods for 2$^{\text{nd}}$ order equations

Newmark method

The Newmark update rule

$$\begin{bmatrix} \mathbf{x}(t+\Delta t) \\ \dot{\mathbf{x}}(t+\Delta t) \\ \ddot{\mathbf{x}}(t+\Delta t) \end{bmatrix} = \mathbf{A}^{-1}\mathbf{B} \begin{bmatrix} \mathbf{x}(t) \\ \dot{\mathbf{x}}(t) \\ \ddot{\mathbf{x}}(t) \end{bmatrix} + \mathbf{A}^{-1}\mathbf{C}\,\mathbf{f}(t+\Delta t).$$

Unconditional stability of the Newmark scheme can be assessed by inspecting the eigenvalues of $\mathbf{A}^{-1}\mathbf{B}$ as $\Delta t \to \infty$.

- This yields the condition $\frac{1}{2} \leq \gamma \leq 2\beta \leq 1$.
- The most commonly used values are $\gamma = \frac{1}{2}$ and $\beta = \frac{1}{4}$ (energy conservation).

Methods for 2nd order equations

Other methods

There exist several variations to the Newmark method that might have better numerical properties, e.g.

- Bossak method
- Hilber-Hughes-Taylor method
- Park-Housner method
- Trujillo method
- space-time finite element method

For more information and references consult

- S. Krenk. Energy conservation in Newmark based time integration algorithms. *Comput. Methods Appl. Mech. Engrg.* 195(44–47):6110–6124, 2006.
- Cz. Bajer, *Time integration methods—still questions*. Theoretical Foundations of Civil Engineering. W. Szcześniak (ed.), vol. 1. pp. 45–54, Warsaw 2002.

Outline

1. Basics
2. Explicit one-step methods
3. Implicit one-step methods
4. Multistep methods
5. Methods for 2^{nd} order equations
6. Homework 5

Homework 5 (20 points)

Numerical integration of ODEs

Zipped HW5[2] contains:

- Simple C++ source code for numerical integration of the equation of a harmonically excited nonlinearized damped pendulum ($\ddot{\theta} + \gamma\dot{\theta} + \sin\theta = a\cos\omega t$) using the Euler method.
- A simple gnuplot[3] script for generating plots (you can also use any other plotting software).
- An example data set computed and plotted for a linear pendulum with the Euler method and **float** datatype at time steps 10^{-2} s, 10^{-3} s, 10^{-4} s and 10^{-5} s.
- The readme file.

[2] Available on `http://info.ippt.pan.pl/~ljank`

[3] `http://www.gnuplot.info/`

Homework 5 (20 points)

Numerical integration of ODEs

1. (6 points) Fill the body of the function printRK2(), which should be similar to printEuler (), but implement the 2$^{\text{nd}}$ order Runge–Kutta method instead of the Euler method.

Homework 5 (20 points)

Numerical integration of ODEs

2. (6 points total) Compute four numerical solutions in the time interval $[0, 50]$ s using the Euler method with the datatype **float** and with the time step sizes of 10^{-2} s, 10^{-3} s, 10^{-4} s and 10^{-5} s. Plot the computed solutions.
    1. (4 points) Which solution seems to be the (more or less) exact solution? Why the solutions computed at the shortest and the longest time step sizes differ so much from the other two solutions?
    2. (2 points) Recompute and replot the solutions using the same time interval, time step sizes and
        1. Euler method and **double** datatype,
        2. 2nd order Runge–Kutta method and **float** datatype,
        3. 2nd order Runge–Kutta method and **double** datatype.

       Comment on the accuracy of the results.

Homework 5 (20 points)

Numerical integration of ODEs

3. (4 points) 2^{nd} order Runge–Kutta method combined with **double** data type seems to be the most reliable from the tested methods, at least in the used time interval of $[0, 50]$ s. Check how far (how many seconds) can you more or less trust this solution at step size 10^{-2} s: …$[0, 75]$ s? …$[0, 100]$ s? …even more? How can you know it?
4. (4 points) Rework the code for the Euler method to tackle the one-variable equation $\dot{x} = -\frac{1}{x}$. Start from $x(0) = 1$ and compute numerical solutions in the interval $[0, 2]$ at different step sizes. What happens and why? What is the exact (analytical) solution?

E-mail your answers, plots and the reworked source code to `ljank@ippt.pan.pl`.