Numerical Simulation with One-Step Integration Methods

Moritz Diehl and Rien Quirynen

Overview

- Problem Statement
- Explicit Euler
- Explicit Runge-Kutta Methods
- Stiff Problems
- Implicit Euler
- Implicit Runge-Kutta Methods
- Collocation Discretization

Problem Statement

- Initial Value Problem (IVP): Regard uncontrolled ODE
$$\boxed{\dot{x} = f(t, x)}$$
with initial value $x(0) = x_0$.
- Aim is to find $x(t)$ on a time horizon of interest, for all $t \in [0, T]$.
- Numerical simulation codes are often called "integrators"

Time Grid and Notation

- Nearly all integration methods divide the time horizon into N intervals. This is called "the time grid".
- For simplicity, we assume all intervals to be of equal length $h := T/N$, with time points $t_n = nh$ for $n = 0, \dots, N$.
- The states $x(t_n)$ on the grid points will be approximated by values $x_n \approx x(t_n)$.
- For convenience, we sometimes use the following shorthand

$$f_n := f(t_n, x_n)$$

- Many integration methods exist, among them the "one-step methods" treated in this talk.

One-Step Integration Methods

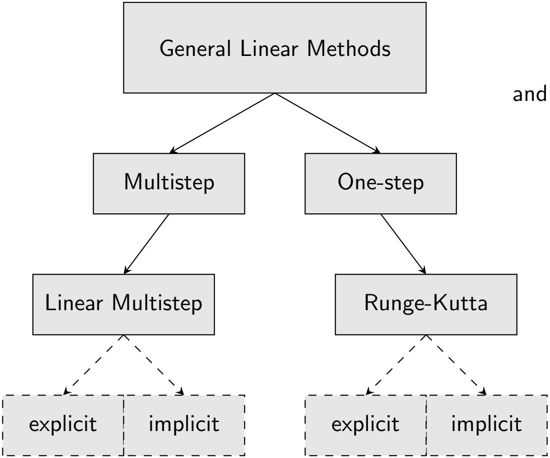
- One-step integration methods are based on a map ϕ that generates the sequence $x_0, x_1, \ldots, x_N$, by a simple recursion, starting at x_0:

$$x_{n+1} = \phi(t_n, x_n), \quad \text{for} \quad n = 0, \ldots, N-1$$

- Examples for one-step integrators are "explicit Euler", "explicit Runge-Kutta", "implicit Euler", "implicit Runge Kutta" (and, as special case of the latter, "Collocation").
- All examples above are special cases of Runge-Kutta methods, which are the focus of this talk.
- A main dividing line in the field of integration methods is between "explicit" and "implicit" methods, and also the Runge-Kutta methods can be divided along these lines.

Overview of Integration Methods

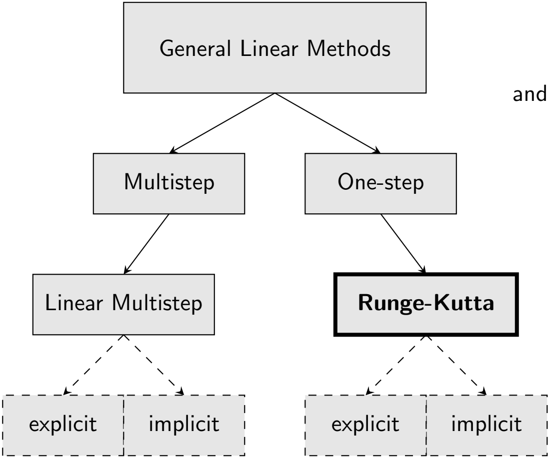
Classes of numerical methods:



and others ...

Overview of Integration Methods

Classes of numerical methods:



and others ...

A simple example for testing integrators

- As an example, we can apply our integrators to the simple scalar ODE
$$\dot{x} = \lambda x$$
with some scalar $\lambda \in \mathbb{R}$ (or, more general, in $\mathbb{C}$), and with initial value $x_0 \in \mathbb{R}$.
- The correct analytic solution is clearly given by
$$x(t) = x_0 e^{\lambda t}$$
- In particular, the last state is given by
$$x(T) = x_0 e^{\lambda T}$$

Convergence and Order

- In the following, we can compare the true solution with the approximate one obtained by the integrators
- We in particular regard the "global error" $e := \max_{n=0,\dots,N} \|x_n - x(t_n)\|$
- An integrator is called "convergent" if, for $N \to \infty$, its approximation converges to the true solution, i.e. $e \to 0$
- The speed of convergence, in terms of the step size h, is called the "order of convergence": we say the integrator is convergent of order p if $e = O(h^p)$.

The Explicit Euler Integrator

- The simplest integrator is the explicit Euler, iterating like

$$x_{n+1} = x_n + hf_n$$

- The Euler integrator is a special case of an "explicit Runge-Kutta (ERK)" method

Explicit Euler integrator applied to simple example

- Applied to $\dot{x} = \lambda x$, the explicit Euler integrator gives the recursion
$$x_{n+1} = x_n + h\lambda x_n = (1 + h\lambda)x_n$$
which has the analytic solution
$$x_n = x_0(1 + h\lambda)^n$$
- For the last state, $n = N$, using $h = \frac{T}{N}$, this gives
$$x_N = x_0 \left(1 + \frac{\lambda T}{N}\right)^N$$
- For $N \to \infty$, this converges to the true solution $x(T) = x_0 e^{\lambda T}$.
- One can show that the order of convergence is $p = 1$.

Explicit Runge-Kutta (ERK) methods

As said, explicit Euler is the simplest ERK method, and it is of order one.

$$x_n = x_{n-1} + h\, f_{n-1}$$

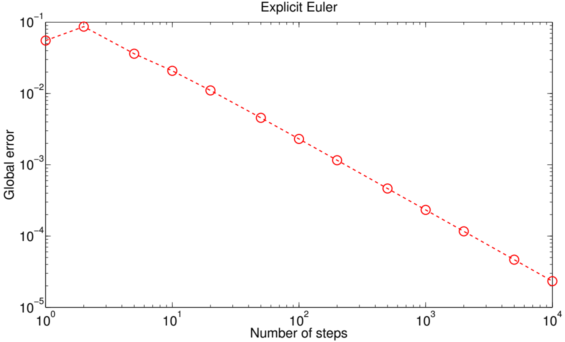
BUT: it is typically not a practical method... Why?

Explicit Runge-Kutta (ERK) methods

As said, explicit Euler is the simplest ERK method, and it is of order one.

$$x_n = x_{n-1} + h\, f_{n-1}$$

BUT: it is typically not a practical method... Why?
Higher order methods need much fewer steps for same accuracy!



Explicit Runge-Kutta (ERK) methods

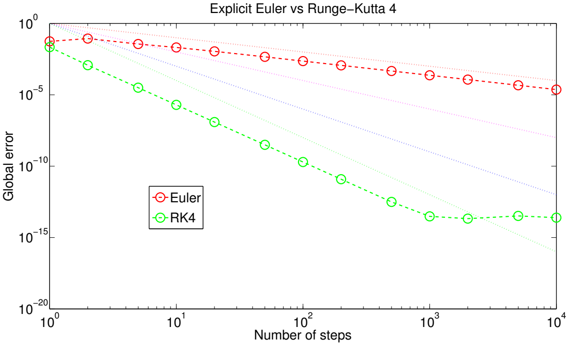
The most popular ERK method is the following 4$^{\text{th}}$ order method

$$
\begin{aligned}
k_1 &= f(t_{n-1}, x_{n-1}) \\
k_2 &= f(t_{n-1} + \frac{h}{2}, x_{n-1} + \frac{h}{2} k_1) \\
k_3 &= f(t_{n-1} + \frac{h}{2}, x_{n-1} + \frac{h}{2} k_2) \\
k_4 &= f(t_{n-1} + h, x_{n-1} + h\, k_3) \\
x_n &= x_{n-1} + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

Explicit Runge-Kutta (ERK) methods

The most popular ERK method is the following 4^{th} order method

$$\begin{aligned}
k_1 &= f(t_{n-1}, x_{n-1}) \\
k_2 &= f(t_{n-1} + \frac{h}{2}, x_{n-1} + \frac{h}{2} k_1) \\
k_3 &= f(t_{n-1} + \frac{h}{2}, x_{n-1} + \frac{h}{2} k_2) \\
k_4 &= f(t_{n-1} + h, x_{n-1} + h\, k_3) \\
x_n &= x_{n-1} + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}$$



Explicit Runge-Kutta (ERK) methods

A general s-stage ERK method

$$
\begin{aligned}
k_1 &= f(t_{n-1}, x_{n-1}) \\
k_2 &= f(t_{n-1} + c_2\, h, x_{n-1} + a_{21}\, h\, k_1) \\
k_3 &= f(t_{n-1} + c_3\, h, x_{n-1} + a_{31}\, h\, k_1 + a_{32}\, h\, k_2) \\
&\ \vdots \\
k_s &= f(t_{n-1} + c_s\, h, x_{n-1} + a_{s1}\, h\, k_1 + a_{s2}\, h\, k_2 + \ldots + a_{s,s-1}\, h\, k_{s-1}) \\
x_n &= x_{n-1} + h \sum_{i=1}^{s} b_i\, k_i
\end{aligned}
$$

NOTE: each Runge-Kutta method is defined by its so called "Butcher table", which contains all coefficients a_{ij}, b_j, c_i

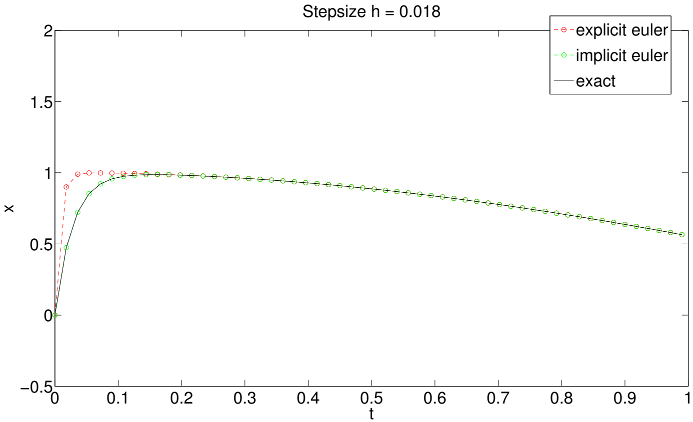
Overview

- Problem Statement
- Explicit Euler
- Explicit Runge-Kutta Methods
- **Stiff Problems**
- Implicit Euler
- Implicit Runge-Kutta Methods
- Collocation Discretization

Stiffness

Let us consider the following simple one-dimensional system

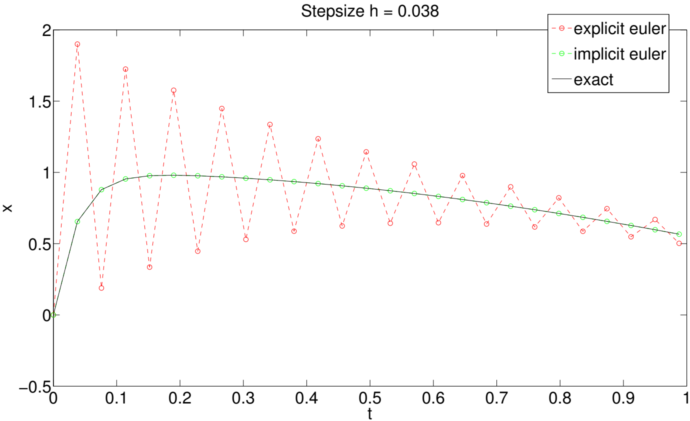
$$\dot{x}(t) = -50(x(t) - cos(t))$$



Stiffness

Let us consider the following simple one-dimensional system

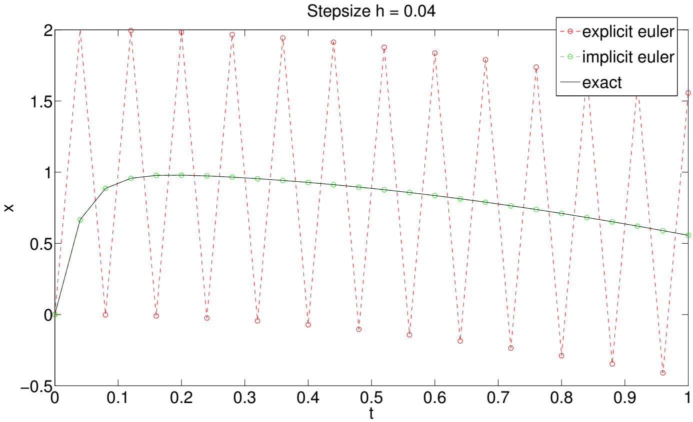
$$\dot{x}(t) = -50(x(t) - cos(t))$$



Stiffness

Let us consider the following simple one-dimensional system

$$\dot{x}(t) = -50(x(t) - cos(t))$$



Stiffness lets all known explicit integrators fail

- A stiff ODE is an ODE where some eigenvalues of the Jacobian $\frac{\partial f}{\partial x}$ are very negative so that some components of the solution decay much faster than the time scale we are interested in
- Explicit methods need excessively many time steps to converge.
- We can illustrate this with the explicit Euler integrator applied to $\dot{x} = \lambda x$ with $\lambda \ll 0$, i.e. for a very stable system. The explicit Euler gives

  $$x_n = x_0(1 + h\lambda)^n,$$

  which only converges if $|1 + h\lambda| < 1$, i.e. if $h < \frac{2}{(-\lambda)}$.
- For large $-\lambda$, we need to choose h extremely small.

The Implicit Euler Integrator

- The simplest implicit integrator is the implicit Euler, iterating like
$$x_{n+1} = x_n + hf_{n+1}$$
or, in, written in more detail:
$$x_{n+1} = x_n + hf(t_{n+1}, x_{n+1})$$
- In each step, a nonlinear equation system needs to be solved, namely the root-finding problem
$$F(x_{n+1}) = 0$$
with $F(x_{n+1}) = x_{n+1} - x_n - hf(t_{n+1}, x_{n+1})$.
- It can be solved e.g. by Newton's method. This needs initialization, matrix factorizations, etc., so each step is more expensive than for explicit methods
- Nevertheless, for stiff problems, implicit methods are cheaper than explicit ones for the usually desired levels of accuracy
- The Euler integrator is a special case of an "implicit Runge-Kutta (IRK)" method

Implicit Euler integrator applied to simple example

- Applied to $\dot{x} = \lambda x$, the implicit Euler integrator gives the recursion
$$x_{n+1} = x_n + h\lambda x_{n+1}$$
which is equivalent to $(1 - h\lambda)x_{n+1} = x_n$.
- This has the analytic solution
$$x_n = x_0 \frac{1}{(1 - h\lambda)^n}$$
- for any negative $\lambda \ll 0$ and any timestep $h > 0$, due to $|1 - h\lambda h| > 1$, this formula converges.
- thus, for stiff problems, the implicit Euler does **not** need excessively many time steps just to ensure convergence
- One can show that the order of convergence is $p = 1$ (like for the explicit Euler).

Implicit Runge-Kutta (IRK) methods

IRK as the natural generalization from ERK methods:

$$
\begin{aligned}
k_1 &= f\left(t_{n-1} + c_1\, h, x_{n-1} + h \sum_{j=1}^{s} a_{1j}\, k_j\right) \\
&\vdots \\
k_s &= f\left(t_{n-1} + c_s\, h, x_{n-1} + h \sum_{j=1}^{s} a_{sj}\, k_j\right) \\
x_n &= x_{n-1} + h \sum_{i=1}^{s} b_i\, k_i
\end{aligned}
$$

pro: nice properties (high order, stability)

Implicit Runge-Kutta (IRK) methods

IRK as the natural generalization from ERK methods:

$$\boldsymbol{k_1} = f\left(t_{n-1} + c_1\, h, x_{n-1} + h \sum_{j=1}^{s} a_{1j}\, \boldsymbol{k_j}\right)$$

$$\vdots$$

$$\boldsymbol{k_s} = f\left(t_{n-1} + c_s\, h, x_{n-1} + h \sum_{j=1}^{s} a_{sj}\, \boldsymbol{k_j}\right)$$

$$x_n = x_{n-1} + h \sum_{i=1}^{s} b_i\, k_i$$

pro: nice properties (high order, stability)

con: large nonlinear system in variables $\boldsymbol{k_1}, \ldots, \boldsymbol{k_s}$

Collocation methods

Important family of IRK methods:

- distinct c_i's (nonconfluent)
- on interval $t \in [t_{n-1}, t_n]$, approximate $x(t)$ by a polynomial $q(t)$ of degree s
- Require that the polynomial starts at x_{n-1} and that it satisfies the "collocation conditions" at the s "collocation nodes" $t_{n-1} + c_j h$, as follows:

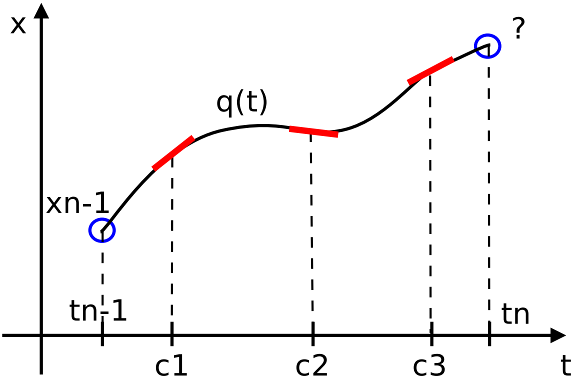
$$
\begin{aligned}
q(t_{n-1}) &= x_{n-1} \\
\dot{q}(t_{n-1} + c_1 h) &= f(t_{n-1} + c_1 h, q(t_{n-1} + c_1 h)) \\
&\vdots \\
\dot{q}(t_{n-1} + c_s h) &= f(t_{n-1} + c_s h, q(t_{n-1} + c_s h))
\end{aligned}
$$

  continuous approximation

  $\Rightarrow \quad x_n = q(t_{n-1} + h)$

NOTE: collocation is very popular in direct optimal control

Visualization of Collocation conditions



Collocation methods

How to implement a collocation method?

$$
\begin{aligned}
q(t_{n-1}) &= x_{n-1} \\
\dot{q}(t_{n-1} + c_1 h) &= f(t_{n-1} + c_1 h, q(t_{n-1} + c_1 h)) \\
&\vdots \\
\dot{q}(t_{n-1} + c_s h) &= f(t_{n-1} + c_s h, q(t_{n-1} + c_s h))
\end{aligned}
$$

Collocation methods

How to implement a collocation method?

$$
\begin{aligned}
q(t_{n-1}) &= x_{n-1} \\
\dot{q}(t_{n-1} + c_1 h) &= f(t_{n-1} + c_1 h, q(t_{n-1} + c_1 h)) \\
&\vdots \\
\dot{q}(t_{n-1} + c_s h) &= f(t_{n-1} + c_s h, q(t_{n-1} + c_s h))
\end{aligned}
$$

This is nothing else than ...

$$
\begin{aligned}
k_1 &= f(t_{n-1} + c_1\, h, x_{n-1} + h \sum_{j=1}^{s} a_{1j}\, k_j) \\
&\vdots \\
k_s &= f(t_{n-1} + c_s\, h, x_{n-1} + h \sum_{j=1}^{s} a_{sj}\, k_j) \\
x_n &= x_{n-1} + h \sum_{i=1}^{s} b_i\, k_i
\end{aligned}
$$

where the Butcher table is defined by the collocation nodes c_i.

Conclusions

- Explicit Runge-Kutta of order 4 (RK4) is an easy and efficient integrator for non-stiff problems
- For stiff problems, one should use implicit integrators, for example collocation methods (a special case of implicit Runge-Kutta)
- Other integrators for stiff systems exist, in particular the BDF methods, a special case of linear multistep methods