Adaptive numerical solvers

for Ordinary Differential Equations

Bruno Guindani
Michele Vidulis

POLITECNICO MILANO 1863

July 22, 2019

https://github.com/poliprojects/apc-project

Introduction

Ordinary differential equations (ODE)

Given $I = [t_0, t_F] \subset \mathbb{R}, \quad f(t, \boldsymbol{y}) : I \times \mathbb{R}^n \to \mathbb{R}^n, \quad f \in C^1$, and $t_0 \in I, \boldsymbol{y}_0 \in \mathbb{R}^n$:

Initial Value Problem (IVP):

find a C^1 function $\boldsymbol{y}(t) : I \to \mathbb{R}^n$ that solves

$$\begin{cases} \boldsymbol{y}'(t) = f(t, \boldsymbol{y}(t)) & \text{with } t \in I \\ \boldsymbol{y}(t_0) = \boldsymbol{y}_0 \end{cases}$$

(first order ODE)

Existence and uniqueness guaranteed under *Lipschitz continuity* of f

Runge-Kutta methods

- Family of **single-step** methods ($\boldsymbol{u}_{k+1}$ depends directly only on $\boldsymbol{u}_k$)
- Weighted average of s evaluations (**stages**) of f:

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + h \sum_{i=1}^{s} b_i \boldsymbol{K}_i \quad \text{with}$$

$$\boldsymbol{K}_i = f(t_0 + c_i h,\ \boldsymbol{u}_k + \sum_{j=1}^{s} a_{ij} \boldsymbol{K}_j)$$

Runge-Kutta methods

- Family of **single-step** methods ($\boldsymbol{u}_{k+1}$ depends directly only on $\boldsymbol{u}_k$)
- Weighted average of s evaluations (**stages**) of f:

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + h \sum_{i=1}^{s} b_i \boldsymbol{K}_i \quad \text{with}$$

$$\boldsymbol{K}_i = f(t_0 + c_i h,\ \boldsymbol{u}_k + \sum_{j=1}^{s} a_{ij} \boldsymbol{K}_j)$$

- *Butcher tableau*:

$$\begin{array}{c|ccc} c_1 & a_{11} & \dots & a_{1s} \\ \vdots & & \ddots & \\ c_s & a_{s1} & & a_{ss} \\ \hline & b_1 & \dots & b_s \end{array}$$

with $c_i = \sum_j a_{ij}$

Runge-Kutta methods

- Family of **single-step** methods ($\boldsymbol{u}_{k+1}$ depends directly only on $\boldsymbol{u}_k$)
- Weighted average of s evaluations (**stages**) of f:

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + h \sum_{i=1}^{s} b_i \boldsymbol{K}_i \quad \text{with}$$

$$\boldsymbol{K}_i = f(t_0 + c_i h,\ \boldsymbol{u}_k + \sum_{j=1}^{s} a_{ij} \boldsymbol{K}_j)$$

- *Butcher tableau*:

$$\begin{array}{c|ccc} c_1 & a_{11} & \dots & a_{1s} \\ \vdots & & \ddots & \\ c_s & a_{s1} & & a_{ss} \\ \hline & b_1 & \dots & b_s \end{array} \qquad \text{with } c_i = \sum_j a_{ij}$$

- $O(sn^2)$ if f linear
- Explicit if the upper triangular part of $[a_{ij}]_{ij}$ is null

Examples of explicit RK variants

- Forward Euler: $a = 0,\ b = 1,\ c = 0$
- RK4:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & & \frac{1}{2} & & \\ 1 & & & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

- Heun:

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

Convergence analysis for RK

- **Convergence** $\rightarrow$ absolute error: $\|\boldsymbol{y}(t_k) - \boldsymbol{u}_k\| \simeq O(h^q)$
- **Consistence** $\rightarrow$ truncation error: $\mathrm{max}_k \|\boldsymbol{\tau}_k(h)\| \simeq O(h^q)$
- Under reasonable Lipschitz continuity assumptions on ϕ, a single-step method which is consistent is also convergent

Convergence analysis for RK

- **Convergence** $\to$ absolute error: $\|\boldsymbol{y}(t_k) - \boldsymbol{u}_k\| \simeq O(h^q)$
- **Consistence** $\to$ truncation error: $\max_k \|\boldsymbol{\tau}_k(h)\| \simeq O(h^q)$
- Under reasonable Lipschitz continuity assumptions on ϕ, a single-step method which is consistent is also convergent
- Runge-Kutta is consistent iff $\sum_i b_i = 1 \implies$ **convergent**
- Steep limitations on order of convergence:
  - Maximum order is the number of stages
  - If $s \geq 5$, equality cannot be achieved in explicit variants

order	5	6	7	8
minimum s	6	7	9	11

Adaptive methods

- Step h is **updated** at every iteration adaptively, i.e. based on the trend of the solution
  - Small h near steep slopes, large h near flat points
  - A posteriori **estimate of error** is needed
  - Compare two-round solution computed with step $\frac{h}{2}$, with single-round solution computed with step h
- No need for input of "correct" step

Error computation for adaptive methods

- Relative error in infinity norm is used:

$$\frac{\|\boldsymbol{u}_{h/2} - \boldsymbol{u}_h\|_\infty}{\|\boldsymbol{u}_{k-1}\|_\infty} < \frac{\varepsilon}{2} \quad \text{(tolerance)}$$

- This guaratees consistency ($\implies$ convergence)

Error computation for adaptive methods

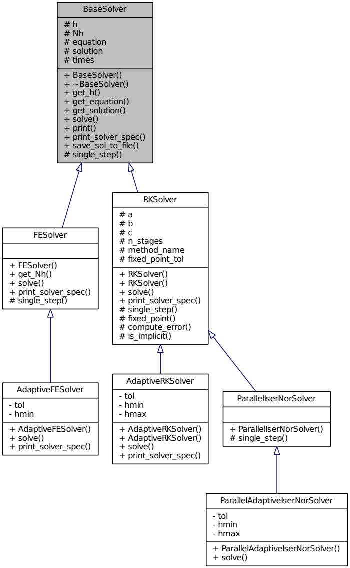
- Relative error in infinity norm is used:

$$\frac{\|\boldsymbol{u}_{h/2} - \boldsymbol{u}_h\|_\infty}{\|\boldsymbol{u}_{k-1}\|_\infty} < \frac{\varepsilon}{2} \quad \text{(tolerance)}$$

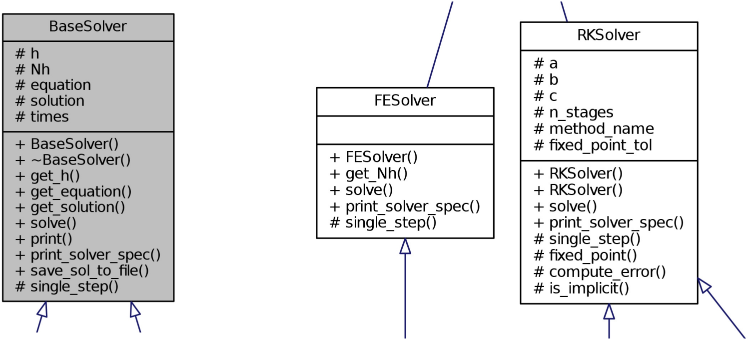
- This guaratees consistency ($\implies$ convergence)
- At each iteration h can be doubled, halved, or unchanged
- h_{min} and h_{max} are required for some methods

Code structure

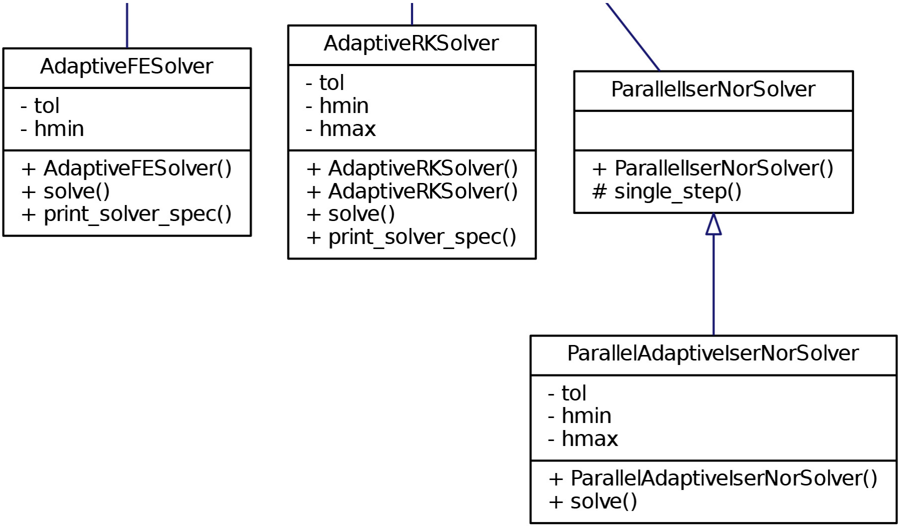
Code structure



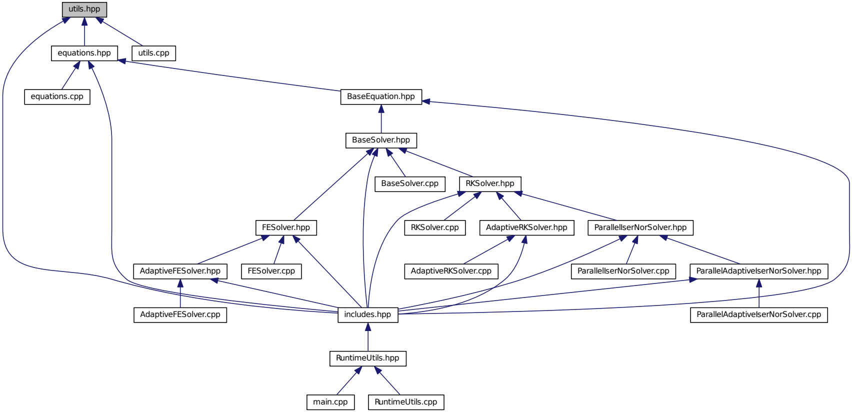
Base classes



Adaptive classes

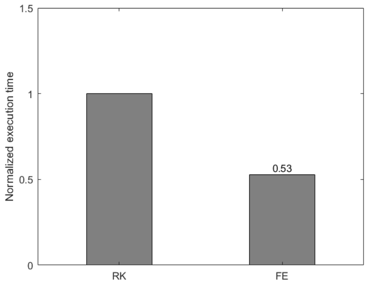


Dependencies



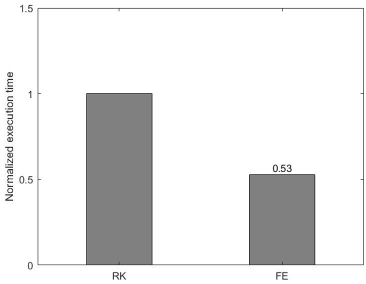
Implementation choices

- Eight test functions are provided
- Separate FE class is much more efficient than RK specialized class:



Implementation choices

- Eight test functions are provided
- Separate FE class is much more efficient than RK specialized class:



- Adaptive `single_step()` class methods are not efficient
- **Fixed point** algorithm was used for implicit methods

Parallel Iserles-Nørsett

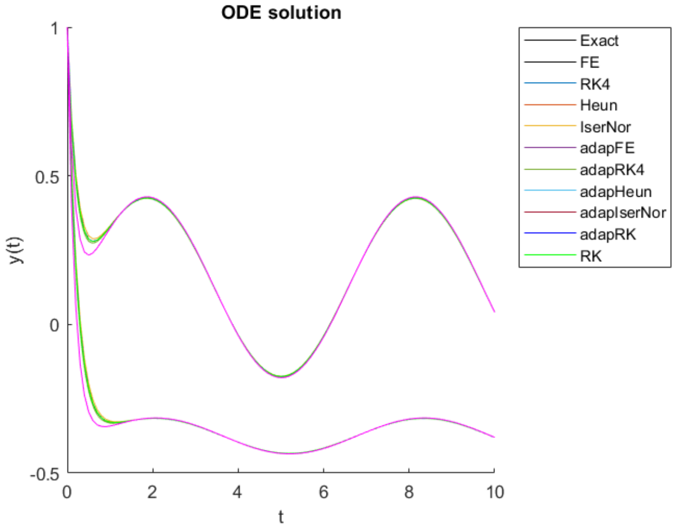
- **Implicit** method:

$$
\begin{array}{c|cccc}
\frac{1}{3} & \frac{1}{3} & & & \\
\frac{2}{3} & \frac{1}{3} & \frac{1}{3} & & \\
\frac{21+\sqrt{57}}{48} & & & \frac{21+\sqrt{57}}{48} & \\
\frac{21-\sqrt{57}}{48} & & & \frac{3-\sqrt{57}}{24} & \frac{21+\sqrt{57}}{48} \\
\hline
 & \frac{9+3\sqrt{57}}{16} & \frac{9+3\sqrt{57}}{16} & -\frac{1+3\sqrt{57}}{16} & -\frac{1+3\sqrt{57}}{16}
\end{array}
$$

- Parellelization exploits **block-diagonal** structure of Butcher array
- The method runs in parallel on **2 processors**, each dealing with one independent 2-by-2 block

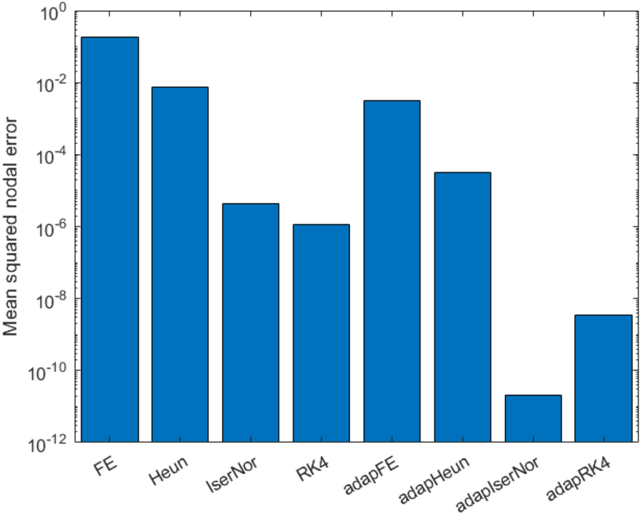
Results

Results



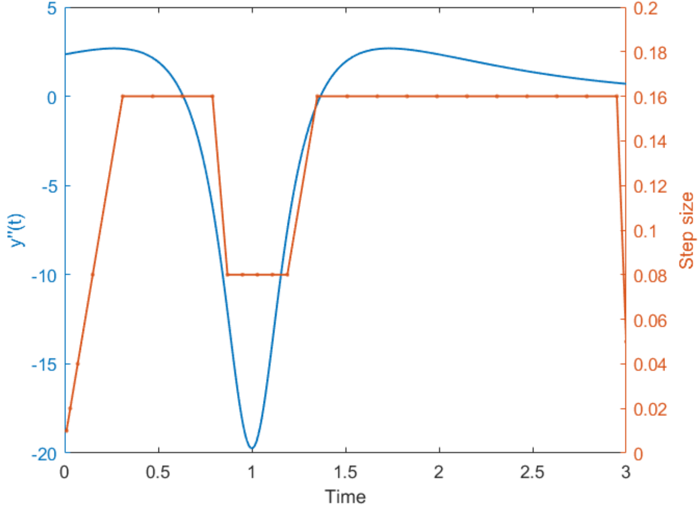
(components of test #4)

Comparison between methods



(logarithm of relative Mean Square Errors in the nodes, test #1)

Trend of step size in adaptive methods



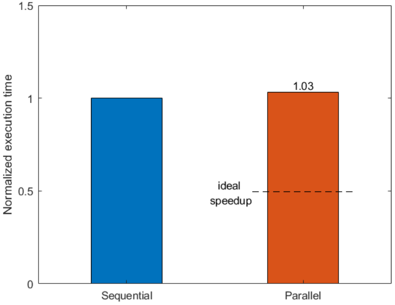
(test #7, adaptive RK4 method)

Actual efficiency of parallelism

Speedup is heavily dependent on the problem function:

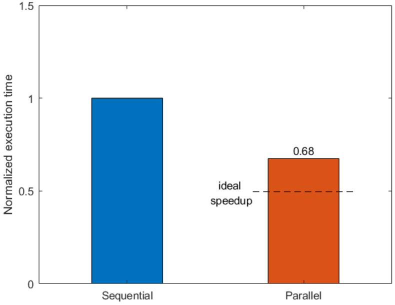
Test #5

$$f(t, y(t)) = -16y(t)$$



Test #6

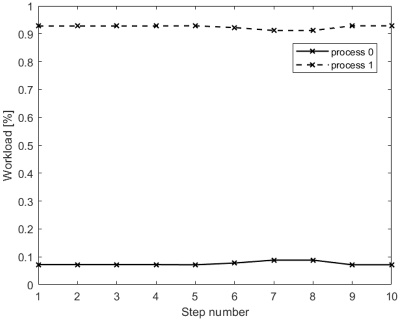
$$f(t, y(t)) = \exp_2(-\frac{y(t)}{4} + 6 + 10t)$$



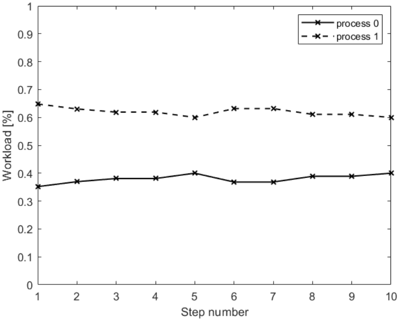
Why?

Workload distribution

Test #5



Test #6

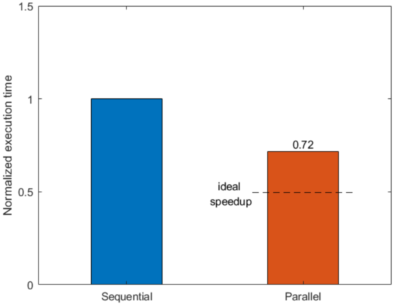


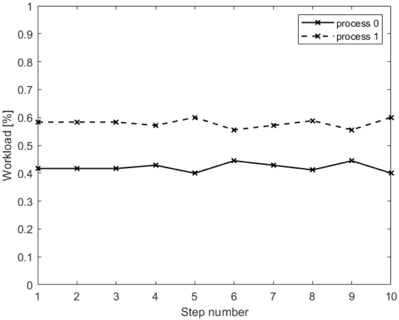
$\Longrightarrow$ **load imbalance**

A vectorial example

- Efficiency still depends on the function
- Here fixed point iterations are well-balanced among processors:

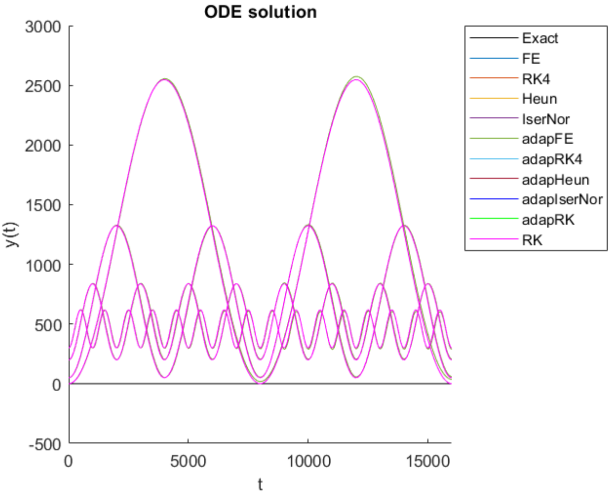
$$\text{Test \#4:} \qquad \boldsymbol{f}(t, \boldsymbol{y}(t)) = \begin{bmatrix} -3 & -1 \\ 1 & -5 \end{bmatrix} \boldsymbol{y}(t) + \begin{bmatrix} \sin(t) \\ -2t \end{bmatrix}$$



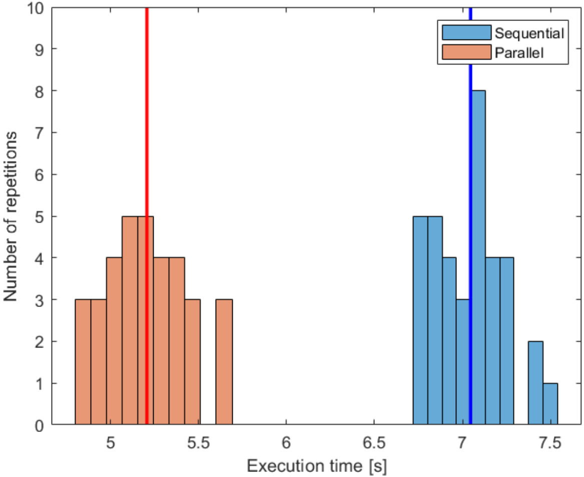


A high-dimensional example

Components of test #8 $\quad (\boldsymbol{y} \in \mathbb{R}^4)$:



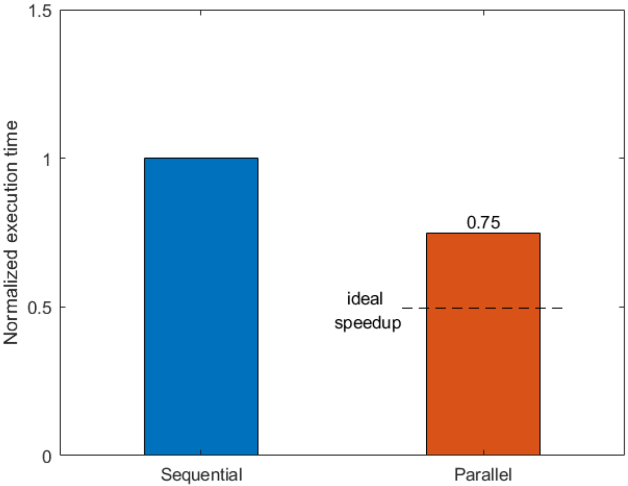
Multiple run results



Mean: $5.206\,s$, SD: $0.227\,s$

Mean: $7.045\,s$, SD: $0.200\,s$

Speedup



1.5
1
0.5
0
Normalized execution time
0.75
ideal
speedup
Sequential
Parallel

Bibliography

- Podhaisky, *Parallel two-step Runge-Kutta methods*
- Quarteroni, Saleri, *Calcolo scientifico*
- Quarteroni, Sacco, Saleri, Gervasio, *Matematica numerica*
- Solodushkin, Iumanova, *Parallel Numerical Methods for Ordinary Differential Equations: a Survey*