D.6.2.6 - State of the art methods for the parallel solution of hybrid ODE systems

sWP 6.2 - Efficient operation and simulation on multi-core platforms

WP6 - Modelling and simulation services

MODRIO (11004)

Version	2.0.1
Date	04/12/2015

Authors

Jean Brac	IFP New Energy
Mongi Ben Gaid	IFP New Energy
Thibaut-Hugues Gallois	IFP New Energy
Abir Ben Khaled	IFP New Energy
Thierry Soriano	Supmeca Toulon

Executive summary

This document is the deliverable of task 6.2.6 in the WP6 of MODRIO. The objective of the MODRIO project is to extend modeling and simulation tools based on open standards from system design to system operation. The major outcome will be a holistic modeling and simulation framework for physical systems design, diagnosis and operation assistance. This environment will be based on computing technology developed in Europe (Modelica, FMI).
One challenge is to allow the execution in real-time for system operation of multiphysics, dynamical system-level models of large scale. These models are usually described as sets of hybrid Ordinary Differential Equations (ODEs) or Differential Algebraic Equations (DAEs). Currently, building high fidelity system-level models of systems in general is a challenging duty. One problem is the prohibitive CPU times observed when such high-fidelity models are run. This is due to the fact that major system-level simulation software are currently unable to exploit multi-core platforms, because they are relying on sequential Ordinary Differential Equation (ODE) and Differential Algebraic equation (DAE) solvers.

The objective of this report is to review the different techniques that were proposed in the literature in order to allow the parallel solution of ODEs. This final (2.0) version contains updates to intermediate (1.0) version delivered on M13, including more details, especially about the waveform relaxation method as well as a proposal for the support of parallel solution of functional mockup units (FMUs). This report was a basis for technical task T6.2.8 in order to allow it to choose the most appropriate approaches with respect to its target.

Table of contents

Executive summary 2

Table of contents 3

1 Introduction 5
1.1 Homogeneous linear systems. 6
1.1.1 1D scalar equation 6
1.1.2 Scalar multi-scaled equation 6
1.1.3 Case where the matrix A is diagonalizable 7
1.2 Basic numerical solutions 7
1.2.1 Explicit, implicit Euler schemes and Crank Nicholson scheme 8
1.3 Resolution of nonlinear problems 10

2 Stability notions 11
2.1 Definitions 11
2.2 Convergency, consistency and stability of a scheme 11
2.2.1 Convergency of a numerical scheme 11
2.2.2 Consistency of a numerical scheme 12
2.2.3 Stability 12
2.2.4 Lax theorem 12
2.3 Stability according to Lyapunov 12
2.4 Stability versus initial conditions 12
2.5 Stability versus iteration convergence 13

3 One step schemes 13
3.1 ERK schemes 14
3.1.1 Schemes definition 14
3.1.2 RK schemes linear stability 16
3.2 DIRK schemes 17
3.3 Fully implicit Runge-Kutta schemes 19
3.3.1 Implicit RK3 or Radau3 scheme 20
3.3.2 Implicit RK5 or Radau5 scheme 20
3.4 Embedded Runge-Kutta schemes 22
3.5 New and simple Runge-Kutta schemes 22

4 Multistep BDF schemes 24
4.1 Illustration for BDF scheme with 2 steps 24
4.2 Stability study of BDF2 scheme 25
4.3 Stability study for BDF3 scheme 26
4.3.1 Stability of the BDF4 scheme 27
4.4 Stability study of the BDF5 scheme 28
4.5 Study of the stability of the BDF6 scheme 29

5 IMEX schemes 31
5.1 Splitting by means of the mesh 31
5.2 Splitting a model into linear and nonlinear parts. 32

6 Hybrid time 33

7 Open software **34**
7.1 DASSL . . . 34
7.2 LSODAR . . . 35
7.3 QSS . . . 36

8 Parallelization approaches for hybrid ODEs resolution **36**
8.1 Introduction . . . 36
8.2 Parallelization across the method . . . 36
8.3 Parallelization across the steps . . . 37
8.4 Parallelization across the model . . . 37
8.4.1 Waveform relaxation . . . 37
8.4.2 Transmission line modeling . . . 39
8.4.3 Modular time-integration or co-simulation . . . 39
8.5 Parallelization with several cores . . . 40
8.6 Parallelization of huge size dynamic problem . . . 41
8.7 Management of a dynamic problem . . . 41

9 Proposal for FMI extension to support parallelization **42**
9.1 Position of the problem . . . 42
9.2 Motivation through an example . . . 42
9.3 Global proposition . . . 43
9.3.1 Suggestion of method of identification . . . 43
9.3.2 New parallel fmi2GetDerivativesPart interface . . . 43

10 Recommendations **44**

References **45**

1 Introduction

This document is the deliverable of task 6.2.6 in the WP6 of MODRIO. The objective of the MODRIO project is to extend modeling and simulation tools based on open standards from system design to system operation. The major outcome will be a holistic modeling and simulation framework for physical systems design, diagnosis and operation assistance. This environment will be based on computing technology developed in Europe (Modelica, FMI).

One challenge is to allow the execution in real-time for system operation of multiphysics, dynamical system-level models of large scale. These models are usually described as sets of hybrid Ordinary Differential Equations (ODEs) or Differential Algebraic Equations (DAEs). Currently, building high fidelity system-level models of systems in general is a challenging duty. One problem is the prohibitive CPU times observed when such high-fidelity models are run. This is due to the fact that major system-level simulation software are currently unable to exploit multi-core platforms, because they are relying on sequential Ordinary Differential Equation (ODE) and Differential Algebraic equation (DAE) solvers.

The objective of this report is to review the different techniques that were proposed in the literature in order to allow the parallel solution of ODEs. This final (2.0) version contains updates to intermediate (1.0) version delivered on M13, including more details, especially about the waveform relaxation method as well as a proposal for the support of parallel solution of functional mockup units (FMUs). This report was a basis for technical task T6.2.8 in order to allow it to choose the most appropriate approaches with respect to its target.

This document is organized as follows. We begin to set the problem and explain the solution of basic linear problems currently met or currently part of more complex problems. In a second item, we recall the definitions of convergence, stability and consistency and analyze the stability under different points of view. Then, we point out the one step schemes as found in the literature; this field is huge and still remains a research topic. The fourth item is about the multi-step schemes centered on Adams and BDF (Backward Differentiation Formula) schemes. Schemes are often built with both explicit and implicit formulas together; moreover, 2 schemes are currently used to assess the local error and to drive the time step-size according to the Richardson idea. An item deals with the hybrid time regarding time event detection and model updating. A quick review of open software is then provided. The last item is about parallelization and acceleration of the simulation process. Finally, we make some recommendations.

Let us consider the following problem depending on the time in the interval I and the initial conditions X_0

$$\begin{cases} t \in I = [t_{start} = 0, t_{end}] \subset \mathbb{R} \\ X \in \mathbb{R}^n \\ \dot{X} = f(t, X) \\ X(t = t_{start}) = X_0 \end{cases} \tag{1}$$

X is any finite-size vector, $\dot{X}$ its time derivative vector and f a regular enough function. This function makes a link between the starting space $I \times \mathbb{R}^n$ and the arrival space $\mathbb{R}^n$.
$\mathbb{R}$ is the field of real numbers and $\mathbb{K}$ the field of complex numbers. We consider the Cauchy's problem on a closed range $[t_{start}, t_{end}]$ included into $\mathbb{R}$; this problem is also called Initial Value Problem (IVP) with $f(t, X(t))$ which is not a mapping from I to $\mathbb{K}^n \times \mathbb{K}^n$. f can include all the couplings and all the conceivable nonlinearities. The problem can be explicitly written as $\dot{X}(t) = A(t, X)\ X + B(t)$.
A solution is a couple $(t, X(t))$ verifying the definition time range, the initial condition and the ODE. It is also called ODE integral curve or **trajectory**.

The system is said **linear in X** if it is possible to write

$$\dot{X}(t) = A(t)\ X + B \tag{2}$$

It is **homogeneous** if B=0.
It is with **constant coefficients** if A doesn't depend on time t.

A more general formulation of this problem is possible. For instance, in the framework of DASSL solver, an implicit formulation using 3 vectors F, X and $\dot{X} \in \mathbb{R}^n$

$$F(t, X(t), X'(t)) = 0 \tag{3}$$
$$X(t=0) = X_0 \tag{4}$$

It is easy, in particular cases, to translate the implicit formulation to an explicit one. Specifically the formulation $A\dot{X} = BX$ can be explicited only if it exists an inverse matrix A^{-1} in order to get the following explicit formulation $\dot{X} = A^{-1}BX$. Obviously, in various cases, the inverse of the matrix A doesn't exist and such explicitness is impossible. The implicit formulation is more general.

Theorem 1.1. *Cauchy-Lipschitz theorem :*
Let us consider a function f lipschitzian with respect to $Y \in \mathbb{R}^n$; thus, it exists a constant L>0 such that
$\forall t \in I = [t_{start}, t_{end}] \subset \mathbb{R}, \forall X, Y \in \mathbb{R}^n, |f(t,X) - f(t,Y)| \leq L\,|X - Y|$
Then, the problem (1) has a unique solution.

The considered problem is very general. Nevertheless, with a great economy of means (f is lipschitzian), it is possible to assert the uniqueness of its solution. Therefore, this problem is well-posed.
Remark: Initial conditions are involved in the problem definition but there are no boundary conditions since it is a time evolution problem.

1.1 Homogeneous linear systems.

The solving of the Ordinary Differential Equation is searched for t in a strip of $\mathbb{R}$ and for the initial condition $X(t=0) = X_0$

$$\dot{X}(t) = A\,X \tag{5}$$

Then, the matrix A is a mapping of $I \to \mathbb{K}^n \times \mathbb{K}^n$.

1.1.1 1D scalar equation

Matrix A is only a scalar; the solution is known and unique

$$X(t) = e^{\lambda(t-t_0)} X_0 \tag{6}$$

with $\lambda = A$. If $\lambda < 0$ then $X(t) \to 0$ when $t \to 0$
If $\lambda = 0$ then X(t) remains constant and equals to the initial condition values
If $\lambda > 0$ then X(t)$\to \pm\infty$ according to the initial conditions

1.1.2 Scalar multi-scaled equation

$$\begin{cases} \dot{X}(t) &= \Lambda X(t) \\ X(t=t_0) &= X_0 \end{cases}$$

with the matrix Λ diagonal.
If the diagonal coefficients are called λ_i, the solution is

$$X(t) = \begin{pmatrix} e^{\lambda_1(t-t_0)} & 0 & \dots & \dots & 0 \\ 0 & e^{\lambda_2(t-t_0)} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & 0 & e^{\lambda_n(t-t_0)} \end{pmatrix} X_0 \tag{7}$$

1.1.3 Case where the matrix A is diagonalizable

$$\begin{cases} \dot{X}(t) &= AX(t) \\ X(t=0) &= X_0 \end{cases} \tag{8}$$

A is diagonalized in the eigenvectors basis and it becomes

$$A = P\Lambda P^{-1} \tag{9}$$

with $\Lambda = \begin{pmatrix} \lambda_1 & 0 & ... & ... & 0 \\ 0 & \lambda_2 & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & ... & ... & 0 & \lambda_n \end{pmatrix}$

Then, the solution is

$$X(t) = P \begin{pmatrix} e^{\lambda_1(t-t_0)} & 0 & ... & ... & 0 \\ 0 & e^{\lambda_2(t-t_0)} & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & ... & ... & 0 & e^{\lambda_n(t-t_0)} \end{pmatrix} P^{-1} X_0 \tag{10}$$

Let A a matrix:

$$\dot{X}(t) = AX + B \tag{11}$$

A and B are assumed to be not depending on time. Assuming that A is invertible, (11) becomes

$$\dot{X}(t) = A(X + A^{-1}B) \tag{12}$$

We notice that (A^{-1}B) is a vector as X is. Since A and B does not depend on the time t and consequently does not change the time derivation

$$\dot{(X(t) + A^{-1}B)} = A(X + A^{-1}B) \tag{13}$$

Setting $Y = X + A^{-1}B$, the homogeneous system becomes

$$\dot{Y}(t) = AY \tag{14}$$

Let $Y(t)$ the solution on the time range and let the initial conditions of the problem. To get back to X, it is only necessary to do $X = Y - A^{-1}B$.

The problem to solve concerns complex systems of large size, non-linear, strongly coupled and defined as mentioned.

1.2 Basic numerical solutions

We look at the simplest schemes:
Let us consider the following ODE $\dfrac{\partial X}{\partial t} = f(X)$ with $X(t=0) = X_0$.
To integrate this equation, we can compute exactly the following integral

$$X(t+dt) = X(t) + \int_t^{t+dt} f(X(\tau), \tau)\, d\tau \tag{15}$$

The purpose is to approximate this integral by a quadrature formula of more or less high order.

1.2.1 Explicit, implicit Euler schemes and Crank Nicholson scheme

The simplest quadrature is obtained by means of a truncated Taylor expansion at the 1^{st} order. It provides the **Euler explicit scheme** (1768)

$$X_{n+1} = X_n + dt f_n \tag{16}$$

Another quadrature provides the **Euler implicit scheme** also of the 1^{est} order

$$X_{n+1} = X_n + dt\ f_{n+1} \tag{17}$$

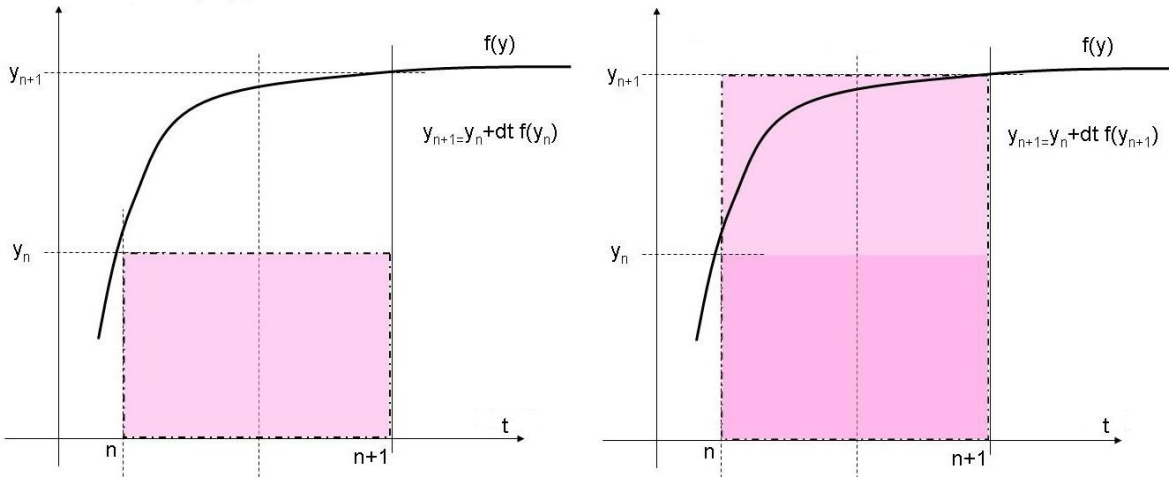

Figure 1: Illustration of the integration method by means of the Euler scheme of the 1^{st} order explicit on the left, implicit on the right.

The Runge's scheme (1895) at order 2 is written as

$$X_{n+1} = X_n + dt\ f(X_n + \frac{dt}{2}\ f_n) \tag{18}$$

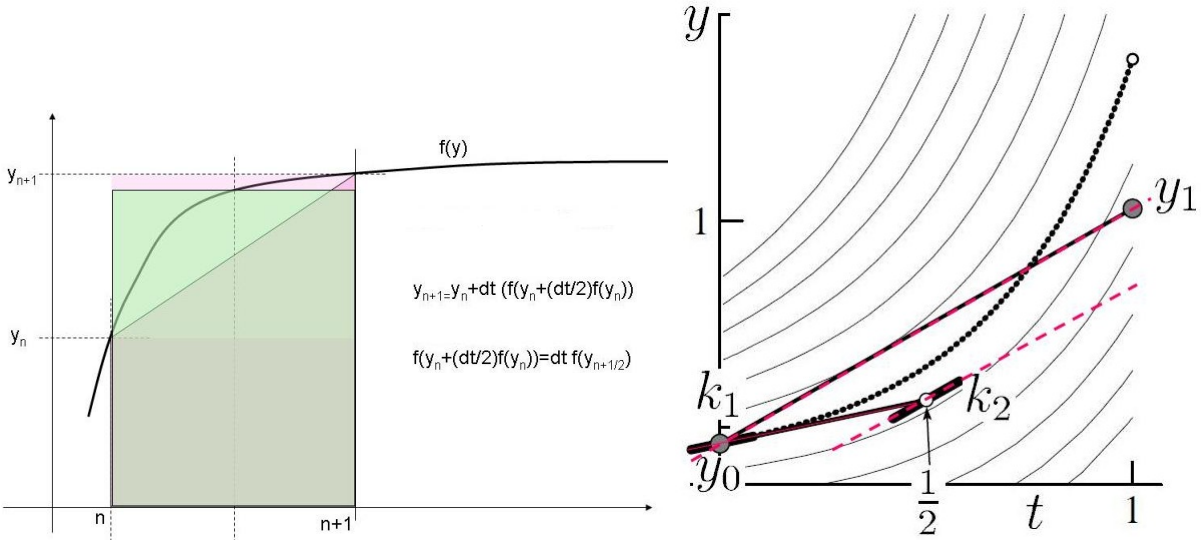

Figure 2: Illustration of the integration method of the modified Euler scheme of order 2.

It is possible to decompose this computation in two steps:

- first, a computation of the predictor with an Euler scheme of order 1,
- then a corrector with the trapezoidal rule.

It is an explicit scheme. It is also called Runge-Kutta scheme of order 2 with 2 steps (s=2).

$$
\begin{aligned}
k_1 &= f(X_n) && \text{initial value} \\
k_2 &= f(X_n + \frac{dt}{2}\, k_1) && \text{prediction Euler} \\
X_{n+1} &= X_n + dt\; k_2 && \text{correction Trapezoidal rule}
\end{aligned}
$$

We match the following tables for the explicit Euler scheme on the left and the Runge scheme on the right.

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

The trapezoidal rule is obtained by averaging the explicit/implicit schemes. This scheme is also called Crank Nicholson scheme.

$$X_{n+1} = X_n + \frac{dt}{2}\left(f(t_n, X_n) + f(t_{n+1}, X_{n+1})\right) \tag{19}$$

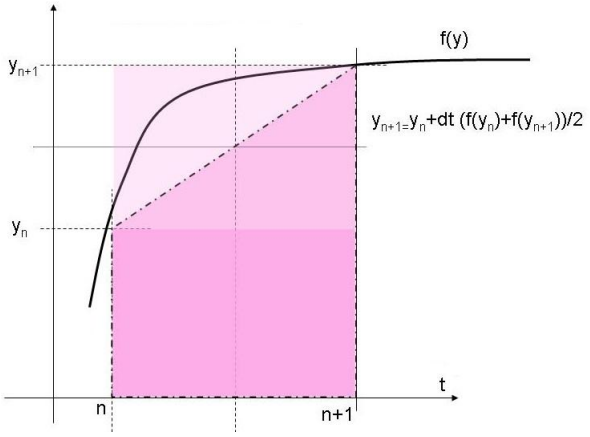

Figure 3: Illustration of the integration method of the Crank Nicholson/trapezoidal scheme of order 2.

Moreover, it can be proved this scheme is of order 2. First, we write two Taylor series of order 2

$$
\begin{aligned}
X_{n+1} &= X_n + dt f_n + \frac{dt^2}{2}\frac{\partial^2}{\partial t^2} X_n \\
X_n &= X_{n+1} - dt f_{n+1} + \frac{dt^2}{2}\frac{\partial^2}{\partial t^2} X_{n+1}
\end{aligned}
$$

We can rewrite these easily

$$X_{n+1} = X_n + dt f_n + \frac{dt^2}{2}\frac{\partial}{\partial t} f_n$$
$$X_n = X_{n+1} - dt f_{n+1} + \frac{dt^2}{2}\frac{\partial}{\partial t} f_{n+1}$$

At the order 2, we have $\frac{\partial^2}{\partial t^2} X_n = \frac{\partial^2}{\partial t^2} X_{n+1}$. Then, we get the scheme by subtracting the two Taylor series.

1.3 Resolution of nonlinear problems

Commonly, the problem is transformed into an iterating process. It is easy enough to manage this process; from an initial Y^0, we repeat the calculation of $Y^{i+1} = F(Y^i)$ until the difference $Y^{i+1} - Y^i < \varepsilon$. This process converges so as the application condition of the fix point theorem is applicable, that is to say, the iterative process F is lipschitzian. The method is well fitted for parallelization.

An elegant improvement of the fix point method is known as Steffensen [26] acceleration based on the Δ^2 Aitken method. Let us consider that the limit of the Y^i expansion be ξ

$$lim_{i\to\infty} Y^i = \xi$$

First of all, since the expansion converges, it comes 2 equations with 2 unknowns ξ and k ($k < 1$ since the expansion is assumed to be convergent)

$$Y^{i+1} - \xi = k\,(Y^i - \xi)$$
$$Y^i - \xi = k\,(Y^{i-1} - \xi)$$

The solution of this nonlinear system is

$$k = \frac{Y^i - Y^{i+1}}{Y^{i-1} - Y^i}$$
$$\xi = \frac{Y^{i+1}Y^{i-1} - (Y^i)^2}{Y^{i+1} - 2Y^i + Y^{i-1}}$$

The differences of first and second ranks are then formed

$$\Delta Y^i = Y^{i+1} - Y^i$$
$$\Delta^{(2)} Y^i = \Delta Y^{i+1} - \Delta Y^i$$

Therefore, the next expansion is built

$$Z^i = Y^i - \frac{(\Delta Y^i)^2}{\Delta^{(2)} Y^i}$$

The interesting result is that:

- iterative process F of order=1, then the new process $Z = G(Z)$ is a process of order ≥ 2,
- process F of order p>1, then G is a process of order 2p-1.

This Steffensen enhancement of the convergence could be fruitfully used for increasing the convergence speed of the waveform relaxation process (see Section 8.4.1).

2 Stability notions

2.1 Definitions

Let us consider u as the exact solution of the following problem

$$\begin{cases} \dot{u} = f(t,u) \\ u(t_n) = y_n \end{cases}$$

Definition 2.1.1. *The local error is defined from t_n by*

$$le_n = u(t_n + h_n) - y_{n+1} \tag{20}$$

Definition 2.1.2. *The global error done by a numerical scheme at the instant t_n is*

$$E_n = y_n - y(t_n) \tag{21}$$

Definition 2.1.3. *The truncation error (sometimes also called consistency error) for a given scheme is defined by*

$$\epsilon_n = y(t_{n+1}) - y(t_n) - h_n\phi(y(t_n), h_n, t_n, f) \tag{22}$$

Definition 2.1.4. *[2] A numerical scheme is of order p when p is the greatest integer verifying for all function f of (1)*

$$\epsilon_n = \boldsymbol{O}(h_n^{p+1}) \tag{23}$$

Then, it comes the following proposition.

Proposition 2.1. *For a scheme of order p, it comes*

$$le_n = \boldsymbol{O}(h_n^{p+1}) \tag{24}$$

Proof. Applying the local error definition to a scheme, it becomes

$$le_n = u(t_n + h_n) - y_{n+1}$$
$$le_n = u(t_n + h_n) - y_n - h_n\phi(y(t_n), h_n, t_n, f)$$

According to the local error definition $y_n = u(t_n)$, we have

$$le_n = u(t_n + h_n) - u_n - h_n\phi(u(t_n), h_n, t_n, f)$$

The scheme is of order p $u(t_n + h_n) - u_n - h_n\phi(u(t_n), h_n, t_n, f) = \mathbf{O}(h^{p+1})$ and then

$$le_n = \mathbf{O}(h^{p+1}) \tag{25}$$

□

2.2 Convergency, consistency and stability of a scheme

2.2.1 Convergency of a numerical scheme

Definition 2.2.1. *Setting $H = \max_n h_n$, a scheme is convergent for the problem (1) if*

$$\lim_{H\to 0} \max_{0\leq n\leq M} ||E_n|| = 0 \tag{26}$$

Moreover, a scheme is said convergent order p if

$$E_n = \boldsymbol{O}(H^p) \tag{27}$$

2.2.2 Consistency of a numerical scheme

Definition 2.2.2. *[6] A numerical scheme is said consistent for the problem (1) if*

$$\lim_{h\to 0} \sum_{n=0}^{n=M} |\epsilon_n| = 0 \tag{28}$$

2.2.3 Stability

Definition 2.2.3. *[2] A numerical scheme is said stable if it exists a constant C not depending on H such as u_n et v_n defined by*

$$u_{n+1} = u_n + h_n \phi(u_n, h_n, t_n, f) \tag{29}$$
$$v_{n+1} = v_n + h_n \phi(v_n, h_n, t_n, f) + \epsilon_n \tag{30}$$

verifies

$$\max_{0\leq n\leq M} ||v_n - u_n|| \leq C(||v_0 - u_0|| + \sum_{n<M} ||\epsilon_n||) \tag{31}$$

2.2.4 Lax theorem

Theorem 2.2. *[6] If a one-step scheme is consistent with the problem 1 and stable then it is convergent.*

Proof. Taking into account the stability definition (2.2.3), we set $z_j = y(t_j)$. Then, the truncation error is $\epsilon_j = |z_{j+1} - z_j - h_j f(t_j, z_j h_j)|$. A vector serie ϵ_j is defined with values in $\mathbb{R}^d$.

By hypothesis, the scheme is stable

$$\exists C \geq 0, \max_{0\leq n\leq M} ||y(t_n) - y_n|| \leq C(||y(t_0) - y_0|| + \sum_{n<M} ||\epsilon_n||)$$

Since the scheme is consistent, it becomes $\lim_{h\to 0} \sum_{n=0}^{n=M} |\epsilon_n| = 0$ and moreover $y_0 = y(t_0)$, then

$$\lim_{h\to 0} \max_{0\leq n\leq M} ||y(t_n) - y_n|| = 0$$

Therefore, the scheme convergence is proved.

□

2.3 Stability according to Lyapunov

A system defined on Ω is stable according to Lyapunov if for any test function f defined on Ω satisfying $f(x,t) > 0$ for all $x \in \Omega$, and $\partial_t f \leqslant 0$; then f is stable according to Lyapunov.
If $\partial_t f < 0$, then, if f is asymptotically stable.

2.4 Stability versus initial conditions

It is the meaning of the stability given by Lyapunov. Let us consider a solution with given initial conditions, if we apply a disruption of a magnitude ε to the initial conditions, then the solution of the

disrupted problem must not be too different from the solution of the non-disrupted problem.
Let us consider the Euler scheme

$$\begin{cases} X_1 = & X_0 + dt\ \lambda X_0 = (1+\lambda dt)\ X_0 \\ X_2 = & X_1 + dt\ \lambda X_1 = (1+\lambda dt)^2\ X_0 \\ & \dots \\ X_{n+1} = & (1+\lambda dt)^n\ X_0 \end{cases} \tag{32}$$

Let us consider a disruption of the initial conditions $\tilde{X}_0 + \delta$, which means $X_{n+1} = (1+\lambda dt)^n\ (X_0 + \delta)$. After n iterations, the error is so $\varepsilon = (1+\lambda dt)^n\ \delta$.
So that the disrupted solution would not be infinitely distant from the non-disrupted solution, we have to $|1+\lambda dt| < 1$. In other terms $-2 < real(\lambda) dt \leq 0$. If this condition is satisfied, the Euler scheme is stable.

2.5 Stability versus iteration convergence

By applying the Dahquist test equation $\dot{y} = \lambda y$ and in taking $y_n = \xi_n$, we get, by gathering the terms, the condition $\zeta(\xi) = \rho(\xi) - h\lambda\sigma(\xi) = 0$. We define therefore $\zeta(\xi)$ as the stability polynomial. We mark the absolute stability region as the region corresponding to the values of λh where $|y_n|$ does not increase with n. So that these conditions are satisfied, all the roots ξ_i of ζ=0 must satisfy $|\xi_i| < 1$.

To study the stability, we use the Dahlquist test equation with $\lambda \in C$ complex and $t \in [t_{start}, t_{fin}] \subset \mathbb{R}$

$$\frac{\partial}{\partial t} X = \lambda X = f(X)$$
$$X(t=0) = X_0$$

We know the analytic solution $X(t) = X_0\ e^{\lambda\, t}$.
For the following, we take $z = \lambda\ dt$.
If $real(\lambda) > 0$, when $t \to \infty$,$X \to \infty$;
On the other hand, if $real(\lambda) < 0$, when $t \to \infty$, $X \to 0$.
It seems so necessary that $real(\lambda) < 0$ so that the system is stable.
The negative half plane is then the stability region for the Dahlquist test.

Definition 2.5.1. *Let us the numerical scheme y_n compared to the Dahlquist test equation and depending on $z = \lambda dt$. The set of values $S := \{z \in \mathbb{K}\ \ \{y_{n>0}\} bound\}$ is called the stability region of the numerical scheme. This scheme is* **A-stable** *if $S \supset \mathbb{K}^-$ with $\mathbb{K}^- = \{z \in \mathbb{K}; Real(z) \leq 0\ \}$*

The Dalhquist barrier says that it doesn't exist A-stable multistep methods of order greater than 2.

Definition 2.5.2. *[17] A method is said* **L-stable** *if it is A-stable and moreover, the magnitude factor $\alpha = \frac{X_{n+1}}{X_n} \to 0$ when $\Re z \to \pm\infty$.*

3 One step schemes

The one step schemes are typically Runge-Kutta schemes (German mathematicians Carl Runge, Martin Wilhelm Kutta - 1901). They are numerous and organized in a family of schemes.

Definition of the explicit Runge Kutta schemes:
The method to find the explicit Runge-Kutta schemes with s intermediate stages is written as

$$\begin{aligned} k_1 &= f(X_n) \\ k_2 &= f(X_n + h a_{21} k_1) \\ &\dots \\ k_s &= f(X_n + h \sum_{i=1}^{s-1} a_{si} k_i) \\ X_{n+1} &= X_n + h \sum_{i=1}^{s} b_i k_i \end{aligned}$$

The coefficients are given by a Butcher table [11] (a_{ij}, b_i, c_i), i=1,s, j=1,s

c_1			
c_2	$a_{2\,1}$		
c_3	$a_{3\,1}$	$a_{3\,2}$	
...			
c_s	$a_{s\,1}$	$a_{s\,2}$	$a_{s\,3}...a_{s\,s-1}$
	b_1	b_2	$b_3...b_{s-1}$ b_s

In case of Explicit Runge Kutta schemes (ERK), the diagonal coefficients are $(a_{ii} = 0)$ and the upper diagonal coefficients are $(a_{ij} = 0, \;\; j > i)$.
The Diagonally Implicit Runge Kutta DIRK schemes have equal diagonal coefficients $a_{ii} \neq 0$.
The fully implicit Runge Kutta schemes have coefficients different from zero upper the diagonal.
The c_i coefficients deals with the time intermediate part of the time step and the a_{ij} provide the spatial intermediate contributions at the i^{th} stage. The b_i enable the computation of the final stage. These coefficients determine all the properties of the schemes: accuracy, stability and efficiency.
These coefficients must verify some connections:

- $c_1 = 0$ and $c_i = \sum_{j=1}^{i-1} a_{ij}$ for i=2,s.
- $\sum_{i=1}^{s} b_i = 1$.

3.1 ERK schemes

We only recall one classical Butcher table for each order. Theses tables are not unique because the connection system to determine the Butcher coefficients is underdetermined.

3.1.1 Schemes definition

- **Case RK1**

$$X_{n+1} = X_n + dt.X_n \tag{33}$$

- **Butcher table for RK2 (s=2)**

0		k_1	$= f(X_n)$
1/2	1/2	k_2	$= f(X_n + \frac{dt}{2} k_1)$
	0 1	X_{n+1}	$= X_n + dt \;\; k_2$

- **Butcher table and then intermediate stages for RK3 (s=3)**

0				k_1	$= f(X_n)$
$1/3$	$1/3$			k_2	$= f(X_n + \frac{dt}{3}\, k_1)$
$2/3$	0	$2/3$		k_3	$= f(X_n + \frac{2\,dt}{3}\, k_2)$
	$1/4$	0	$3/4$	X_{n+1}	$= X_n + \frac{dt}{4}(k_1 + 3k_3)$

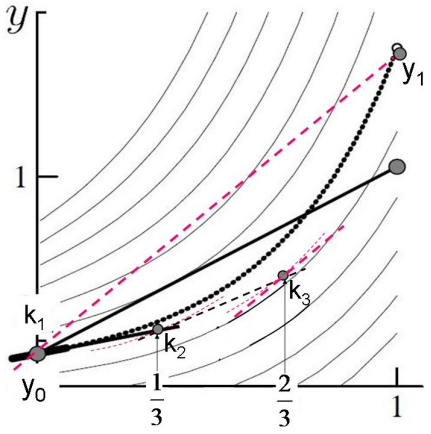

Figure 4: Illustration of the integration RK3 method. The end of the segment corresponding to order 2 (thick right line) has not been removed to point out the accuracy enhancement due to the order increase.

- **Butcher table for RK4 (s=4)**

0					k_1	$= f(X_n)$
$1/2$	$1/2$				k_2	$= f(X_n + \frac{1}{2}k_1 dt)$
$1/2$	0	$1/2$			k_3	$= f(X_n + \frac{1}{2}k_2 dt)$
1	0	0	1		k_4	$= f(X_n + k_3 dt)$
	$1/6$	$2/6$	$2/6$	$1/6$	X_{n+1}	$= X_n + \frac{1}{6}dt(k_1 + 2k_2 + 2k_3 + k_4)$

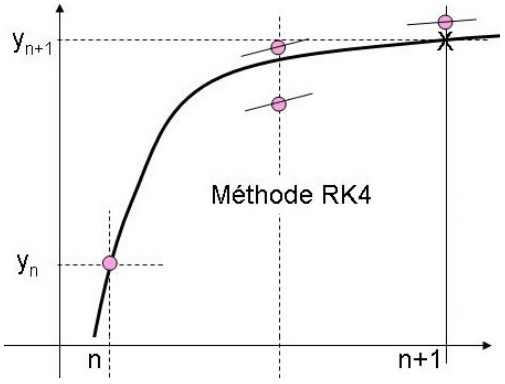

Figure 5: Illustration of the integration RK4 method with 4 points inside the step. The intermediate points of derivative calculus are indicated: one at the beginning, 2 inside the step and the last one at the right end of the step.

3.1.2 RK schemes linear stability

The linear stability can be studied by means of the stability polynomial $S(z) = \frac{X_{n+1}}{X_n}$, easily obtained in case of one step scheme in putting the Dalhquist model in the RK scheme with $z = \lambda dt$. Depending on the order of the RK scheme (indicated between brackets in upper case), here are the stability polynomial

$$S^{(1)}(z) = 1 - z \tag{34}$$

$$S^{(2)}(z) = 1 - z + \frac{1}{2}z^2 \tag{35}$$

$$S^{(3)}(z) = 1 - z + \frac{1}{2}z^2 - \frac{1}{6}z^3 \tag{36}$$

$$S^{(4)}(z) = 1 - z + \frac{1}{2}z^2 - \frac{1}{6}z^3 + \frac{1}{24}z^4 \tag{37}$$

$$S^{(5)}(z) = 1 - z + \frac{1}{2}z^2 - \frac{1}{6}z^3 + \frac{1}{24}z^4 + \frac{1}{120}z^5 \tag{38}$$

$$S^{(6)}(z) = 1 - z + \frac{1}{2}z^2 - \frac{1}{6}z^3 + \frac{1}{24}z^4 + \frac{1}{120}z^5 + \frac{1}{720}z^6 \tag{39}$$

It is important to notice that the stability polynomial does not depend on the Butcher table which is not unique for a given order. These polynomials are the truncated Taylor expansion of $e^{-\lambda dt}$. It is the exact solution of the Dalhquist problem for the considered order. The stability function is a polynomial and consequently it cannot be A-stable. The stability area is inside the closed curve given by the stability polynomial. The curves have been computed and plotted by means of Maple Software.

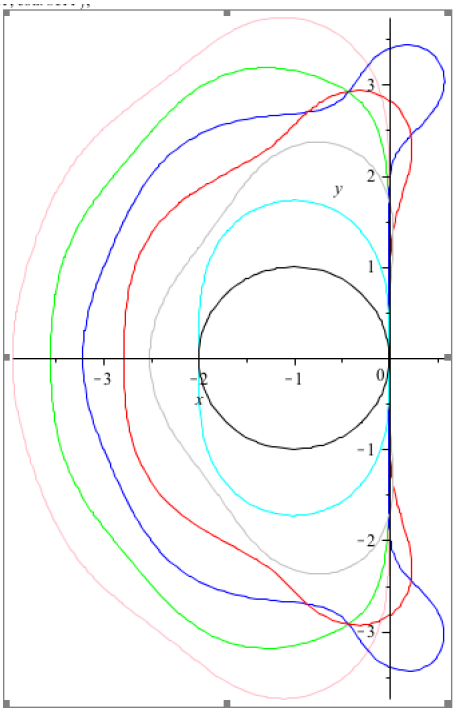

Figure 6: The stability region of order ranging from 1 to 6 is inside the closed curve defined as the points where $|S(z)| \leq 1$. Moreover, for the orders 3 and 4, 2 small zones where $real(z) \leq 0$ have to be excluded.

3.2 DIRK schemes

The Diagonal Implicit Runge Kutta schemes are commonly used when integrating stiff linear and nonlinear ODEs. Their study began originally by [8] [10]. They are a well-balanced tradeoff between accuracy and computational effort. Let begin to give the more famous DIRK schemes. These schemes are also called Singly Diagonally Implicit Runge Kutta SDIRK.
Using the Butcher's notation, the s stages SDIRK scheme is developed as

$$\begin{cases} k_i = X_n + dt \sum_{j=1}^{i} a_{ij} f(k_i) \\ X_{n+1} = X_n + dt \sum_{j=1}^{s} b_i f(k_i) \end{cases}$$

Diagonal implicit coefficient shape makes easier the computation of the stage. At each stage number i, assessment of only $f(k_i)$ have to be achieved, the values $f(k_j)$ with j<i being inherited from the previous stages. A fixed point or a Newton method can be suitable to solve the implicit equation of a given stage.
Let q be the number of stages and p the order of the scheme called (q,p) scheme.

- A DIRK (1,2) scheme is well known as the implicit mid-point scheme:

  $$\begin{array}{c|c|l} \frac{1}{2} & \frac{1}{2} & k_1 = X_n + \frac{dt}{2} f(k_1) \\ \hline & 1 & X_{n+1} = X_n + dt\ f(k_1) \end{array}$$

  The stability polynomial $R^{(2)}(z)$ is written as

  $$\begin{cases} R^{(2)}(z) = 1 + \dfrac{z}{1 - \frac{z}{2}} \end{cases}$$

- A DIRK (2,2) scheme depending on the parameter α can be developed as

  $$\begin{array}{c|cc|l} \alpha & \alpha & & k_1 = X_n + dt\ \alpha f(k_1) \\ 1-\alpha & 1-2\alpha & \alpha & k_2 = X_n + dt\ ((1-2\alpha\ f(k_1) + \alpha\ f(k_2)) \\ \hline & \frac{1}{2} & \frac{1}{2} & X_{n+1} = X_n + \frac{dt}{2}(f(k_1) + f(k_2)) \end{array}$$

  The stability polynomial $R^{(2)}(z)$ is written as

  $$\begin{cases} k_1 = \dfrac{1}{1-\alpha\ z} \\ k_2 = \dfrac{1 + z(1-3\alpha)}{1 - 2\alpha z + \alpha^2\ z^2} \\ R^{(2)}(z) = 1 + \dfrac{z}{1-\alpha\ z}\left(1 + \dfrac{1 + (1-3\alpha)z}{1-\alpha\ z}\right) \end{cases}$$

  Then, we look for the limit of $R^{(2)}(z)$ when $z \to \infty$ in order to know whether the DIRK2 scheme is L-stable or not.

  $$\lim_{z\to\infty} R^{(2)}(z) = \frac{1 - 4\alpha + 2\alpha^2}{2\alpha^2} \tag{40}$$

  This equation ($\lim = 0$) admits two roots $\alpha = 1 \pm \frac{1}{\sqrt{2}}$ and the root $\alpha = 1 + \frac{1}{\sqrt{2}}$ corresponds to the smallest area and an error of $1.37dt^3$ while the truncation error for $\alpha = 1 - \frac{1}{\sqrt{2}}$ is only $-0.04dt^3$. Therefore, we will use $\alpha = 1 - \frac{1}{\sqrt{2}}$ to get better accuracy.

  Moreover, it is possible to choose α such that the scheme is of order 3: $\alpha = \frac{1}{2} \pm \frac{1}{2\sqrt{3}}$. We will use $\alpha = \frac{1}{2} + \frac{1}{2\sqrt{3}}$ since the scheme is then A-stable and becomes a DIRK3 scheme; unfortunately, this scheme is not yet L-stable. (For the other root (pink curve), the scheme is not

A-stable.)

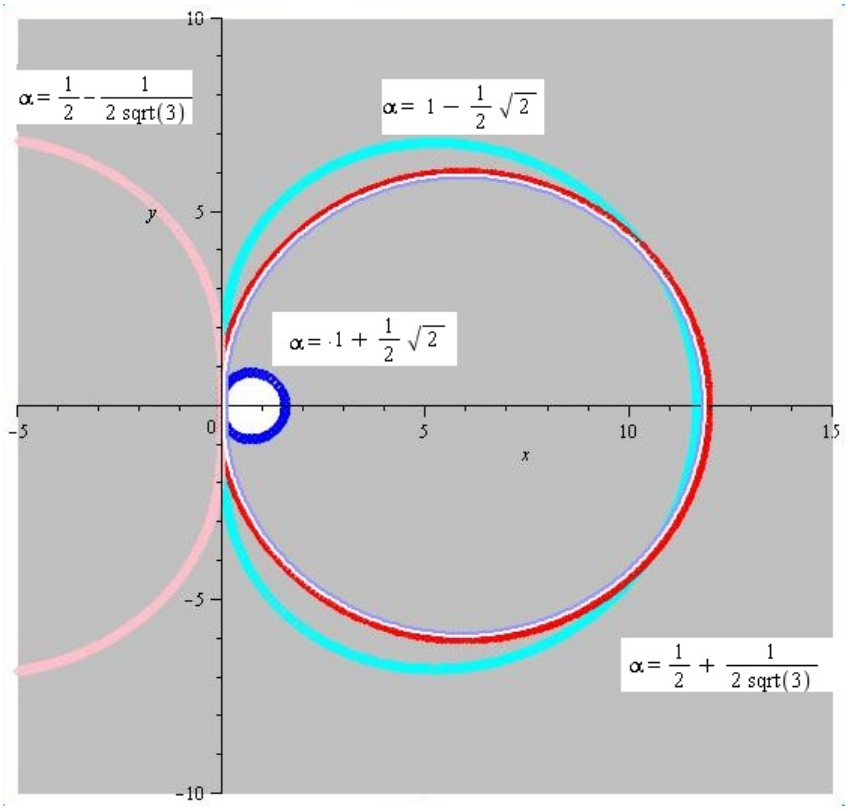

Figure 7: The stability region of the DIRK2 schemes is the outside of the closed blue and cyan curves defined as the points where $|R(z)| \leq 1$. Each curve corresponds to the values of α which makes the scheme L-stable. The red and pink curves correspond to values of α such that the scheme with only 2 stages is of order 3; for $\alpha = 1 + \frac{1}{2} + \frac{1}{2\sqrt{3}}$, the scheme is L-stable and is a SSP3 scheme.

Another scheme of the same shape with 2 stages and of order 2 can be found with $\alpha = 1 + \sqrt{2}$. This scheme is also L-stable:

$$\begin{array}{c|cc} \frac{1+\alpha}{2} & \frac{1+\alpha}{2} & \\ 1-\frac{\alpha}{2} & -\alpha & \frac{1+\alpha}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

- A DIRK (3,3) scheme with parameter γ is formulated as

$$\begin{array}{c|ccc} \gamma & \gamma & & \\ 1-\gamma & 1-2\gamma & \gamma & \\ \frac{1}{2} & \frac{1}{2}-\gamma & 0 & \gamma \\ \hline & \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \end{array} \qquad \begin{array}{ll} k_1 & = X_n + dt\ \gamma f(k_1) \\ k_2 & = X_n + dt\ ((1-2\gamma) f(k_1) + \gamma f(k_2)) \\ k_3 & = X_n + dt\ ((\frac{1}{2}-\gamma) f(k_1) + \gamma f(k_3)) \\ \hline X_{n+1} & = X_n + \frac{dt}{6}\ (f(k_1) + f(k_2) + 4 f(k_3)) \end{array}$$

The limit of the stability polynomial is

$$\lim_{z\to\infty} R^{(3)}(z) = \frac{1-4\beta+2\beta^2}{2\beta^2} \tag{41}$$

This equation admits two roots $a = 1 \pm \frac{1}{\sqrt{2}}$ and the root $a = 1 + \frac{1}{\sqrt{2}}$ corresponds to the smallest area.

- Another (3,3) DIRK scheme of degree 3 with $\alpha = \frac{2}{\sqrt{3}}\cos(\frac{\pi}{18})$ is written as

$$\begin{array}{c|ccc|l}
\frac{1+\alpha}{2} & \frac{1+\alpha}{2} & & & k_1 = X_n + dt\ \frac{1+\alpha}{2} f(k_1) \\
\frac{1}{2} & -\frac{\alpha}{2} & \frac{1+\alpha}{2} & & k_2 = X_n + dt\ (-\frac{\alpha}{2} f(k_1) + \frac{1+\alpha}{2} f(k_2)) \\
\frac{1+\alpha}{2} & 1+\alpha & -(1+2\alpha) & \frac{1+\alpha}{2} & k_3 = X_n + dt\ ((1+\alpha)f(k_1) - (1+2\alpha)f(k_2) + \frac{1+\alpha}{2} f(k_3)) \\
\hline
 & \frac{1}{6\alpha^2} & 1-\frac{1}{3\alpha^2} & \frac{1}{6\alpha^2} & X_{n+1} = X_n + dt\ (\frac{1}{6\alpha^2} f(k_1) + (1-\frac{1}{3\alpha^2}) f(k_2) + \frac{1}{6\alpha^2} f(k_3))
\end{array}$$

The value of the diagonal terms $\frac{1+\alpha}{2}$ is 1.068 and consequently, the intermediate stages are solved with a range bigger than the stepsize.

An interesting theorem sets that there is no DIRK formula with (q,p)=(4,5).
The following figure points out the stability area of the Dirk2, 3 and 4. It is clear that these schemes are A-stable.

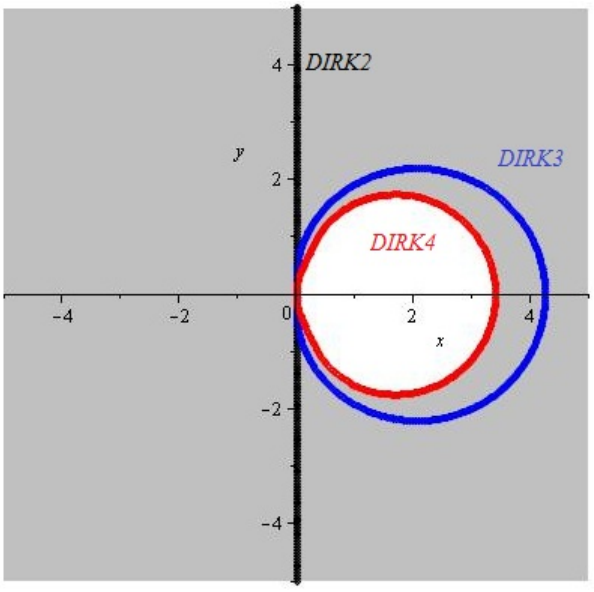

Figure 8: The stability region of order ranging from 2 to 4 is the outside of the closed curves defined as the points where $|R(z)| \leq 1$. The Dirk schemes are A-stable.

3.3 Fully implicit Runge-Kutta schemes

The fully implicit Runge-Kutta schemes, that is to say schemes which the corresponding Butcher table is not yet lower triangular, have very interesting stability properties. We investigate two famous schemes (for the first order, the Radau scheme is just the implicit Euler scheme)

3.3.1 Implicit RK3 or Radau3 scheme

The corresponding Butcher table (on the left) and the steps (on the right) are

$1/3$	$\frac{5}{12}$	$-\frac{1}{12}$
1	$\frac{3}{4}$	$\frac{1}{4}$
	$\frac{3}{4}$	$\frac{1}{4}$

$$k_1 = X_n + \frac{dt}{12}(5f(k_1) - f(k_2))$$
$$k_2 = X_n + \frac{dt}{4}(3f(k_1) + f(k_2))$$
$$X_{n+1} = X_n + dt\frac{dt}{4}(3\ f(k_1) + f(k_2))$$

It is necessary to solve the following linear system to get the stability polynomial

$$\begin{pmatrix} 1 - \frac{5}{12}z & \frac{1}{12}z \\ -\frac{3}{4}z & 1 - \frac{1}{4}z \end{pmatrix} \begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{42}$$

the solution of which is

$$k_1 = -2 * y0\frac{-3 + z}{z^2 + 6 - 4 * z}$$
$$k_2 = 2 * y0\frac{3 + z}{z^2 + 6 - 4 * z}$$

The roots of the stability polynomial are with $z = \lambda\ dt$

$$r = 2\ \frac{3 + z}{6 - 4 * z + z^2} \tag{43}$$

(The error is $\varepsilon = \frac{1}{72}h^4$)

3.3.2 Implicit RK5 or Radau5 scheme

The RK5 implicit Butcher table is written as

$\frac{4-\sqrt{6}}{10}$	$\frac{88-7\sqrt{6}}{360}$	$\frac{269-169\sqrt{6}}{360}$	$\frac{-2+3\sqrt{6}}{225}$
$\frac{4+\sqrt{6}}{10}$	$\frac{269+169\sqrt{6}}{360}$	$\frac{88+7\sqrt{6}}{360}$	$\frac{-2-3\sqrt{6}}{225}$
1	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$
	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$

The linear system due to the implicitation has to be solved

$$\begin{pmatrix} 1 - z\frac{88-7\sqrt{6}}{360} & -z\frac{296-169\sqrt{6}}{1800} & -z\frac{-2+3\sqrt{6}}{225} \\ -z\frac{296+169\sqrt{6}}{1800} & 1 - z\frac{88+7\sqrt{6}}{360} & -z\frac{-2-3\sqrt{6}}{225} \\ -z\frac{16-\sqrt{6}}{36} & -z\frac{16+\sqrt{6}}{36} & 1 - \frac{z}{9} \end{pmatrix} \begin{pmatrix} k_1 \\ k_2 \\ k_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{44}$$

(The error is $\varepsilon = \frac{1}{7200}h^6$)

Once k_1, k_2 and k_3 are computed, the equation giving X_{n+1} provides the stability polynomial

$$r = \frac{3(20 + 8z + z^2)}{60 - 36z + 9z^2 - z^3} \tag{45}$$

The following figure shows that the Radau schemes of order 1, 3 and 5 are A-stable.

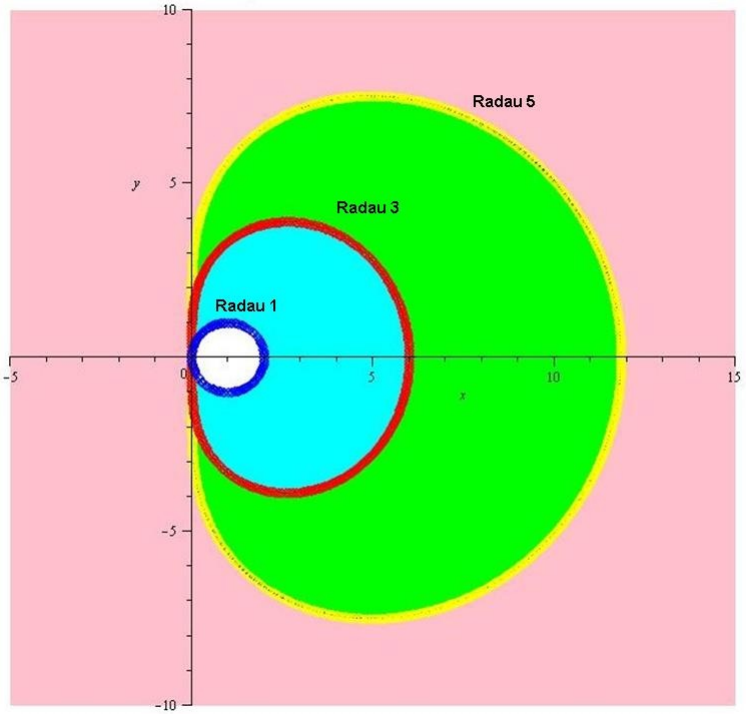

Figure 9: Stability of Radau1, Radau3 and Radau5 implicit schemes. These schemes are A-stable because their stability region are outside the closed curves which remains totally on the right half plane (real(z)>0).

Finally, it exists another IRK scheme of different class. It is the Lobatto IIIC which is of order 4

$$\begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

(The error is $\varepsilon = \frac{1}{480}h^5$)

The IRK schemes needs large computation to solve the implicit problem at each intermediate stage. Therefore, the benefit in stability remains to be pointed out in comparison to the DIRK scheme but

especially to the explicit RK schemes.

3.4 Embedded Runge-Kutta schemes

As we explain in the error estimation section, it is necessary to assess X_{n+1} for two consecutive orders in order to estimate the local and the global error due to the numerical scheme. To do these double assessments, some particular Butcher tables have been computed. To manage an economical computation of these two assessments, some people succeed in establishing 2 butcher tables for order n and n-1 with the same sequence of k_i. Thus, the k_i are computed once time and from an order to the following, only the linear combination of the c_i changes. The second serie of c_i coefficients is the last row of the Butcher RK4-2 table:

0					
$1/3$	$1/3$				
$2/3$	$-1/3$	1			
1	1	-1	1		
	$1/8$	$3/8$	$3/8$	$1/8$	
	$1/12$	$1/2$	$1/4$	0	$1/6$

Second row of coefficients is useful to compute XX, the order $k-1$ assessment.

$$
\begin{aligned}
k_1 &= f(X_n) \\
k_2 &= f(X_n + \frac{1}{3}k_1 dt) \\
k_3 &= f(X_n + \frac{1}{3}(-k_1 + 3k_2 dt) \\
k_4 &= f(X_n + k_1 - k_2 + k_3 dt) \\
X^{(4)}_{n+1} &= X_n + dt\frac{1}{8}(k_1 + 3k_2 + 3k_3 + k_4) \\
XX^{(3)}_{n+1} &= X_n + dt\frac{1}{12}(k_1 + 6k_2 + 3k_3 + 0k_4 + 2f(X_{n+1}))
\end{aligned}
$$

$X^{(4)}_{n+1}$ et $XX^{(3)}_{n+1}$ represent assessments respectively for orders 4 and 3. Generally, the operator sets the relative tolerance $rtol$ and/or the absolute tolerance $atol$. Some authors recommend to use $Tol = atol + rtol\ max|X_i|$. Consequently, it becomes possible to compute the time step according to the requested accuracy and with less computing cost because the embedded schemes for orders k and $k-1$.

3.5 New and simple Runge-Kutta schemes

The Butcher coefficients have to satisfy several conditions; thus, it is possible to build an underdetermined system to compute them.
- The sum of b_i must be equal to 1
- the relation has to be verified $c_i = \sum_{j=2,s} a_{ij}$.

Let us consider first, the following RK2 scheme

$$X_{n+1} = X_n + hb_1k_1 + hb_2k_2 \tag{46}$$

with

$$k_1 = f(t_n + hX_n) \tag{47}$$
$$k_1 = f(t_n + hc_2, X_n + ha_{21}k_1) \tag{48}$$

The coefficients of RKx scheme are not unique and in order to show how it is possible to find the coefficients of a RK2 scheme, we consider the Taylor expansion of X_{n+1}.

$$X_{n+1} = X_n + hf_n + \frac{1}{2}h^2 \left(\frac{df}{dt}\right)_n + O(h^3) \tag{49}$$

By expanding, we have

$$X_{n+1} = X_n + hf_n + \frac{1}{2}h^2 \left(\left(\frac{\partial f}{\partial t}\right)_n + \left(\frac{\partial f}{\partial X}\right)_n f_n\right) + O(h^3) \tag{50}$$

So we have

$$\begin{aligned} k_1 &= f(t_n, X_n) = f_n \\ k_2 &= f(t_n + hc_2, X_n + ha_{21}k_1) \\ &= f(t_n, X_n + ha_{21}k_1) + hc_2 \frac{d}{dt} f(t_n, X_n + ha_{21}k_1) + O(h^2) \\ &= f_n + hc_2 \left(\frac{\partial f}{\partial t}\right)_n + ha_{21} \left(\frac{\partial f}{\partial X}\right)_n f_n + O(h^2) \end{aligned}$$

We can then rewrite the equation (46)

$$X_{n+1} = X_n + hb_1 f_n + hb_2 \left(f_n + hc_2 \left(\frac{\partial f}{\partial t}\right)_n + ha_{21} \left(\frac{\partial f}{\partial X}\right)_n f_n\right) + O(h^3) \tag{51}$$

By equalization of the coefficients of 50 and of 51, it becomes

$$\begin{aligned} b_1 + b_2 &= 1 \\ b_2 c_2 &= \frac{1}{2} \\ b_2 a_{21} &= \frac{1}{2} \end{aligned}$$

We have three equations and four unknowns, so we can solve the system by setting a parameter, for example b_2, so we have

$$\begin{aligned} X_{n+1} = X_n + h[(1 - b_2)k_1 + b_2 k_2], \\ k_1 = f(t_n, X_n), \\ k_2 = f\left(t_n + \frac{h}{2b_2}, X_n + \frac{h}{2b_2}k_1\right) \end{aligned}$$

Butcher tables of RK schemes of orders 2, 3, 4, 5 et 6

The idea is to have $(\forall i,\ a_{ii} = \frac{1}{2})$ and the terms under diagonal equal to zero. Then, it remains to compute the b_i by identifying with the exponential expansion.

0		
$1/2$	$1/2$	
	0	1

0					
$1/2$	$1/2$				
$1/2$	0	$1/2$			
	0	$1/3$	$2/3$		

0				
$1/2$	$1/2$			
$1/2$	0	$1/2$		
$1/2$	0	0	$1/2$	
	0	$1/3$	$1/3$	$1/3$

0					
$1/2$	$1/2$				
$1/2$	0	$1/2$			
$1/2$	0	0	$1/2$		
$1/2$	0	0	0	$1/2$	
	0	$1/3$	$1/3$	$1/5$	$2/15$

0						
$1/2$	$1/2$					
$1/2$	0	$1/2$				
$1/2$	0	0	$1/2$			
$1/2$	0	0	0	$1/2$		
$1/2$	0	0	0	0	$1/2$	
	0	$1/3$	$1/3$	$1/5$	$4/45$	$2/45$

The stability polynomials of all these schemes are the exponential expansions truncated at the scheme order.

4 Multistep BDF schemes

In opposition to the Adams method family determined by integrating an approximation polynomial, BDF methods are obtained by differentiating this polynomial φ(t).

4.1 Illustration for BDF scheme with 2 steps

The polynomial with 2 steps (X_{n+1}, X_n et X_{n-1}) is

$$X(t) \approx \varphi(t) = X_n + \frac{t - t_n}{t_n - t_{n-1}}(X_n - X_{n-1}) + \frac{(t - t_n)(t - t_{n-1})}{2(t_n - t_{n-1})^2}(X_n - X_{n-1} - X_{n-1} + X_{n-2}) \quad (52)$$

By deriving this expression with respect to t, it becomes

$$X'(t) = f(t, X) \approx \varphi'(t) = \frac{X_n - X_{n-1}}{t_n - t_{n-1}} + \frac{2t - t_n - t_{n-1}}{(t_n - t_{n-1})^2}(X_n - X_{n-1} - X_{n-1} + X_{n-2}) \quad (53)$$

$$f(t_n, X_n) = \frac{X_n - X_{n-1}}{t_n - t_{n-1}} + \frac{2t_n - t_n - t_{n-1}}{2(t_n - t_{n-1})^2}(X_n - X_{n-1} - X_{n-1} + X_{n-2}) \quad (54)$$

By simplifying and collecting, it remains

$$f(t_n, X_n) = \frac{X_n - X_{n-1}}{t_n - t_{n-1}} + \frac{X_n - 2X_{n-1} + X_{n-2}}{2(t_n - t_{n-1})} \quad (55)$$

And by isolating X_n and by setting $h = t_n$ - t_{n-1}, it comes the two steps BDF scheme:

$$X_n = \frac{4}{3}X_{n-1} - \frac{1}{3}X_{n-2} + \frac{2}{3}hf(X_n, t_n) \quad (56)$$

The **Gear or BDF** schemes are well adapted to stiff systems as their stability regions are the largest than the previous schemes. It is the implicit side which gives them these large stability regions. They have the following constraints:

$$p = k \qquad et \qquad b_0 = b_1 = ... = b_{k-1} = 0 \tag{57}$$

$$X_{n+i} = \sum_{i=0}^{k} a_i X_{n-i} + dt\ b_0\ f(X_{n+1}) \tag{58}$$

X_{n+1} depends on the previous values of X; the scheme is implicit since it depends on $f(x_{n+1})$. At orders 2, 3, 4, 5 and 6, here is the list of the schemes

$$order1 \quad X_{n+1} = X_n + dt\ f_{n+1}$$
$$order2 \quad X_{n+1} = \frac{1}{3}(4X_n - X_{n-1} + 2dt\ f_{n+1})$$
$$order3 \quad ; X_{n+1} = \frac{1}{11}(18X_n - 9X_{n-1} + 2X_{n-2} + 6dt\ f_{n+1})$$
$$order4 \quad X_{n+1} = \frac{1}{25}(48X_n - 36X_{n-1} + 16X_{n-2} - 3X_{n-3} + 12dt\ f_{n+1})$$
$$order5 \quad X_{n+1} = \frac{1}{137}(300X_n - 300X_{n-1} + 200X_{n-2} - 75X_{n-3} + 12X_{n-4} + 60dt\ f_{n+1})$$
$$order6 \quad X_{n+1} = \frac{1}{147}(360X_n - 450X_{n-1} + 400X_{n-2} - 225X_{n-3} + 72X_{n-4} - 10X_{n-5} + 60dt\ f_{n+1})$$

4.2 Stability study of BDF2 scheme

From the scheme $X_{n+1} = \frac{1}{3}(4X_n - X_{n-1} + 2dtf_{n+1})$, it is possible to write easily

$$w^2 = \frac{1}{3}(4w - 1 + 2\lambda w^2) \tag{59}$$

By setting $z = \lambda dt$, it appears a second degree polynomial
$(1 - \frac{2\lambda}{3})w^2 - \frac{4}{3}w + \frac{1}{3} = 0$
The roots are $w_{1/2} = \frac{-2 \pm \sqrt{1 + 2x + i2y}}{-3 + 2x + i2y}$
The scheme is stable when $|w_i| <= 1$ for i=1,2. The stability region corresponds to the grey region on the following figure. We can prove that in all the left half-plane (x<0), the scheme is stable; the scheme is so A-stable.

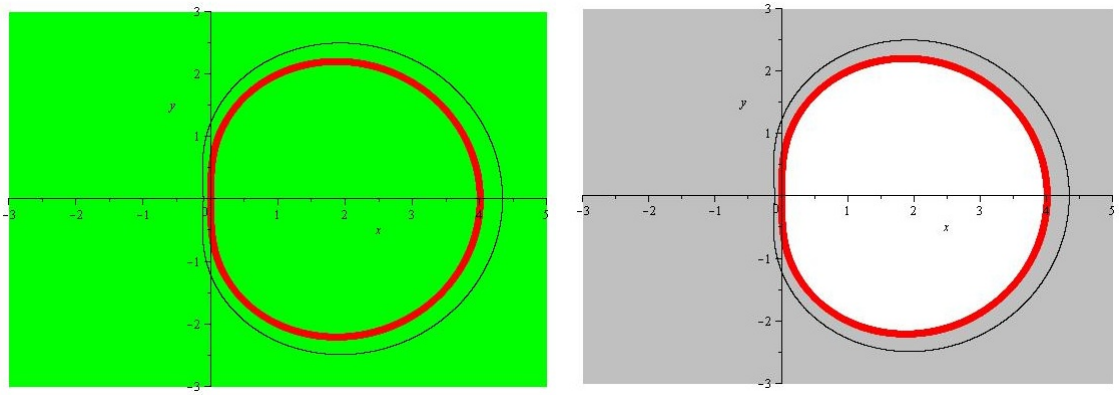

Figure 10: Stability region of the BDF scheme of order 2 for both roots of the stability polynomial. We notice that one of the roots is always smaller than 1. 2 contour lines have been plotted: a red one for w=1 and a black one for w=0.9; only the inside is not stable.

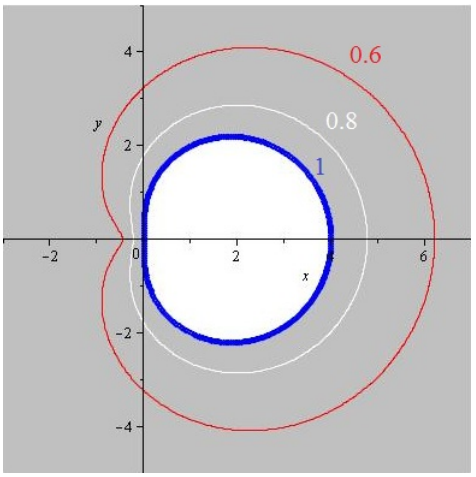

Figure 11: The stability region of the BDF scheme for order 2 is outside the blue curve - grey region. We have a lose plotted the contour lines for 0.8 and 0.6. The instability region is the white region inside: The BDF2 scheme is A-stable.

4.3 Stability study for BDF3 scheme

Reminder of BDF3 scheme:

$$X_{n+1} = \frac{6}{11}\left(3y_n - \frac{3}{2}X_{n-1} + \frac{1}{3}X_{n-2} + dtf(X_{n+1})\right) \tag{60}$$

The characteristic equation of BDF3 scheme:

$$(1 - \frac{6}{11}z)w^3 - \frac{18}{11}w^2 + \frac{9}{11}w - \frac{2}{11}w = 0 \tag{61}$$

The following figure shows the 2 regions of stability of the 2 roots of this stability polynomial; the third root is always smaller than 1. We have almost A-stability but unfortunately there is a small region of instability in the left half plane. The stability regions are grey tint outside the closed curves (instability region in white). The instability region for the scheme is so the conjoining of both instability regions.

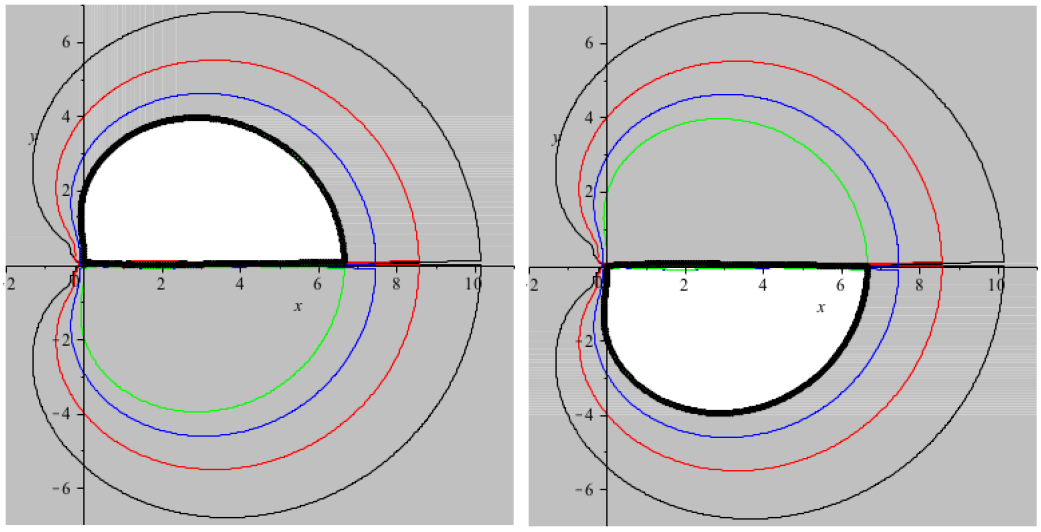

Figure 12: Regions of stability of the 2 roots of the polynomial stability of the BDF3 scheme; the third root is always smaller than 1. The contour lines of the roots of 0.9, 0.8 and 0.7 have been plotted in the stability region.

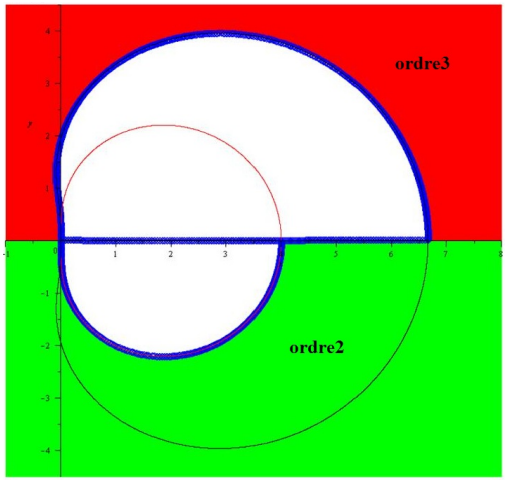

Figure 13: The stability region of BDF2 and BDF3 schemes - green and red regions respectively; We have plotted only the half of the plots to make it easier: has to be completed by symmetry around the axis of x. We have plotted also the contour lines of 0.8 and 0.6. The instability region is the white region inside the curves respectively: the BDF2 scheme is A-stable but the BDF3 scheme is not.

The BDF scheme for a given order seems to be the one which have the largest stability region and so the most well adapted for stiff systems.

4.3.1 Stability of the BDF4 scheme

The BDF4 scheme is the following

$$X_{n+1} = \frac{12}{25}\left(4X_n - 3X_{n-1} + \frac{4}{3}X_{n-2} + \frac{1}{4}X_{n-3} + dt\ f(X_{n+1}\right) \tag{62}$$

His characteristic equation is a 4th order polynomial

$$w^4 * (1 - \frac{12}{25}z) - \frac{48}{25}w^3 + \frac{36}{25}w^2 - \frac{16}{25}w + \frac{3}{25} = 0 \tag{63}$$

One of the roots is always positive and the following figure gives the regions in pink where they are smaller than 1.

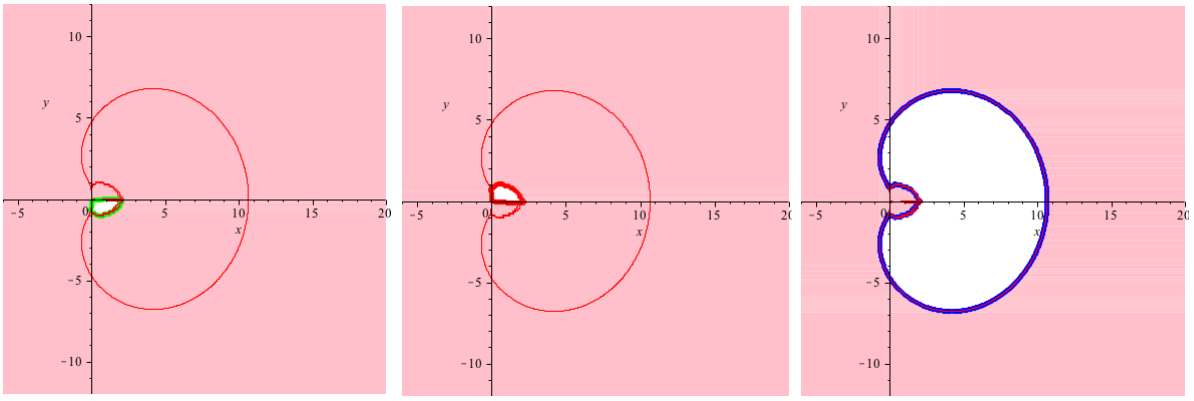

Figure 14: The stability regions have 4 parts, each one is linked to a root of the characteristic polynomial. One of the roots is always smaller than 1. The outside of the closed curve corresponds to regions where the roots are smaller than 1.

4.4 Stability study of the BDF5 scheme

The BDF5 scheme is the following

$$X_{n+1} = \frac{1}{137}(300X_{n+1} - 300X_n + 200X_{n-1} - 75X_{n-2} + 12X_{n-3} + 60dtf_{n+1}) \tag{64}$$

It has the following characteristic equation:

$$w^5 * (137 - 60 * z) - 300 * w^4 + 300 * w^3 - 200 * w^2 + 75 * w - 12 = 0 \tag{65}$$

It has 5 roots and one is always smaller than 1

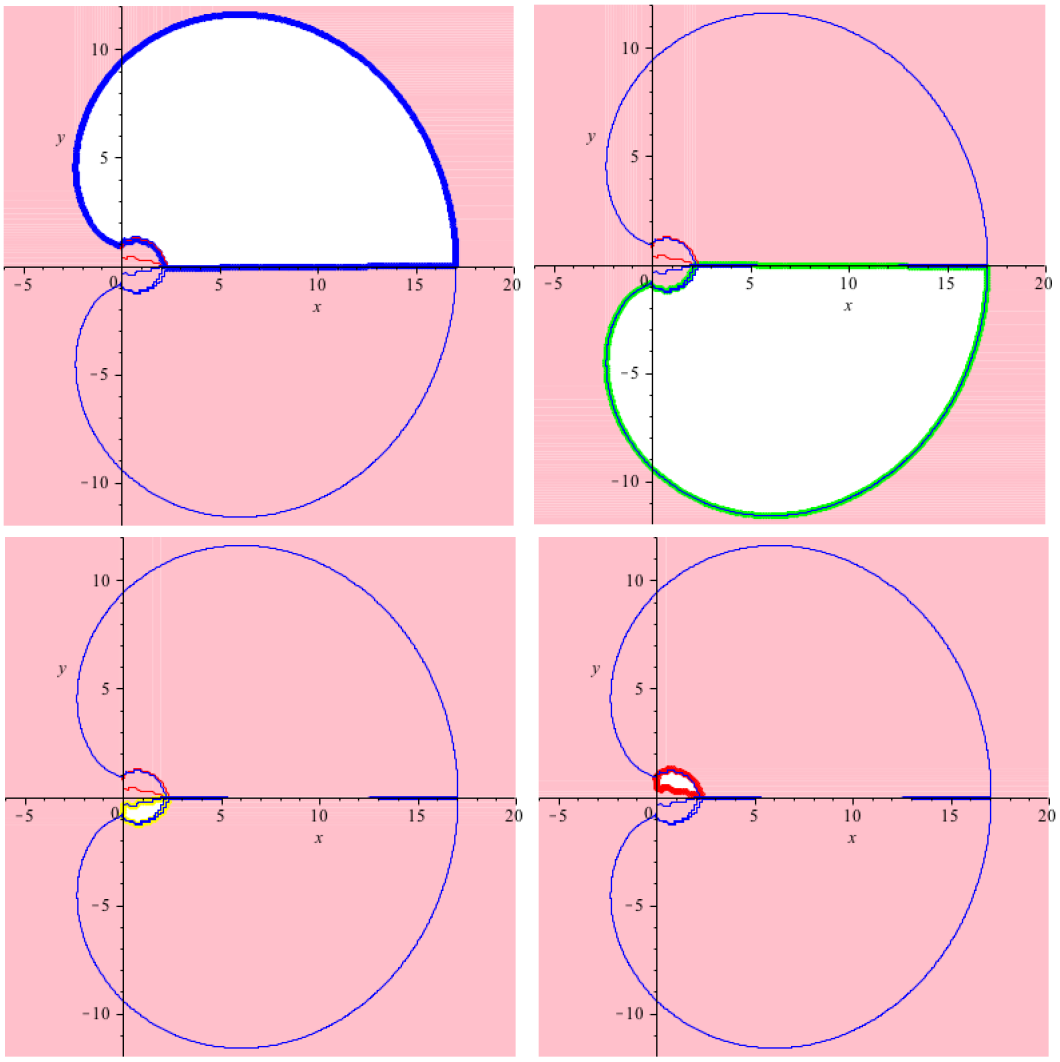

Figure 15: The regions of stability of the BDF5 scheme have 5 parts, each one is linked to a root of the characteristic polynomial. one of the roots is always smaller than 1. The outside of the closed curve corresponds to regions where the roots are smaller than 1.

The scheme is not stable because of the instability region in the left half plane. However, the outside of the closed curve is the common stability region of all 5 roots.

4.5 Study of the stability of the BDF6 scheme

The BDF6 scheme is the following:

$$X_{n+1} = \frac{1}{147}(360X_{n+1} - 450X_n + 400X_{n-1} - 225X_{n-2} + 72X_{n-3} - 10X_{n-4} + 60dtf_{n+1}) \tag{66}$$

His characteristic equation is the following

$$(147 - 60z)w^6 - 360w^5 + 450w^4 - 400w^3 + 225w^2 - 72w + 10 = 0 \tag{67}$$

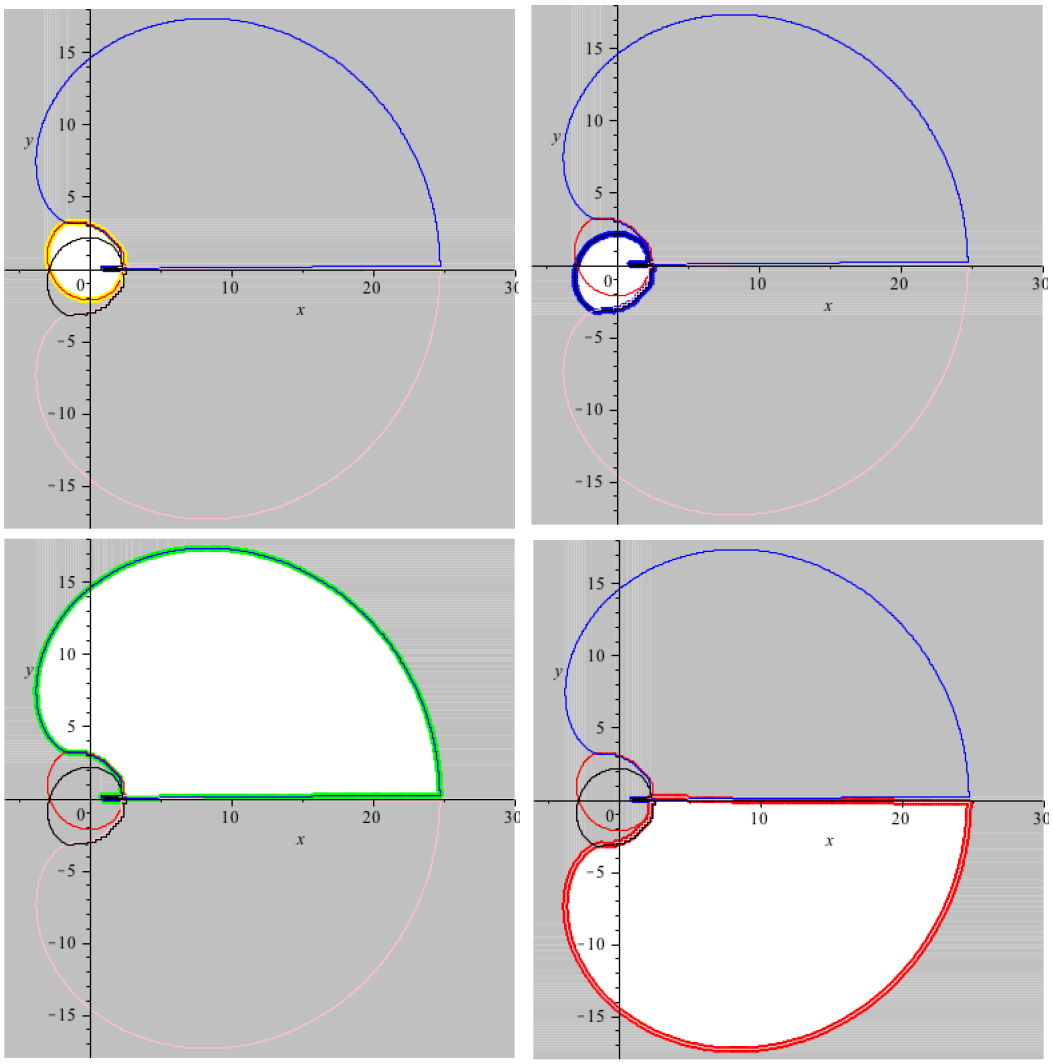

Figure 16: The stability regions of the BDF6 scheme have 4 parts, each one is linked to a root of the characteristic polynomial. 2 roots are always smaller than 1. The outside of the closed curves corresponds to stability regions where the roots are smaller than 1.

The BDF scheme for a given order seems to be the one which has the largest stability region and so the most well adapted for stiff problems

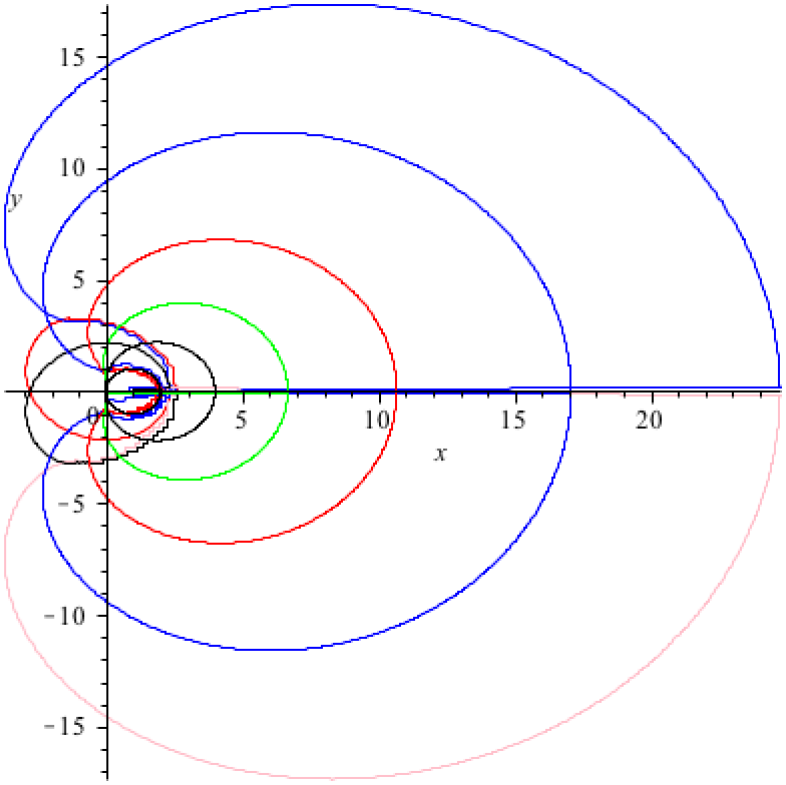

Figure 17: Comparison of the size of the regions of stability for the BDF schemes of order 1, 2, 3, 4, 5 and 6. The stability is outside the closed curves which are embedded.

5 IMEX schemes

The ODE's problems often trade with complex systems in exploitation. They are related to parameter estimation or predictive evolution of the complex system depending of internal parameters. The resolution of such problem allows to know whether the system can see its performances dissipated or even break down or become dangerous.
Another kind of complex problem appears in the literature to solve physical problem with partial differential equations. These problems are reduced to ODE's problems by means of the Method Of Line. It consists of discretization of the spatial derivative with a scheme at the given current time and to remain with only a temporal integration problem. Several problems are presented in the literature and the ODE problems are seen as modeling problem for what a special care is brought to the time integration.

5.1 Splitting by means of the mesh

First of all, [38] [39] presents a general fluid dynamics problem with turbulence solved by means of Large Eddy Scale modeling and an equation for heat balance. The spatial scheme uses a discontinuous Galerkin approach and the time integration proceeds with an Implicit Explicit scheme. The problem is split into a stiff part $g(X)$ and a non-stiff part $f(X)$ such a way as

$$\frac{dX}{dt} = f(X) + g(X) \tag{68}$$

A ERK scheme is applied to the non-stiff part and a DIRK scheme to the stiff part of the model. The interest is huge since it is not yet necessary to decrease drastically the time step if using everywhere an ERK scheme; the total cost of simulating is 3 times lower.
The difficulty is to set the splitting; it is driven by the meshing. Indeed at the proximity of the boundaries, there is a strong gradient of velocity which makes the problem stiff in the vicinity of the viscous layer. The splitting is displayed on the following figure.

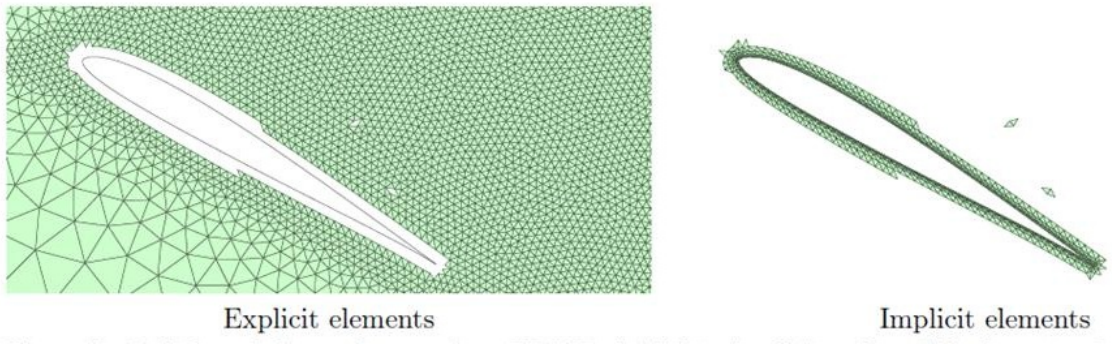

Figure 18: Stiff et non-stiff areas for the wing meshing.

Same idea is developed in [31] which investigated the Karman street behind a circular obstacle. The splitting is still driven by the meshing of the boundary layer as illustrated on the following figure

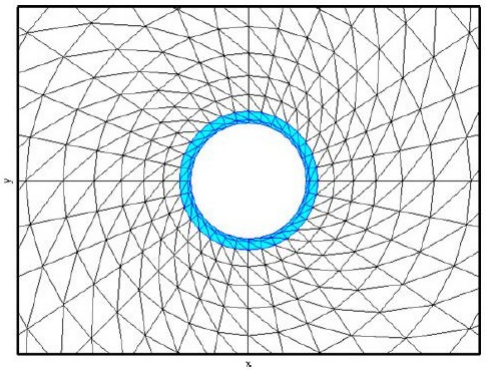

Figure 19: Stiff (blue area) and nonstiff areas around the obstacle.

5.2 Splitting a model into linear and nonlinear parts.

In [17], atmospheric processes are examined in the following form

$$\frac{\partial u}{\partial t} = Lu + f(u) \tag{69}$$

L is a matrix representing the linear part of the model. The remaining terms are included in $f(u)$. The short time scales are assumed to be gathered in L and if some fast processes are not linear, they are linearized. Therefore f(u) contains the longer wave lengths.
The numerical approximation can be expressed in the form

$$\sum_{M-1}^{1} \alpha_k u^{n+k} = \Delta t \left(\sum_{k=-M}^{0} \beta_k f(u^{n+k}) + \sum_{k=-M}^{1} \nu_k L u^{n+k} \right) \tag{70}$$

(α_k, β_k) define the explicit method and (α_k, β_k) the implicit one; the choice of a single set of α_k restricts the possibilities of the IMEX schemes.

6 Hybrid time

Hybrid dynamic systems are described by means of a continuous time behavior and a discrete event behavior. The continuous behavior is based on differential equations $\dot{X} = f(X, u, t)$. And the discrete events are defined by means of **indicator functions** $g(X, u, t)$; when an indicator function changes of sign (from positive to negative or vice versa), a discrete event δ happens and it is referred to as a zero-crossing. In algorithm words, there are a first step of zero-crossing detection, a second step of zero-crossing location and finally a model update.
Zero-crossing detection consists of watching the sign of the indicator. Zero-crossing location makes currently use of bisectional search until a value of $g(X, u, t)$ less than a given threshold [45]. At the third step, the model has to be updated, either by modifying some differential equations or in adding some ones. Then the solver is reset and starts again from the last detected event time with new initial conditions to be determined.
In [45], several pathological issues for the zero-crossing are analyzed.
The first case is described with an indicator function of dimension 2: $\{g_1, g_2\}$ respectively of red and green colors in the next figure. Inside the time interval $[T_{n-1}, t_n]$, the first component crosses twice and the second only one time. It appears easy to detect $\delta 2$ and subsequently $\delta 1$ whereas the third crossing escapes from detection.

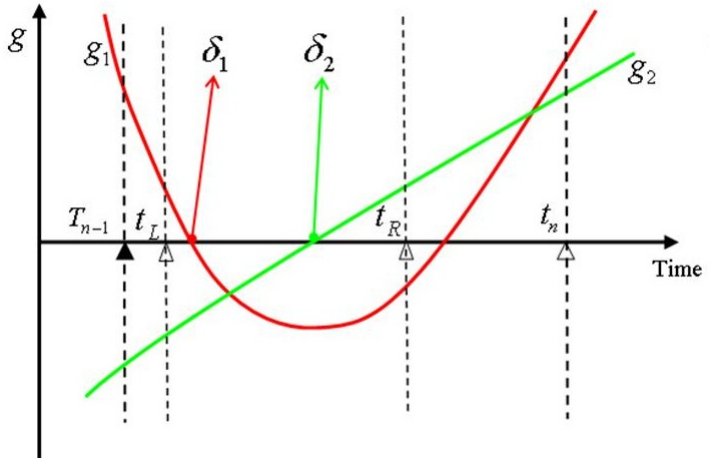

Figure 20: Masked even roots problem extracted from [24].

Another case of difficulty for zero-crossing is the chattering. Then, the zero-crossings become closer until the model is as in halt; an infinite number of transitions take place in a finite time interval. The model has then a zeno behavior; there is also a Zeno neighborhood around the Zeno point in case of Zeno execution. Zeno phenomenon is not only a mathematical trick, and a model Zeno behavior can be caused by model simplifications. All these concepts are mathematized in [24] and [32]. Such behavior can be clearly illustrated by means of the bouncing ball close the end of its movement. A Zeno neighborhood criterion is defined in [45]:

- First, all the indicator function values have to be less than a given threshold ε inside one integration interval.

- Second, before T_{n-1} an excess of consecutive zero-crossings must be detected. With these two conditions, a Zeno point is detected and the algorithm can be halted.

In [32] and [1], 3 examples are described and some regularization means are tested. In the case of the bouncing ball, 2 regularizations are carried out:

- A temporal regularization is given in applying a time delay before bouncing. This regularization is proved to be efficient.

- A dynamical regularization where the ground is modeled as a stiff spring, is also efficient. Both regularizations are consistent and physically meaningful.

7 Open software

7.1 DASSL

Differential-Algebraic Stiff System Solver DASSL is a software developed by Sandia National Laboratory [40] for solving ordinary differential and differential-algebraic equations on implicit form; it is one of the most successful solvers. An interface for ODE and DAE is at the user disposal.
This algorithm uses Adams and BDF implicit schemes; there is a permanent adaptation of the time step since the schemes are expressed on variable time steps. That damages the accuracy of the algorithm whereas high orders are used.
Currently, Adams Bashforth scheme is used as a predictor step of the Adams-Moulton scheme used as a corrector step. However, the integration begins with a ERK2 from the initial conditions.
The DAE such as $g(X, \dot{X}, t) = 0$ are solved using a Newton-Raphson scheme. Let us consider the Jacobian $J = \frac{\partial g}{\partial X} + c\frac{\partial g}{\partial \dot{X}}$. The Newton-Raphson iterations are driven as

$$X_{k+1}^n = X_k^n - (J(X_k^n))^{-1} g(X_k^n, \dot{X}_k^n, t_n) \tag{71}$$

It is necessary that the Jacobian be invertible. The error computation is managed from the following formula

$$e_i = \frac{Err_i}{rtol * |X_i| + atol} \tag{72}$$

If e_i is less than 1, the solution is kept. Else, the time step is shortened. Err_i is the local error done for the current time step and $atol$ is the machine error. In case of multiple components, the final value is computed by means of RMS or infinite norm. Obviously, the result accuracy and consumed time depends a lot on the chosen $rtol$.

It is known that DASSL is inefficient enough in case of non-stiff equation system and well appropriated for stiff and large scale models.

In [34], there is the description of a rewritten code in C^{++} of DASSL and some rules of well programming are noted. The authors of CDASSL stress on a rigorous management of the various errors able to appear during execution and the need to build a uniform calling interface to structure the data exchange.

In [14], we found some remarks about programming. In the seventies, models were low order and devoted to limited complexity models as Van der Pol model. They currently written in Fortran.
Presently, the programming state of art is drastically changed. The increasing "clientele" enlarged the field of involved applications and the computer performances have exploded. Some convergent

remarks can be found in [34]. A revision of Fortran Dassl Petzold version is described: object-oriented programming with 3 targets: proper data storage, programming error and data consistency, numerical error. The authors stress on the usefulness of uniform interfaces. The uniform calling interfaces consist of a structure which bundles the information by a class of functions, a constructor and a destructor of structure, all of that referenced by a pointer of type void.

7.2 LSODAR

- LSODE (Livermore Solver for Ordinary Differential Equations)[40] written by Alan C. Hindmarsh, is a basic solver written originally in standard Fortran 77 for the initial value problem for ordinary differential equation systems. It has eight variants that are suitable for both stiff and non-stiff systems. It uses Adams methods (predictor-corrector) in the non-stiff case, and Backward Differentiation Formula (BDF) methods in the stiff case.
  In the stiff cases, it treats the Jacobian matrix $\frac{df}{dX_n}$ as a dense (full) or a banded matrix or as user-supplied or internally approximated by difference quotients. It includes solvers for systems given in explicit form, as like as LSODE, and also solvers for systems given in linearly implicit form, as like as LSODI.
  For explicit form solvers, it is assumed that ODEs are given explicitly, so that the system can be written in the form $\frac{dX_n}{dt} = f(t_n, X_n)$, where X_n is the vector of dependent variables, and t_n is the independent variable.

- LSODES, is like LSODE, but in the stiff case the Jacobian matrix is assumed to be sparse, and treated with sparse routines;

- LSODA, solves systems $\frac{dX_n}{dt} = f$ with a dense or banded Jacobian when the problem is stiff, but it automatically selects between non-stiff and stiff methods. It uses the non-stiff method initially, and dynamically monitors data in order to decide which method to use;

- LSODAR is like LSODE, but includes the ability to detect and solve for solutions of a related set of algebraic equations. It is a variant of LSODA with a root-finding capability added, it finds the roots of any of a set of given functions of the form g(tn, Xn). This is often useful for finding stop conditions, or for finding points at which a switch is to be made in the function;

- LSODPK, is similar to LSODE, but uses preconditioned Krylov space iterative methods for the linear equation solvers;

- LSODKR includes the root-finding ability of LSODA, and the Krylov solvers of LSODPK. Solvers for linearly implicit systems These kind of solvers treat systems in the linearly implicit form $A(t_n, X_n)\frac{dX_n}{dt} = g(t_n, X_n)$, A is a square matrix, i.e. with the derivative dX_n/dt implicit, but linearly so;

- LSODI, solves the implicit system $A(t_n, X_n)\frac{dX_n}{dt} = g(t_n, X_n)$, it assumes the matrix A is either dense or banded;

- LSOIBT, is similar to LSODI, but assumes the matrix A is block tridiagonal.

7.3 QSS

Classically, the solver advances in time from a step to another. The underlying idea of the Quantized State System consist of state step and to look for time providing such state advance while keeping the time continous. This approach of state slicing wrong-foots all the classical methods based on the time slicing.
The QSS methods described in [33] of fairly recent vintage, are developed at order 2 and 3 as QSS2 and QSS3 respectively. These algorithms are naturally asynchronous since the target value of each state is reached at various times. The authors note also the symplectic properties of these schemes in particular cases. The parallelization of QSS algorithms is studied in [23].

8 Parallelization approaches for hybrid ODEs resolution

When complex hybrid ODEs have to be solved, simulation duration becomes of primary concern. To allow a speedup of this simulation duration and to avoid model reduction, parallelization of the simulation resolution is of interest.

8.1 Introduction

Parallel computing is employed for solving large problems that could be partitioned into many independent small parts in order to be solved by multiple processing elements in a simultaneous way. There are different types of parallelism: bit-level, instruction level, data and task or thread parallelism.

Bit-level parallelism is directly related to the processor word size. In fact, when variables sizes are greater than the length of the word, the processor have to execute an operation in at least two instructions. So increasing the word size will reduce the number of instructions.

Instruction-level parallelism is directly related to the processor pipeline size. In fact, the number of stages in the pipeline represents the potential parallelism for the instructions. In order to have an efficient parallelism, there have to be many independent instructions that could be re-ordered and grouped together into a pipeline. Recent processor have super-scalar capabilities allowing them to be able to execute several instructions in parallel, using algorithms like Tomasulo algorithm.

Data parallelism is directly related to program loops. It is possible only when there is no dependencies in the iteration loop. The iterations could be then divided into the number of available processors and executed in parallel without disturbing the data.

Task parallelism consists in distributing threads (or process) across multiple core or processors. It allows different calculations on the same or different sets of data, unlike the data parallelism which only allows the same calculation on them.

Different approaches have been proposed for the parallelization of ODEs and hybrid ODEs:

8.2 Parallelization across the method

The main approach to obtain a parallel numerical scheme for ODE systems consists of parallelizing "across the method". The main idea is to exploit concurrent function evaluations within a step. Burrage provides an excellent review of these methods in [9].

Explicit multistage Runge-Kutta methods are sequential. However, in some cases two or more stages of DIRK methods can be executed in parallel if some coefficients a_{ij} of the strictly lower triangular matrix associated to the method are zero. The parallelization of these methods were studied in [29]. The conclusions is that the parallelization potential is limited.

In the sixties, Miranker and Liniger [36] have proposed an approach allowing to devise parallel predictor/corrector methods. The later generalization of these approaches is reviewed in the surveys

by [30] and [9]. Like the previous approaches, these methods offer a limited parallelization potential. One drawback of these methods is their relatively small stability regions.

Other approaches that can fit this classification rely on parallelizing matrix inversions, which are needed when using an implicit method [44, 26] or and parallelizing operations on vectors for ODEs resolution by separating them into modules (see PVODE solver [12] implemented using MPI (Message-Passing Interface) technology)

Domain specific approaches were also studied. In [15], the parallel execution of multibody simualations on multi-core or shared memory multiprocessors were studied. Using OpenMP, the proposed approach relies on the parallelization of matrix/vector operations. This approach was implemented on the MBSim tool and evaluated on several simple and complex case-studies.

8.3 Parallelization across the steps

Another approach, which is the time decomposition method, originally introduced by researchers from the multi-grid field [27] and applied to solve PDEs. Moreover, two methods were introduced in this category: the Parareal scheme proposed in [35] and PITA algorithm described in [21], where both of them were derived from the multiple shooting method [18]. They follow the approach of splitting the time domain in sub-domains by considering two levels of time grids. A first parallel computation of a predicted solution is performed with a fine time grid. After that, at each end of time of sub-domains, the solution makes a jump with the previous initial boundary value (IBV) of the next time sub-domain. A correction of the IBV for the next fine grid is then computed on the coarse time grid. [21] shows that the method converges at least in a finite number of iterations. Nevertheless, this approach seems to have difficulties for stiff nonlinear problems. In [26], adaptivity in the tolerance of the time slice integrator and the number of sub-domains was studied and some improvements has been shown towards stiff ODEs. However, for very stiff problems, difficulties still appears. So another parallel solver has been proposed for stiff ODEs based on Richardson Extrapolation.

8.4 Parallelization across the model

Numerically integrating PDEs in parallel is facilitated by the need to solve across their spatial dimensions, which naturally leads to data parallelism. However, this is not the case for ODEs/DAEs, when both models and solvers exhibit a strong sequential nature.

Decoupling this apparently sequential computation is an important issue for distributed simulation, because the way of decoupling a system could significantly affect the simulation results. Some important methods exploiting the parallelization across the model include the relaxation algorithm and the transmission-line modeling method.

8.4.1 Waveform relaxation

The Relaxation algorithm uses an iterative process to solve equations. Its principle consists in, first, decomposing the system into several subsystems, then, the relaxation algorithm is applied to solve each of them with an initial condition. In each iteration, subsystems are solved separately. The iteration stops when the solutions converge. In order to achieve a reasonable convergence rate, the system should be partitioned at specific locations where the coupling is weak. This is a trade-off for the designer because if there are too many tightly coupled elements in a same subsystem, the advantage of using the relaxation algorithm is weakened. In [28], three types of coupling methods were compared to show how they affect the convergence of the solution.

The waveform relaxation (WFR) method is a decomposition method useful to solve a large ODE system defined on a time interval $I = [t_{start},\ t_{end}] \subset \mathbb{R}$. Its is based on a Jacobi iterative scheme in the framework of which each relaxation elements are called a wave. The global system is partitioned into p pieces when we use p processors. Each subdomain is solved separately and then, they

exchange the computed waves; the algorithm is repeated until global convergence. Each dynamical subsystem is solved for the entire time interval I.
The state vector of dimension N is divided into p joined but without overlapping sub vectors $X = \{X_1,\ X_2,\ \ldots, X_{p-1},\ X_p\}$. Each subvector is of dimension N/p.
The system is assumed to be of the shape $\dot{X} = A.X$ and the waveform decomposition on 4 processors is driven as

$$A(X)\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix}\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix}$$

Figure 21: Decomposition on 4 processors in freezing 3/4 of the components in each of the 4 processors.

From a mathematical point of view, the WFR algorithm is interesting since the convergence is proved by means of the two-point contraction mapping theorem.

Definition 8.4.1. *A mapping $f : D \subset \mathbb{R}^n \times \mathbb{R}^n$ is two point contractive on a closed set $D_0 \subset D$ if there are $\alpha_0 \in [0, 1]$ and $\alpha_1 \in [0, 1]$ such that*

$$||f(x_0, x_1) - f(y_0, y_1)|| \leq \alpha_0\ ||x_0 - y_0|| + \alpha_1\ ||x_1 - x_1|| \tag{73}$$

Theorem 8.1. *Two point contraction mapping theorem :*
Suppose that $f : D \subset \mathbb{R}^n \times \mathbb{R}^n$ is two point contractive on a closed set $D_0 \subset D$ and that $f(D_0, D_0) \subset D_0$. Thus, f has a unique fixed point x^* in D_0. Furthermore, for any initial guess $x^1, x^0 \subset D_0$ and x^1, x^0 linearly independent, the sequence $\{x^{k+1} = f(x^k, x^{k-1})\}$ converges uniformly to x^*.

This theorem is important since it states the convergence of the WFR method under no restrictive assumptions.

Take care about convergence.
Crow [16] indicates an example of model of dimension 5 which converges straightforwardly. He shows that the WFR method destroys the solvability of the model. Indeed, the splitting points out jacobians in the sub matrix which are evanescent around the equilibrium point. Therefore, the only efficient method is to solve one shot the entire problem. It seems there is no rule to detect such drawbacks.

First enhancement of the method : Overlapping.
An enhancement of the method consists of overlapping the splitting. This method becomes clear by means of illustration of an example. Let us consider a dynamic system $\dot{X} + AX = B$ where the matrix A is partitioned in the following manner

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix} = A_1 + A_2 = \left(\begin{array}{cc|cc} 2 & -1 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ \hline 0 & 0 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{array}\right) - \left(\begin{array}{cc|cc} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \hline 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right) \tag{74}$$

The splitting allows the computation on two processors. Matrix A_1 consists of two diagonal blocks which are not coupled and matrix A_2 couples the components X_2 and X_3. Standard WFR method needs iterations in order to get convergence on these two unknowns; processor number one computes X_1 and X_2 whereas X_3 and X_4 are frozen and processor number two computes X_3 and X_4

whereas X_1 and X_2 are frozen. At the end of each iteration, the processors exchange the computed unknowns.
The idea of the overlapping is to compute more in each processor in order to increase the speed velocity of the convergence. Each processor compute a ghost unknown tied up to a ghost line. The new sub matrix are

$$A1 = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \end{pmatrix} \quad A2 = \begin{pmatrix} -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix} \tag{75}$$

Thus the processor number one computes the unknowns X_1, X_2 and X_3 with X_4 frozen and the processor number two compute X_2, X_3 and X_4 with X_1 frozen. Therefore the components X_2 and X_3 are computed twice; their update after exchange of the waves, is managed by means of the average of their values to get quicker convergence.

Second enhancement : Steffensen acceleration.
This method is already described in the subsection *Resolution of nonlinear problems.* The methods aims at building a new series which converges more quickly than the natural series of the WFR method.

Third enhancement : Windowing.
It can be shown [46] that the WFR method converges superlinearly on a finite interval of length T. The uniform convergence is reached in the exponential norm

$$||X||_{\xi,T} = max_{t\in[0,T]} \, |e^{\xi t} X(t)| \tag{76}$$

In other words, WFR method converges quickly on short intervals than in longer intervals. Therefore, it can be advised to cut the time interval into subintervals, called windows. However, it is not clear in the literature there exists an optimal length and the optimal window seems to be problem- and machine-dependent. Substantial computations can be saved using small subwindows.

For more information about WFR method, please consult deliverable D6.2.7.

8.4.2 Transmission line modeling

The transmission line modeling (TLM) technique [28],[7] is a discrete modeling method that provides a general approach for decoupling systems in distributed simulation. It represents the physical process by a transmission-line graph which is solvable by a computer. According to this method, the decoupling point should be chosen where variables change slowly and the time-step of the solver should be relatively small. Consequently, the decoupled subsystems are seen as they were connected by constant variables and the error due to time delays can be significantly decreased. Other advantages of this technique are the ability to increase the efficiency and the accuracy of the solution, in fact the discrete model is derived with TLM directly from the physical system (elimination of the discretization error).

The Hopsan [20] simulation tool, used primarily for hydro-mechanical simulation, allows multi-domain system modeling and simulation. It integrates the TLM method allowing performing multi-threaded and multi-core simulations. The integration of TLM method in Modelica was studied in [42],[41].

8.4.3 Modular time-integration or co-simulation

A modular time integration method, also called co-simulation, sees the system to be integrated as a connection of several subsystems. Its proceeds in macro steps of stepsize H. The data exchange between subsystems is restricted to the discrete synchronization points. Between these synchronization points the subsystems are integrated completely independent of each other.

The xMOD tool [3] supports this method. In xMOD, each subsystem (that can be an FMU or a model from an authoring tool like Simulink), is assigned to a thread, with an associated communication step-size and solver. This method was benchmarked by Faure et al. [22] in the context of Hardware-In-The-Loop. Several alternative methods were proposed in order to perform real-time simulation of complex physical models. However, the study was focused only on fixed-step solver without treating the case of variable-step solver In [5], the modular time integration involving the LSODAR solver was studied. Tests results on engine model showed that, with the model partitioning, it is possible to use efficiently and locally variable-step solvers thanks to the decrease of the number of discontinuities, so the number of integration interrupts, in each subsystem, achieving a speedup of 3.9 in a 4-core machine with respect to the single solver approach. Numerical stability of these methods was studied in [13].

8.5 Parallelization with several cores

Since the cores have a shared memory, it is judicious to use OpenMP parallelization. Open-Multi-Processing is an API which supports multi-platform and shared memory multiprocessing, (see [43] for more details).

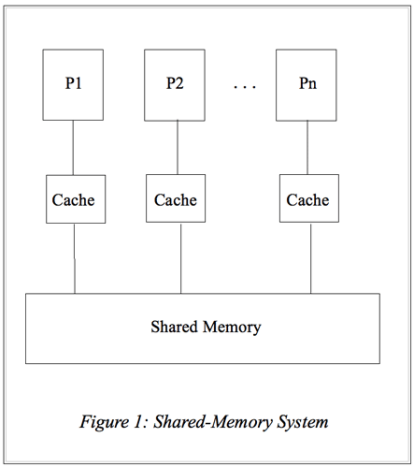

Figure 22: Shared memory within several cores

Thus, the computation of the derivative $\dot{y}(t)$ is parallelized a directive is typed just before the "for" loop:

```
#pragma omp parallel for
for (icomp=1; icomp<Neq; icomp++) {
        ydot[icomp] = ... ;          }
```

The computation of each component of the derivative vector is parallelized and the "for" loop is split in smaller "for" loops. The number of small for loops is equal to the number of cores set in the machine. All the cores can access to the vector y which is stored in the shared memory. That is why y is not split between the different cores.

The pros of OpenMP are the following :

- simple directives to add to the code
- the program can run either sequential either parallel without changing anything of the code
- a small program can be parallelized without changing the remaining parts.

- decomposition of the date is automatic

The cons of OpenMP :

- needs a shared memory
- requires a compiler which supports OpenMP
- no error handling
- useless for program with a short running time

We can expect a Speedup of N if we use N processors but in reality there are several reasons why it does not occur :

- processes must wait if there is dependency
- a process must wait if it accesses to a data which is shared
- the whole code is not parallelized
- there is limits with the memory bandwidth

8.6 Parallelization of huge size dynamic problem

When the size of dynamic problems is huge the parallelization within several cores is not enough. More and more computers have several processors with several cores. The parallelization within the different processors is useful for large scale systems. However contrary to the cores the processors have their own memory and do not share it with the other. That is why the processors exchange data trough messages. The best way to exchange those messages is to use MPI (Message Passing Interface).

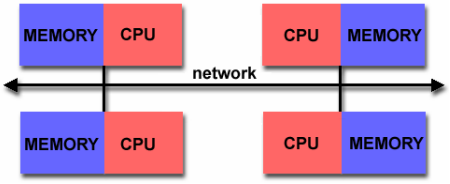

Figure 23: Distributed memory within several processors connected

"MPI is a message-passing application programmer interface, together with protocol and semantic specifications for how its features must behave in any implementation" [37].

The goals of MPI are high performance, scalability and portability. Typically each CPU is assigned to a single process. The messages consist in send/receive data and combine partial results of computations (gather and reduce operations) and synchronizing nodes.

8.7 Management of a dynamic problem

Most of the time, the equation system is very sparse (i.e. only a few of the variables are present in any one equation). The matrix A of $\dot{X} = A.X$ includes only some percent of non-zero elements. If sparsity is not taken into account, the computer looses its time to make multiplication by zero. Therefore the road of the efficiency passes through sparsity management and the corresponding algebra. It is usual to also consider the special case of diagonal type or diagonal block type matrix. In [19], we can find some more sophisticated techniques called relaxing and tearing approaches. They lack of generality since they are efficient in specific domains

9 Proposal for FMI extension to support parallelization

9.1 Position of the problem

Modelica models contain systems of ODE/DAE defined on a time horizon $t \in [t_{start} = 0, t_{end}] \subset \mathbb{R}$ with a state vector $X(t) \in \mathbb{R}^n$. The mapping F is associated to the ordinary differential equations and the mapping G to the algebraic equations also called constraints, such that the problem is formulated as

$$\begin{aligned} X(t=0) = X_0 \quad &, \quad X \in \mathbb{R}^n \\ \dot{X} = F(X,Y,t) \quad &, \quad F : \mathbb{R}^{n+m+1} \longrightarrow \mathbb{R}^n \\ 0 = G(X,Y,t) \quad &, \quad Y \in \mathbb{R}^m, G : \mathbb{R}^{n+m+1} \longrightarrow \mathbb{R}^m. \end{aligned}$$

This is the simplest formulation of ODE/DAE problems said explicit, but there exists a more general implicit formulation

$$\begin{aligned} X(t=0) = X_0 \quad &, \quad X \in \mathbb{R}^n \\ H(\dot{X},X,Y,t) = 0 \quad &, \quad H : \mathbb{R}^{n+m+1} \longrightarrow \mathbb{R}^n \\ 0 = G(X,Y,t) \quad &, \quad Y \in \mathbb{R}^m, G : \mathbb{R}^{n+m+1} \longrightarrow \mathbb{R}^m. \end{aligned}$$

To the former problem, it is possible to write it in a 'matrix form'. F can be split into a matrix A_l representing the linear part of F, and a formal array A_n which represents the nonlinear remaining part of the considered mapping. G can be split in the same way into a matrix T_l representing the linear part and a formal array T_n representing the nonlinear remaining part such that

$$\begin{aligned} \dot{X} = F(X,Y,t) \quad &\Longrightarrow \quad \dot{X} = A_l X + A_n(X) + A_0 \\ 0 = G(X,Y,t) \quad &\Longrightarrow \quad 0 = T_l X + T_n(X) + T_0. \end{aligned}$$

9.2 Motivation through an example

We consider a problem of dimension $n = 1000$ without DAE. It is a linear problem of heat conduction. The calculations are done in standalone by means of C language code. This code is structured in order to mimic the communication of the solver with the FMU. It consists of two parts:

- a solver which is a time integrator based on BDF and DIRK schemes,
- a routine '**fex**' which computes the appropriate time derivatives of the state vector on solver request.

The calculations of **fex** and solver are parallelized by means of Open MP pragmas. The following curves display the variations of the dimensionless time $\frac{T_{fex}}{T_{fex}+T_{Solver}}$ with respect to the non-zero elements of matrix $A = A_l$ for a given number of cores. It appears clearly that the most important contribution to the total consumed time is due to the calculation of **fex** and not to the solver calculation. We think this result is general and still true in case of nonlinear ODE/DAE where more complex functions are involved.

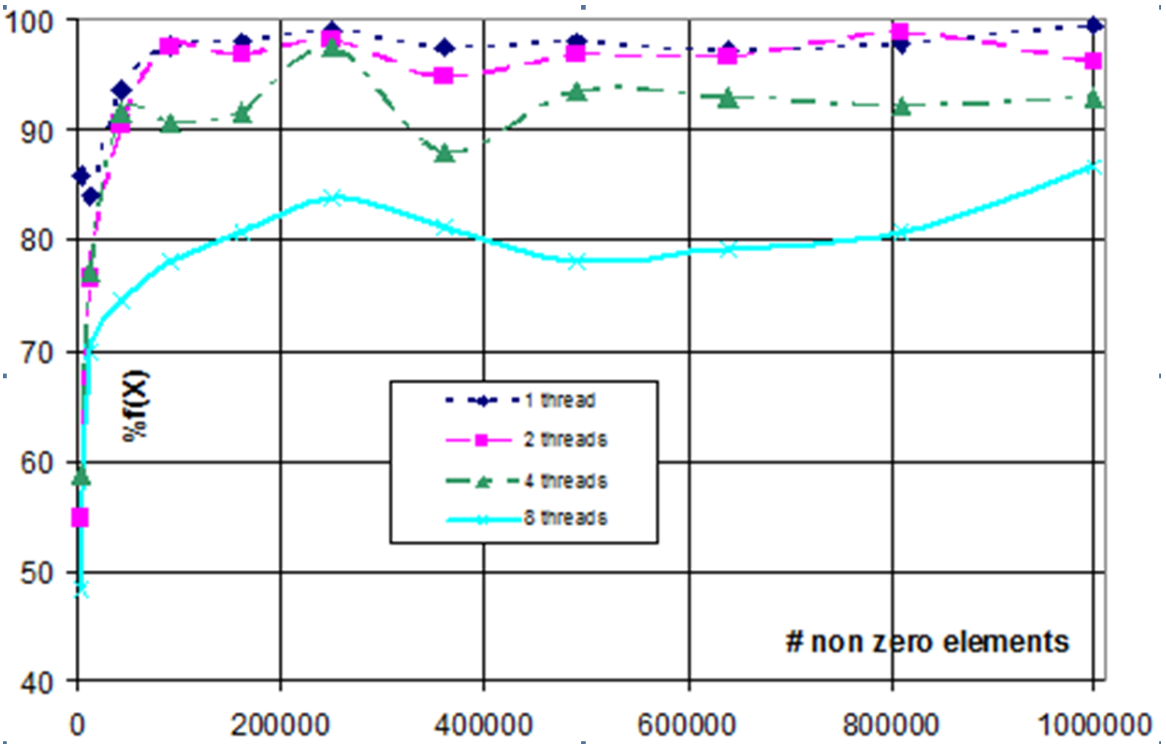

Figure 24: $\frac{T_{fex}}{T_{fex}+T_{Solver}}$ in % as a function of the number of non-zero elements

9.3 Global proposition

The matrices and vectors are considered in the framework of large systems; therefore, they must be implemented with sparse structure CSR or CSC.

9.3.1 Suggestion of method of identification

First of all, we need to carry out the identification of matrices A_l, T_l, and vectors A_0, T_0. An appropriate call request will allow to inform the solver of the constant coefficient values of the matrices A_l, T_l, and vectors A_0, T_0, only known inside the FMU. Currently, the introduced support of Jacobian (API function `fmi2GetDirectionalDerivatives`) as well as ModelStructure XML in FMI 2.0 allows the identification of the A_l matrix. The identification of vector A_0 can be performed by means of unit sequence $(0, 0, ..., 0)$. Currenlty FMI 2.0 only exposes an interface for handling ODEs. DAE problems are 'hidden' from the interface and handled internally inside the FMU. Handling matrices T_l and T_0 can be expressed once a specification of DAE handling with FMI is released.

9.3.2 New parallel fmi2GetDerivativesPart interface

Secondly, a specific enhanced call for the calculation of the derivative is needed. Indeed, currently, the derivative values are returned from the FMU to the solver as a monolithic block. This point penalizes heavily the parallel approach and the research works about. It is important to separate the ODE calculation from the DAE calculations, moreover, these calculations have to be split in order to satisfy the request of the threads and not a global request only appropriate in sequential calculations.

Old call :

```
fmi2GetDerivatives(fmi2Component c, fmi2Real derivatives[], size_t nx);
```

`fmi2GetDerivatives` allow to recover, in an atomic way, the value of all the derivatives in a vector of dimension nx.

Proposal of a new call :

```
fmi2GetDerivativesPart(fmi2Component c, fmi2Real derivativesPart[], size_t n1, size_t n2,
size_t nx)
```

- n1 : number of the first requested row (n1>0 and n1<=n2)
- n2 : number of the last requested row (n2 can be equal to n1, never <n1, always n1<=n2)
- nx: integer of total number of ODE in the system
- derivativesPart : output values of the derivative, with dimension n2-n1+1.

This call is an enabler to compute, in parallel, the **fex** function. To get effective parallel solution, FMI ModelStructure XML part, especially dependencies between $\dot{X}$ and X needs to be taken into account, so that effective parallel requests to `fmi2GetDerivativesPart` are made. The FMU needs to organize the components of variables X and U as illustrated in paper [4], to maximize the potential parallelism that can be obtained from this new call.

10 Recommendations

The improvement of the efficiency of solvers passes mainly through the improvement of the computation of model derivatives $\dot{X} = f(X)$. This computation is an important part of the computer load. Some considerations drive to dense, diagonal and sparse matrix/vector formulations; they have to be fitted with parallelization. The minimum is the use of Pragma Open MP. Any improvements of the solver efficiency by means of parallelization for instance, would be obstructed without the parallelization of derivatives computations. One possible solution is to extend the scope of the FMI specification to allow the parallel computation of $\dot{X} = f(X)$. Even if the solver is highly parallelized, the scheme performance fully depends of the very numerous f evaluations. An access to compute separately each component is necessary since each core computes only one partition of the state vector.

In conclusion of this point, FMI extension towards the parallel resolution of the derivatives function will be of great interest to our problem. This can be done either by extending the FMI derivatives computation interface to allow it to compute selected components of state derivatives, or by keeping the same interface while transparently parallelizing this resolution. Moreover, in many cases, f can be represented by a very sparse matrix. This sparsity has to be taken into account for f evaluation and storage.

For starting the scheme at the beginning or after each event which has stopped the marching time step, it is important to use an implicit scheme as IRK 3 in order to avoid numerous steps to recover an optimal time step. The following figure shows the difference between an Explicit Euler and an IRK3 to start from initial conditions.

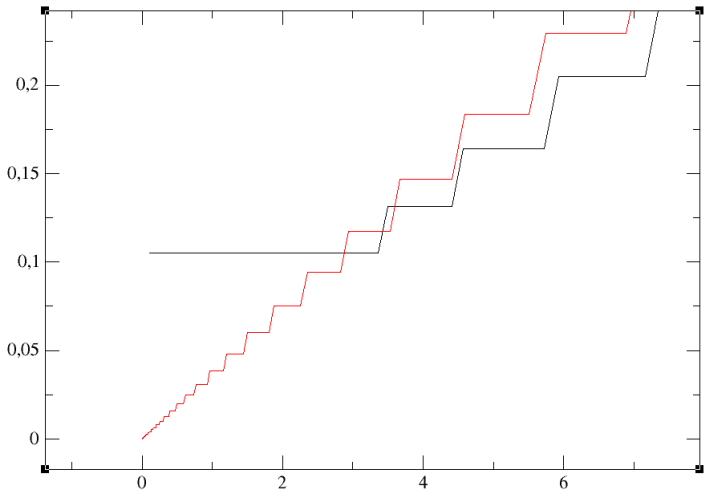

Figure 25: This curve represents the size of the time step depending on the time in seconds. Starting with explicit Euler (red curve) or IRK3 scheme (black curve). In last case, the time step appropriate to the given RTOL is correctly determined from the beginning.

Many considerations about the stability of the time solver lead to enlighten the benefit to use fully implicit Runge Kutta schemes. Several suggestions have been pointed out about the implementation of the Waveform Relaxation method. They aim was at improving the efficiency of the parallelization.

Necessity of large size benchmarks : Benchmarks are required at several levels. First, it is useful to determine the accuracy of an algorithmic process along the time. Second, the quantitative quality of the solution depending on events is also an important point. For both cases, benchmarks are required and for the second case, we indicate the bouncing ball problem. In [25], there is a complete test of the event detection and model update. A large size benchmark would be worldwide useful to test and compare the parallelization results.

References

[1] Ames Aaron, Zheng Haiyang, Gregg Robert, and Sastry Sankar. Is there life after Zeno? Taking executions past the breaking (zeno) point. *American control conference, Minneapolis*, june 14-16 2006.

[2] René Aïd. *Contribution à l'estimation asymptotique de l'erreur globale des méthodes d'intégration numérique à un pas. Application à la simulation des réseaux électriques*. PhD thesis, Institut National Polytechnique de Grenoble, 1998.

[3] M. Ben Gaid, G. Corde, A. Chasse, B. Lety, R. De La Rubia, and M. Ould Abdellahi. Heterogeneous model integration and virtual experimentation using xmod: Application to hybrid powertrain design and validation. In *7th EUROSIM Congress on Modeling and Simulation*, Prague, Czech Republic, Sep. 2010.

[4] Abir Ben Khaled, Mongi Ben Gaïd, and Daniel Simon. Parallelization approaches for the time-efficient simulation of hybrid dynamical systems: Application to combustion modeling. In *EOOLT'2013: 5th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, page 10, 2013.

[5] Abir Ben Khaled, Mongi Ben Gaïd, Daniel Simon, and Gregory Font. Multicore simulation of powertrains using weakly synchronized model partitioning. In *E-COSM'12 IFAC Workshop on Engine and Powertrain Control Simulation and Modeling*, Rueil-Malmaison, France, October 2012.

[6] Benjamin Bouvier. Résolution numérique des équations différentielles. *math.univ-lyon1.fr*, 2010.

[7] Robert Braun. *Multi-threaded distributed system simulations: Using bi-lateral delay lines*. Lic thesis No 1576, Linkoping University, 2013.

[8] Kevin Burrage. Efficiently implementable algebraically stable runge-kutta method. *Siam J. Numer. Anal.*, 19:245–258, 1982.

[9] Kevin Burrage. Parallel methods for initial value problems. *Applied Numerical Mathematics*, 11(1-3):5–25, January 1993.

[10] Kevin Burrage, Carolyn Dyke, and Bert Pohl. On the performance of parallel waveform relaxations for differential systems. *Applied Numerical Mathematics*, 20:39–55, February 1996.

[11] J C Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, Ltd, 2008.

[12] George D. Byrne and Alan C. Hindmarsh. PVODE, an ODE solver for parallel computers. *International Journal of High-Performance Computing Applications*, 13(4):254–365, 1999.

[13] George D. Byrne and Alan C. Hindmarsh. Stability of sequential modular time integration methods for coupled multibody system models. *Journal of computational and nonlinear dynamics*, 5(3), 2010.

[14] F.E. Cellier. How to enhance the robustness of simulation software. *Syst. Anal. Model*, Simul.1(1):55–61, 1984.

[15] Jan Clauberg and Heinz Ulbrich. An adaptive internal parallelization method for multibody simulations. In *In Proc. 12th Pan-American Congress of Applied Mechanics*, January 2012.

[16] M. L. Crow and M. D. Ilic. The waveform relaxation method for systems of differential/algebraic equations. *Mathl. Comput. Modelling*, 19(12):67–84, 1994.

[17] Peter Blossey Dale Durran. Implicit-explicit multistep methods for fast-wave slow-wave. *Monthly Weather Review*, march 30th 2011.

[18] P. Deuflhard. *Newton Methods for Nonlinear Problems*, volume 35 of *Springer Series in Computational Mathematics*. Springer, 2004.

[19] H. Elmqvist and M. Otter. Methods for tearing systems of equations in object-oriented modeling. In *In Proc. European Simulation Multiconference*, Barcelona, Spain, June 1994.

[20] Björn Eriksson, Peter Nordin, and Petter Krus. Hopsan ng, a c++ implementation using the tlm simulation technique. In *Proceedings of The 51st Conference on Simulation and Modelling*, Oulu, Finland, Oct. 2010.

[21] Charbel Farhat and Marion Chandesris. Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid-structure applications. *International Journal for Numerical Methods in Engineering*, 58(9):1397–1434, 2003.

[22] Cyril Faure, Mongi Ben Gaid, Nicolas Pernet, Morgan Fremovici, Font Gregory, and Gilles Corde. Methods for real-time simulation of cyber-physical systems: application to automotive domain. In *Proceedings of the First IEEE Workshop on Design, Modeling and Evaluation of Cyber Physical Systems*, Istanbul, Turkey, July 2011.

[23] Xenofon Floros. *Exploiting model structure for efficient hybrid dynamical systems simulation*. PhD thesis, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 21871, 2014, 2014.

[24] Zhang Fu, Karl Johansson, Lygeros John, and Sastry Shankar. Dynamical systems revisited: hybrid systems with zeno executions. *HSCC 2000, LNCS 1790*, pages 451–464, 2000.

[25] T.-H. Gallois, J Brac, and T. Soriano. A new method of event handling for ode solvers and application to a benchmark. In *27th European Simulation and Modeling Conference*, Lancaster UK, oct 2013. EUROSIS.

[26] David Guibert. *Analyse de méthodes de résolution parallèles d'EDO/EDA raides*. PhD thesis, Université Claude Bernard - Lyon, 2009.

[27] G. Horton and S. Vandewalle. A space-time multigrid method for parabolic pdes. Technical report, Universitaet Erlangen, 1993.

[28] S. Y. R. Hui and C. Christopoulos. Numerical simulation of power circuits using transmission-line modelling. *IEE proceedings. Part A. Physical science, Measurements and instrumentation, Management and education, Reviews*, 137(6):379–384, 1990.

[29] A. Iserles and S.P. Nørsett. On the theory of parallel Runge-Kutta methods. *IMA Journal of numerical analysis*, 10(4):463–488, 1990.

[30] Kenneth R. Jackson. A survey of parallel numerical methods for initial value problems for ordinary differential equations. *IEEE Transations on Magnetics*, 27(5):3792–3797, September 1991.

[31] A. Kanevsky, M.H. Carpenter, D. Gottlieb, and J.S. Hesthaven. Application of implicit-explicit high order runge-kutta methods to discontinuous-galerkin schemes. *Journal of Computational physics 225*, pages 1753–1781, march 2007.

[32] Johansson Karl H, Egerstedt Magnus, John Lygeros, and Shankar Sastry. On the regularisation of zeno hybrid automata. *Systems and control letters 38*, pages 141–150, june 14-16 1999.

[33] François Cellier Ernesto Kofman, Gustavo Migoni, and Mario Bortolotto. Quantized state system simulation, proc gcms'08, grand challenge in modeling and simualtion part of scsc'08. *Inst. Computational Science CAB G82.1*, pages 504–510, 2010.

[34] Alvin Leung, Anthony Skjellum, and Geoffrey Fox. Concurrent dassl : A second generation; dae solver library. *Scal. Parallel Librairies Conference*, 1993.

[35] J. Lions, Y. Maday, and G. Turinici. Résolution d'edp par un schéma en temps "parallèle". *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 332(7):661–668, April 2001.

[36] Willard. L. Miranker and Werner Liniger. Parallel methods for the numerical integration of ordinary differential equations. *Mathematics of Computation*, 21(99):303–320, July 1967.

[37] Peter S. Pacheco. *Parallel Programming with MPI*. Morgan Kaufmann Publishers Inc., 1996.

[38] Per-Olof Persson. High-order les simulations using implicit-explicit runge-kutta schemes. *49th AIAA Aerospace Sciences Meeting-Orlando*, 2011.

[39] Per-Olof Persson. High-order les simulations using implicit-explicit runge-kutta schemes. *50th AIAA Aerospace Sciences Meeting-Nashville*, 2012.

[40] Linda Petzold. Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM J. SCI. Stat. Comput.*, 4(1):136–148, 1983.

[41] Martin Sjölund. *Tools and Methods for Analysis, Debugging, and Performance Improvement of Equation-Based Models*. Doctoral thesis No 1664, Linkoping University, Department of Computer and Information Science, 2015.

[42] Martin Sjölund, Robert Braun, Peter Fritzson, and Petter Krus. Towards efficient distributed simulation in modelica using transmission line modeling. In *Proceeding of the 3rd International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Oslo, Norway, Oct. 2010.

[43] Edward Valeev. *Introduction to OpenMP*, 2009.

[44] P. J. van der Houwen and B. P. Sommeijer. Parallel iteration of high-order runge-kutta methods with stepsize control. *J. Comput. Appl. Math.*, 29:111–127, January 1990.

[45] Fu Zhang, Murali Yeddanapudi, and Pieter Mosterman. Zero-crossing location and detection algorithms for hybrid system simulation. *17th IFAC world congress*, pages 7967–7972, 2008.

[46] Hong Zhang. A note on windowing for the waveform relaxation method. *Applied Mathematics and Computation*, 76(1):49 – 63, 1996.