MAP551 - PC 5 - Numerical Integration of Ordinary Differential Equations (Part III)

Paul Calot

November 8, 2020

0.1 Introduction

0.2 Use of RADAU solver

0.2.1 Integration of Brusselator model

0.2.1.1

Table 1 gives the evolution of the number of time steps required before convering and for a given tolerance. As expected, the less the tolerance, the more time steps are required. Interestingly enough, when dividing the tolerance by 10, we multiply the number of required steps by roughly 1.6.

tolerance	time steps number
1e-3	62
1e-4	99
1e-5	166
1e-6	278
1e-7	483
1e-8	849

Table 1: Time steps required before converging as a function of tolerance and for Radau5.

0.2.1.2

When comparing the different methods accuracy and computational cost as a function of tolerance, we can note include the fixed time-step RK4 method which does not required any tolerance. In addition, we will use the integration time instead of any other indicators of complexity as this is the only one that we can compute easily for each scheme and use to compare to others. The first graph 1 shows the integration time as a function of tolerance for the following schemes : Radau 5, RK embedded, Dopris 5 and finally Radau 5 computed with *fortran*. The second graph 2 shows the accuracy as a function of the integration time for each scheme.

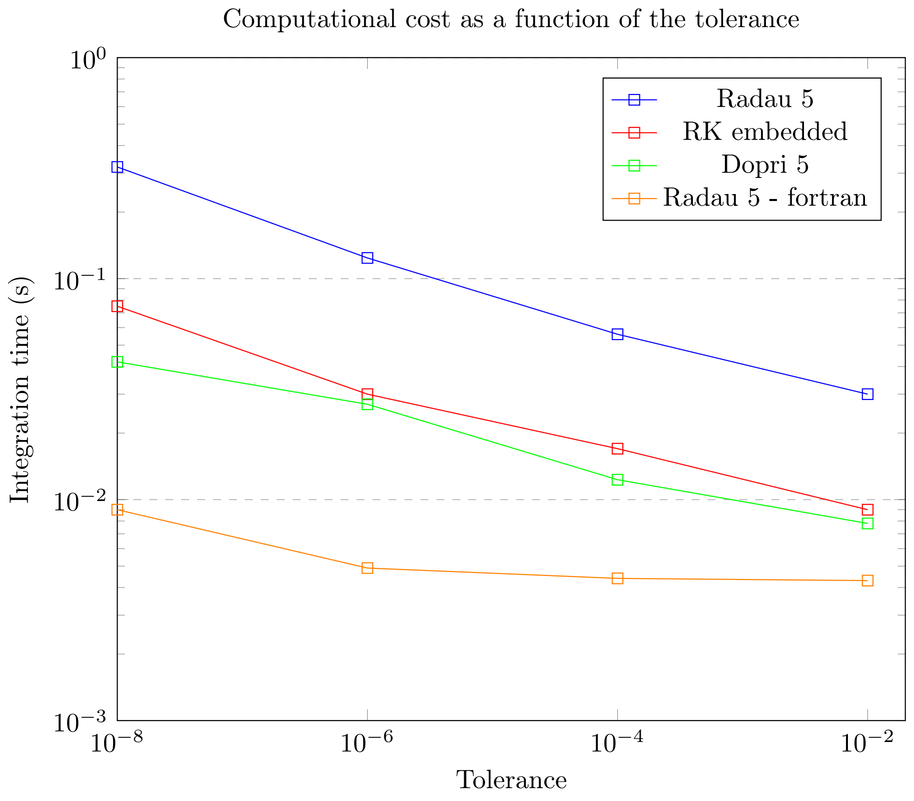

Figure 1: As one could except from the only implicit sheme coded in *python*, the implicit Runge-Kutta scheme **Radau 5** (blue curve) requires more time, at fixed tolerance, to compute the solution than the other python-coded schemes. The **Dopri 5** scheme, which is an computational-optimized version of the **RK embedded** fourth order sheme, as it requires less function evaluation for a step, uses slightly less time that the RK embedded scheme. Finally, **Radau 5** coded in *fortran* beats by a roughly 10-factor **Dopri 5**. Thus, for very time consuming application, it can be worth making the effort to use *fortran* instead of *python*.

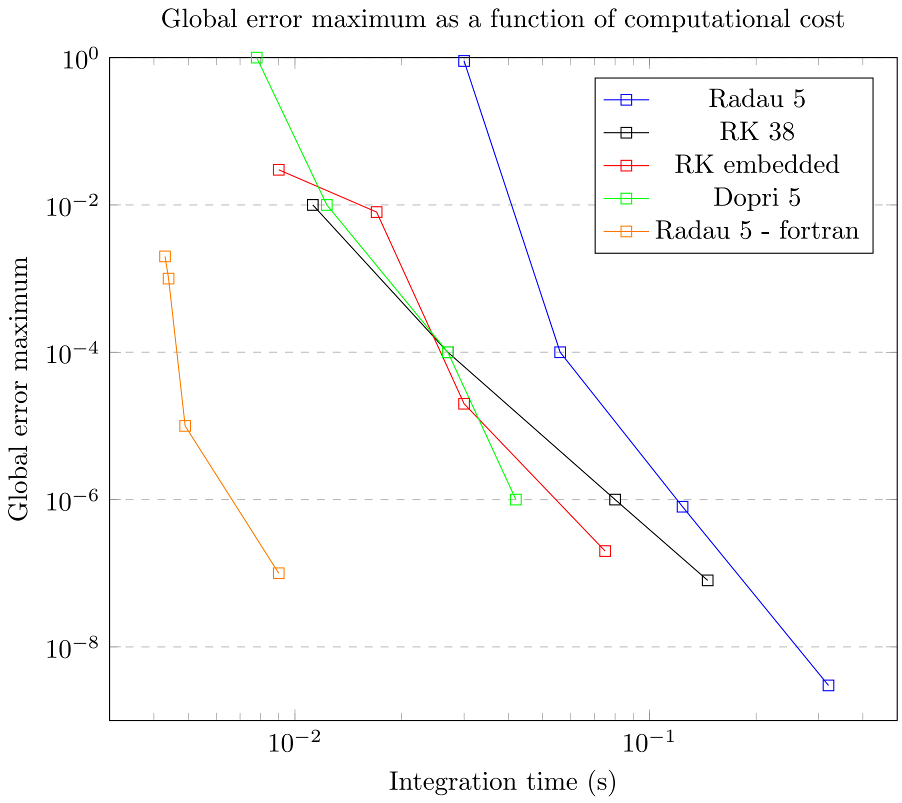

Figure 2: The computational cost that was chosen here is the integration time as it is easier, even if less reliable, to compare different schemes. A better way would be to average the integration time over a N simulations ($N = 10$ for example) which was not done here however. We see is that there is no big difference between *Dopri 5*, *RK 38* (explicit scheme with fixed time step) and *RK embedded*. *Radau*5, the only implicit scheme here, is only better in the case of a *fortran*-based code. In the *python*-based code, the *Radau 5* scheme performs poorly relatively to the other one.

Remark : the fortran-based Radau5 scheme is, for the same tolerance, much less accurate. It's also much faster. This can indicate it was coded (very) differently that the python version.

0.2.1.3

Considering only the number of function evaluations, one can see that for a global error of $7e-7$ and 2206 function evaluations for the **Radau 5** scheme, the other schemes reaches the same accuracy with : 7196, 2917 and 1766 function evaluations (RK38, RK-embedded and Dopri 5 respectively). This is no representative of the integration time as the implicit scheme requires computing the jacobian and some LU decompositions too (here : 51 computations of jacobian and 228 of LU decompositions).

The implicit schemes requires a lot of extra-work to compute the result making it generally speaking slower for the same accuracy (as long as there is no optimization that is performed as using a *fortran*-based code for example).

0.2.1.4

The main advantage of implicit shemes lies in its A-stability. Consequently, a very stiff system that could not be integrated using a explicit scheme should be integrated using a implicit one. However, for the systems that allow the use of explicit schemes (meaning : there never are eigenvalues that are out of the A-stability domain for the chosen sheme), explicit schemes are much better because they allow a faster integration for an equivalent accuracy.

0.2.2 Integration of van der Pol oscillator model

0.2.2.1

In order the illustrate the use of implicit schemes, we are going to set $\epsilon = 20$. This way, we have a very stiff system.

tolerance	number of time steps
1e-2	205
1e-4	438
1e-6	1088
1e-8	3303

Table 2: Time steps required before converging as a function of tolerance, for Radau 5 and the Vanderpol oscillator.

0.2.2.2

Generally speaking, the lesser the tolerance, the bigger the time step and thus the more likely the explicit schemes will be unstable. The vanderpol system, with $\epsilon = 20$ is very stiff. That means we expect the explicit schemes with non-fixed time step to begin to take more time as to not diverge even for low-tolerance. This is what is observe in 3.

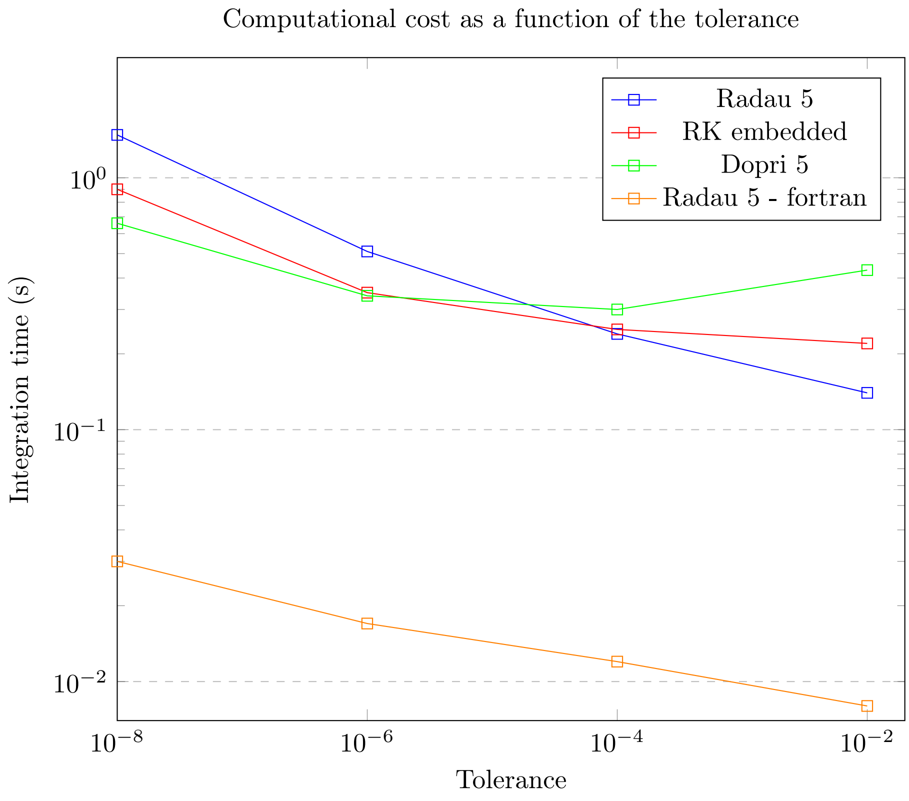

Figure 3: This graph illustrates how the implicit schemes are more likely to start diverging for lower tolerance as they near the negative boundary of their A-stability domain. Generally speaking, and if we ignore the *Radau 5 - fortran* based scheme, the explicit schemes are slighly quicker for lower tolerance. However, it does not yet allow to compare the accuracy between each scheme.

The figure 4 illustrate the accuracy for each scheme as a function of the integration time.

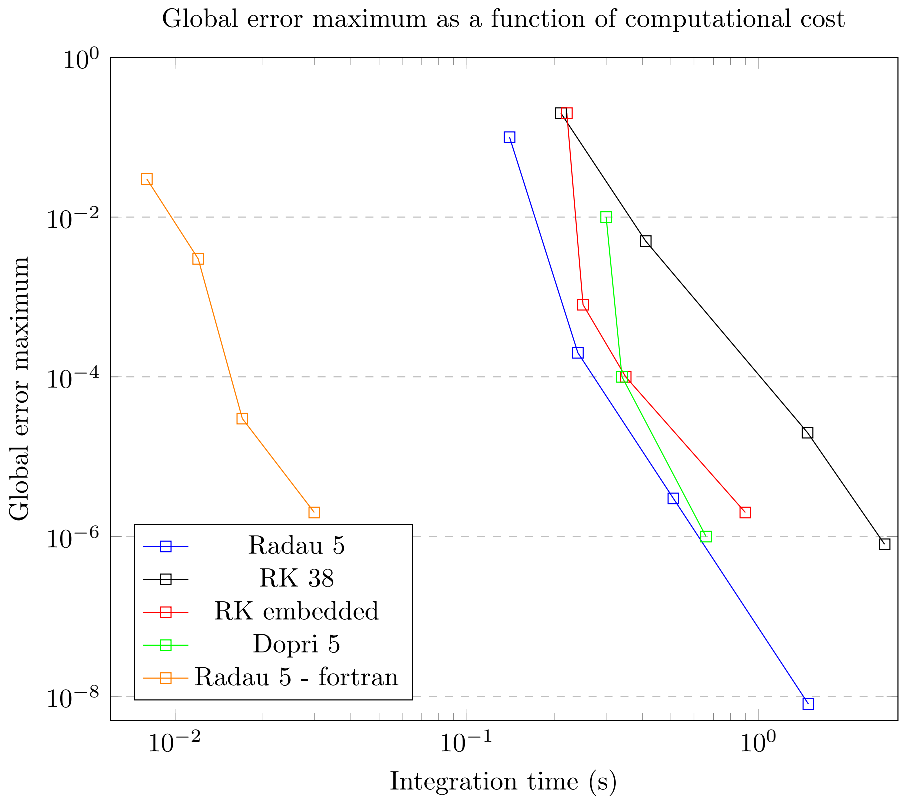

Figure 4: Contrary to the brusselator, the Vanderpol system is very stiff for $\epsilon = 20$. Consequently, the explicit schemes are expected to requires much more computations, especially for very low tolerances. This is what we see here. *Radau 5* scheme is better that the other explicit schemes, the worse being the *RK38* fixed time step scheme.

0.2.2.3

TODO: peut être plot accuracy en fonction de la tolérance ? Ou simplement commenter sur l'explosion de l'erreur pour explicit plutôt et pas pour implicit.

0.2.2.4

To achieve a very accurate integration, one needs for a very stiff system, a implicit scheme, as explicit-ones requires much more steps to achieve similar results and it becomes simply unfeasible to use one.

When there is no need for a very accurate integration however, a explicit scheme can be enough. It should required more step, but it does not have to do the extra work (jacobian evaluations and LU decompositions) as a implicit scheme needs to do.

0.3 Operator splitting techniques for stiff ODEs

0.3.1 Singular pertubation analysis and various forms of the system involving two operators

0.3.1.1

With the given notations :

$$d_\tau U = \begin{cases} y2 - y1 \\ \frac{qy_3 - y_3y_2 + y_2(1-y_2)}{\epsilon} \\ \frac{-qy_3 - y_3y_2 + fy_1}{\mu} \end{cases} = A_1(U) + \frac{1}{\mu}B_1(U)$$

with :

$$\begin{cases} A_1(U) & = (y2 - y1, \frac{qy_3 - y_3y_2 + y_2(1-y_2)}{\epsilon}, 0)^t \\ B_1(U) & = (0, 0, -qy_3 - y_3y_2 + fy_1)^t \end{cases}$$

In this equation and considering that $\mu << \epsilon$, B_1 holds the bigger part of the stiffness : the more $\mu << 1$, the stiffer the system is.

0.3.1.2

For $\mu \to 0$, we have $-q\bar{y}_3 - \bar{y}_3\bar{y}_2 + f\bar{y}_1 \approx 0$. Consequently, we can write : $\bar{y}_3 = \frac{f\bar{y}_1}{q+\bar{y}_2}$.

Then we can replace $\bar{y}_3$ by its value in the other equations yielding (9), (8) remaining as is.

The figures 5 and 6 illustrate the results for two different values of μ.

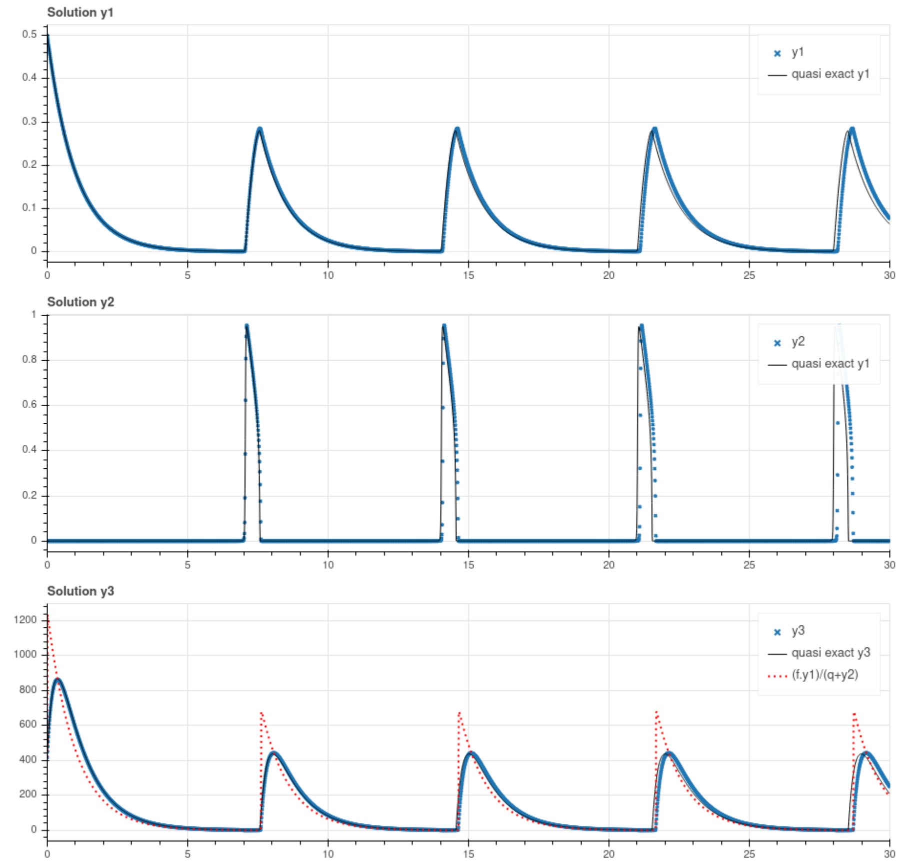

Figure 5: Solution of the system using first form with Lie splitting and with $\mu = 10^{-4}$. We can see that the integration of the system using the first form of Lie splitting is slower than the true solution (meaning the period is slighlty bigger than what is should be). At the same time, y_3 can not follow very well the sharp increase that occurs every 7 seconds or so in the system.

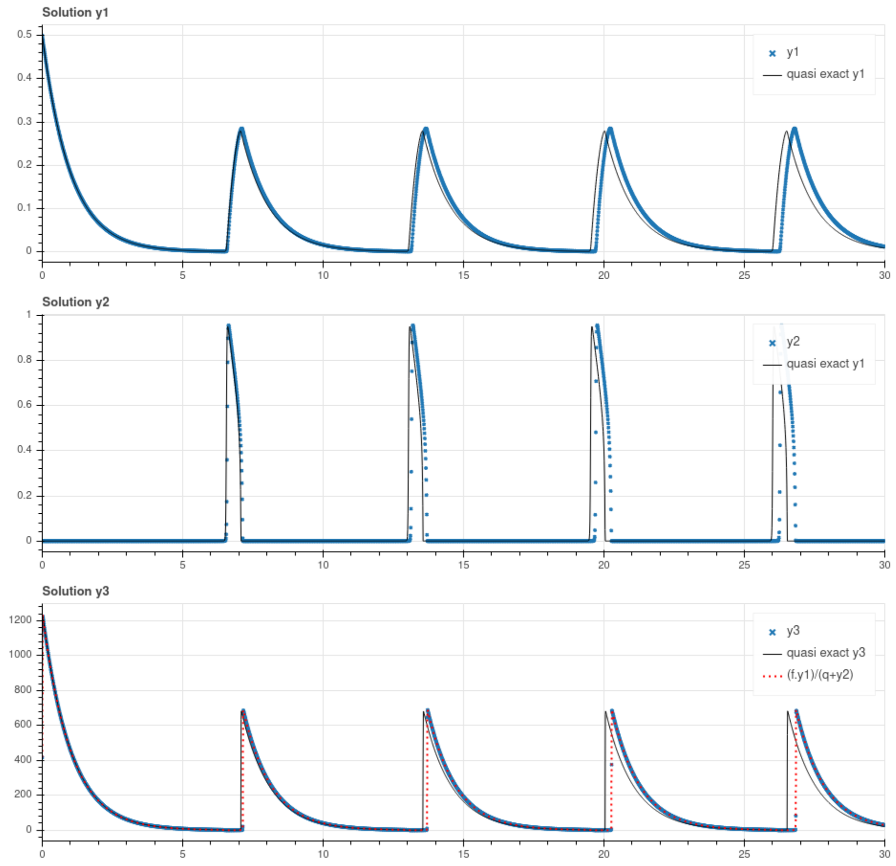

Figure 6: Solution of the system using first form with Lie splitting and with $\mu = 10^{-6}$. Contrary to the previous solution for $\mu = 10^{-4}$, we have an even bigger period and thus the current solution is slower than the previous one which was already slower than the true solution. At the same time, and if we ignore the period issue, $\bar{y_3}$ is much more accurate and follows pretty well the sharp increase.

The approximation becomes better as μ nears 0 and the shape of the solution is better. However, the period increases with μ nearing 0, meaning the slow dynamics is diverging from the full dynamics.

If we want to have a slow-dynamic that has a period close to the one of the full dynamic, it is better to choose $\mu = 10^{-4}$ even though the period-issue remains (it at least seems to lower).

This period related issue is due to the fact that when we compute y_1 and y_2, we consider that y_3 is fixed and then we update y_3. Theses splitted calculations introduce an error visible here on the period. This is the splitting error.

0.3.1.3

We write : $d_\tau y_2 = \frac{1}{\epsilon}\big((q - y_2)\frac{fy_1}{q+y_2} + y_2(1-y_2)\big) + \frac{1}{\epsilon}((q-y_2)(y_3 - \frac{fy_1}{q+y2})$

Which yields : $d_\tau U = A_2(U) + B_2(U)$ with :

$$\begin{cases} A_1(U) & = (y2 - y1, \frac{1}{\epsilon}\big((q-y_2)\frac{fy_1}{q+y_2} + y_2(1-y_2)\big)^t \\ B_1(U) & = (0, \frac{1}{\epsilon}((q-y_2)(y_3 - \frac{fy_1}{q+y2}), -qy_3 - y_3y_2 + fy_1)^t \end{cases}$$

The figures 7 and 8 illustrate the results for two different values of μ.

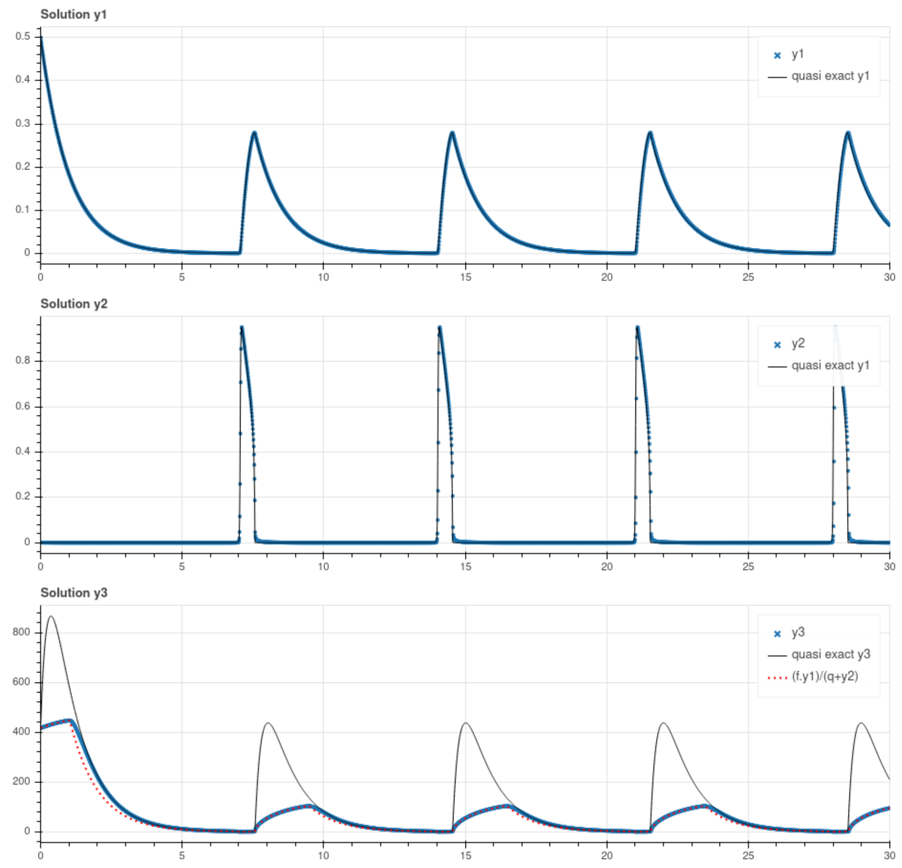

Figure 7: Solution of the system using second form with Lie splitting and with $\mu = 10^{-4}$.

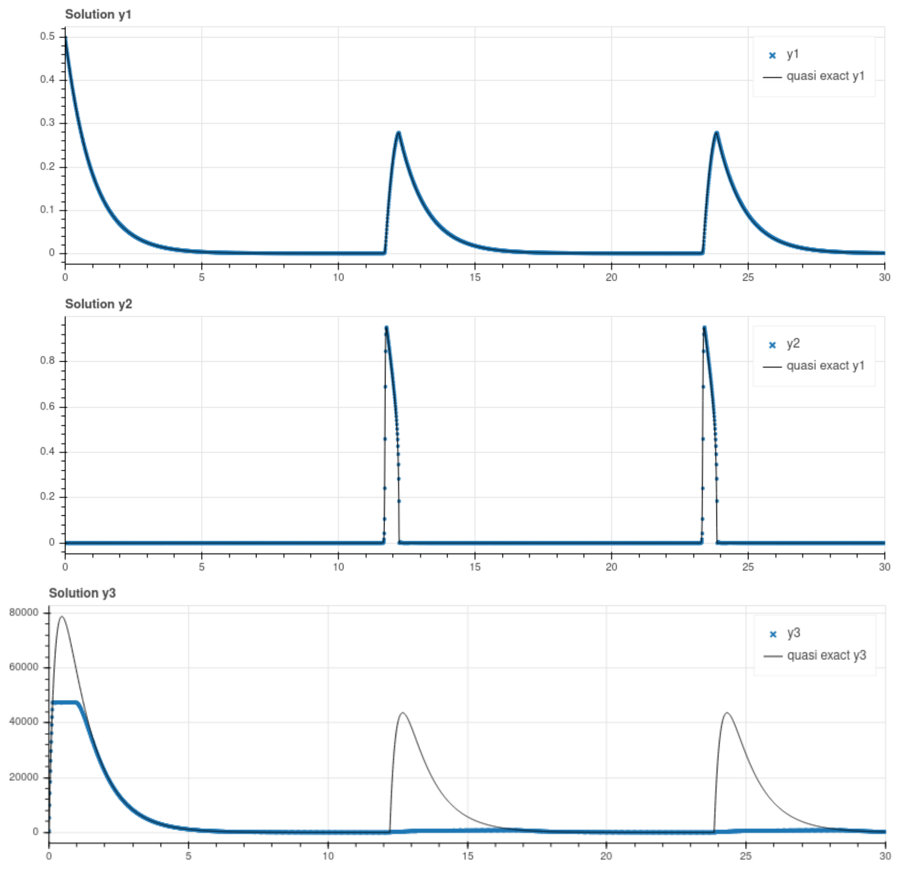

Figure 8: Solution of the system using second form with Lie splitting and with $\mu = 10^{-6}$.

The solution y_1 and y_2 almost seem perfect for both μ (it is much better than for the first splitting). However, y_3 is much worse than before : it does not follow at all the full dynamics. Note that $\frac{(fy_1)}{q+y_2}$ was not plotted for $\mu = 10^{-6}$ as it was exploding.

This second Lie splitting seems more adequate in the case where μ is far from 0.

0.4 An ODE system derived from the semi-discretization in space of the heat equation

0.4.1 Properties of the original system of PDEs and of the discretized one

0.4.1.1

Using (13) and $u(t,x) = \sum_{m=0}^{\infty} a_m(t)cos(m\frac{x+a}{2a}\pi)$, we can get the first derivative along t and the second derivative along x and use the Neumann conditions and then identify in the basis composed of the $cos(m\frac{x+a}{2a}\pi)$, the factors to get :

$$d_t a_m + \frac{m^2\pi^2}{4a^2}a_m(t)$$

which yield immediately to : $a_m(t) = a^0{}_m e^{-\frac{m^2\pi^2}{4a^2}t}$.

For small times and big m, we can have a system that has sharp transition has the exponential will decrease very fast.

0.4.1.2

This is a constant-recursive sequence of order 2.

For $\lambda \in]-4, 0[$, we have $\Delta = (2+\lambda)^2 - 4 < 0$ thus yielding the following solution to $(e) r^2 - (2+\lambda)r + 1 = 0$:

$$r_{1/2} = \frac{2 + b \pm i\sqrt{4-(2+\lambda)^2}}{2}$$

Which yields : $|r_{1/2}| = 1 \Rightarrow r_{1/2} = e^{\pm i\theta}$.

And thus : $v_n = \alpha cos(n_\lambda \theta) + \beta sin(n_\lambda \theta)$, which, since $v_1 = v_{-1}$ yields $\beta = 0$.

And thus $v_n = \alpha cos(n_\lambda \theta)$ and since $\alpha \neq 0$ and $v_{N-1} = v_{N+1}$ yields : $N\theta = 0[\pi]$.

Consequently, $\theta_m = \frac{m\pi}{N}[\pi]$ for $m = 0, ..., N$.

Which yields : $\lambda_m = -2(1 - cos(\theta_m)) = -4 sin^2(\frac{m\pi}{2N})$.

And since $(\Delta x)^2 A_N$ is the matrix corresponding to v_n (and $\Delta x = \frac{2a}{N}$) , the spectrum of A_N is given by : $\lambda_m = -\frac{N^2}{a^2} sin^2(\frac{m\pi}{2N}), m = 0, ..., N$.

0.4.1.3

The stiffness of (15) is linked to the eigen-values of A_N which we just found. These eigen-values depends on N and thus the stiffness of (15) depends on N. Indeed, when $N >> \frac{m\pi}{2}$, then $\lambda_m \approx -\frac{m^2\pi^2}{4a^2}$ which can be very negative thus introducing a very stiff dynamic for the system. This is also the eigenvalues of the continuous Laplacian from question 0.4.1.1.

0.4.2 Integration of the dynamics for a given space discretization

0.4.2.1

From lesson 5, we know that the Rock4 scheme is better suited to our problem. Indeed, contrary to RK4, which has a A-stability domain which becomes bigger for imaginary number, the A-stability domain of Rock4 allows for much negative real eigen-values.

Rock4 also uses a adaptative step which, along with the previous remark, makes it less computationally extensive : 375 for Rock4 vs 8306 for RK45 for the number of time the function is evaluated. The accuracy being roughly the same.

0.4.2.2

In both case, we start with a initial time step of roughly 10^{-6} but very quickly we reach 10^{-2}. The firt initial condition makes the time step converge more quickly toward the final time step.

Refining spatial discretizationn, meaning augmenting N, yields to lower time step. We saw previously that the bigger the N, the bigger the eigenvalues of the matrix, and thus the stiffer the system. Consequently, it makes sense that we need lower time step to allow for the same local error.