MAP551 - PC 4

Paul Calot

October 30, 2020

0.1 Introduction

0.2 Stability diagrams: graphical representations and their use

0.2.1

Figure 1 shows the stability diagram of Runge Kutta schemes of order 1, 2, 3 and 4. Generally speaking, the stability domain size increases with the order of the scheme. For the higher-order scheme, the real part of z being positive means that the numerical scheme will seem to converge whereas the system will diverge for such z (for example for a solution which form is : $e^{\lambda t}$).

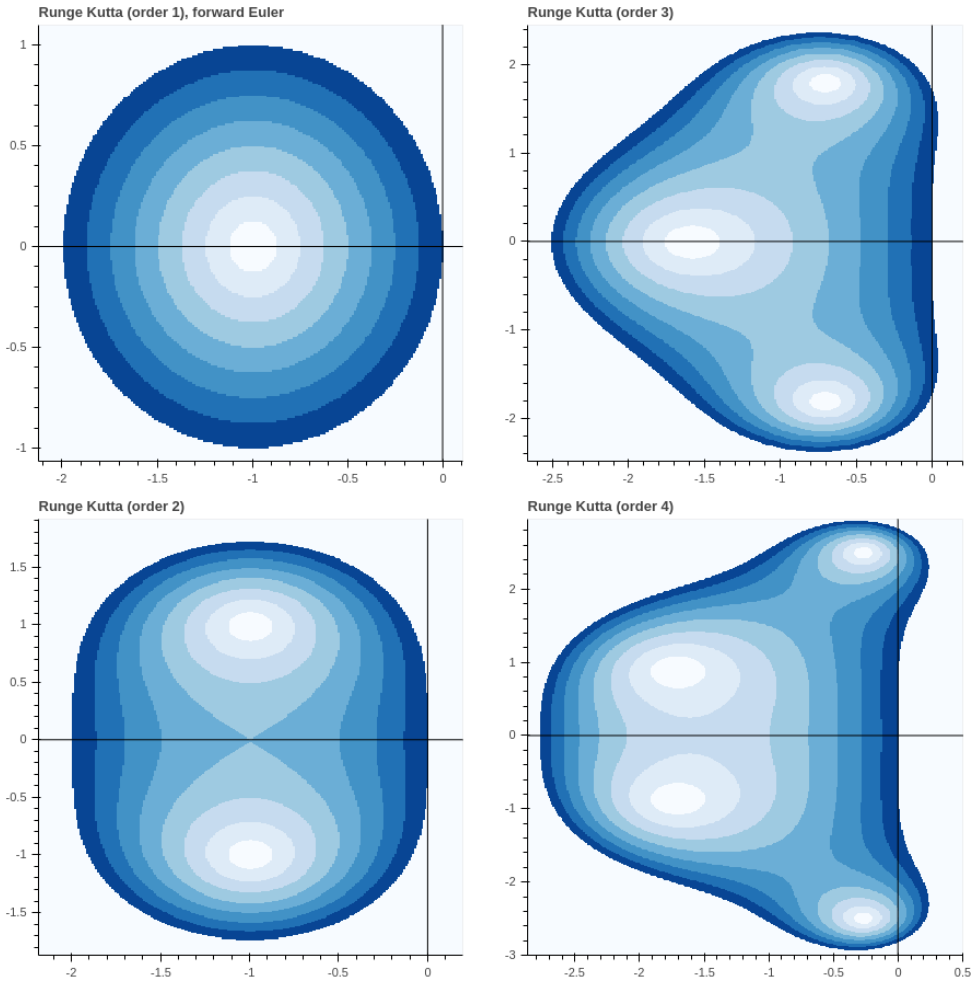

Figure 1: Stability diagrams for Runge Kutta schemes of order : 1, 2, 3 and 4 (left to right, top to bottom). The stability domain increases with the order. However, stability to some eigen-values with positive real part occurs for order 3 and 4 schemes. This means that while the system diverges for such eigen values, the scheme will remain stable and seem to converge.

0.2.2

Choosing a time step close to the boundary but still in the stability domain of the scheme meant having oscillations for the Curtiss and Hirschfelder model. An example is given in figure 2.

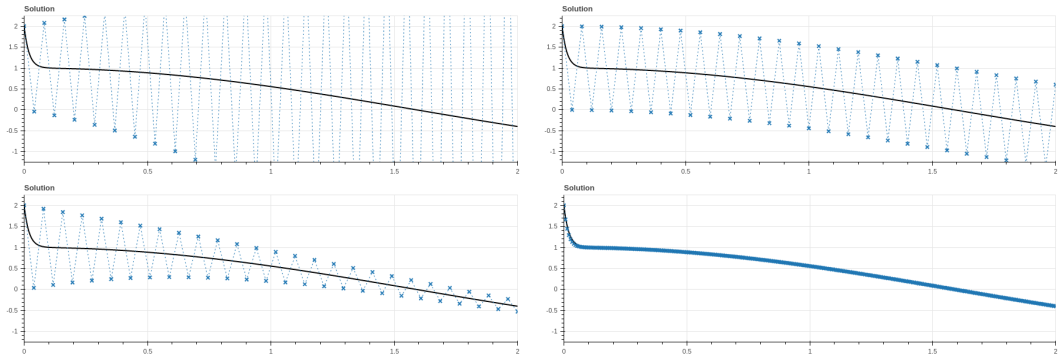

Figure 2: Examples of the Curtiss and Hirschfelder model for $k = 50$, $T = 2$ sec and $n_t = 50, 51, 52, 300$ (left to right, top to bottom). The first example shows a diverging scheme, n_t was choosen too little. The second example, with $n_t = 51$, seems like the scheme is right on its stability boundary, as oscillations are neither decayed nor amplified. The two following ones show the scheme bahavior close to the boundary ($n_t = 52$, lots of oscillations) and far from it ($n_t = 300$, no oscillations).

Stablity and accuracy are two distincts notions (as figure 2 illustrates). One stable scheme may be far from the solution even if it should converge for $t >> 1$. A unstable scheme will never be accurate however.

0.2.3

The figure 3 shows the stability domain of the DOPRI methods.

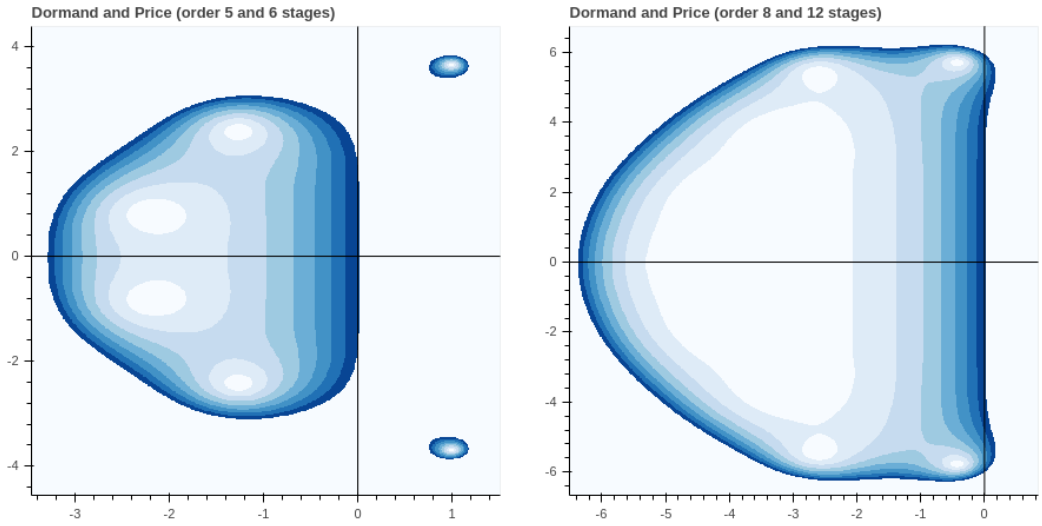

Figure 3: In comparison with the RK schemes, these schemes allow for bigger negative eigen values real part (-3.2 and -6.2 *vs* -2.8 at best previously). DOPRI schemes also cover a bigger imaginary axis than the RK schemes. However, one should be careful not to have system eigen-values z with $\Re(z) > 0$ that falls in the stability domain.

TODO : Resolution of the system ??

0.3 High order RK methods for BZ reaction dynamics integration

0.3.1 Stiffness ?

0.3.1.1

The evolution of the two eigen values on the interval $[0, 40]$ is shown in figure 4. We see that λ_2 reaches very negative eigen-values real part every 8 seconds or so. This characterised a stiff system.

And right before this stiffness, λ_1 goes up to 13 which means that y_1 should increase very quickly.

That is indeed what we see on the quasi-exact numerical solution obtained with *scipy* given in figure 5

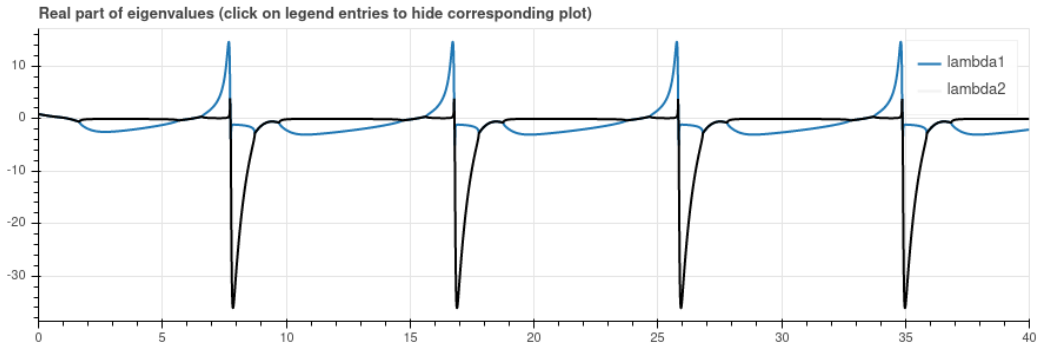

Figure 4: Characterisation of stiffness : read part of eigen-values are given at each time step. Stiffness is characterized by high negative real part of eigen-values and is visible in this system around time $8, 17, 26$ and 35 seconds which goes down to roughly -33.

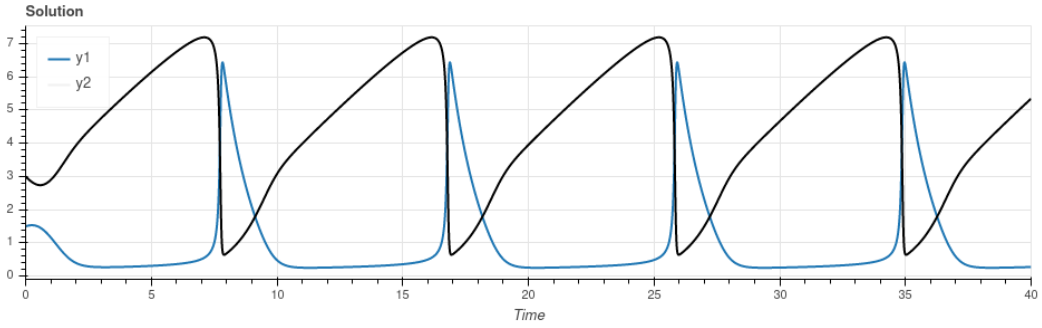

Figure 5: Quasi-exact numerical solution of the system. The system state changes very abruptly at the same times (roughly $8, 1, 26$ and 35 seconds) as the large eigen-values real parts appear (either negative for y_2 or positive for y_2).

0.3.1.2

Changing the initial conditions yields two remarks :

1. Starting close to the limit cycle leads to roughly the same cycle straight away. This is the previous case.

2. Starting "far" from the limit cycle create a very huge stiffness in the system early on. The system converges toward the limit cycle nonetheless and the previous stiffness can be observed again. The figure 6 illustrates this case with an initial condition equals to $(y_1(0), y_2(0)) = (10, 10)$.

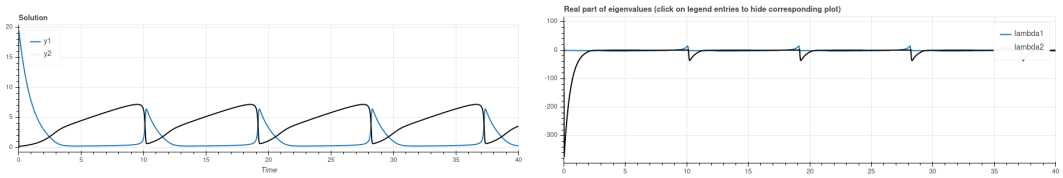

Figure 6: Numerical quasi-exact solution (left) and eigen values of the system (left) for a initial condition that is "far" from the limite cycle : $(y_1(0), y_2(0)) = (10, 10)$. The initial stiffness is much bigger that the one observed in the limit cycle. Note that starting with an even further initial condition, for example $(y_1(0), y_2(0)) = (100, 100)$ yields to an even bigger stiffness but still converges towards the limit cycle.

0.3.2 Integration of the system using high order RK methods with fixed time steps.

0.3.2.1 May be use some pictures ?

Absolute stability for explicit schemes can be reach by increasing the time step. In such case, we close on the absolute stability limite. When the time step becomes too large to garuantee the absolute stability, oscillations and then pure divergence can be observed on the numerical solution.

The absolute stability criterion is valid only for the contracting dynamic parts of the system which is supposed to yield to some kind of convergence that will not be observed for time steps that are too big. For those parts, small time steps are required. However, there is no need for such small time steps in the slower evolving part of the system.

0.3.2.2

Aller voir le cours pour une formule théorique de la stabilité absolue selon l'ordre des RK.

0.3.2.3

The bigger the order of the scheme, the better the absolute stability. Graphs from figure 1 illustrates it. As for the accuracy, the definition of an order of magnitude p means that $\epsilon_n = O(\Delta)t^{p+1}$ where ϵ_n is the error at time n, that is valid for Δt sufficiently small. In such conditions, the bigger the order, the better the accuracy by definition. For bigger Δt, the constant K in $\epsilon_n = K\Delta t^{p+1}$ has its importance to compute the accuracy. The schemes of higher order generally requires more execution of the function f and consequently will be more time consuming that small order schemes. However, increasing the order generally have for consequence a smaller number of time steps required to have the same accuracy result as for the smaller-order schemes.

0.3.2.4

The figure 7 shows for *RK1* the global error.

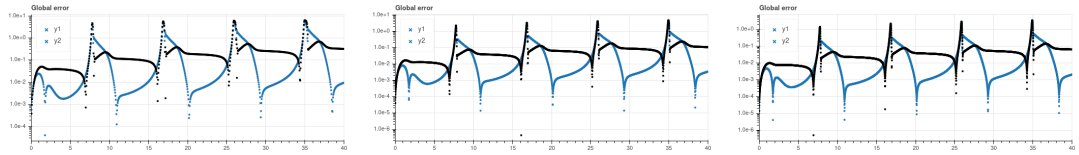

Figure 7: Plot of the global error for *RK1* for several number of points : 1000, 3000 and 5000. Decreasing the time step diminishes the global error. Indeed, if we look at the max of the global error for y_2 at $t \approx 7$ secs, we roughly have : 5, 2 and 1 for 1000, 3000 and 5000 respectively. This figure shows that this local maxima of the global error increases slightly with time and for all number of time steps. This maxima is obtained at the same time at the stiffness in the system increases. The global error is bigger for quick dynamic variations.

This accuracy is however not related to the absolute stability of the system. Indeed, for all the chosen number of time steps, the absolute stability of the system is verified. Both accuracy and absolute stability depends (at least partly) on the stiffness of the system.

Note that what has been said for *RK1* extends to *RK2, RK3 and RK4* for which the global error is even lower with the order becoming bigger.

0.3.2.5

As shown previously, errors don't build up that much.

Indeed, the limit cycle is attracting the trajectories that are close enough. Consequently, the errors are always contracted towards the limit cycle.

Interestingly enough, the numerical schemes seem slower than the real solution having for consequences to give the right behavior but at the wrong times. The global error can thus be much larger because of a phase between the real solution and the numerical solution. Figure ..TODO.. illustrates that.

0.3.2.6

For positive eigen-values real parts, meaning when the system is supposed to explode, the implicit method does not manage to reproduce the behavior. Indeed, for the DONNER LE NOM ICI system : $Y_{n+1} = Y_n + \Delta t k Y_{n+1} \Rightarrow Y_{n+1} = \frac{1}{1-k\Delta t} Y_n$.

If Δt is big enough (it depends on the system), then : $\frac{1}{1-k\Delta t} << 1$ whereas k is positive. Consequently, the sheme is going to converge whereas the system is temporarily exploding. This is especially problematic as the numerical solution can seem to converge and one can mistaken this behavior for the true behavior of the system.

Here, and as shown in figure 8 the system loses the dynamic of the system and can't reproduce the abrupt increase of the solution.

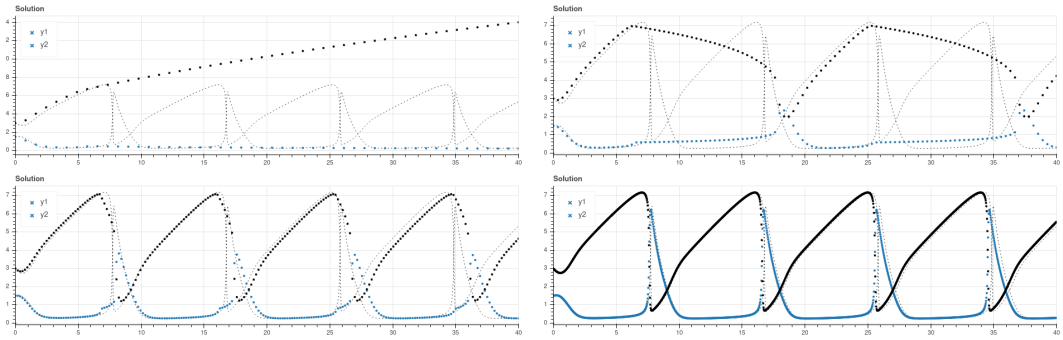

Figure 8: Implicit numerical solution obtained for various increasing n_t values : 50,110,200 and 1500. Note how the obtained y_2 can not follow the abrupt increase of the system for lower n_t. For $n_t = 200$, it never reaches the maximum value and for lower n_t values, the scheme does not manage to reproduce the right behavior of the system.

0.3.3 Integration of the system using adaptative time stepping

0.3.3.1

The stiffness of the system is linked to the time step as seen previously. Adapting the time step to have at each time n, the required Δt_n to have the stability of the scheme is therefore directly linked to the stiffness of the system for explicit schemes.

The figure 9 illustrates and clearly shows the time step value decreasing when stiffness increases.

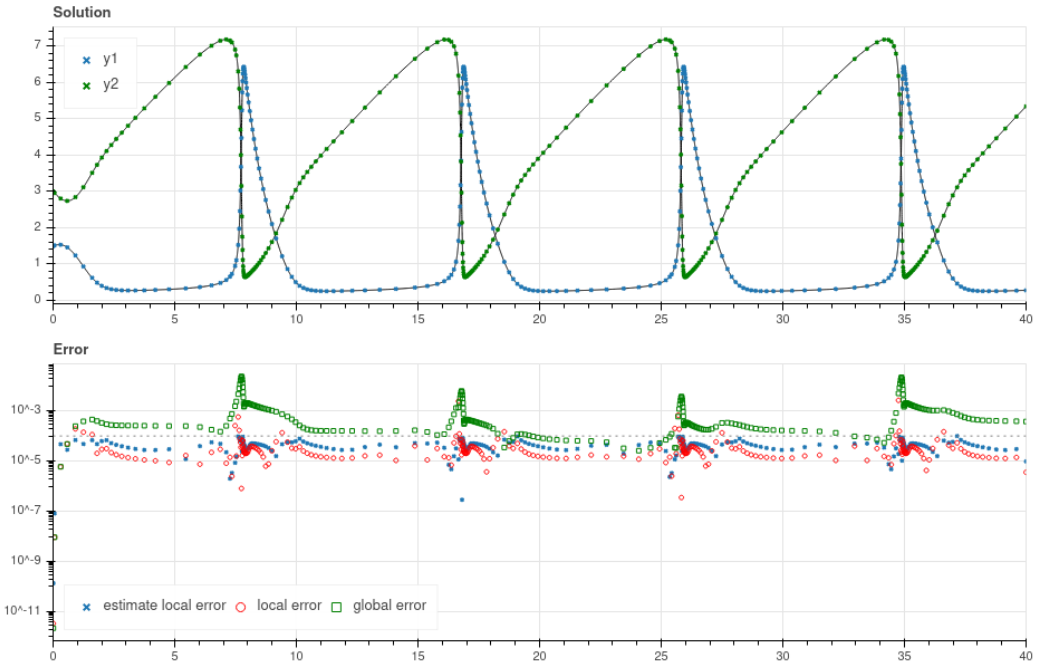

Figure 9: Numerical solution obtained with an embedded Runge-Kutta scheme with a tolerance of 1^{-4}. It shows the relation between time-stepping and the stiffness of the system. The number of points is increasing with the stiffness increasing. This numerical solution has been computed using 257 time steps and 1293 function evaluations for a global error that is below 0.05 as shown on the bottom figure. The meaning of the different errors is described later on.

The number of time step required for $RK38$ and for such an accuracy is much higher as illustrated in figure 10.

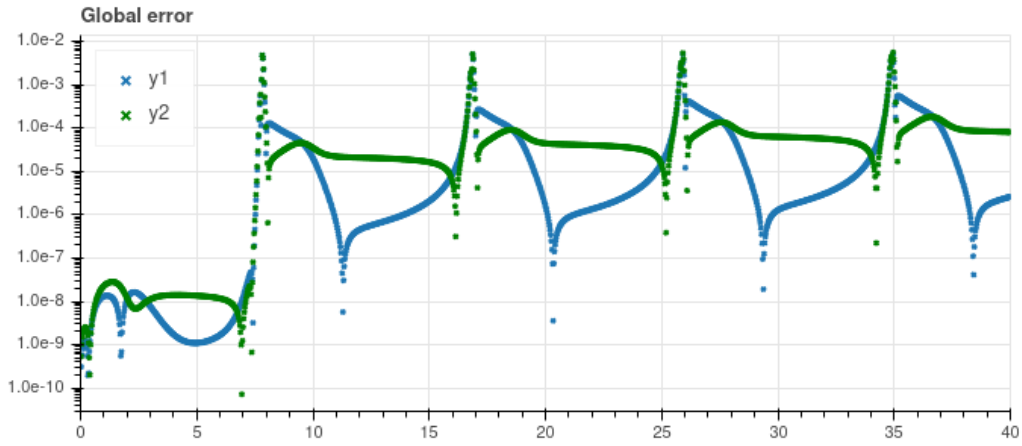

Figure 10: Error obtained with the fourth-order Runge-Kutta scheme based on the 3/8 rule and with 1500 points. It displays an error that is slightly higher than in figure 9 and with almost 6 times more time steps used.

Note that while the extrema of the global error for the embedded sheme are relatively close (roughly a factor 100), the extrema for the RK38 scheme are much more separated (roughly a factor 10^{-7}). This was to be expected as for the first method, the time step is adapted to the requirements whereas it's not in the second case and thus allow for a very low error for the slower-evolving part of the system.

0.3.3.2

The number of time steps is not the only issue. Even though using adaptative time stepping allows for less time steps for the same accuracy, the number of times f is computed at each time step is higher when we are looking for the best time steps and are not finding it immediatly. Indeed, before finding the right time step, several can be rejected as shown in figure 11.

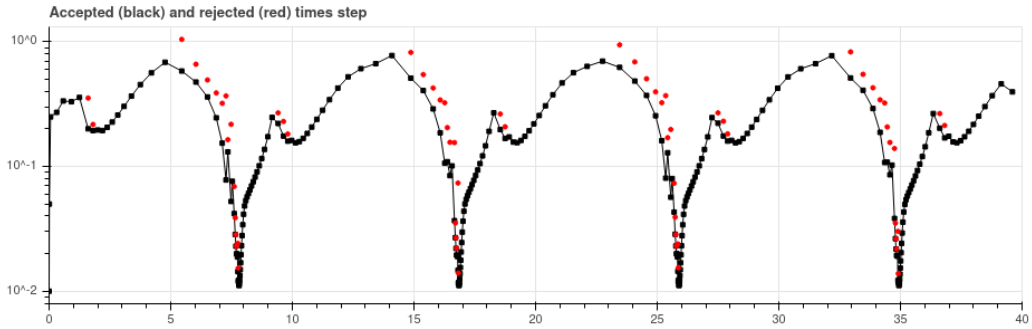

Figure 11: Rejected time steps as a function of time for a tolerance of $1e-4$. We see that a fast-evolving dynamic of the system requires re-computing the correct time steps several times as the density of red dots increases. We also see that for slower dynamic, the time step can increase up to roughly 0.8 while for a fast evolving dynamic, the time step reaches 10^{-2}.

Therefore, to evaluate the actual cost of the scheme, it is required to count the number of time the function f has been evaluated accross all time.

The figure 12 shows how the embedded method evaluates less the function to achieve better accuracy.

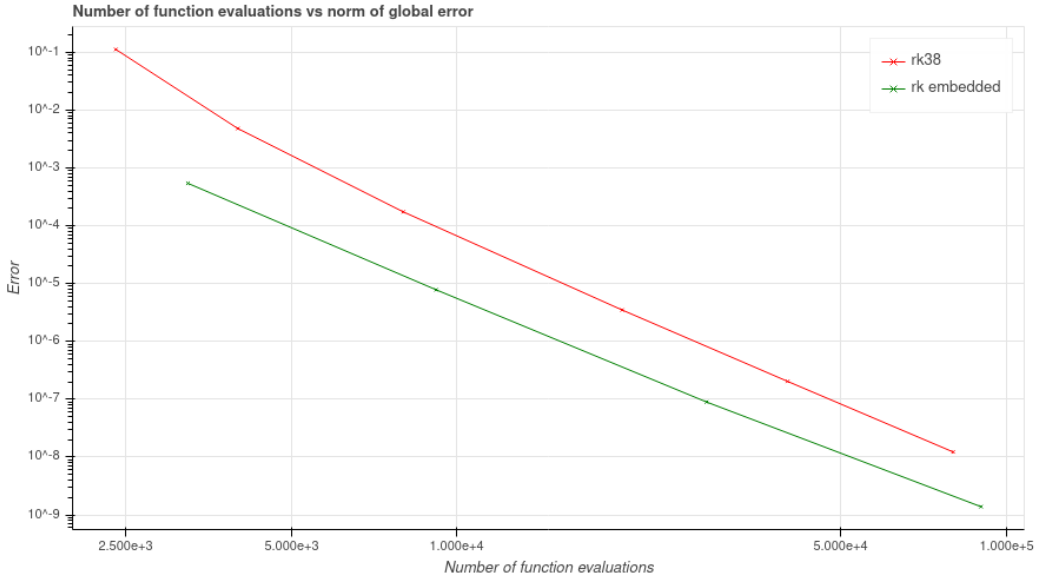

Figure 12: Comparison of the number of time steps as a function of the global error for the embedded RK scheme and the classical RK4. In average, there appear to be a little bit more than a factor ten between the two methods.

0.3.3.3

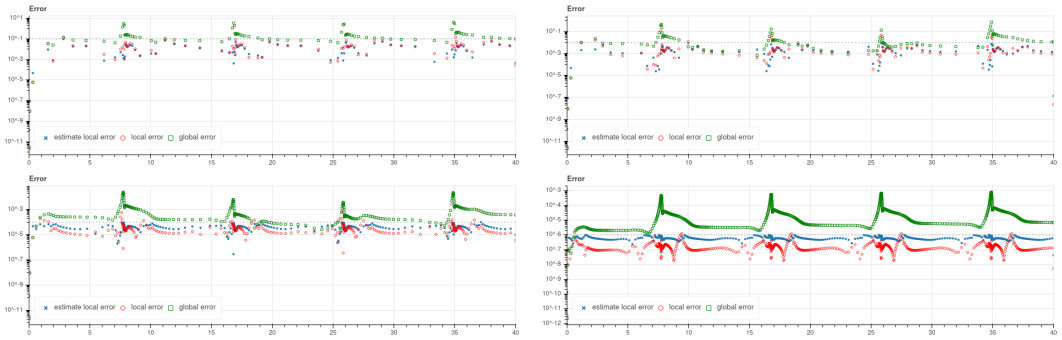

Figure 13: Errors are plotted for four different tolerances (from left to right, top to bottom) : 10^{-1}, 10^{-2}, 10^{-4} and 10^{-6}. The number of function evaluations required are respectively : 505 ,621, 1293 and 3245. On each of these graphs, we see that the "exact" local error, which is computed by comparison to the quasi-exact *scipy RK45* scheme, is generally very close to the local error estimate made by comparing to the higher-order scheme used in the adaptive time stepping method. The only exception is for the tolerance 10^{-6} where the local estimate is close to 10 times higher than the "exact" local error.

0.3.3.4

tolerance ($\approx$ max local error)	# of steps - fixed	# of steps - adaptive	computational cost - adaptive
10^{-1}	2400	88	505
10^{-2}	3800	111	621
10^{-3}	6000	168	897
10^{-4}	9500	257	1293
10^{-5}	15000	427	2077
10^{-6}	24000	733	3245

Table 1: In this table are reported the number of steps and computational cost (number of time f has been evaluated) required for the completion of the scheme with the given tolerance and for the embedded RK scheme. It shows that decreasing the tolerance of a factor 10 increases the number of steps and computational cost "roughly" of the same factor which is roughly 1.5 (once again it depends on the tolerance, this factor seems to increase with the tolerance decreasing). For the fixed time step methods, and if we associate the tolerance to the maximum local error allowed, we know that for a fourth-order RK38 method, decreasing time step of a factor 10, meaning increasing the number of steps of a factor 10, decreases the local error of a factor 10^5 which is roughly the same as for the embedded RK scheme. Note that the given values for the fixed time step method are not the exact one (there were read using figure 12). Figure 12 shows that the evolution is roughly the same for the two methods.

0.3.4 Integration of the system using Dormand and Price method

0.3.4.1

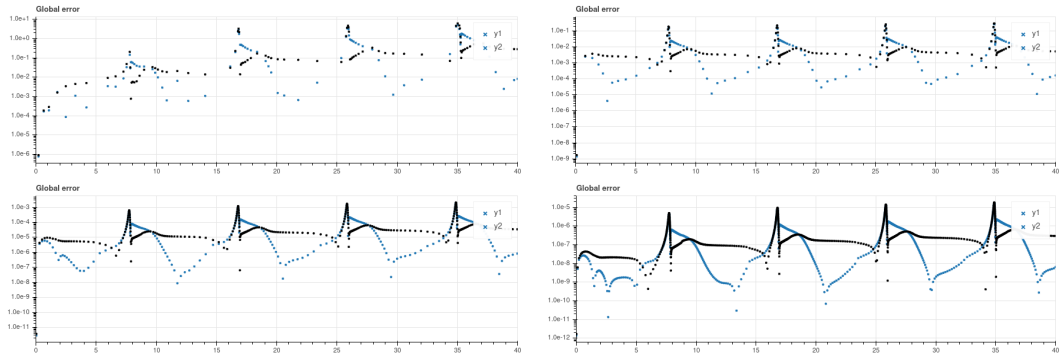

Figure 14: Global errors for the Dormand and Price method with various tolerance (left to right, top to bottom) : 10^{-2}, 10^{-4}, 10^{-6} and 10^{-8} Time steps need for each tolerance are : 90, 149, 322 and 748. Compared to the previous adaptive RK scheme, this is much better. We have, for the same number of time steps needed, a tolerance that is 10 to a 100 times better (cf. 1). As for the number of function evaluations, we have : 698, 1160, 2384 and 4952 which is better for the lower tolerances only (e.g. $\neq 10^{-4}$) than the previous method.

0.3.4.2

The impact of using a higher order and more precise method is that it allows, when a very low tolerance is required, a quicker simulation (all things being equal for both methods).

0.3.4.3

0.4 The van der Pol oscillator : a discriminating test-case

0.4.1 Stiffness ?

0.4.1.1

The figure 15 shows the computed solution for several ϵ.

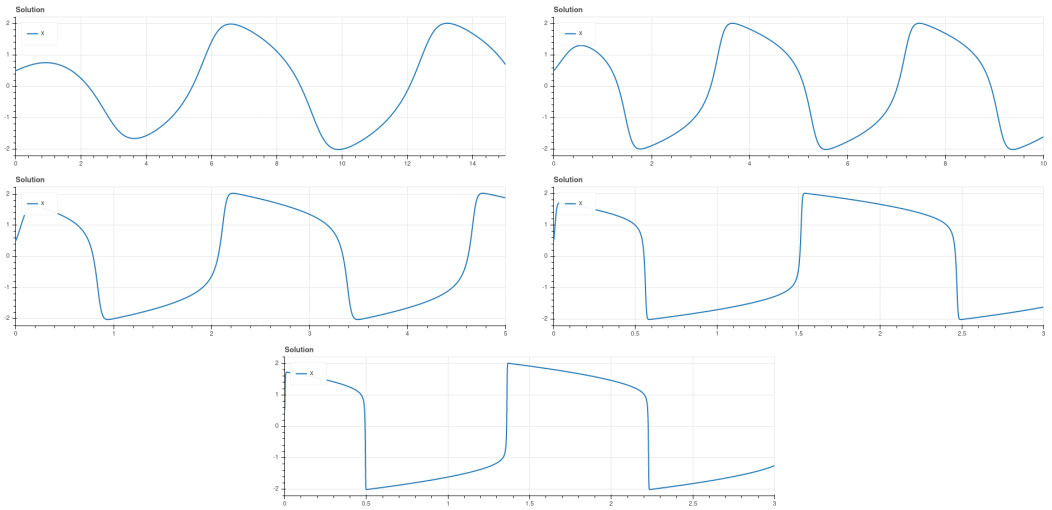

Figure 15: From left to right, top to bottom, are displayed the numerical solution for various ϵ and $y_{ini} = (0.5, 0)$: 1, 2, 4, 10 and 20. We see that doubling ϵ roughly divide by two the time required to reach the limit cycle.

The initial behavior is not stiffer than the rest of the dynamic as some very fast evolution can be seen later on and with all ϵ. The system studied here is composed of so-called "slow system" and "fast-system". To find those systems, several variables changes are required. Anyway, Fenichel's theory tells us that the system is well-approximated by merging the fast and the slow system together. This tells us that the behavior of the resulting system is composed of a slow evolving part for a given (x, y) domain and a fast evolving part for another (x, y) domain, hence the stiffness.

0.4.1.2

The figure 16 gives the quasi-exact solution for the brusselator compared to the numerical ones for the fast-slow system and for various ϵ : 1, 10 and 20.

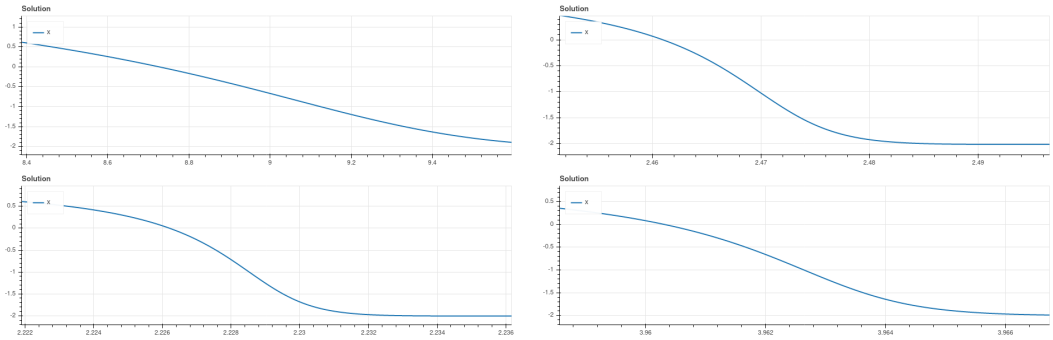

Figure 16: The figure 16 gives the quasi-exact solution for the brusselator compared to the numerical ones for the fast-slow system and for various ϵ : 1, 10 and 20. When comparing x and y_2, we see that the max gradient is roughly 20 for the