Application of Multirate Partitioned Runge–Kutta method in neuron modeling

Aleksei Iupinov

Introduction

Modeling of biological systems resolves to modeling of multitude of processes running on various temporal and spatial scales; sometimes models are fixed at one scale for simplicity. If we are to model a system on several time scales, we have to use appropriate numerical method for every subsystem, and make sure the results are coupled among them. When dealing with such problems, a need for partitioned numerical methods arises.
One such problem is simulation of interaction between chemical and electric signals in neurons: injected current triggers release of calcium, that in turn takes the chemical system from one steady state to another. Numerical methods have to model robustly such a tightly coupled biological process.

Problem

The previous work on the subject has been done by Jerker Nilsson in his master thesis work for KTH on multiscale co-simulation in neuroscience[1]. In his work, he implemented the corresponding differential equations system in Matlab with coupled electrical and chemical subsystems, as well as implemented and compared several combinations of numerical solvers (the Runge–Kutta 4th order (RK4) method for the chemical subsystem, Hines method for the electric subsystem (fixed step size); BDF for both subsystems (with adaptive and fixed step sizes)).
The scope of this project was to implement the multirate partioned Runge–Kutta method of the 4th order (MPRK) in Matlab, and to evaluate its applicability to this specific simulation. The method is described in the article by M. Günther, A. Kværnø, P. Rentrop[2]. The expected result would be that it provides a robust alternative solver, able to compete with integrated Matlab solvers (ode15s/ode23s).

Method

The assumption of the MPRK method is that in the state vector of the equation system, all components are strictly divided into latent and active parts:

$$\dot{y}_L(t)=f_L(y_A,y_L,t) \quad y_L(t_0)=y_{L,0}$$
$$\dot{y}_A(t)=f_A(y_A,y_L,t) \quad y_A(t_0)=y_{A,0}$$

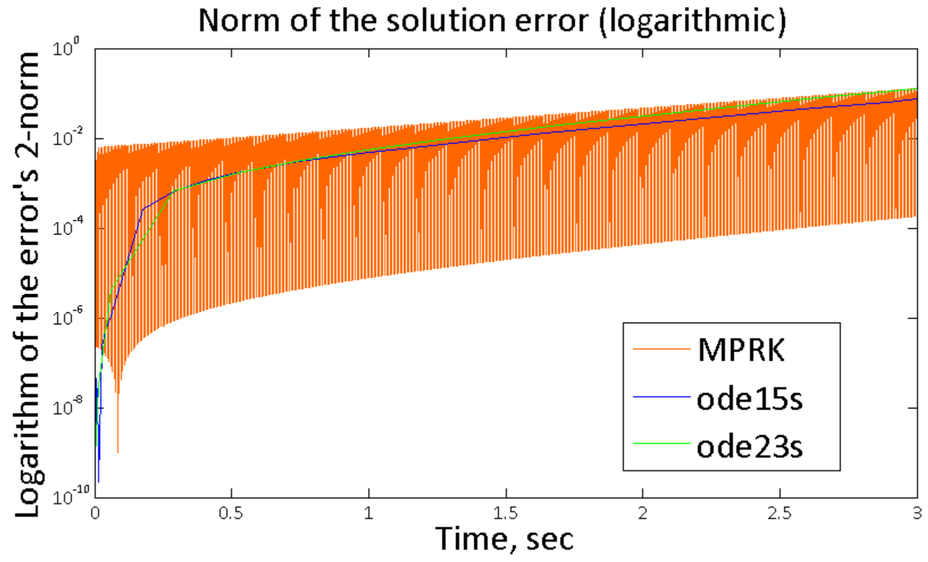
As the MPRK method assumes that stiffness is contained in the latent part (f_L) only, it uses implicit RK4 method to solve the latent part, and explicit RK4 with much smaller step size for the active part. At the end of each latent step state vector is synchronized. In our case, for the purposes of the method, electric system is assumed to be strictly latent and stiff, while the chemical one is active.
The embedded Runge–Kutta methods of the 3rd order are used for error estimation. Based on local errors, sizes of active and latent steps are adjusted at the end of each latent step. Step control, while not specifically mentioned in the article[2], was described in another article[3] by Peter Rentrop:

$$h_{new}=h_{old}\cdot 0.9\cdot\left(\frac{absolute\ tolerance}{local\ error}\right)^{\frac{1}{4}}$$

The resulting step size change coefficients are kept within certain ranges [0.5; 1.5] to avoid “zigzagging” behavior.

Results

The method has been implemented and tested with a simple weakly coupled stiff problem (1 latent and 1 active component) at first. The method's accuracy is comparable with the standard stiff ODE solvers:



However, the adequate performance has not been achieved. The method has been compared with standard solvers by number of steps (and corresponding numbers of function calls):

Solver stats	ode15s	ode23s	MPRK
Time steps	87	62	635 latent 85620 active
Function evaluations	182	317	3185 latent 341040 active
Time elapsed, sec	0.026	0.019	18.549

Even with the sample problem the MPRK method takes about 10-20 times more macro-steps than ode15s/ode23s. What is worse, the amount of active part function computations and time elapsed is larger by 3 orders of magnitude.
Following with the subject problem, which has 18 latent and 22 active components, the method is also significantly slower in the same way. While ode15s takes about 40 seconds to model the 5 second time span corresponding to the current injection, the MPRK predicted computation time was around 25 hours, and the method actually did not finish as it ran out of memory.

Conclusion

The result is negative: the current implementation of the MPRK method is not fit for modeling electrochemical processes in neurons due to low performance, which might stem from the stability of this specialized method (developed with the goal of simulating latent electric circuits).
One of the reasons for the low performance might be that in the model the numbers of subsystems' components are similar (22 and 18), while the assumption by authors[2] is that the number of active components is significantly smaller than the number of latent components.
A possibility of improving the performance lies within the reassigning mechanism, in which values of local errors are used to dynamically re-label components as latent or active during the step size control. However, the architecture of the system being studied was not flexible enough for such undertaking.
Parallelization of the computation of active steps within every latent step is also feasible; it was not attempted, as it is likely to affect performance significantly only when the method runs on a massively parallel system.

References

[1] Jerker Nillson. Investigation and implementations of efficient algorithms for multiscale co-simulation in neuroscience. Master of Science Thesis, Royal Institute of Technology, Stockholm, Sweden, 2014
[2] M. Günther, A. Kværnø, P. Rentrop. Multirate Partitioned Runge-Kutta Methods. BIT Numerical Mathematics, Volume 41, Issue 3, pp 504-514, 2001
[3] Peter Rentrop. Partitioned Runge-Kutta methods with stiffness detection and stepsize control. Numerische Mathematik, Volume 47, Issue 4, pp 545-564, 1985