21st International Symposium on
Mathematical Theory of Networks and Systems
July 7-11, 2014. Groningen, The Netherlands

Reducing run-times of excitable cell models by replacing computationally expensive functions with splines

Michael Clerx and Pieter Collins

***Abstract*—Numerical simulation of muscle cells and tissue is an established tool in cardiac electrophysiology, where the electrical behavior of excitable heart muscle cells is commonly modeled as a stiff, non-linear system of ordinary differential equations. A common feature of this system's right-hand side is the heavy use of computationally expensive univariate functions of the membrane potential. In this article, we investigate the performance benefits of replacing these functions with cubic spline approximations in an automated model simplification process. Clear performance gains were found when evaluating the right-hand side in isolation and when performing multi-cellular simulations using a simple forward Euler method. Single cell simulations run with an adaptive method saw smaller gains due to a higher overhead from the solver. A parallel multi-cellular simulation was also investigated, but the overhead of the implementation overshadowed the evaluation time of the right-hand side.**

***Index Terms*—Biological Systems; Numerical and Symbolic; Systems Biology**

I. Introduction

Numerical simulations based on models of ion channels and transporters are an established tool in cellular cardiac electrophysiology [1, 2]. This field, which studies the electrical behavior of heart muscle cells (myocytes) and its relation to muscle contraction, can deepen our knowledge of the fundamental processes underlying our heart beats and provides insights into life-threatening conditions of arrhythmia. The main components of these models: ion concentrations, channels, transporters and fluxes can be measured in isolation and described using ordinary differential equations (ODEs). Combining these components into cell models allows the effects of molecular differences to be investigated on a cellular level. For example, the effects of genetic mutations on channel proteins can be measured in vitro, leading to an updated ion channel model that can then be incorporated into existing models of the cell. By coupling adjacent cells through a current invoked by the difference in their potentials, multi-cellular models of muscle fibers or patches of tissue can be created.

Michael Clerx is with the Department of Knowledge Engineering and the Cardiovascular Research Institute Maastricht, Maastricht University, The Netherlands. michael.clerx@maastrichtuniversity.nl

Pieter Collins is with the Department of Knowledge Engineering, Maastricht University, The Netherlands.
pieter.collins@maastrichtuniversity.nl

Since their introduction in the last half of the 20th century, ODE models of myocytes have become increasingly complex. From a computational efficiency point of view modern models contain many expensive-to-evaluate functions, especially univariate functions of the membrane potential V making heavy use of the exponential function [3, 4, 5, 6]. As more is learned about the complexity of the mammalian heart, the number of equations used to model a single cell is growing. This conflicts with the need to run simulations fast and the desire to create personalized models of the whole human heart, which contains roughly $5 \cdot 10^9$ of such cells [7].

In this paper we investigated the performance gains to be had from replacing computationally expensive parts of the right-hand-side (RHS) of the differential equations with easier-to-evaluate approximations. Numerical experiments were run to show the effects of simplifying with cubic spline approximations on the RHS evaluation times. We examined the influence of using a simplified RHS in three scenarios: single cell simulations using an advanced numerical solver, multi-cell simulations without parallelization, and multi-cell simulations run in parallel on a graphics processing unit (GPU).

To the best of our knowledge, no similar attempts have been reported. Although Mirin et al. [8] report using rational function approximates for the same purpose as part of a larger effort to simplify the model by ten Tusscher et al. [9], our approach differs in two ways: Firstly, in our approach we use splines, which are more stable and flexible when approximating functions with strong localized variations and may evaluate faster, and secondly, instead of simplifying one specific model, we have described and implemented a generic approach.

II. Preliminary considerations

A cell model's RHS can be broken down into two parts:

$$\begin{aligned} \frac{dV}{dt} &= -\frac{1}{C}\left[I_{\text{ion}}\left(V, \mathbf{u}\right) + I_{\text{stim}}\right] \\ \frac{d\mathbf{u}}{dt} &= f\left(V, \mathbf{u}\right) \end{aligned}$$

(see also [10]). Here, V is the cell's membrane potential, C is the membrane capacitance and the vector $\mathbf{u}$ is a collection of channel protein states, ion concentrations, and sometimes other variables such as fractions of bound enzymes. From

ISBN: 978-90-367-6321-9

these variables a set of ionic currents is calculated as $I_i = g_i(V, \mathbf{u})(V - E_i)$ where E_i is an equilibrium potential for the ion type and $g_i(V, \mathbf{u})$ is a model for the ion channel's *conductance*[1]. An external stimulus I_{stim} is periodically applied to drive the system. It should be noted that $f(V, \mathbf{u})$ is non-linear and typically stiff. In electrophysiological terms, it contains slow and fast currents. In all models, the *fast sodium current* rises and falls in a time span of < 5ms, whereas other currents, for example the *slow delayed rectifier potassium current* take several seconds to reach their peak. For a classic example of a cardiac myocyte model and the accompanying equations see [11] or [12], a good example of the complexity of modern models is found in the detailed appendix to [3]. An example of a reduced cell model is given in this paper as Table I. These equations describe the Morris-Lecar model [13], a reduced form of the original neuronal model by Hodgkin and Huxley [14] which contains only two state variables. In this example, the external input forcing the system is given as the dimensionless value $pacing$.

Table I
THE REDUCED MORRIS-LECAR MODEL

$$\frac{dV}{dt} = -\frac{1}{C}\left[I_{ion} + I_{\text{stim}}\right] \qquad \frac{dw}{dt} = 0.04\,\frac{w_{\text{inf}} - w}{w_{\text{tau}}}$$
$$C = 20 \qquad I_{Ca} = g_{Ca}(V - E_{Ca})$$
$$I_{ion} = I_{\text{K}} + I_{\text{Ca}} + I_{\text{leak}} \qquad E_{Ca} = 120$$
$$I_K = g_K(V - E_K) \qquad g_{Ca} = 4.4 m_{\text{inf}}$$
$$E_K = -84 \qquad m_{\text{inf}} = \tfrac{1}{2}\left[1 + \tanh\left(\tfrac{V+1.2}{18}\right)\right]$$
$$g_K = 8w \qquad I_{\text{leak}} = 2(V + 60)$$
$$w_{\text{inf}} = \tfrac{1}{2}\left[1 + \tanh\left(\tfrac{V-2}{30}\right)\right] \qquad I_{\text{stim}} = -80 \times pacing$$
$$w_{\text{tau}} = 1/\left[\cosh\left(\tfrac{V-2}{60}\right)\right]$$
$$V(t=0) = -60.86 \qquad W(t=0) = 0.015$$

An important feature of these models is that they contain many computationally expensive functions depending only on the membrane potential V. We automatically identify these functions and replace them by spline approximations that are faster to evaluate. Most of these functions are found in ion channel models, which are a phenomenological class of models so that no valuable information is lost by altering the form of the expression. Furthermore, the precision of cell models is limited, and, as we illustrate in the next two sections, the stability of the ODE integrator takes precedence over the accuracy of the evaluation of the RHS.

A. Single cell simulations

For detailed modern single-cell models the RHS is a costly function to evaluate and the system of equations is stiff. As a result, higher order adaptive schemes that require multiple RHS evaluations or suffer from reduced stability fail to produce any performance benefits. Implicit methods offer greater stability and therefore bigger step sizes, but the need to approximate the next solution often involves more RHS evaluations which can counteract this benefit[2]. As a result, the explicit forward Euler method has remained a competitive choice. A difficult to implement, but much faster approach is to use an implicit adaptive multi-step method, e.g. CVODE [15]. The multi-step approach means only a limited number of RHS evaluations per time step are required; in most cases a single evaluation per step is sufficient (see Figure 2, part D). Myocyte models are typically *paced* (forced) periodically with a block wave stimulus, which introduces two discontinuities per cycle. At these points, it is necessary to reset the simulation routine, leading to a higher number of steps around those points.

B. Multi-cell simulations

Single cell models can be used for multi-cellular simulations by simply duplicating the single cell state vector. Without any interaction between the cells, the system's Jacobian takes on a diagonal block structure. Connections between the cells are then introduced in the form of a diffusion current. For each cell i:

$$I_{\text{diff},i} = \sum_j g_{ij}(V_i - V_j)$$

Where the sum is over all neighboring cells and the cell-to-cell conduction g_{ij} is assumed constant. This diffusion current connects all membrane potentials to that of their neighbors, but leaves the term $\frac{d\mathbf{u}}{dt} = f(V, \mathbf{u})$ untouched.[3] The system now becomes:

$$\begin{aligned}\frac{dV_i}{dt} &= -\frac{1}{C}\left[I_{\text{ion}}(V_i, \mathbf{u}_i) + I_{\text{stim},i} + I_{\text{diff},i}\right] \\ \frac{d\mathbf{u}_i}{dt} &= f(V_i, \mathbf{u}_i)\end{aligned}$$

The most common strategy for running multi-cell simulations is the explicit forward Euler method. There are a number of reasons for this. First of all, in a multi-cell simulation, the evaluation time of the duplicated RHS becomes extremely high, so that methods requiring multiple evaluations all suffer from poor performance. Secondly, in the single cell case the system's fast dynamics are highly localized in time (about 5ms per 1000ms for a heart rate of 60bpm) and occur at the onset of excitation. In a multi-cell scenario each cell's excitation triggers its neighbors' excitation with a small delay, causing the fast dynamics (and the need for a small step size) to be spread out over a much larger time. In extreme cases (which are often of high scientific interest) some part of the tissue is undergoing rapid changes at any given time. This limits the utility of adaptive methods. Finally, when performing parallelized multi-cellular simulations on GPU devices the cost of memory access and synchronization is such that a simple forward Euler scheme is

[1] In fact, this is Ohm's law using conductance $g = 1/R$ instead of resistance R.

[2] The Jacobian of a cell model is typically not available analytically. For some good examples see [3, 4, 5].

[3] This is a simplification, as the diffusion current must be carried by one or more species of ion. Taking this into account, any ionic concentrations in the term $\mathbf{u}$ would also be affected by the I_{diff}.

almost always faster than a more delicate approach requiring the storage of multiple state or derivative vectors.

Garcia et al. [16] report experimenting with different adaptive Runge-Kutta pairs before selecting the simplest tested method: an explicit Euler / Heun pair. They report a 25% speed-up over fixed size forward Euler. Unfortunately, little details are given and we were unable to reproduce this result.

C. Software & model implementations

All simulations and benchmarks were carried out using the Myokit modeling framework.[4] Models used were either implemented following the equations published in source files and appendices or imported from the CellML model repository [17]. Myokit is implemented in Python, but uses code generation and on-the-fly compilation of C-extensions to achieve high performance. Simulations are run in three stages: first a model file is read and parsed into a structured, symbolic representation of the model's equations. This symbolic model is then passed on to a simulation class that uses it to generate, compile and link to a custom simulation object. Finally, the experiment is run and a pointer to the results is passed back to the caller. An advantage of this method is that it allows symbolic manipulation of the model's equations before simulation, without the drawbacks of actually running simulations using the symbolic types.

III. Results

A. Spline fitting procedure

A cubic spline function $g(V)$ was fit to each selected function $f(V)$ on the interval $I = [-100\text{mV}, 150\text{mV}]$. Spline coefficients were set by imposing $\frac{df}{dV} = \frac{dg}{dV}$ at the endpoints of I, setting $g(x_i) = f(x_i)$ at each knot x_i and then solving the resulting linear system. The absolute and relative error in the fit were estimated by sampling both functions at 1000 evenly spaced points on the selected interval and calculating

$$\begin{aligned} e_{\text{abs}} &= \max|f-g| \\ e_{\text{rel}} &= \frac{e_{\text{abs}}}{\max f - \min f} \end{aligned}$$

over the full sample. The tolerance was set to $e_{\text{abs}} \leq 10^{-3}$ and $e_{\text{rel}} \leq 10^{-5}$. Whenever these limits were exceeded an extra knot was added at the point with the highest error. If a set maximum number of pieces (100) was reached the procedure was halted and an error returned, marking the function as unfit for approximation.

To show the performance advantage of spline approximations, cubic spline approximations using 32-piece[5] splines were created for the following functions:

$$\begin{aligned} f_0(V) &= \exp(V/100) \\ f_1(V) &= \exp(0.01V) \\ f_2(V) &= 1/[1+\exp((V+40)/-10)] \\ f_3(V) &= 1/[7\exp((V+12)/35) + 9\exp(-(V+77)/6)] \\ f_4(V) &= 1/[1+\exp(-0.1(V+40))] \\ f_5(V) &= 1/[7\exp(0.03(V+12)) + 9\exp(-0.2(V+77))] \end{aligned}$$

The first function is a minimal example of an $\exp$ function, scaled to the range I. The second function shows the performance benefit of multiplication over division, compared to the run-time of an $\exp$ evaluation. The third and fourth equations are adapted from the fast sodium channel formulation of [4] and are typical examples of the kind of equation we hoped to simplify: The first, f_2 represents the steady-state value of one of the state variables, m, and defines a sigmoid curve. The second, f_3 is the voltage dependent time constant with which m approaches its steady state value and defines a "hat" or "bell" shape over I. The final two equations show the effect of replacing division operations with multiplication in f_2 and f_3.

Table II
Performance gains for single function approximation

Function	T_{original} (ns)	T_{spline} (ns)	$T_{\text{spline}}/T_{\text{original}}$
f_0	25.9	5.08	20%
f_1	23.8	5.07	21%
f_2	27.4	5.10	19%
f_3	53.8	5.07	9%
f_4	24.4	5.10	21%
f_5	48.3	5.10	11%

The performance gains are shown in Table II. As can be seen, the performance of the splines is invariant with respect to the run-time of the original function, leading to large speed-ups for complex expressions. This illustrates the potential advantage of spline approximations in the RHS.

B. Isolating expressions for approximation

A symbolic version of the model's equations was obtained and scanned for univariate functions of V. Care was taken to include functions both directly dependent only on V and directly dependent on other functions themselves a function of only V. For the selected functions, a spline was calculated using the methods described earlier. A piecewise polynomial with the calculated coefficients was then inserted into the symbolic model representation and finally the model was exported to standard C or OpenCL.[6]

To estimate the maximum speed gain from this method, the model's RHS was evaluated either in full or without the selected equations. The ratio between run-times, $F_{\text{null}} = T_{\text{RHS,null}}/T_{\text{RHS}}$, is an estimate of fraction of the RHS evaluation time unaffected by the selected expressions. This estimate serves as a baseline value to compare the

[4]Myokit, or the Maastricht Myocyte Toolkit, is an open-source Python-based software package designed to simplify the use of numerical models in the analysis of cardiac myocytes developed at Maastricht University. Source files and documentation are available from `http://myokit.org`

[5]Splines with 32 pieces provided accurate fits in each of these examples. As can be seen in part B of Figure 1, the evaluation time of a full spline depends only weakly on the number of pieces used and using, for example, a 64-piece spline gives roughly the same results.

[6]OpenCL™ (Open Computing Language) is a standard for parallel programming available from `http://www.khronos.org/opencl/`

actual performance F_{splines} against. The results are shown in Table III.

In this table, N_s and N_e are the number of state variables and equations in each model respectively, while N_r is the number of those equations that could be simplified. F_{null} is an estimate of the best performance and F_{splines} is the actual performance of the RHS evaluation time reduction measured as $F_{\text{splines}} = T_{\text{RHS,splines}}/T_{\text{RHS}}$. The run time of an unoptimized one-second single cell simulation is given as T_0 and the performance using splines is given as $F_0 = T_{0,\text{splines}}/T_0$. The number of RHS evaluations made by the adaptive method in these simulations is given as M for the unaltered RHS and M_{splines} for the version using splines. Finally, $F_1 = T_{1,\text{splines}}/T_1$ represents the speed-up gained in an unparallelized fixed step size cable simulation. Note that the *smaller* the F-value, the *larger* the performance gain using splines. The time taken to perform the spline approximations was not included in any of the benchmarks.

C. Accuracy & performance

Experiments were run to asses the trade-off between spline accuracy and run-time using the reduced Morris-Lecar model for excitable cells (see Table I). The expressions for w_{inf}, w_{tau} and m_{inf} were each selected for simplification: spline approximations were generated for each with an increasing number of segments (from 2 to 200). For each generated spline, a single cell simulation was run using CVODE. All simulations were started from a common set of initial values.[7] Results are shown in Figure 1.

As can be seen, increasing the number of pieces reduces the error of fit almost without affecting the RHS evaluation time, which shows a reduction to about 13% of the evaluation time of non-optimized case. The difference between state variable w with and without the approximation was tracked and seen to stabilize around 10 pieces. Similarly, the number of steps taken by the adaptive solving scheme and the total number of RHS evaluations performed in a simulation reach a stable average at about 5 pieces.

From these results we concluded that (1) Increasing the number of pieces in a spline below the minimum number required to achieve acceptable accuracy doesn't dramatically affect RHS evaluation time. As a result, no special strategy to keep the number of pieces in a spline to a minimum is required. (2) Using spline approximations increases the number of steps taken by the adaptive solver, but the increase does not seem affected by the number of pieces. (3) The accuracy of the produced result increases dramatically as spline quality increases, but then stabilizes once the maximum relative error of fit has gone below 10^{-3}.

We speculate that the unusually high and seemingly random variation in the number of RHS evaluations and integrator steps was caused by the higher order discontinuities at the spline knots. This is briefly discussed in Section III-E.

[7] A similar test was run where models were "pre-paced" for 10^5 periods before the steps taken were measured, but no significant difference with the pre-pacing free case was found.

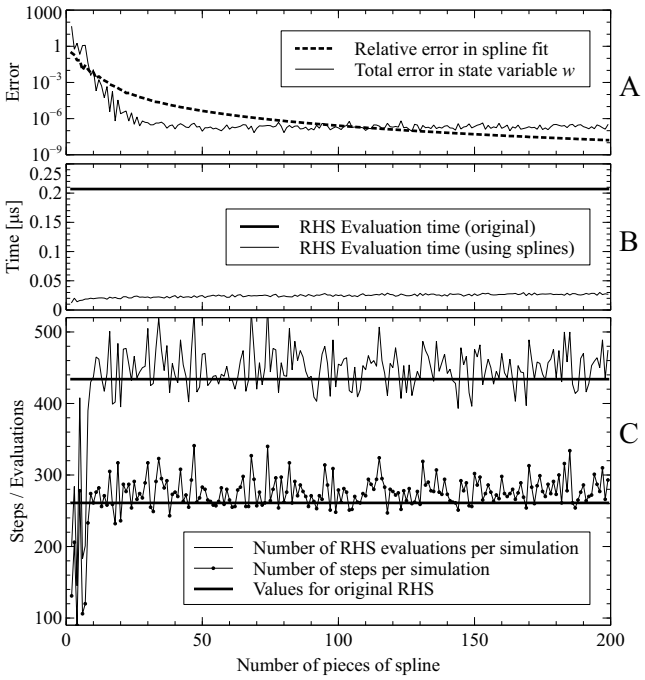

Figure 1. Effects of the number of pieces in a spline. *(A)* The relative error in the spline approximation and the resulting total error in state variable w during a full simulation, calculated as $e_w = \sum_i (w_i - w_{i,\text{Splines}})$ where w_i and $w_{i,\text{Splines}}$ are the values of w at a 1000 linearly spaced points during the simulation. *(B)* The average RHS evaluation time with approximations using an increasing number of pieces and for the unmodified RHS. *(C)* The number of steps taken by an adaptive solver and the number of RHS evaluations performed during a simulation. Thick lines indicate the corresponding values for the unoptimized RHS.

D. RHS Evaluation times

Right-hand-side evaluation times were measured using the following procedure: A simulation was run and the state vector was saved for every position. Next, a benchmarking tool revisited each state multiple times and measured the average execution time per call. The results show that employing spline approximation decreased the RHS evaluation time in each case. However, the utility of function approximation is seen to decrease with increasing model complexity. This can be attributed to two factors: the inclusion of more multivariate equations (for example signaling or intracellular diffusion processes) and the inclusion of more fast-to-evaluate equations (for example Markov formulations of ion channels).

For small models, the performance of the spline-based RHS sometimes exceeds our calculated estimate. This can be attributed to the difficulty of measuring small run times and the many optimizations employed by modern compilers and hardware, which make it difficult to get consistent benchmarking results when comparing small run times.

E. Single cell simulation

Single cell simulations were run using a simulation built on Myokit's `Simulation` class, which creates, compiles and executes C code using CVODE to integrate the ODE. The

Table III
MODEL RUN TIMES AND PERFORMANCE

Model	N_s	N_e	N_r	F_{null}	F_{splines}	T_0	F_0	M	M_{aplines}	F_1
2009, Tran [18]	4	24	11	28%	17%	0.5ms	69%	359	350	-
1991, Luo-Rudy [12]	8	37	17	30%	24%	1.7ms	56%	847	773	25%
2009, Stewart [6]	20	106	38	50%	48%	3.7ms	80%	942	991	51%
2011, O'Hara [4]	41	249	67	53%	64%	12ms	98%	1483	2586	-
2009, Decker [19]	48	204	48	74%	72%	13ms	88%	1646	1594	73%
2010, Sampson [20]	81	299	107	39%	85%	38ms	97%	3277	3209	78%
2011, Heijman [3]	145	514	61	80%	88%	83ms	98%	4165	4071	89%

results show a smaller speed-up than expected on the basis of the RHS performance alone. We initially hypothesized that this may be due to the higher order discontinuities created at the knots, but this would suggest a strong relation between the number of pieces used and the number of steps taken. As can be seen in Table III no such relation was found. Tentative experiments using splines with higher orders of continuity failed to show a consistent performance gain in terms of number of steps. By comparing the number of RHS evaluations taken in simulations with and without spline approximations (given in Table III as M and M_{Splines}) it can be seen that even with a *reduced* number of evaluations the performance gain is not proportional to that in the RHS evaluation times. This suggests overhead of the method is a bigger factor in these simulations and the impact of RHS run-time reduction is reduced.

To gain insight into the adaptive behavior of the CVODE solver, we implemented a tracking mode that logs the number of evaluations taken and the system time at each step of a simulation. Figure 2 shows the results for a 1000ms simulation using the model by Decker et al. [19]. As can be seen from the top half of the figure, most of the work done during a simulation occurs during the action potential (AP), the period when the membrane potential is elevated from its usual resting state of around -80mV. This example, which is typical for ventricular myocyte models, shows how most of the system's fast dynamics are localized around the start and finish of the AP. The lower half of the figure clearly shows why this method is efficient in terms of RHS evaluations: the most common number of evaluations needed at each step is 1 with occasional short burst of around 50 evaluations, leading to an average of 3.1 evaluations per step throughout the simulation. This average ranges from 1.6 to 6.0 evaluations per step for the models described in Table III.

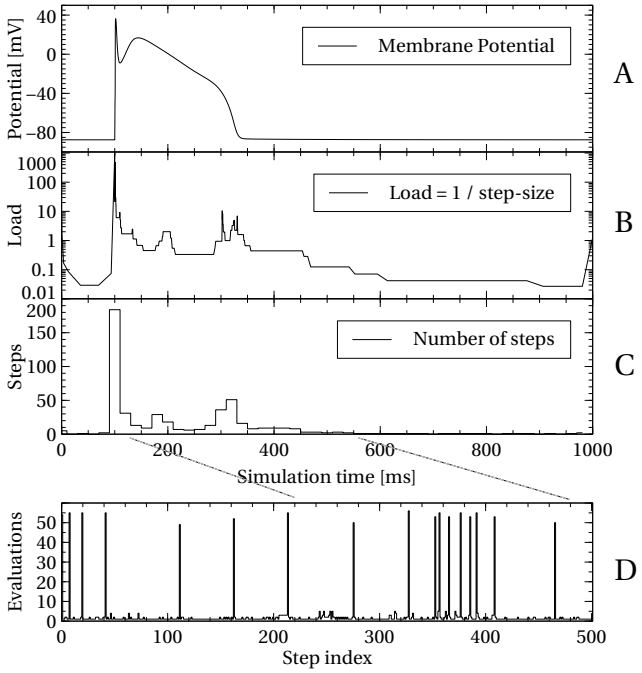

Figure 2. Number of steps and RHS evaluations taken by the CVODE adaptive solver during a 1000ms simulation using Decker et al. (A) The simulated membrane potential plotted against the simulated time. (B) The load on the solver, calculated as the inverse of the used step size at each simulated point in time, shown on a logarithmic axis. (C) A histogram of the number of steps taken by the solver plotted against the simulated time. (D) The number of RHS evaluations taken at each step of the simulation. The full simulation is shown; note that the x-axis does not scale linearly to the axes of the figures above.

F. Multi-cell simulation

Non-parallel multi-cellular simulations were run using Myokit's Simulation1D class, which uses an explicit forward Euler method to integrate the ODE. As described in Section II-B, this is often the best approach for multi-cellular simulations. Without a step-size choosing algorithm or other complicating factors, it can be expected that the performance boost in this scenario matches that seen in single RHS evaluation times. A small number of tests were run which clearly confirmed this notion so non-parallel data was not included in Table III.

An OpenCL-based parallel version of this simulation was implemented, but was unable to produce any speed-ups on a GPU. Indeed, we found that eliminating the simplified calculations from the RHS entirely only lead to marginal speed-ups, indicating that the RHS evaluation time is of little importance in our parallel implementation. We suspect that the overhead of memory access and synchronization in this case is a bottleneck in our calculations. Further work is needed to reduce this overhead before the benefits of reduced RHS evaluation time for parallel cell simulation can be judged.

IV. CONCLUSIONS & FURTHER WORK

Splines are a natural candidate for providing simplified approximations to univariate functions occurring in the right-

hand side of a differential equation model of excitable cells. In this paper, we showed that the use of cubic spline approximations can reduce the time needed to evaluate the RHS of an excitable cell model's ODE. Further, the non-smoothness of the approximation did not appear to negatively affect the number of RHS evaluations needed by the sophisticated implicit multi-step method used by the package CVODE. The time benefit of this optimization was seen directly in unparallelized multi-cell simulations, but failed to materialize fully in single cell simulations using CVODE, and had no effect on parallel multi-cell simulations using the forward Euler method run on the GPU. We suggest that the RHS evaluation time is overshadowed by overhead resulting from the solver in the former case, and from overhead of memory access due to the hardware architecture in the latter.

Given the speedups obtained in the evaluation of the RHS, we believe that this method may still be useful in situations where RHS evaluation time is a dominant factor, either in the field of cellular electrophysiology or in other domains. Simplification of multivariate functions using splines is possible, but significantly more complicated. Further work is needed to optimize our parallel solver to the point that the benefits of RHS simplification become visible. Work is in progress on computing optimal spline approximations of various degrees in the uniform norm.

ACKNOWLEDGEMENTS

The authors would like to thank Dr. Enno de Lange for our valuable discussions on cell models, model implementations and writing for the GPU.

REFERENCES

[1] D. Noble and Y. Rudy, "Models of cardiac ventricular action potentials: iterative interaction between experiment and simulation," *Philosophical Transactions of the Royal Society A*, vol. 359, pp. 1127–1142, June 2001.

[2] N. P. Smith, E. J. Crampin, S. A. Niederer, J. B. Bassingthwaighte, and D. A. Beard, "Computational biology of cardiac myocytes: proposed standards for the physiome.," *The Journal of experimental biology*, vol. 210, pp. 1576–1583, May 2007.

[3] J. Heijman, P. G. Volders, R. L. Westra, and Y. Rudy, "Local control of β-adrenergic stimulation: Effects on ventricular myocyte electrophysiology and Ca(2+)-transient," *Journal of molecular and cellular cardiology*, vol. 50, no. 5, pp. 863–871, 2011.

[4] T. O'Hara, L. Virág, A. Varró, and Y. Rudy, "Simulation of the undiseased human cardiac ventricular action potential: model formulation and experimental validation.," *PLoS computational biology*, vol. 7, pp. e1002061+, May 2011.

[5] A. Mahajan, Y. Shiferaw, D. Sato, A. Baher, R. Olcese, L.-H. Xie, M.-J. Yang, P.-S. Chen, J. G. Restrepo, A. Karma, A. Garfinkel, Z. Qu, and J. N. Weiss, "A rabbit ventricular action potential model replicating cardiac dynamics at rapid heart rates," *Biophysical Journal*, vol. 94, pp. 392–410, Jan. 2008.

[6] P. Stewart, O. V. Aslanidi, D. Noble, P. J. Noble, M. R. Boyett, and H. Zhang, "Mathematical models of the electrical action potential of Purkinje fibre cells.," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 367, pp. 2225–2255, June 2009.

[7] G. Olivetti, G. Giordano, D. Corradi, M. Melissari, C. Lagrasta, Gambert, and P. Anversa, "Gender differences and aging: effects on the human heart," *J Am Coll Cardiol*, vol. 26, pp. 1068–1079, Oct. 1995.

[8] A. A. Mirin, D. F. Richards, J. N. Glosli, E. W. Draeger, B. Chan, J.-l. Fattebert, W. D. Krauss, T. Oppelstrup, J. J. Rice, J. A. Gunnels, *et al.*, "Toward real-time modeling of human heart ventricles at cellular resolution: simulation of drug-induced arrhythmias," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, p. 2, IEEE Computer Society Press, 2012.

[9] K. H. ten Tusscher and A. V. Panfilov, "Alternans and spiral breakup in a human ventricular tissue model," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 291, no. 3, pp. H1088–H1100, 2006.

[10] J. Cooper, G. R. Mirams, and S. A. Niederer, "High-throughput functional curation of cellular electrophysiology models," *Progress in biophysics and molecular biology*, vol. 107, no. 1, pp. 11–20, 2011.

[11] G. W. Beeler and H. Reuter, "Reconstruction of the action potential of ventricular myocardial fibres.," *J Physiol*, vol. 268, pp. 177–210, June 1977.

[12] C. H. Luo and Y. Rudy, "A model of the ventricular cardiac action potential. Depolarization, repolarization, and their interaction.," *Circ Res*, vol. 68, pp. 1501–1526, June 1991.

[13] C. Morris and H. Lecar, "Voltage oscillations in the barnacle giant muscle fiber," *Biophysical journal*, vol. 35, no. 1, pp. 193–213, 1981.

[14] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve.," *The Journal of physiology*, vol. 117, pp. 500–544, Aug. 1952.

[15] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers," *ACM Trans. Math. Softw.*, vol. 31, pp. 363–396, Sept. 2005.

[16] V. M. Garcia, A. Liberos, A. M. Climent, A. Vidal, J. Millet, and A. Gonzalez, "An adaptive step size gpu ode solver for simulating the electric cardiac activity," in *Computing in Cardiology, 2011*, pp. 233–236, IEEE, 2011.

[17] A. A. Cuellar, C. M. Lloyd, P. F. Nielsen, D. P. Bullivant, D. P. Nickerson, and P. J. Hunter, "An Overview of CellML 1.1, a Biological Model Description Language," *SIMULATION*, vol. 79, pp. 740–747, Dec. 2003.

[18] D. X. Tran, D. Sato, A. Yochelis, J. N. Weiss, A. Garfinkel, and Z. Qu, "Bifurcation and chaos in a model of cardiac early afterdepolarizations.," *Phys Rev Lett*, vol. 102, no. 25, p. 258103, 2009.

[19] K. F. Decker, J. Heijman, J. R. Silva, T. J. Hund, and Y. Rudy, "Properties and ionic mechanisms of action potential adaptation, restitution, and accommodation in canine epicardium.," *Am J Physiol Heart Circ Physiol*, vol. 296, pp. H1017–H1026, Apr. 2009.

[20] K. J. Sampson, V. Iyer, A. R. Marks, and R. S. Kass, "A computational model of purkinje fibre single cell electrophysiology: implications for the long qt syndrome," *The Journal of physiology*, vol. 588, no. 14, pp. 2643–2655, 2010.