# Speeding up Cardiac Simulations with Parallel-in-Time Solvers

Maxfield R Comstock[1], Elizabeth M Cherry[1]

[1]School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA, USA

## Abstract

*Cardiac models are an important tool for understanding the causes of arrhythmias and other types of heart disease. Numerical simulations of these models require small time steps to make accurate predictions, leading to long simulation times even with programs parallelized over space. Parallelization over time presents a promising approach to increase the speed of cardiac simulations and to fully utilize highly parallel modern architectures. This study evaluates the use of the parareal algorithm, a parallel-in-time integration method for solving differential equations, for simulations of cardiac cells and tissue. The parareal algorithm estimates the system state at fixed times, then iteratively refines these estimates by solving from the previous time estimate in parallel. Thus, fast convergence to an accurate solution is necessary for this method to be viable. We perform simulations of the Beeler-Reuter (BR) model to evaluate the accuracy and speed of this method in comparison to sequential algorithms. We demonstrate that the parareal algorithm converges exponentially to the true solution in the single-cell case and accurately reproduces APD dynamics in tissue simulations, while attaining speedup relative to the serial algorithm in both cases.*

## 1. Introduction

Parallel hardware is widely available in modern computers in the form of multi-core central processing units (CPUs) and graphics processing units (GPUs). Taking full advantage of these resources to solve computationally-intensive problems requires the use of parallel algorithms. A common and effective strategy for simulations of cardiac electrophysiology is to parallelize over the spatial dimensions of the problem domain. However, spatial parallelization lacks impact when the domain is small relative to the number of available processors and is not applicable to single-cell simulations.

Although computers are now fast enough to perform many types of cardiac simulation in real time or faster [1], further speedup of cardiac simulations may facilitate studying problems that require long time scales. Examples of such problems include the effects of circadian rhythms, which have been shown to change the behavior of ion channels in cardiac cells throughout the day [2], and tissue remodeling, which has been shown to have proarrhythmic effects over the course of weeks to months [3, 4].

Parallel-in-time algorithms offer a promising addition to the toolbox of parallelization techniques in cases where parallel-in-space methods are either not applicable or of limited benefit. Here, we focus on the parareal algorithm, an iterative method based on existing serial methods of solving differential equations.

## 2. Methods

### 2.1. Parareal algorithm

The parareal algorithm is a method of solving initial-value problems that allows some computations to be performed in parallel [5]. Notably, an approximate solution is found at a given time independently from a fully accurate solution at previous times [6]. The method solves systems of ODEs (or spatially-discretized PDEs) of the form

$$\dot{\mathbf{u}} = \mathbf{f}(\mathbf{u}) \quad \text{over} \quad t \in [t_0, t_N] \quad \text{given} \quad \mathbf{u}(t_0) = \mathbf{u}_0 \quad (1)$$

at equally spaced time points $t_0, t_1, \ldots, t_N$.

Parareal requires two propagation operators: the coarse operator $G(t_1, t_2, \mathbf{u}_1)$ and the fine operator $F(t_1, t_2, \mathbf{u}_1)$. Each provides an approximate solution to $\mathbf{u}(t_2)$ from the initial condition $\mathbf{u}(t_1) = \mathbf{u}_1$; however, the fine operator uses a finer time step and thus is assumed to give a more accurate approximation than the coarse operator.

The first step of the parareal algorithm is to compute an initial approximation $\mathbf{U}_n^0$ at each coarse time point $t_n$ by sequentially calculating

$$\mathbf{U}_{n+1}^0 = G(t_n, t_{n+1}, \mathbf{U}_n^0) \quad \text{with} \quad \mathbf{U}_0^0 = \mathbf{u}_0. \quad (2)$$

Further iterations for a given time step of the parareal algorithm are computed in parallel for $k = 0, 1, 2, \ldots$ using the correction step

$$\begin{aligned} \mathbf{U}_{n+1}^{k+1} = {} & G(t_n, t_{n+1}, \mathbf{U}_n^{k+1}) + F(t_n, t_{n+1}, \mathbf{U}_n^k) \\ & - G(t_n, t_{n+1}, \mathbf{U}_n^k). \end{aligned} \quad (3)$$

The values of $\mathbf{U}_n^k$ converge for large enough $k$ to the accuracy of the fine operator $\mathbf{U}_{n+1} = F(t_n, t_{n+1}, \mathbf{U}_n)$ [6].

## 2.2. Implementation

All simulations use the Beeler-Reuter model of cardiac membrane action potentials [7], implemented in Fortran with its original parameter values. The fine and coarse propagators use a semi-implicit Euler time-stepping method with time steps of 0.02 ms and 0.1 ms for the fine and coarse operators, respectively. Our implementation of the parareal algorithm uses MPI to compute $F(t_n, t_{n+1}, \mathbf{U}_n^k)$ in Equation 3 in parallel when iterating over $n$.

One-dimensional tissue simulations use a spatial domain discretized into 512 grid points with a spacing of 0.025 cm, diffusion coefficient 0.0011 cm$^2$/ms, and no-flux boundary conditions. Simulations begin at a rest state and are paced at regular intervals starting at time $t = 0$ s with a pacing stimulus of 26.4 µA/cm$^2$ applied for 2 ms. In the case of the one-dimensional tissue simulation, the stimulation occurs at three tissue points at one end of the domain.

# 3. Results

## 3.1. Single-cell simulations

The parareal algorithm is well-suited to parallelizing single-cell simulations, as they have no spatial domain to exploit for parallelism. Figure 1 shows a comparison of the time series for the last two seconds of two different 120-second simulations: one serial (black solid), the other using two iterations of the parareal algorithm (red dashed). The results are in good agreement with very small differences. The most significant difference in the parareal solution is that the action potential upstroke lags slightly behind that of the serial solution, although some discrepancy also occurs during repolarization. The error in the repolarization phase occurs in cases, such as in the figure, where some alternans is present. When no alternans is present, as in the 400 ms case, the error is concentrated almost entirely in the upstroke. As the number of parareal iterations increases, the maximum error at any point in time decreases at an exponential rate, as seen in Figure 2. This convergence allows straightforward estimation of the number of parareal iterations necessary to achieve a desired accuracy.

## 3.2. One-dimensional tissue simulations

In one-dimensional tissue, the parareal algorithm does not attain the same rapid convergence as with the single-cell case. As shown in Figure 3, significant error arises at the front of each excitation wave, which can only be eliminated by a large number of parareal iterations (on the

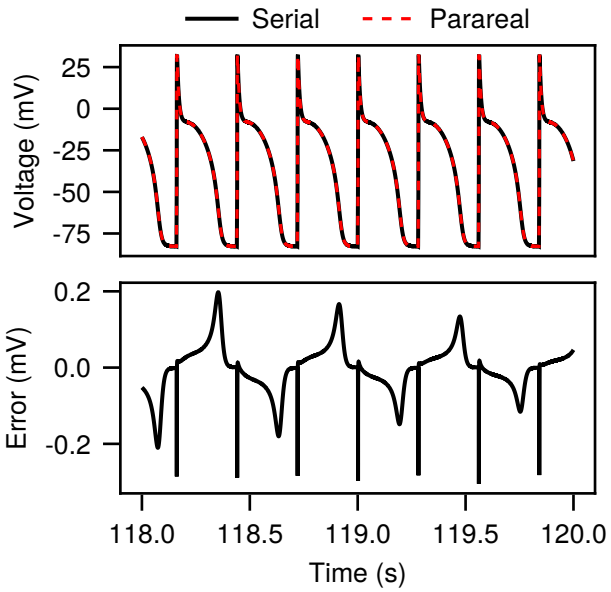


Figure 1. **Above:** Time series of the voltage for the last 2 s of a 120 s simulation with cycle length 280 ms. The parareal solution after two iterations is shown. **Below:** Error from the same simulation measured as the difference between the parareal solution and the serial solution.

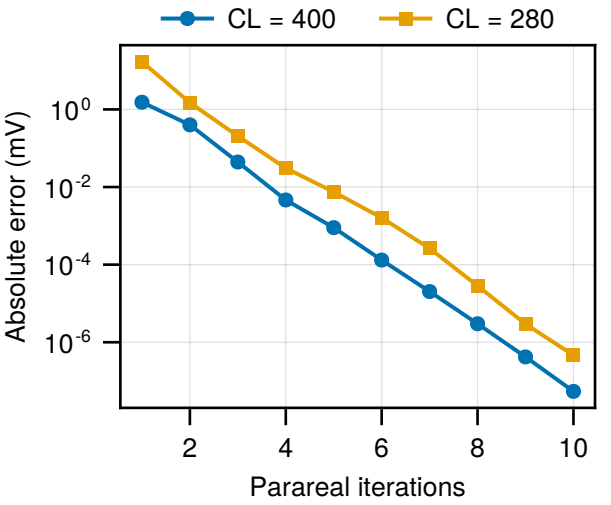


Figure 2. Maximum difference between the parareal solution and the serial solution, plotted on a logarithmic scale, for varying numbers of parareal iterations. A shorter cycle length of 280 ms (squares) results in greater error than a cycle length of 400 ms (circles), but both converge exponentially to the serial solution. The simulation time for both cases is 120 s.

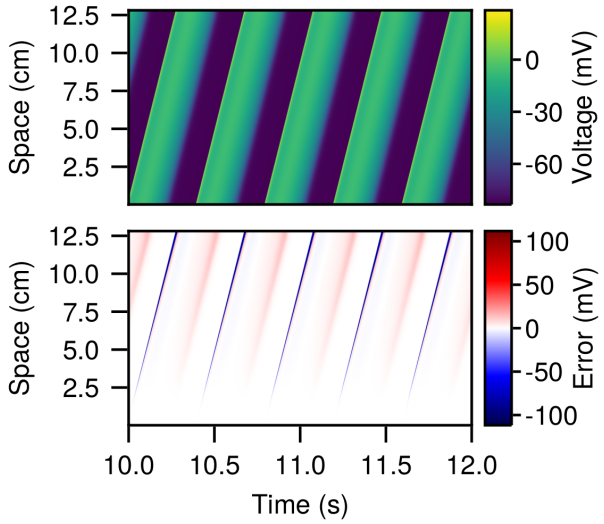


Figure 3. **Above:** Heatmap showing the voltage over a one-dimensional domain for the last 2 s of a 12 s simulation using three iterations of the parareal algorithm and a cycle length of 400 ms. **Below:** Heatmap of the error of the parareal solution relative to the serial solution. The error is concentrated along the front, and to a lesser extent the back, of the propagating wave, which travels slower in the parareal simulation.

order of $10^1$ to $10^2$, and increasing with domain size). This effect is the result of the parareal solution attaining slower wave propagation than the serial solution by about 5.4 %. Increasing the number of parareal iterations results in a decrease in the wave propogation speed error, however, this convergence is linear, so many iterations are required to eliminate the error entirely. When no alternans is present (cycle length 400 ms), the difference in APD between the two simulations is less than 1 % when measured at a threshold of −60 mV, indicating that the qualitative behavior is captured well by the parareal simulation. However, in the alternans case (cycle length 350 ms), the difference in APD is larger because of the role of CV during alternans dynamics for this model.

### 3.3. Speedup

Additional iterations of the parareal algorithm require additional computation time and the speedup provided by parallelism becomes less significant. A plot of the speedup relative to the serial algorithm for varying numbers of iterations is presented in the left plot of Figure 4. The parareal solver is faster for all cases shown, but becomes slower with enough iterations. Consequently, parareal is only practical in circumstances where it quickly converges to the serial solution.

The speedup provided by the parareal algorithm increases with the number of processors, up to the limit imposed by the serial component of the parareal algorithm, which consists of running the coarse propogator. The right plot in Figure 4 shows the speedup relative to the serial algorithm resulting from increasing the number of processors up to 16. The results are consistent for both single-cell and tissue simulations. In both cases, at least 6 processors are required for the parareal simulation to run faster than the serial simulation.

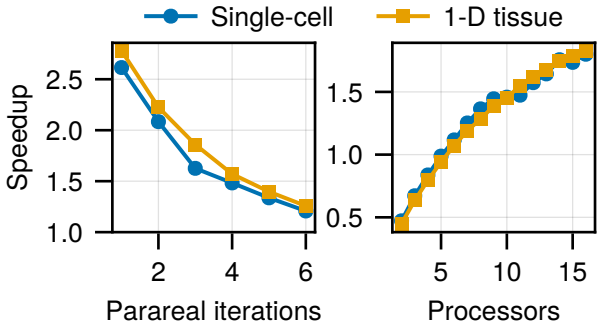


Figure 4. Speedup of the parareal simulation as a factor of the time required to run the serial simulation. The single-cell simulations occur over 120 s, and the one-dimensional simulations over 12 s. **Left:** Speedup of the parareal algorithm running on 16 cores for varying numbers of iterations. **Right:** Speedup of the parareal algorithm running for three iterations on varying numbers of processors.

## 4. Discussion

We find that the parareal algorithm is capable of parallelizing simulations of cardiac tissue across the time domain, with convergence to the serial solution occurring quickly for the case of a single cell and leading to significant speedup on a 16-core processor. Parareal also achieves good accuracy and speedup in terms of APD in the one-dimensional tissue simulation; when discordant alternans is present, the slightly reduced wavespeed in the parareal case can lead to larger APD errors.

The parareal algorithm requires significantly more computation than the corresponding serial algorithm. Therefore, parareal is most useful in cases where parallel hardware is already available but not utilized, as is often the case for computers with multi-core CPUs or GPUs running serial algorithms. A further restriction on the efficiency of parareal is that each iteration requires additional computation, so the rate of convergence to the serial solution must be fast in practice for parareal to produce accurate results while providing significant speedup.

In cases where parallelism over the spatial domain is possible, the amount of spatial parallelization is limited

by the domain size, as eventually the computational cost of communication will outweigh the benefits of increased parallelism. The time dimension, on the other hand, may be extended arbitrarily without increased communication costs. An advantage of parareal is that it can be used in combination with spatial parallelization, so an optimal resource allocation may be attained when large-scale parallelism is available by parallelizing across space and time domains simultaneously. This strategy requires careful application depending on the specific problem to solve and the hardware available to solve it.

The efficiency of the parareal algorithm may be improved by shifting the relative computational cost of the coarse and fine solvers. Parareal is most efficient when the fine solver is more costly than the coarse solver. Therefore, using stable implicit or semi-implicit methods that allow the largest possible coarse time step will result in greater speedup of the parareal algorithm, even if the accuracy of the solver at this time step is low. Similarly, in cases where a high degree of accuracy is desired, requiring either a costly high-order integration method or a very small time step, the fine solver must become more costly to meet these criteria. If the coarse solver remains unchanged, the resulting parareal method will become faster relative to the serial method.

We have shown that the parareal algorithm is particularly effective for the single-cell case, where it attains fast convergence to the serial solution and few alternatives for leveraging parallel hardware are available. The one-dimensional tissue simulation does not achieve the same convergence rate due to reduced wave propagation speed; this issue is likely caused by the relatively large time step size of the coarse operator [8]. Further investigation is required to find a way to mitigate this reduced wave speed.

Parareal is only one of many methods for parallelization in time [9–11]. Further investigation of alternative methods is needed to compare their accuracy and efficiency for cardiac problems with parareal. Promising applications of these methods include problems that require long simulation times, such as the effects of circadian rhythms, development of tissue remodeling, and long-term drug effects.

## Acknowledgments

This work was supported by the NIH InQuBATE training grant T32GM142616 and by NSF grant CNS-2028677.

## References

[1] Kaboudian A, Cherry EM, Fenton FH. Real-time interactive simulations of large-scale systems on personal computers and cell phones: Toward patient-specific heart modeling and other applications. Science advances 2019; 5(3):eaav6019.

[2] Chirico N, Van Laake LW, Sluijter JP, van Mil A, Dierickx P. Cardiac circadian rhythms in time and space: the future is in 4d. Current Opinion in Pharmacology 2021;57:49–59.

[3] Lu Z, Scherlag BJ, Lin J, Niu G, Fung KM, Zhao L, Ghias M, Jackman WM, Lazzara R, Jiang H, et al. Atrial fibrillation begets atrial fibrillation: autonomic mechanism for atrial electrical remodeling induced by short-term rapid atrial pacing. Circulation Arrhythmia and Electrophysiology 2008;1(3):184–192.

[4] Tracy E, Rowe G, LeBlanc AJ. Cardiac tissue remodeling in healthy aging: the road to pathology. American Journal of Physiology Cell Physiology 2020;319(1):C166–C182.

[5] Lions JL, Maday Y, Turinici G. A "parareal" in time discretization of PDEs. Comptes Rendus de lAcademie des Sciences Series I Mathematics 2001;332(7):661–668.

[6] Gander MJ, Vandewalle S. Analysis of the parareal time-parallel time-integration method. SIAM Journal on Scientific Computing 2007;29(2):556–578.

[7] Beeler GW, Reuter H. Reconstruction of the action potential of ventricular myocardial fibres. The Journal of physiology 1977;268(1):177–210.

[8] Cherry EM, Greenside HS, Henriquez CS. Efficient simulation of three-dimensional anisotropic cardiac tissue using an adaptive mesh refinement method. Chaos An Interdisciplinary Journal of Nonlinear Science 2003;13(3):853–865.

[9] Lelarasmee E, Ruehli AE, Sangiovanni-Vincentelli AL. The waveform relaxation method for time-domain analysis of large scale integrated circuits. IEEE transactions on computer aided design of integrated circuits and systems 1982; 1(3):131–145.

[10] Gander MJ, Liu J, Wu SL, Yue X, Zhou T. Paradiag: Parallel-in-time algorithms based on the diagonalization technique. arXiv preprint arXiv200509158 2020;.

[11] Gander MJ, Lunet T, Ruprecht D, Speck R. A unified analysis framework for iterative parallel-in-time algorithms. arXiv preprint arXiv220316069 2022;.

Address for correspondence:

Maxfield R Comstock
School of Computational Science and Engineering
Georgia Institute of Technology
756 West Peachtree Street Northwest
Atlanta GA 30332-4017
mcomstock3@gatech.edu