Detailed Numerical Analyses of the Aliev-Panfilov Model on GPGPU

Xing Cai[1,2] Didem Unat[3] Scott Baden[3]

Simula Research Laboratory, Norway[1]

Department of Informatics, University of Oslo[2]

Department of Computer Science & Engineering, UCSD[3]

PARA2010, Reykjavik, June 2010

Outline

- The Aliev-Panfilov model
- Numerical algorithms
- Simulations on CPU
- Simulations on GPU

We want to study the applicability of GPU for Aliev-Panfilov simulations: accuracy & speed

Motivation

- The Aliev-Panfilov model can simulate very complex spiral waves in cardiac tissues
- High resolution is needed for long-term simulations
- Simulations can be vulnerable to numerical errors
- GPUs are known to possess great computing power
- Are GPUs suitable for Aliev-Panfilov simulations?
    - Accuracy?
    - Single-precision or double-precision?
    - How large is the speed advantage over CPUs?

The Aliev-Panfilov model of cardiac excitation

$$\frac{\partial e}{\partial t} = \delta\nabla^2 e - ke(e-a)(e-1) - er \tag{1}$$

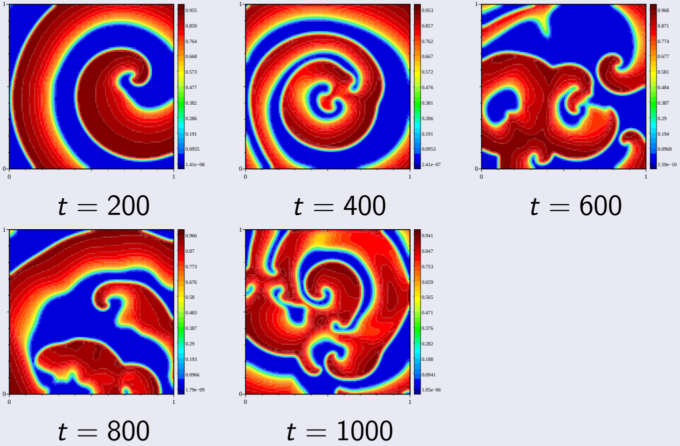
$$\frac{\partial r}{\partial t} = -\left[\varepsilon + \frac{\mu_1 r}{\mu_2 + e}\right][r + ke(e-b-1)] \tag{2}$$

e — scaled transmembrane potential

r — a state variable responsible for cardiac tissue recovery

Model parameters: $\mu_1 = 0.07$, $\mu_2 = 0.3$, $k = 8$, $\varepsilon = 0.01$, $b = 0.1$, $\delta = 5 \times 10^{-5}$

Spiral wave formation and breakup



First-order explicit scheme

During time step $\ell + 1$:

$$\begin{aligned}
e_{i,j}^{\ell+\frac{1}{2}} &= e_{i,j}^{\ell} + \Delta t \delta \left(\frac{e_{i-1,j}^{\ell} - 2e_{i,j}^{\ell} + e_{i+1,j}^{\ell}}{\Delta x^2} + \frac{e_{i,j-1}^{\ell} - 2e_{i,j}^{\ell} + e_{i,j-1}^{\ell}}{\Delta y^2} \right), \\
e_{i,j}^{\ell+1} &= e_{i,j}^{\ell+\frac{1}{2}} - \Delta t \left(k e_{i,j}^{\ell+\frac{1}{2}} (e_{i,j}^{\ell+\frac{1}{2}} - a)(e_{i,j}^{\ell+\frac{1}{2}} - 1) - e_{i,j}^{\ell+\frac{1}{2}} r_{i,j}^{\ell} \right), \\
r_{i,j}^{\ell+1} &= r_{i,j}^{\ell} - \Delta t \left[\varepsilon + \frac{\mu_1 r_{i,j}^{\ell}}{\mu_2 + e_{i,j}^{\ell+1}} \right] [r_{i,j}^{\ell} + k e_{i,j}^{\ell+1} (e_{i,j}^{\ell+1} - b - 1)].
\end{aligned}$$

Very strict stability requirement $\Delta t \sim \mathcal{O}(\Delta x^2)$

Second-order implicit scheme

During time step $\ell + 1$:

- Solve the ODE part from t_ℓ to $t_{\ell+\frac{1}{2}}$ by second-order SDIRK method
- Solve the PDE $\dfrac{\partial e}{\partial t} = \delta \nabla^2 e$ from t_ℓ to $t_{\ell+1}$ by Crank-Nicolson
- Solve the ODE part from $t_{\ell+\frac{1}{2}}$ to $t_{\ell+1}$ by second-order SDIRK method

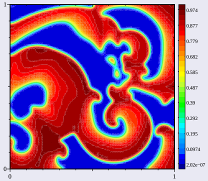
Stability requirement $\Delta t \sim O(\Delta x)$

Comparison of the two schemes

- The first-order explicit scheme
  - straightforward to implement and parallelize (using MPI)
  - efficient computaion per time step
  - strict stability requirement $\Delta t \sim \mathcal{O}(\Delta x^2)$
- The second-order implicit scheme
  - more computations involved
  - need to solve the PDE through a linear system per time step
  - need to solve the ODE part twice by a second-order ODE solver
  - more difficult to implement and parallelize
  - less strict stability requirement $\Delta t \sim O(\Delta x)$

High resolution is needed!

1600×1600 mesh

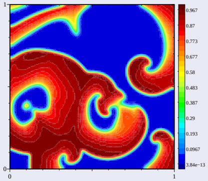


$t = 600$

Double precision

$$\left\| e - e^{\mathrm{ref}} \right\|_{L_2} = 0.42$$

3200×3200 mesh

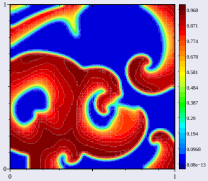


$t = 600$

Double precision

$$\left\| e - e^{\mathrm{ref}} \right\|_{L_2} = 0.16$$

6400×6400 mesh



$t = 600$

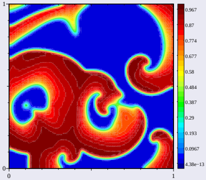
Double precision

$$\left\| e - e^{\mathrm{ref}} \right\|_{L_2} = 0.06$$

Results from the first-order explicit scheme

Single-precision vs. double-precision

3200 × 3200 mesh

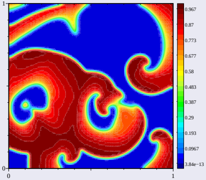


$t = 600$

Single precision

$$\left\| e - e^{\text{ref}} \right\|_{L_2} = 0.13$$

3200 × 3200 mesh



$t = 600$

Double precision

$$\left\| e - e^{\text{ref}} \right\|_{L_2} = 0.16$$

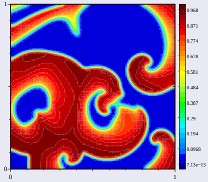
Results from the first-order explicit scheme

GPU implementation

- We´ve made a CUDA implementation of the first-order explicit scheme
- ```
  __global__ void panfilov(
                          REAL* E, const REAL* E_prev, REAL *R,
                          const REAL alpha, const int N, const REAL dt
                          )
  ```
- Capable of both single-precision and double-precision computations
- MPI for communication between multiple GPUs
- More details about GPU programming and optimization in Didem Unat's talk: *Optimizing Aliev-Panfilov model of cardiac excitation on heterogeneous systems*
  ```

Accuracy of GPU computations

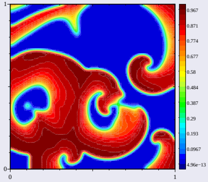
4048 × 4048 mesh



$t = 600$

Single precision

4048 × 4048 mesh



$t = 600$

Double precision

$\left\| e - e^{\mathrm{ref}} \right\|_{L_2} = 0.07$ $\left\| e - e^{\mathrm{ref}} \right\|_{L_2} = 0.11$

Closer comparison of accuracy

Single-precision computations

Mesh	CPU 3200×3200	GPU 4048×4048	CPU 6400×6400
$t = 100$	0.0009	0.0013	0.0009
$t = 200$	0.0012	0.0022	0.0017
$t = 300$	0.0022	0.0024	0.0014
$t = 400$	0.0170	0.0097	0.0023
$t = 500$	0.0619	0.0279	0.0038
$t = 600$	0.1347	0.0747	0.0350

Values of $\left\| e - e^{\mathrm{ref}} \right\|_{L_2}$

Closer comparison of accuracy (cont'd)

Double-precision computations

Mesh	CPU 3200×3200	GPU 4048×4048	CPU 6400×6400
$t = 100$	0.0009	0.0012	0.0007
$t = 200$	0.0012	0.0017	0.0011
$t = 300$	0.0025	0.0024	0.0014
$t = 400$	0.0208	0.0135	0.0064
$t = 500$	0.0756	0.0472	0.0203
$t = 600$	0.1554	0.1121	0.0572

Values of $\left\| e - e^{\mathrm{ref}} \right\|_{L_2}$

Comparison of speed

Single-precision computations

	64 CPU cores MPI	1 GPU Cuda
Mesh	3200×3200	4048×4048
Steps	2149659	3541300
Time usage	31569	13366
GFLOP/s	32.94	124.48

- Dual Xeon quad-core L5420 2.5GHz processors + Gbit/s ethernet
- GPU: nVidia Tesla C1060

Concluding remarks

- For the Aliev-Panfilov simulations, GPUs seem to produce similar accuracy as CPUs
  - both single-precision and double-precision computations
- The computing speed of GPU is a great advantage
- GPUs are poised for high-resolution Aliev-Panfilov simulations
  - using the first-order explicit numerical scheme
  - Fermi is better suited for double-precision simulations
  - Efficient use of cluster of GPUs?

Acknowledgement

We thank the following colleagues:

- Robert Artebrant
- Monica Hanslien
- Glenn Lines
- Aslak Tveito