Hemodynamic Simulation with Lattice Boltzmann

Harvard IACS AC 290R

Michael S. Emanuel

Jonathan Guillotte-Blouin

Yue Sun

Module II in Context of AC 290R

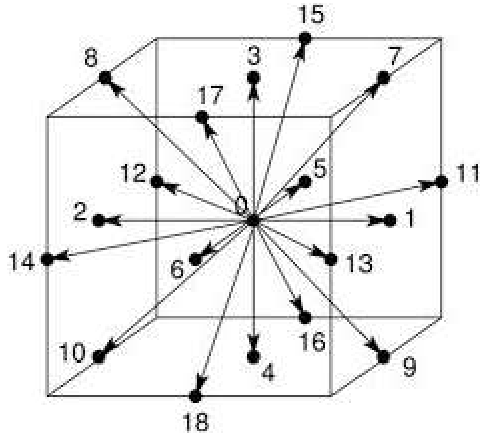
- Goals of AC 290R: learn techniques in Extreme Computing applied to application domain of Fluid Dynamics
- Module I covered the continuum description and Navier Stokes; we simulated Rayleigh-Bénard Convection using the CPU-centric Drekar Code
- Module II shifted to the mesoscale description, which is well suited to the life sciences in particular
- We tackled a prototypical problem: a hemodynamics simulation
- We also learned about GPU computing
- While MUPHY can make good use of GPUs, our simulation ran on 1024 CPUs only

Problem Statement & Motivation

- We simulated the release of a therapeutic drug to treat a stenotic artery
- Stenosis is a narrowing of the artery, often caused by atherosclerosis
- We used an idealized geometry, modeling the artery as a cylinder with a narrowing in the stenotic region
- We wanted to model the dispersal of the drug agent into the stenotic region over a time scale in the range of 1-4 seconds (1 second is roughly 1 heartbeat)
- We attempted two simulations
    - Baseline - Reynolds number 10, 0% hematocrit (red blood cell content)
    - More Realistic - Same geometry and Re as baseline, but with 30% hematocrit

Overview of Numerical Methods Used

- The Boltzmann Equation (Ludwig Boltzmann, 1872) is a triumph of classical thermodynamics and statistical physics
- The Lattice Boltzmann Method (LBM) is a CFD simulation technique based on the Boltmann Equation
- The fluid is modeled as a distribution of *populations* of particles on a grid; each grid point tracks *p* counts of particles with different discrete velocities
- We used the common D3Q19 approach; 3D space is discretized into cubes
- Each cube has 19 neighbors: 1 at distance 0, 6 at distance 1, 12 at distance 2



Credit: research.tue.nl

Equations of Lattice Boltzmann

Bhatnagar-Gross-Krook Update Rule:

$$f_p(x + hc_p, t + h) = f_p(x,t) + \omega(x,t)h\left[f_p^{eq}(\rho, \mathbf{u} - f_p)(x,t) + w_p \frac{c_p \cdot \mathbf{g}}{c_s^2}\right]$$

Equilibrium Population:

$$f_p^{eq}(\rho, \mathbf{u}) = w_p\rho\left[1 + \frac{\mathbf{u}\cdot c_p}{c_2^2} + \frac{(\mathbf{u}\cdot c_p)^2 - c_s^2u^2}{2c_2^4}\right]$$

- f_p is the population with discrete velocity *p*
- u is the velocity
- c_p is the displace $\mathbf{u}(\mathbf{x},t) = \frac{1}{\rho(\mathbf{x},t)}\sum_p c_p f_p(\mathbf{x},t)$ discrete velocity, e.g. (1,0,0)
- ω is the relaxation frequency, related to viscosity by $\nu = c_s^2\left(\frac{1}{\omega} - \frac{1}{2}\right)$
- ρ is the density at this grid point, the sum of the f_p
- w_p are the LBM weights; ⅓, ⅙ and 1/18 for 0th, 1st and 2nd neighbors
- c_s is the lattice speed of sound, √3/3

Description of Code

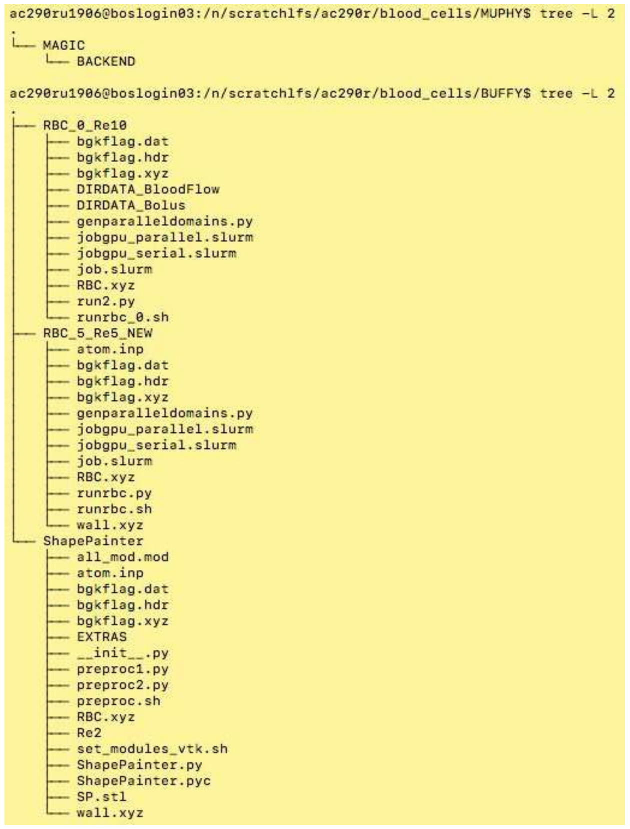
- Workhorse is the back end fluid simulator MUPHY/MOEBIUS
- MUPHY is a ~10 year old multi-physics simulator using LBM with an emphasis on biological applications; guest lecturer Simone Melchionna was a lead developer
- MOEBIUS is a commercial code developed by Dr. Melchionna's company Lexma
- MUPHY simulation engine written in C/C++ and Fortran for maximum speed
- Front end is in Python for convenience in specifying and running simulations
- `ShapePainter.py` generated the geometry for our problem; 8.6m points, 7.0m cells
- `run2.py` invokes the simulation in MUPHY, using MPI to run in parallel
- `runrbc.sh` is a shell script that runs our job on Odyssey with suitable parameters

Post-Processing: Visualization & Analysis

- Performance intensive visualization (e.g. Paraview) was run remotely on Odyssey due to huge size of simulation output (~2GB / frame)
- Analysis and plots carried out on a handful of frames were performed locally on frames downloaded every 100 ms from 0.1 to 1.0 seconds
- VTK outputs (.vtu and .pvtu) were converted to numpy using vtki library
- Drug volume was computing by summing concentration in cells
- Plots of velocity were made using points data with matplotlib

Refactoring

- MUPHY/MAGIC: Backend library
- BUFFY: Each subdirectory represents each simulation with different RBC and Re
- Workflow: Create shapes in ShapePainter, copy the output files into their respective RBC_X_ReX folder, submit batch scripts
- Simulation attempts:
  - Re = 10: 0% RBC, 5% RBC (❌), 30% RBC (❌)
  - Re = 5: 5% RBC, 10% RBC (❌)



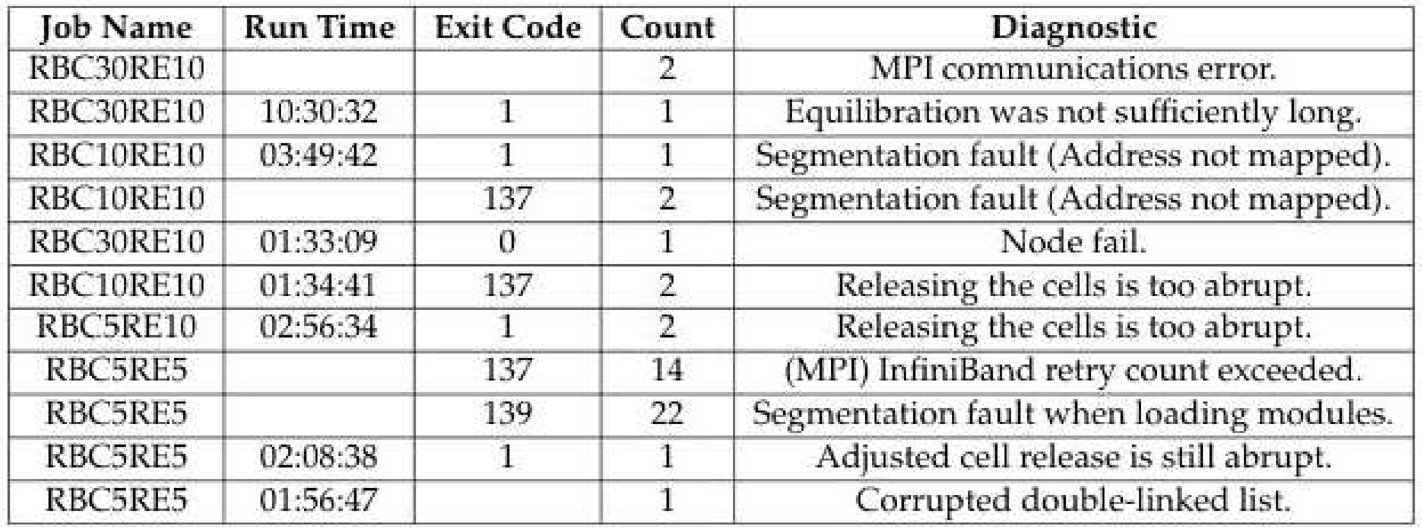
```
ac290ru1906@boslogin03:/n/scratchlfs/ac290r/blood_cells/MUPHY$ tree -L 2
.
└── MAGIC
    └── BACKEND

ac290ru1906@boslogin03:/n/scratchlfs/ac290r/blood_cells/BUFFY$ tree -L 2
.
├── RBC_0_Re10
│   ├── bgkflag.dat
│   ├── bgkflag.hdr
│   ├── bgkflag.xyz
│   ├── DIRDATA_BloodFlow
│   ├── DIRDATA_Bolus
│   ├── genparalleldomains.py
│   ├── jobgpu_parallel.slurm
│   ├── jobgpu_serial.slurm
│   ├── job.slurm
│   ├── RBC.xyz
│   ├── run2.py
│   └── runrbc_0.sh
├── RBC_5_Re5_NEW
│   ├── atom.inp
│   ├── bgkflag.dat
│   ├── bgkflag.hdr
│   ├── bgkflag.xyz
│   ├── genparalleldomains.py
│   ├── jobgpu_parallel.slurm
│   ├── jobgpu_serial.slurm
│   ├── job.slurm
│   ├── RBC.xyz
│   ├── runrbc.py
│   ├── runrbc.sh
│   └── wall.xyz
└── ShapePainter
    ├── all_mod.mod
    ├── atom.inp
    ├── bgkflag.dat
    ├── bgkflag.hdr
    ├── bgkflag.xyz
    ├── EXTRAS
    ├── __init__.py
    ├── preproc1.py
    ├── preproc2.py
    ├── preproc.sh
    ├── RBC.xyz
    ├── Re2
    ├── set_modules_vtk.sh
    ├── ShapePainter.py
    ├── ShapePainter.pyc
    ├── SP.stl
    └── wall.xyz
```

Parameters of the Simulation

- Simulation 1:
  - Dimensions: Re=10.0, Pe=10.0, Length L=1000.0, Radius =50.0
  - Red Blood Cells: 0%
  - Drug Release Time: 100000
- Simulation 2:
  - Dimensions: Re=5.0, Pe=10.0, Length L=500.0, Radius =25.0
  - Red Blood Cells: 5%
  - Drug Release Time: 50000
  - Blood Unfreeze Time: 2000 (3000)
- Density $\rho_0 = 1.0$
- Viscosity: $\nu = 0.1$
- Average Velocity: $\bar{u} = 0.01$

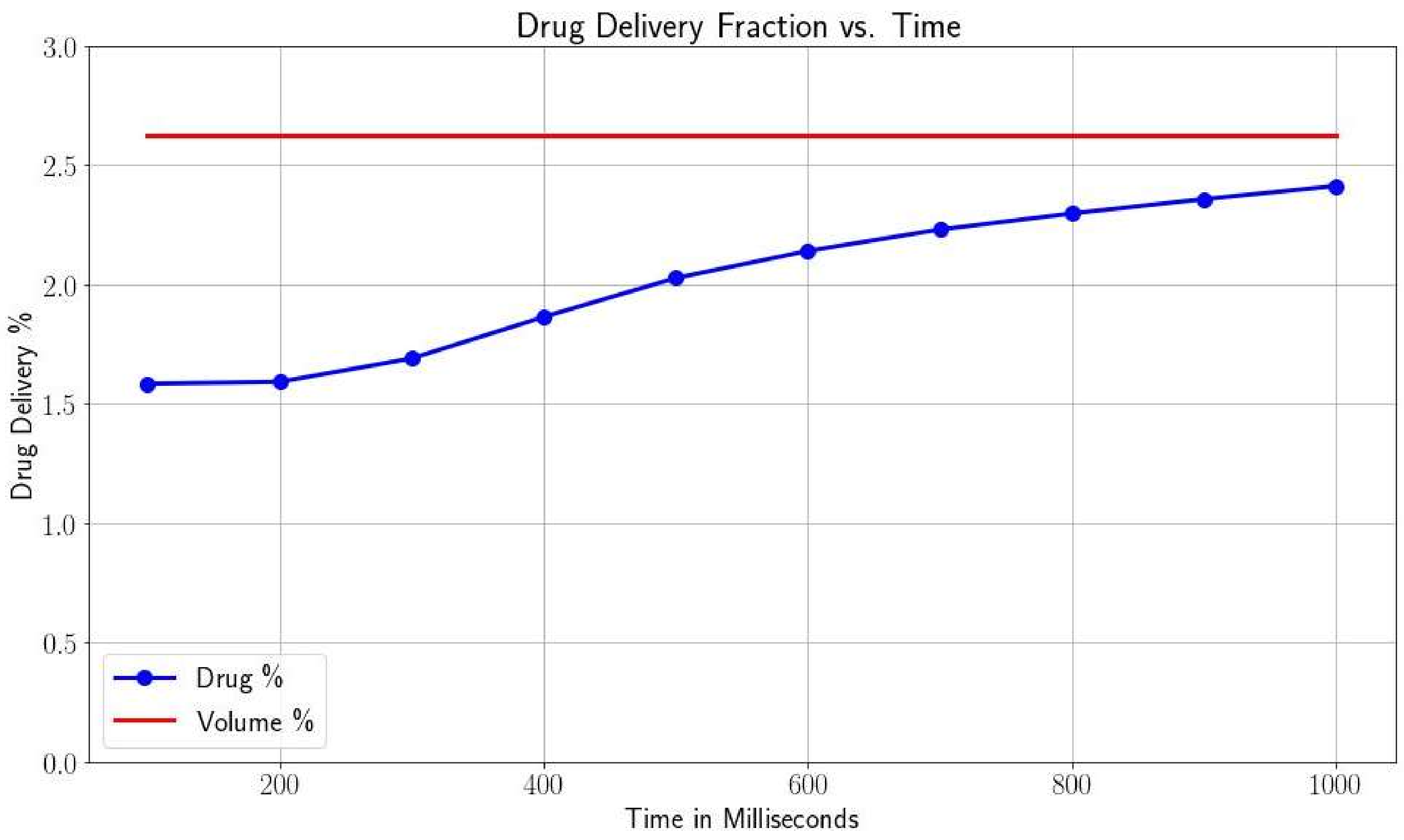
The “Odyssey”



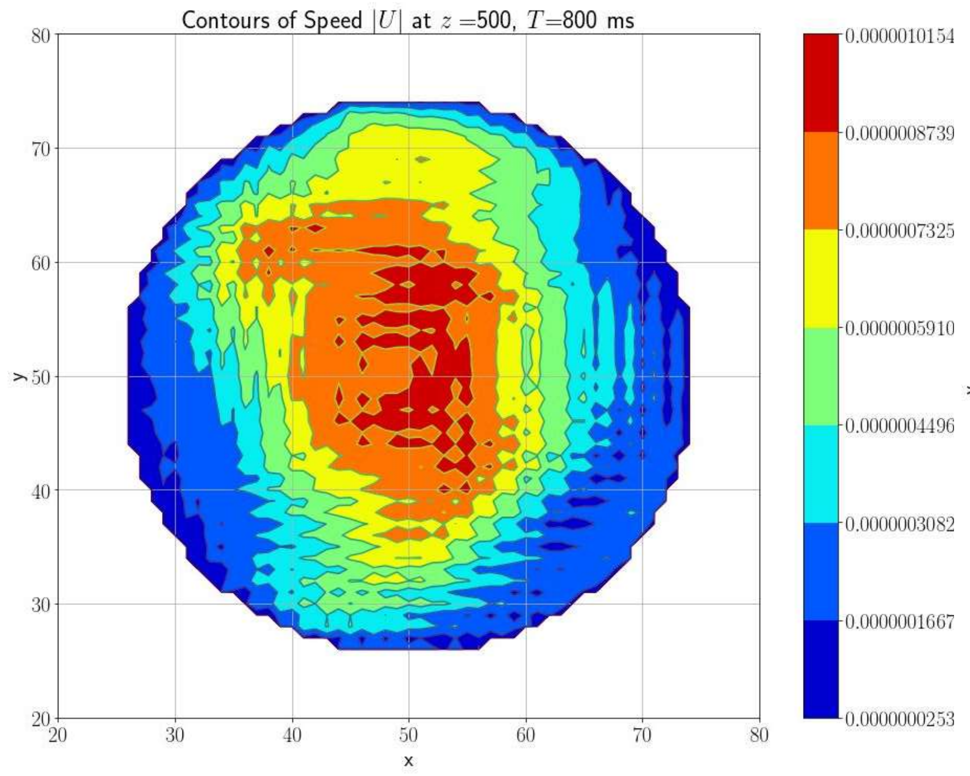
Job Name	Run Time	Exit Code	Count	Diagnostic
RBC30RE10			2	MPI communications error.
RBC30RE10	10:30:32	1	1	Equilibration was not sufficiently long.
RBC10RE10	03:49:42	1	1	Segmentation fault (Address not mapped).
RBC10RE10		137	2	Segmentation fault (Address not mapped).
RBC30RE10	01:33:09	0	1	Node fail.
RBC10RE10	01:34:41	137	2	Releasing the cells is too abrupt.
RBC5RE10	02:56:34	1	2	Releasing the cells is too abrupt.
RBC5RE5		137	14	(MPI) InfiniBand retry count exceeded.
RBC5RE5		139	22	Segmentation fault when loading modules.
RBC5RE5	02:08:38	1	1	Adjusted cell release is still abrupt.
RBC5RE5	01:56:47		1	Corrupted double-linked list.

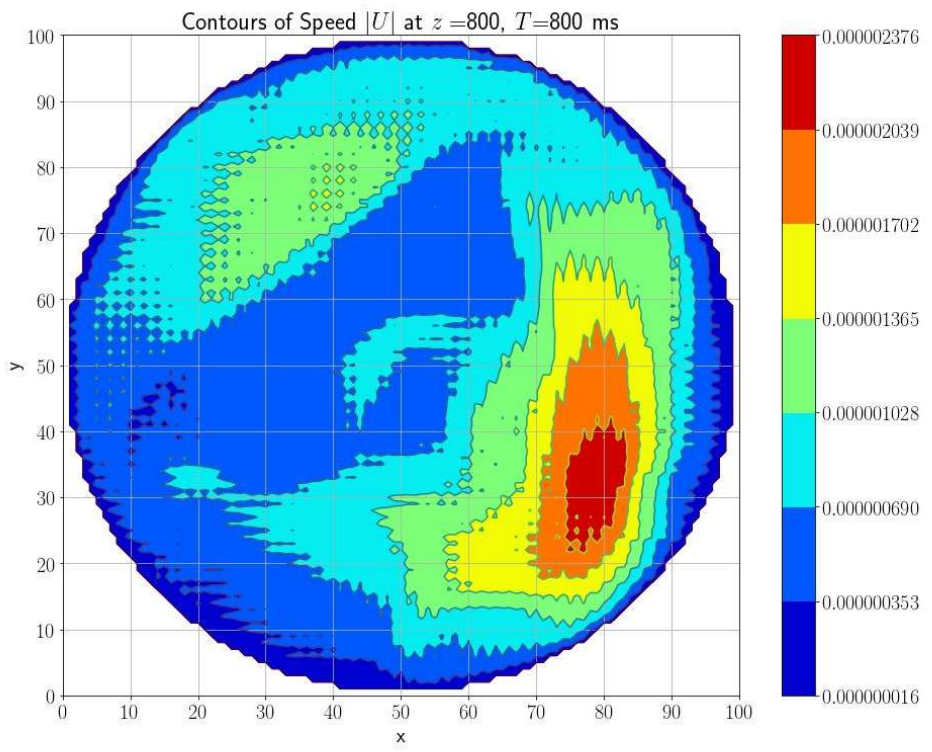
- Always run smaller test cases before launching the actual full-scale simulation
- Some errors are unexpected, and submitting jobs repeatedly may help

Drug Delivery Over Time

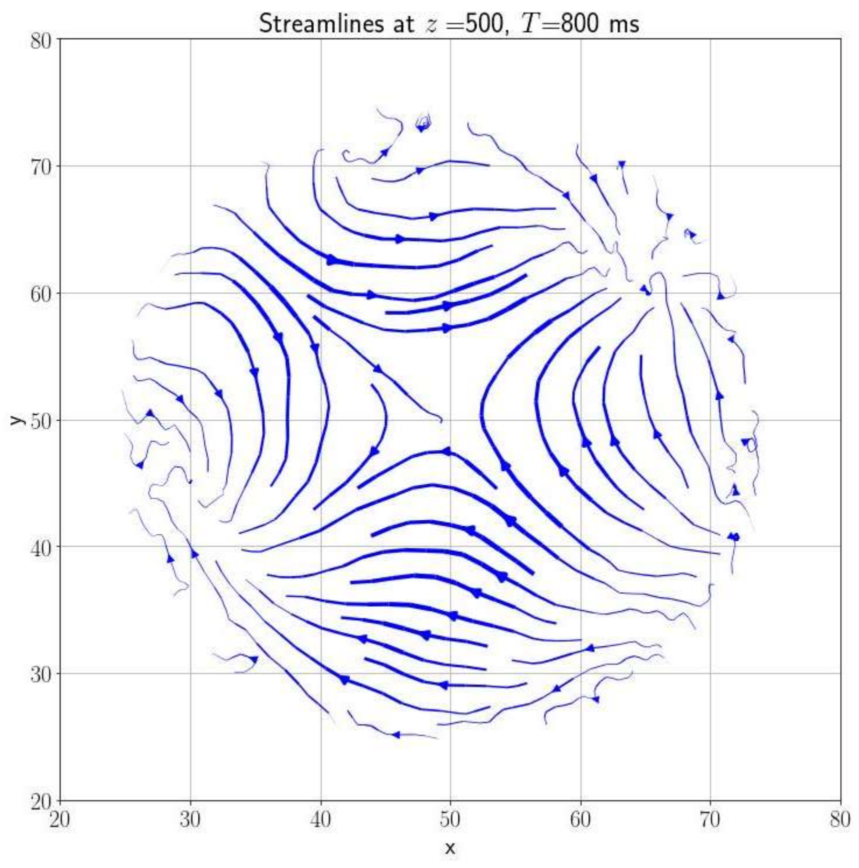


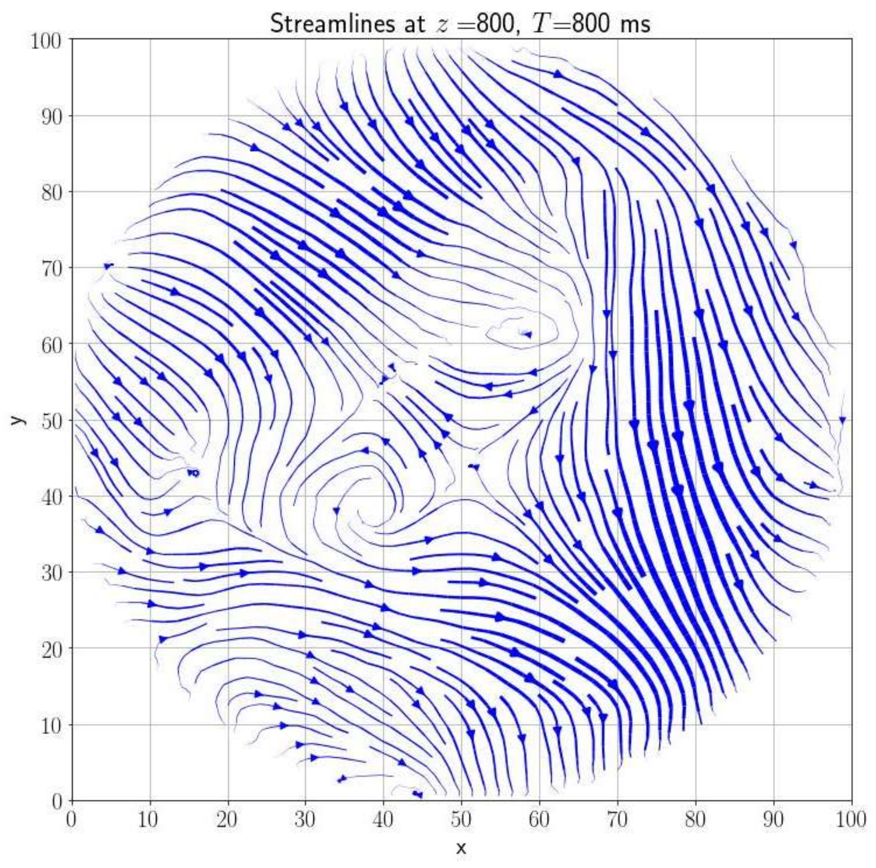
Contour Plots of Speed



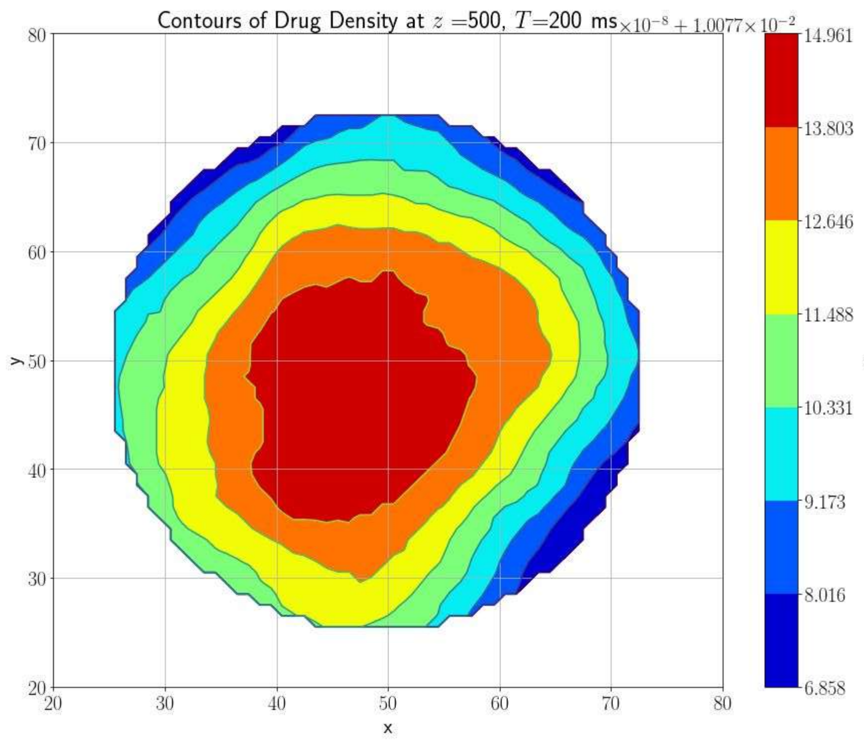


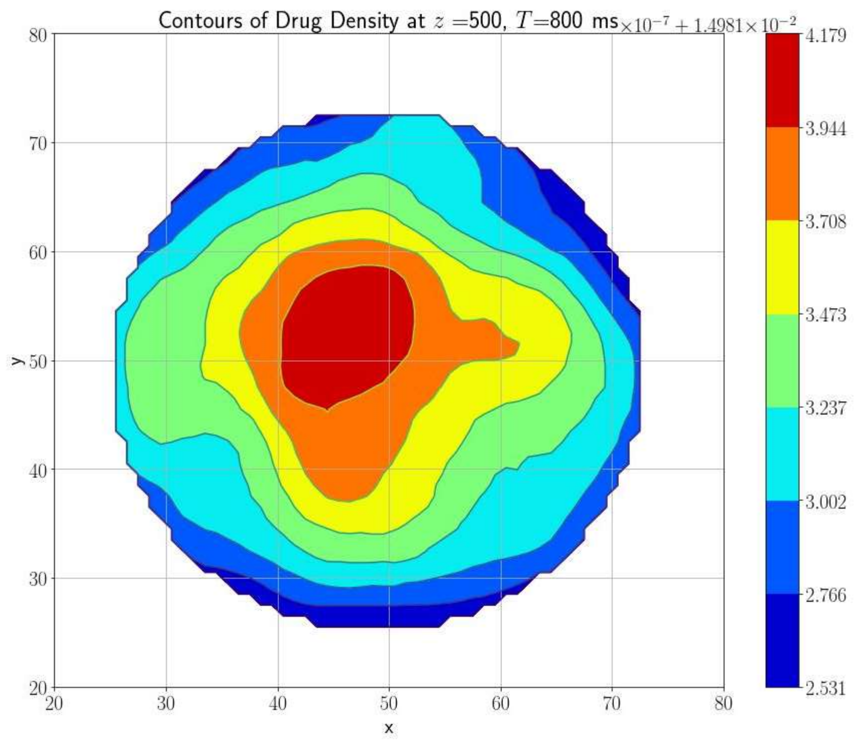
Streamlines



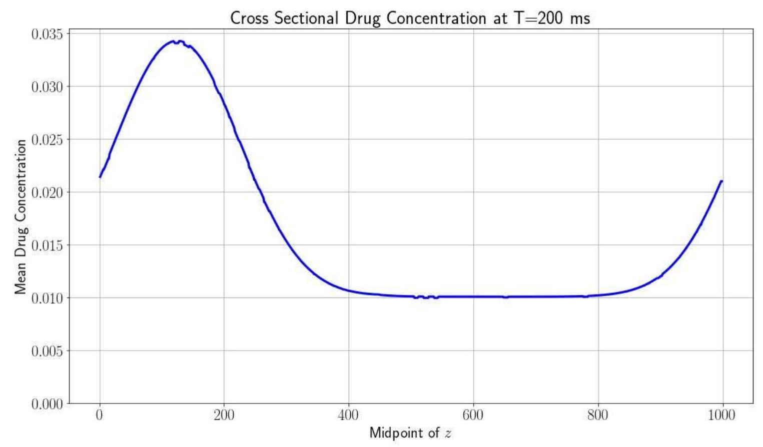


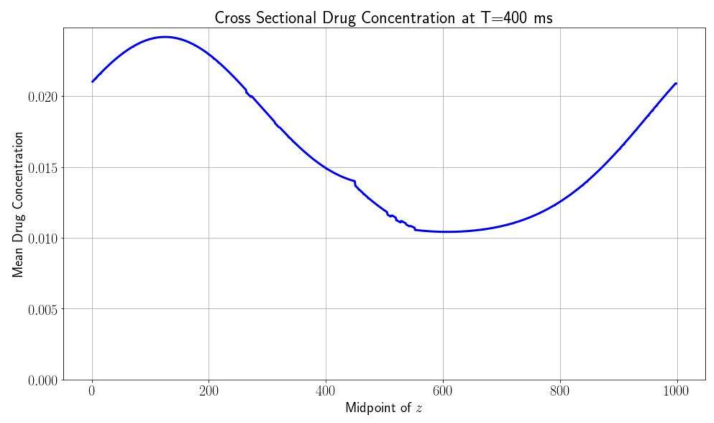
Contour Plot of Drug Density

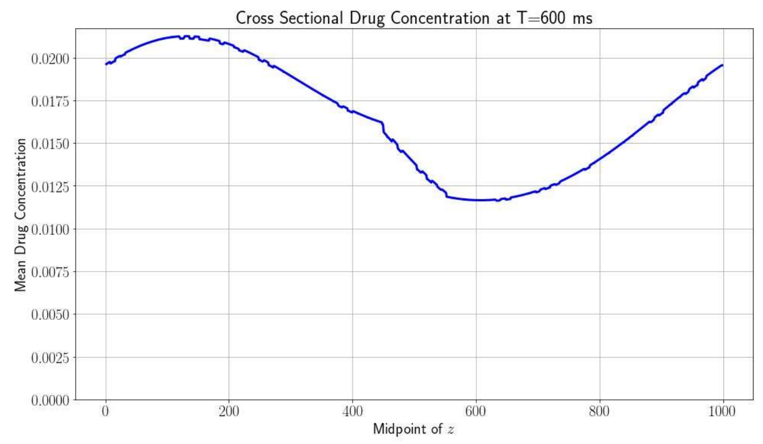


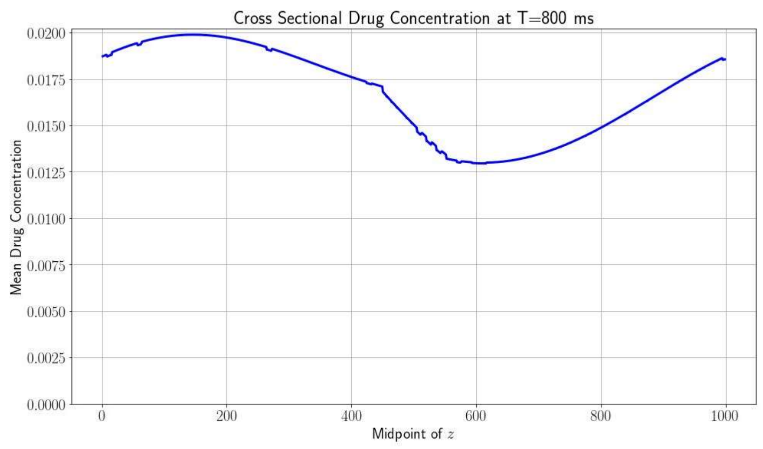


Longitudinal Drug Profile









Movie

-

Conclusions: Hemodynamic System

- Baseline simulation suggests that the drug diffuses to equilibrium levels rapidly (~1 second) and specific geometry not too important
- Since the RBC runs failed, we couldn't learn about their effect on the simplified system, though we expect it to be small in a large artery
- To refine the simulation, we need a more accurate geometry, ideally a scan of a patient; probably more important than RBC for accuracy
- Suggested directions for future work:
  - More accurate geometry; replace period boundaries with heart & veins
  - Shift from CPU to GPU computing
  - More accurate biochemistry model

Conclusions: Extreme Computing

Grazie Mille, Grazie Ragazzi