Simulations for Hybrid models of Tumour growth

Nikolaos Dimitriou

Department of Bioengineering, McGill University, Montréal, Canada.

October 18, 2021

Contents

1. Mathematical model
2. Numerical methods
3. Implementation

Continuum model

A simplified mechanistic, Keller-Segel type model:

u: cell density
f: chemo-attractant density

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u + su(1-u) - \chi \nabla \cdot (u(1-u)\nabla f)$$

$$\frac{\partial f}{\partial t} = D_f \nabla^2 f + rfu(1-u) \text{ , Neumann B.C.}$$

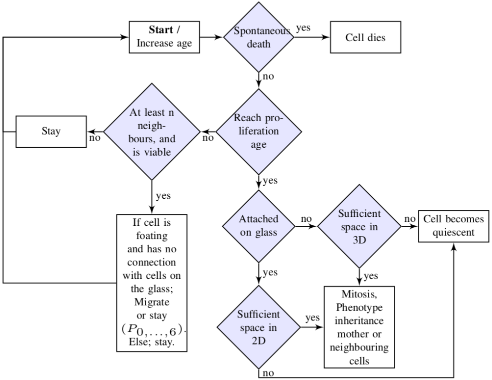
- $\nabla^2 u$: diffusion
- $u(1-u)$: proliferation
- $\nabla \cdot (u(1-u)\nabla f)$: advection

Discrete model

Discretize continuum model using central differences and Euler method:

$$u^{n+1}_{i,j,k} = P_0 u^n_{i,j,k} + P_1 u^n_{i+1,j,k} + P_2 u^n_{i-1,j,k}$$
$$+ P_3 u^n_{i,j+1,k} + P_4 u^n_{i,j-1,k} + P_5 u^n_{i,j,k+1} + P_6 u^n_{i,j,k-1}$$

- u: cell
- P_i: moving probabilities



Remarks

- Both continuum and discrete run at the same time $\leftarrow$ slow $\approx$ 20 minutes
- The continuum can work without the discrete model $\leftarrow$ fast $\approx$ 1-5 minutes
- The discrete model cannot work without the continuum model

Remarks

Thoughts on calibration with experimental data

- Calibrate the continuum
- Pass the calibrated continuum to the discrete
- Fix the rest of the parameters

Numerical methods

Space and time discretizations for the simulation of the Continuum model

Explicit vs Implicit time stepping

Explicit	**Implicit**
$u^n = au^{n-1}$	$u^n + \nabla u^n = au^{n-1}$
Conditionally stable	Unconditionally stable
Simple implementation	Difficult implementation

Numerical methods

- Parabolic problems e.g. heat equation

$$\frac{\partial u}{\partial t} - \nabla^2 u = f$$

- Hyperbolic problems e.g. transport equation

$$\frac{\partial u}{\partial t} + \nabla u = f$$

- Mixed or other problems

Implicit time stepping

Explicit time stepping

?

Numerical methods

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u + su(1-u) - \chi \nabla \cdot (u(1-u)\nabla f)$$

$$\frac{\partial f}{\partial t} = D_f \nabla^2 f + rfu(1-u) \text{ , Neumann B.C.}$$

- treat $\nabla^2 u$ implicitly $\implies$ ADI Douglas-Gunn scheme (2nd order accurate)
- treat $\nabla \cdot (u(1-u)\nabla f)$ explicitly $\implies$ Lax-Wendroff with Flux Limiter (2nd order accurate)

This technique is called **IMEX** or **operator splitting**

Numerical methods

What remains to be done?

– Update the variable by adding the operators. **Caveat**: Accuracy drops to **1st order**

Solution: Strang-splitting scheme:

- Update **explicitly** treated operator for $\mathrm{d}t/2$
- Update **implicitly** treated operator for $\mathrm{d}t$
- Update **explicitly** treated operator for $\mathrm{d}t/2$

Implementation - ADI Douglas-Gunn scheme

Subproblem:

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u$$

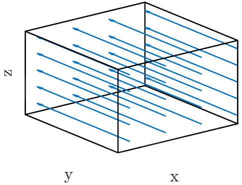
Scheme:

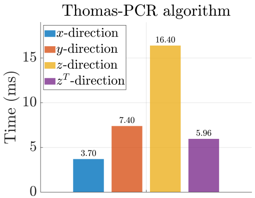
$$\begin{aligned}\left(1 - \frac{1}{2} v\delta_x^2\right) u^{n,*} &= (1 + \frac{1}{2} v\delta_x^2 + v\delta_y^2 + v\delta_z^2) u^n \\ \left(1 - \frac{1}{2} v\delta_y^2\right) u^{n,**} &= u^{n,*} - \frac{1}{2} v\delta_y^2 u^n \\ \left(1 - \frac{1}{2} v\delta_z^2\right) u^{n+1} &= u^{n,**} - \frac{1}{2} v\delta_z^2 u^n\end{aligned}$$

where, $v = \frac{D_u \mathrm{d}t}{2h^2}$, $h = \mathrm{d}x = \mathrm{d}y = \mathrm{d}z$, and $\delta_i^2 = \frac{\partial^2}{\partial x_i^2}$

Implementation - ADI Douglas-Gunn scheme in GPUs

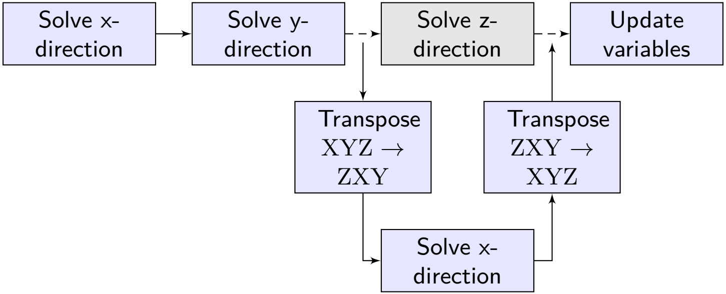
- u is 3D tensor
- For every step:
  - Load u as *pencils* aligned with x-axis
  - y-axis
  - z-axis
- Solve the linear system using Thomas-PCR algorithm





Implementation - ADI Douglas-Gunn scheme in GPUs

Summary



Implementation of Lax-Wendrof with Flux limiter

Subproblem:

$$\frac{\partial u}{\partial t} = -\chi \nabla u$$

Scheme:

$$u^{n+1}_{i,j,k} = u^{n}_{i,j,k} + \chi \frac{\mathrm{d}t}{h}(F_{i-1/2} - F_{i+1/2} + F_{j-1/2} - F_{j+1/2} + F_{k-1/2} - F_{k+1/2}) \qquad (1)$$

Here, $F_{i\pm 1/2}$ are defined as follows

$$F_{i-1/2} = (u\nabla f)_{i-1} + \phi_{-}\frac{1}{2}\mathrm{sign}((\nabla f)_i)(1-c)[u_i(\nabla f)_i - u_{i-1}(\nabla f)_{i-1}] \qquad (2)$$

$$F_{i+1/2} = (u\nabla f)_{i} + \phi_{+}\frac{1}{2}\mathrm{sign}((\nabla f)_i)(1-c)[u_{i+1}(\nabla f)_{i+1} - u_{i}(\nabla f)_{i}] \qquad (3)$$

Implementation of Lax-Wendrof with Flux limiter

where $c = \chi \frac{\mathrm{dt}}{h}$,

$$(u\nabla f)_i = u_i \max(0, (\nabla f)_i) - u_{i+1} \max(0, -(\nabla f)_{i+1}) \tag{4}$$
$$(u\nabla f)_{i-1} = u_{i-1} \max(0, (\nabla f)_{i-1}) - u_i \max(0, -(\nabla f)_i) \tag{5}$$

and

$$(\nabla f)_i = \frac{f_i - f_{i-1}}{h}, \quad (\nabla f)_{i-1} = \frac{f_{i-1} - f_{i-2}}{h}, \quad (\nabla f)_{i+1} = \frac{f_{i+1} - f_i}{h} \tag{6}$$

The $\phi_\pm$ are the flux limiter variables and are defined as follows

$$\phi_\pm = \phi(r_{i\pm 1/2}) = \max(0, \min(2r_{i\pm 1/2}, \frac{1}{2}(r_{i\pm 1/2} + 1), 2)) \tag{7}$$

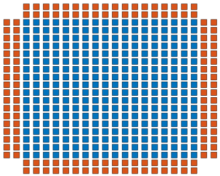
where,

$$r_{i-1/2} = \frac{u_I - u_{I-1}}{u_i - u_{i-1}}, \; r_{i+1/2} = \frac{u_{I+1} - u_I}{u_{i+1} - u_i} \tag{8}$$

and $I = i - \mathrm{sign}((\nabla f)_i)$

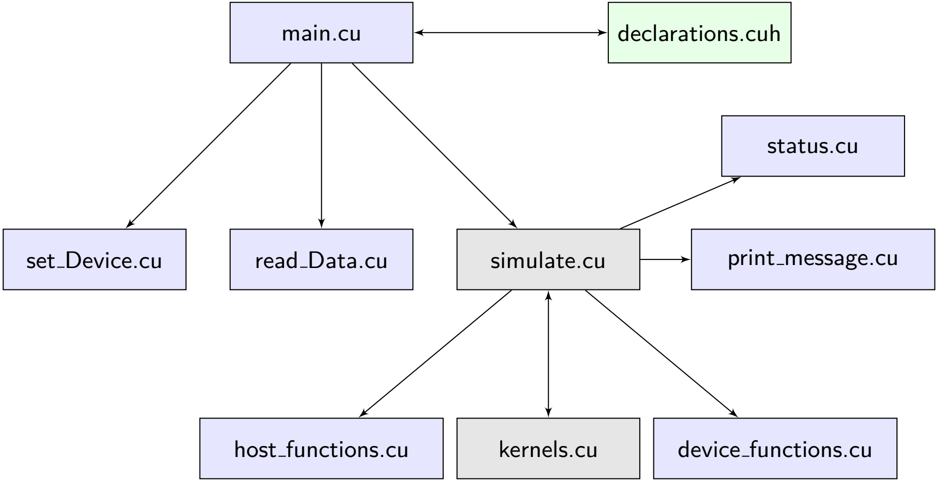
Implementation of Lax-Wendrof with Flux limiter in GPUs

- For every step in z-direction
  - Load the xy-plane, *tile*,
  - the regions above and below
  - the ghost regions



Structure of the code

Main simulator code:



Structure of the code

Further improvements:

- IC loading is time consuming (10-20 sec)
- Simulation may be run many times (e.g. calibration)

Nodal shared memory:

- Stores IC into shared memory of a node
- Instead of IC (300 MB per file), we give the pointer to the address ($\sim$ KB)
- Same for experimental data ($\sim$ GB)