GPU Implementation of Lattice Boltzmann Method with Immersed Boundary: observations and results

Alistair Revell

University of Manchester
Modelling and Simulation Centre
Twitter: @Dr_Revell

The Oxford e-Research Centre Many-Core Seminar Series
5th June 2013

The University of Manchester

Acknowledgements

- Colleagues contributing to this work
  - Mr. Mark Mawson (PhD student @ Univ. Manchester)
  - Dr. Julien Favier (Univ. Marseilles)
  - Pr. Alfredo Pinelli (City Univ. London; prev. CIEMAT, Madrid)
  - Mr. Pedro Valero Lara PhD student @ CIEMAT)
  - Mr. George Leaver (Visualization, Univ. Manchester)
- Thanks to the UKCOMES community (UK Consortium Mesoscale Engineering Sciences), which led to this opportunity.

Overview of Seminar

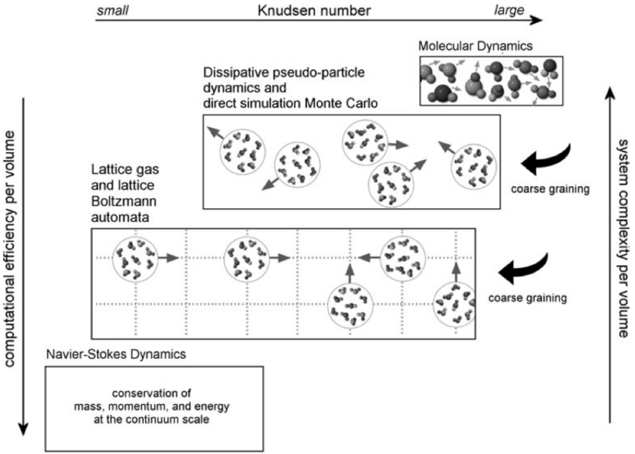
- Lattice-Boltzmann method
  - An overview of the method
  - Some GPU-related optimizations
  - Validation / Results
- Immersed Boundary method
  - Overview of the method
  - Implementation of IB into LB
  - Some GPU optimisations
  - Results
- Realtime simulation
  - OpenGL tweaks
  - Live demos

Lattice Boltzmann Method: Introduction

The Lattice Boltzmann Method might be considered to be a ‘Mesoscale’ approach

- **Macroscale approaches:**
  - In the limit of low Knudsen number one can assume a ‘Continuum’.
  - The Navier Stokes equation can be applied to infinitesimal elements
- **Microscale approaches:**
  - In the limit of high Knudsen, one might resort to Molecular Dynamics
  - While this approach is impractical for macroscale applications
- **Mesoscale:**
  - Broadly, one can consider that Lattice Boltzmann Method operates between these constraints.
  - on one side it can be extended to macroscale problems, whilst retaining a strong underlying element of particle behaviour.

Lattice Boltzmann Method: Introduction

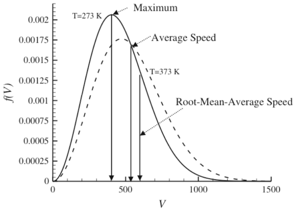


[Raabe, 2004,]

Lattice Boltzmann Method: Introduction

- Focus on a *distribution* of particles $f(r, c, t)$
  - Characterises the particles without realising their individual dynamics
  - Instead considers the distribution of particle velocities
- For given equilbrium gas, one can obtain the Maxwell-Boltzmann distribution function :

$$f^{(eq)} = 4\pi \left(\frac{m}{2\pi kT}\right)^{\frac{3}{2}} c^2 e^{\frac{-mc^2}{2kT}}$$



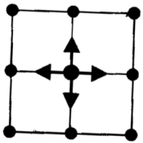
- Macroscopic quantities recovered by integration
  - Boltzmann equation describes the return of $f(\mathbf{r}, \mathbf{c}, t)$ to equilibrium conditions $f^{(eq)}(\mathbf{r}, \mathbf{c}, t)$
    - the cornerstone of the Lattice Boltzmann Method

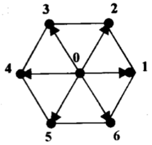
$$f(\mathbf{r} + \mathbf{c}dt, \mathbf{c} + \mathbf{F}dt, t + dt)d\mathbf{r}d\mathbf{c} - f(\mathbf{r}, \mathbf{c}, t)d\mathbf{r}d\mathbf{c} = \Omega(f)d\mathbf{c}dt$$

MANCHESTER 1824 The University of Manchester

Lattice Boltzmann Method: Introduction

- LBM has origins in Lattice Gas Cellular Automata
  - Hardy, Pomeau, de Pazzis [Hardy et al., 1973,]
    - proposed a square lattice arrangement
    - though not rotationally invariant and produced 'square vorticies'!
  - Frisch Hasslacher, Pomeau [Frisch et al., 1986,]
    - proposed a hexagonal lattice; ensured realisitic fluid dynamics
    - included a 'random element' and used look up tables.
- basic conservation laws applied
  - Particles can move only along one of the directions
  - Particles move only to next node in one timestep.
  - No particles at same site can move in same direction





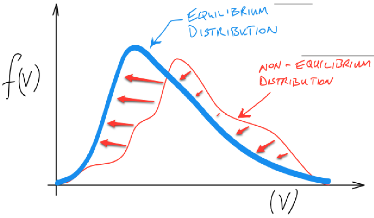
Lattice Boltzmann Method: Introduction

- Boltzmann equation can be written as:

$$\partial_t f + c \cdot \nabla_r f + F \cdot \nabla_c f = \Omega$$

- We use the BGK collision operator from [Bhatnagar et al., 1954,] to approximate Ω

$$\Omega = \frac{1}{\tau}\left(f^{(eq)} - f\right)$$



- represents relaxation back to equilibrium distribution in timescale τ.
  - And use expansion of $f^{(eq)}$ to be 2nd order accurate.

$$f^{(eq)} = \left(\frac{1}{2\pi c_s^2}\right) e^{\frac{-c^2}{2c_s^2}} \left[1 + \frac{\mathbf{c}\cdot\mathbf{u}}{c_s^2} + \frac{(\mathbf{c}\cdot\mathbf{u})^2}{2c_s^4} - \frac{u^2}{c_s^2}\right]$$

Lattice Boltzmann Method: Introduction

- Spatial discretization on lattice is provided by Gaussian Quadrature
- DnQm models used for n spatial dimensions and m discrete velocities
- Here we use D2Q9 and D3Q19
- Force term implemented following [Guo et al., 2002,]:

$$f_i(x+e_i, t+1) = f_i(x,t) - \frac{1}{\tau}(f_i(x,t) - f_i^{(0)}(x,t)) + \left(1 - \frac{1}{2\tau}\omega_i(\frac{e_i - u}{c_s^2} + \frac{c_i \cdot u}{c_s^4}c_i\right) \cdot f_{ib}$$

  - f_{ib} eventually used for immersed boundary

Validity of LBM for different flows

- Based on 13-moment theory of [Grad, 1949,] the distribution function may be expanded over velocity and space using orthonormal Hermite polynomials [Shan et al., 2006,].

$$f^N(\mathbf{c}, \mathbf{c}, t) = \omega(c) = \sum_{n=0}^{N} \frac{1}{n} a^{(n)}, \qquad a^{(n)} = \int f \mathcal{H}^{(n)}(c) dc$$

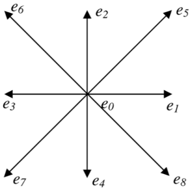
- Coefficients of the Hermite polynomial match the moments of the macroscopic variables

H(n)	Resulting model	N-S momentum only	Burnett	N-S+Energy	Suitable physics
4	D2Q9 D3Q19	OK at small Ma	Not valid	Not valid	Incompressible non thermal flow
6	D2Q17 D3Q39	OK	OK at small Ma	OK at small temp variation	Supersonic regime
8	D2Q37 D3Q121	OK	OK	OK	All !

[Latt, 2013,]

Difference Lattice models

D2Q9

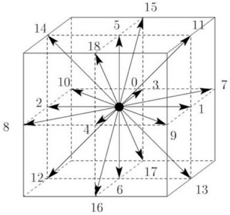


$$w_0 = \frac{4}{9}$$

$$w_{1-4} = \frac{1}{9}$$

$$w_{5-8} = \frac{1}{36}$$

D3Q19

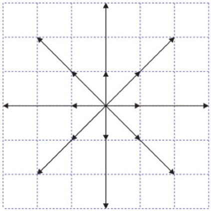


$$w_0 = \frac{12}{36}$$

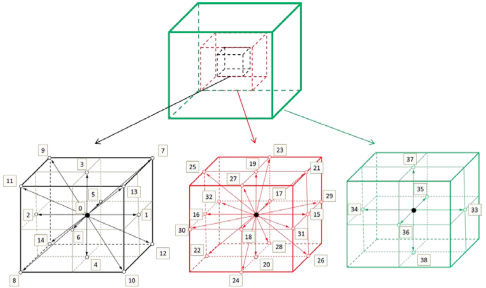
$$w_{1-6} = \frac{2}{36}$$

$$w_{7-18} = \frac{1}{36}$$

D2Q17

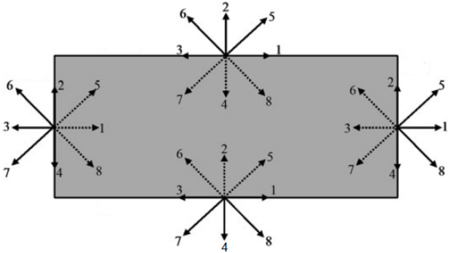


D3Q39[Nie and Chen, 2009,]



Boundary Conditions

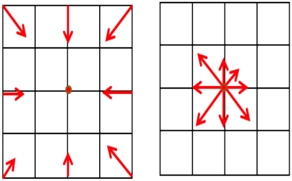
- 1st order bounce-back conditions are the most simple
- 2nd order Zou-He conditions have been implemented [Zou, 1997,]
- some ‘issues’ at corners and along edges where problem is ‘underdefined’



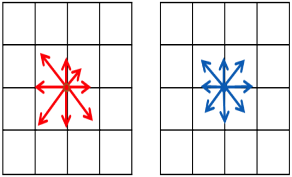
Lattice Boltzmann Method: Algorithm

1. initialise
2. compute forces
3. compute equilibrium function
4. stream & collide
5. imposed boundary condition
6. compute macroscopic quantities
7. $\longrightarrow$ loop to 2

Stream (non-local)



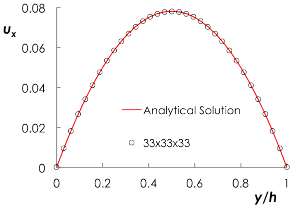
Collide(local)

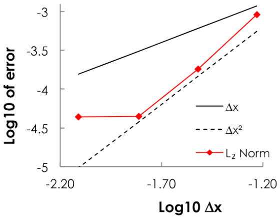


Validation of 3D solver & boundary conditions

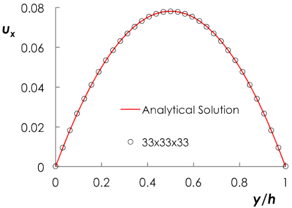
Poiseuille Flow

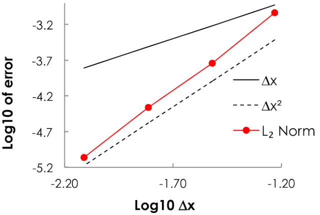
single precision





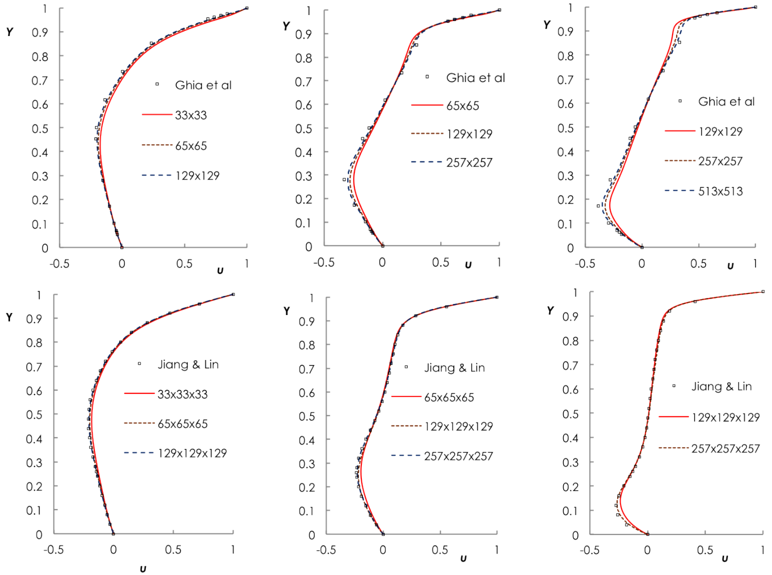
double precision





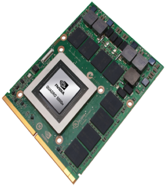
Validation of 3D solver & boundary conditions

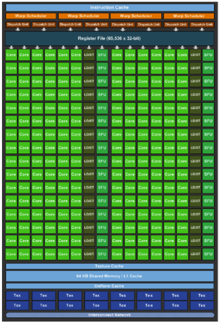
Lid Driven Cavity 2D: [Ghia et al., 1982,], 3D: [Jiang et al., 1994,]



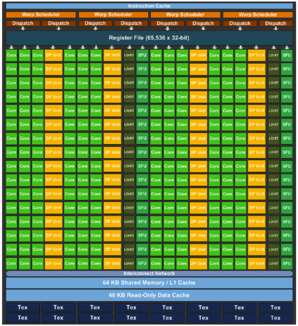
Implementation on GPU

Hardware Tested





(a) GK104



(b) GK110

Feature	FK104 :K5000M	GK110: K20c
cores (SMX x cores/SMC)	1344 (7 x 192)	2496 (13 x 192)
regs / thread	63	255
DRAM	4GB	4.7GB
SP/DP ratio	24:1	3:1
Peak performance (single precision)	1.6 TFLOPS	3.5 TFLOPS
DRAM Bandwidth	66 GB/s (measured)	143 GB/s (measured)

Lattice Boltzmann Method: Algorithm

PUSH

1. initialise
2. compute forces
3. compute $f^{(eq)}$
4. collide (local)
5. stream (non-local)
   - i.e. requires synchronisation
6. impose bcs.
7. compute macroscopic quantities

PULL (see [Rinaldi et al., 2012,])

1. initialise
5. stream
   i.e. read values from host into new location
6. impose bcs.
7. compute macroscopic quantities
2. compute forces
3. compute $f^{(eq)}$
4. collide

Lattice Boltzmann Method: GPU implementation

CPU implementation: push

```
for(int dir = 0; dir < 9; ++dir) {
  Xnew=X+cx[dir]; // Stream x `PUSH'
  Ynew=Y+cy[dir]; // Stream y
  pop [dir][Xnew][Ynew] = pop_old[dir][X][Y] * (1 - omega)
      + pop_eq[dir] * omega + force_latt[dir];  // Collide
}
```

GPU implementation: pull

```
int size=Nx*Ny;
for(dir = 0; dir < 9; ++dir) {
Xnew=X-d_cx[dir]; // Stream x `PULL'
Ynew=Y-d_cy[dir];
pop_local[dir]  = pop_old[dir*size+Ynew*Nx+Xnew];
}
```

Memory Arrangement

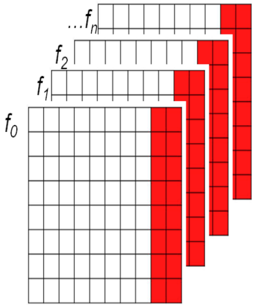
- Code is parallelized such that one thread will perform the complete LBM algorithm at one spatial location $f(x)_i$ in the fluid domain.
- Each thread stores values of $f(x)_i$ and an integer tag denoting boundary type. Density ρ and velocity u_i are only stored if output.
- All is stored in a struct within registers to minimise high latency access to DRAM once initially loaded
- within DRAM it is common practise to 'flatten' multiple dimension

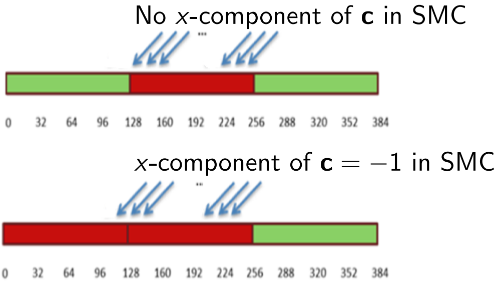
```
f[dir*Nx*Ny+Y*Nx+X]
```

Instruction Level Parallelism

More ILP at expense of occupancy improves performance [Volkov, 2010,]

- Operating on all indices of f in one thread helps to hide latency
- Use structs of arrays access to f. Coalesced access therefore only depends on the x component of the discrete velocity direction ($\mathbf{c}$)
- Cache hit rate is low as we don't have repeat accesses ($< 7\%$ in L2)
- .. so instead maximise register use (which lowers occupancy)





Maximising use of registers

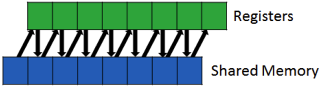
- maximum of 65536 registers and can host up to 2048 threads

$$\frac{\text{registers}}{\text{thread}} \times \frac{\text{threads}}{\text{block}} \times \frac{\text{blocks}}{\text{SMX}} \leq 65536$$

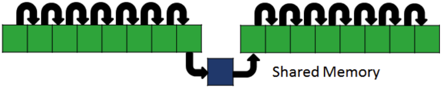
- in 3D we need 19 loads for $f(x)$ and 19 stores
    - we also need 1 integer (boundaries) and 4 macroscopic quantities
    - so total of 43 registers needed per thread
- for high registers/thread, large block sizes are impractical.
    - e.g block size of 1024 means only a single block would run
    - [Obrecht et al., 2013,] recommend maximum block size of 256

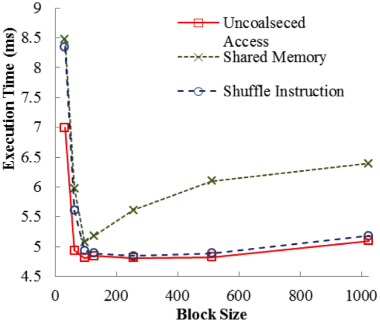
Shared mem shuffle operation on Keplers

Shared Mem

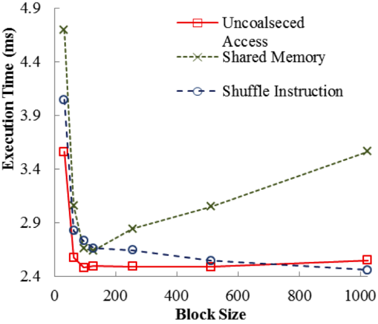


Shuffle



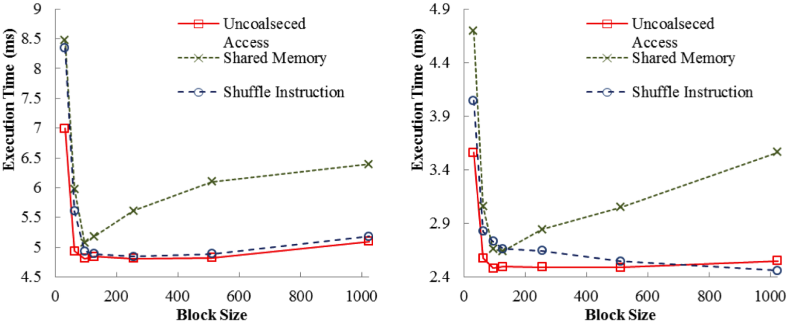


(a) K5000M



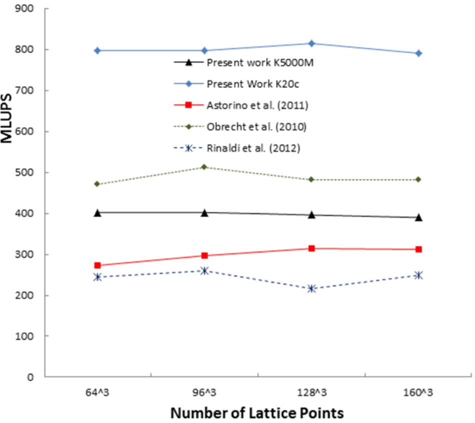
(b) K20c

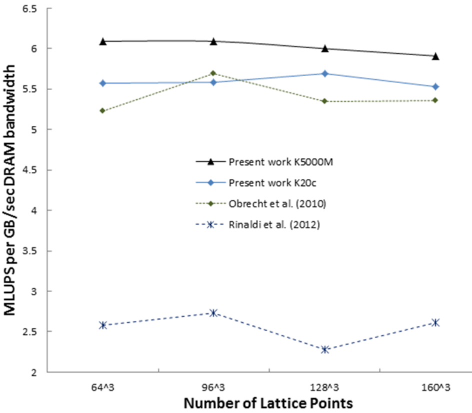
Shared mem shuffle operation on Keplers



- ILP is more efficient than Shared memory or shuffle
- large block sizes are impractical for LBM

Overall Performance





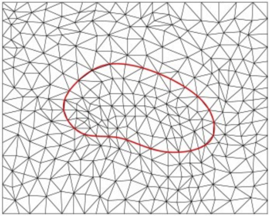
- Peak 814MLUPS and 402MLUPS for K20c and K5000M respectively
    - vs theoretical max of 892MLUPS and 412MLUPS based on measured bandwidth
- Compared well to other implementations (albeit on other h/w)
- [Rinaldi et al., 2012,] and [Astorino et al., 2011,] use Shared Mem.

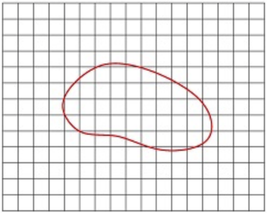
Immersed Boundary method

Immersed Boundaries (1977-)

Fluid equations solved on an Eulerian mesh. The geometrical complexity is embedded using an Lagrangian mesh via a system of singular forces.

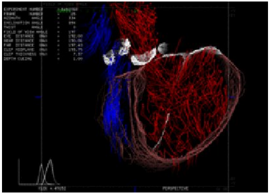
- Original Motivation of Charles Peskin [Peskin, 1977,]
  - Preserve efficient high order (Cartesian) solver
  - Define arbitrary mesh shape
- An alternative to body fitted mesh
  - Not a replacement, but a valuable tool





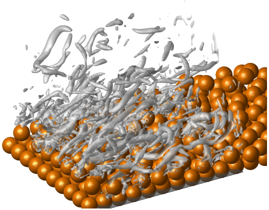
Original vs current approach

- [Peskin, 1977,]



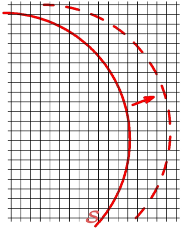
- Applied to a complete heart fluid system
- Peskin idea: model the boundary as a system of inextensible spring & dampers
- The *elastic* forces that restore the *true* and *actual* boundary position are supplied to the *rhs* of the momentum equations in terms of body forces.

- [Pinelli et al., 2010,] (no springs) has been introduced and applied (CIEMAT): no k stiffness introduced



- Modified approach has some advantages:
  - ◇ Moving boundary
  - ◇ Deformable boundary
- Sharing a drawback:
  - ◇ pressure correction error

Immersed Boundary: basics (I)



Along $\mathcal{S}$ $u = u^{(d)}$

$$\frac{\partial u}{\partial t} = rhs + f, \quad f = \frac{u^{(d)} - u^n}{\Delta t} - rhs \ \ \forall \vec{x} \ \in \mathcal{S}$$

algorithm:

1. given f update position of Lagrangian markers (boundary shape)
2. advance momentum equation without boundary induced body forces (u^* on $\mathcal{S}$)
3. compute f as a function of the difference $u^{(d)} - u^*$
4. repeat momentum advancement with f
5. compute for pressure correction, project velocity field
6. goto 1.

Immersed Boundary: basics (II)

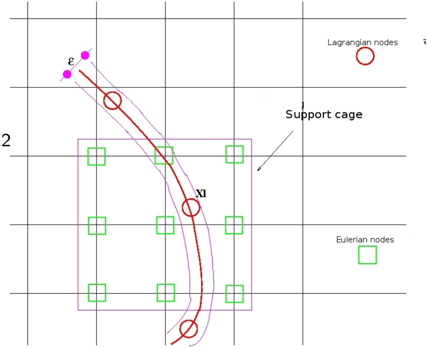
In discrete form in 2D

Interpolation

$$U^*(x_l) = \sum_{i,j} u^*(x_{i,j})\delta_h(x_{i,j} - X_l)\Delta x^2$$

Spread (convolution)

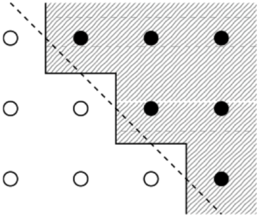
$$f(x_{i,j}) = \sum_{l\in\Omega_l} F(X_l)\delta_h(x_{i,j} - X_l)\epsilon\Delta l$$



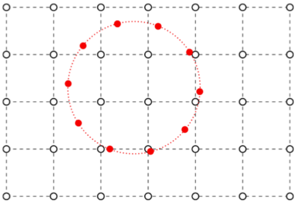
- Epsilon is the key to the accuracy
- it can be considered to represent the physical width of the surface.
- and it's computation guarantees that *interpolation*(*spread*(**F**))=**F**

Coupling IB with LB

- Bounceback does not offer high accuracy
  - assumes ‘stair-step’ surface
  - and is problematic for moving/flexible boundaries



- Inherent suitability of LB to IB
  - Lattice already uniform Cartesian
  - Poisson-free ! $\longrightarrow$ No pressure correction drawback



Lattice Boltzmann Method: Algorithm

1. initialise
2. set forces to zero
3. find velocity field in absence of object (call LBM)
4. compute support not required if not moving
5. find epsilon Costly, & also not required if not moving
6. interpolate fluid velocity onto Lagrangian marker
7. compute required force for object
8. spread force onto lattice
9. (solve other equations: collision, tension, bending, inextensibility)
10. find velocity field with object (call LBM again)
11. compute macroscopic quantities

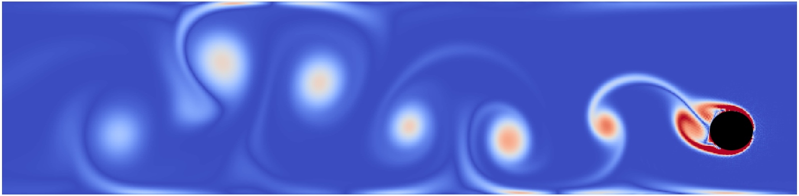
(see [Favier et al., 2013,] for full details)

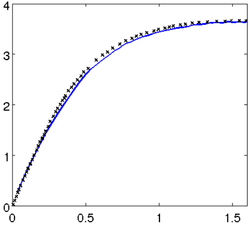
Rigid Particles: validation 1

Validation on finite-differences DNS [Uhlmann, 2005,]

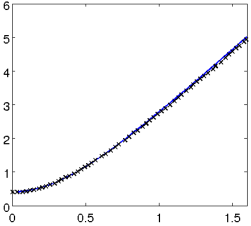
Falling particle under gravity $\rho_p/\rho_f = 8$, $Re = 165$

Starting at rest, no slip walls, gravity along x





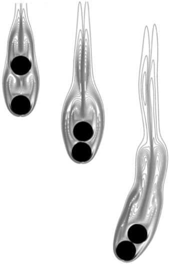
settling velocity

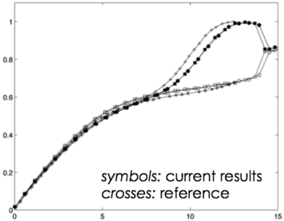


x-position

Rigid Particles: validation 2

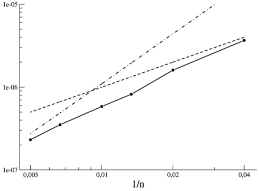
Kissing/Tumbing particles [Uhlmann, 2005,]

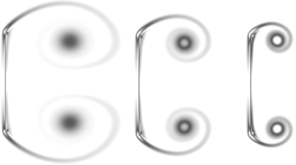


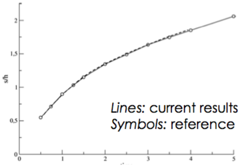


(a) u/U_{max} (parallel to gravity).

Impulsively moved flat plate [Koumoutsakos and Shiels, 1996,]

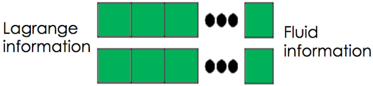




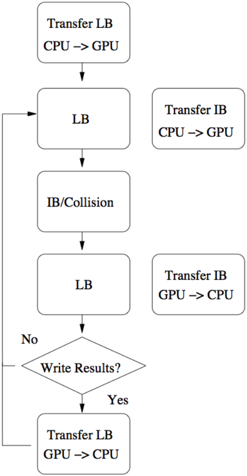


Immersed boundary + GPU

- Transactions for each point of the object are coalesced
  - each point only needs information about itself
- Transactions moving data between fluid and boundary are random with much higher cache use.

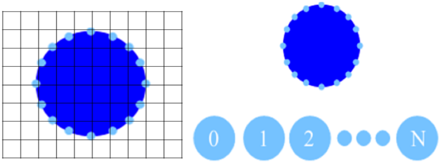


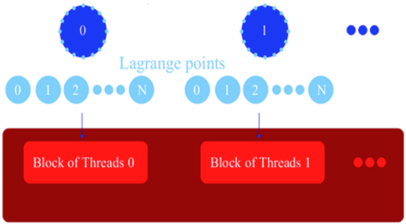
- Transfer from host to IBM kernel can be started simultaneously to hide latency
  - exploit capability to overlap memory transfers with executions
- spreading operation is a problem (atomic add)



Small vs. large objects

- Objects are treated according to size; number of Lagrangian markers (nlag)
- For small objects (nlag $<$ 1024) we assign one block per object
  - generally the case for 2D
- For large objects (nlag $>$ 1024), we need to launch a kernel for each object
  - e.g. for a sphere radius $r = 20$ we need nlag 4000





Flexible beating filament I

Apply inextensibility condition to the filament

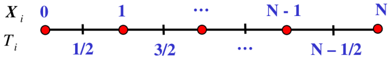
$$\begin{cases} \dfrac{\partial \mathbf{X}}{\partial t} = Interpol(\mathbf{u}) & \text{Kinematic condition} \\ \mathbf{F}_h = -Interpol(\mathbf{f}) & \text{Immersed boundary} \\ \rho \dfrac{\partial^2 \mathbf{X}}{\partial t^2} = \dfrac{\partial}{\partial s}(T \dfrac{\partial \mathbf{X}}{\partial s}) - \dfrac{\partial^2}{\partial s^2}(K_B \dfrac{\partial^2 \mathbf{X}}{\partial s^2}) + \rho \mathbf{g} - \mathbf{F}_h & \text{Solid momentum} \\ \nabla \cdot \mathbf{u} = 0 & \text{Incompressible condition} \\ \dfrac{\partial \mathbf{X}}{\partial s} \cdot \dfrac{\partial \mathbf{X}}{\partial s} = 1 & \text{Inextensibility condition} \end{cases}$$

following method of [Huang et al., 2007,]

- $\mathbf{f}$ is the force required by the fluid to verify the b.c
- $\mathbf{F}_h$ is the hydrodynamic force resulting from having applied the b.c.

Flexible beating filament II

- Application to Lattice-Boltzmann solver
- Staggered discretization of the Lagrangian space ($\mathbf{X}$ and T)



$$\frac{\mathbf{X}^{n+1} - 2\mathbf{X}^{n} + \mathbf{X}^{n-1}}{\Delta t^2} = [D_s(T^{n+1} D_s \mathbf{X}^{n+1})] + (F_b) + Fr\frac{\mathbf{g}}{g} - \mathbf{F}^n$$

$$D_s\mathbf{X}^{n+1} \cdot D_s\mathbf{X}^{n+1} = 1;$$

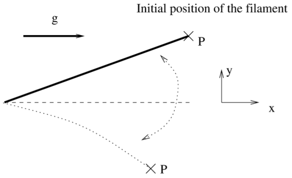
- Tension computed via iterative Newton-Raphson loop (by computing the exact Jacobian)
- For the initial guess, it's possible to derive a very good estimate for the tension, by incorporating the inextensibility condition in the momentum equations:

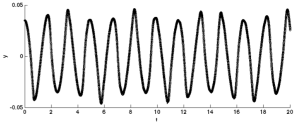
$$\frac{\partial^2 T^{n+1}}{\partial s^2} - \left(\frac{\partial^2 \mathbf{X}^n}{\partial s^2} \cdot \frac{\partial^2 \mathbf{X}^n}{\partial s^2}\right) T^{n+1} = -\frac{\partial \mathbf{X}^n}{\partial s} \cdot \frac{d\mathbf{F}_h^n}{ds} - \rho \frac{\partial \dot{\mathbf{X}}^n}{\partial s} \cdot \frac{\partial \dot{\mathbf{X}}^n}{\partial s} + \frac{\partial \mathbf{X}^n}{\partial s} \cdot \frac{\partial^3}{\partial s^3}\left(K_B \frac{\partial^2 \mathbf{X}^n}{\partial s^2}\right)$$

$$\Longrightarrow AT^{n+1} = rhs^n \quad \text{where A is a tridiagonal matrix}$$

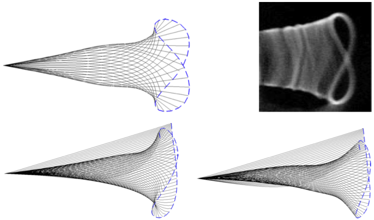
flexible filaments: validation 1

Without fluid (falling under its own weight)





With fluid (for different rigidities we observe correct 'kick' of free end)

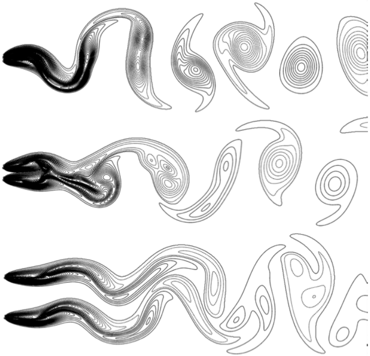


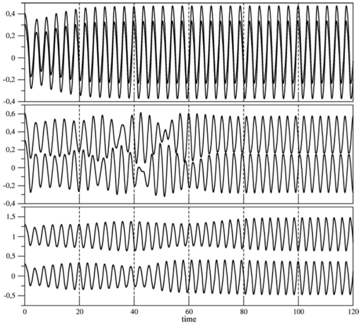
(a) Totally flexible filament ($K_B = 0$). (b) Rigid filament ($K_B = 0.001$).

(see [Favier et al., 2013,] for full details)

flexible filaments: validation 2

Filament interactions: 2 filaments



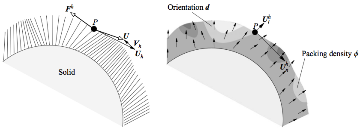


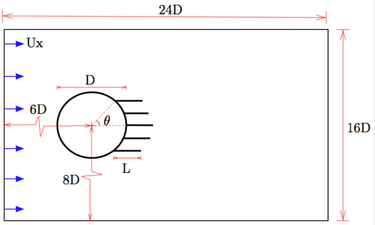
correct phase dependence on separation of filaments

(see [Favier et al., 2013,] for full details)

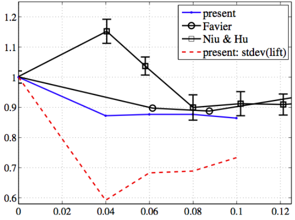
flexible filaments: investigation

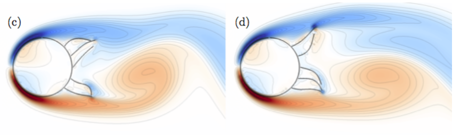
Cylinder coated with filaments: drag reduction [Revell et al., 2013,]





[Favier et al., 2009,]





EU funded project on this topic just started: PELSKIN

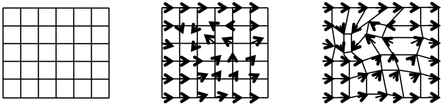
(temp web address : http://195.83.116.187/pelskin_web/)

Realtime LBM

Realtime output

several OpenGL visualization techniques are implemented

- Contour colour map (for velocity magnitude) is stored on the device
- Image Based Flow Visualization (IBFV) simulates advection of particles through an unsteady vector field (macroscopic u field)
  - instead of particle seeding which can be hit or miss
  - noise textures are used to represent dense set of particles, which are advected forwards using texture mapping
- previous frame M is textured onto a distorted mesh and blended with random noise texture N according to blending factor α



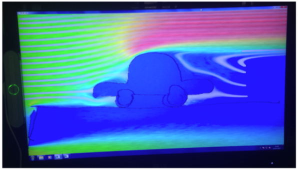
  - velocities used to displace mesh vertices using forwards integration
  - noise texture does not affect flow
  - mesh resolution may be coarser than lattice

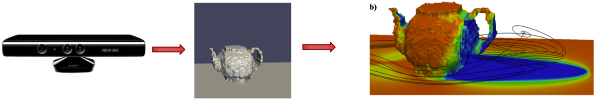
$$M_i(\mathbf{x}) = (1-\alpha)M_{i-1}\left(\mathbf{x} + \mathbf{u_i}\left(\mathbf{x}, t_i\right)\delta t\right) + \alpha N_i(\mathbf{x})$$

- Dye injection uses a similar method
  - can be more intuitive

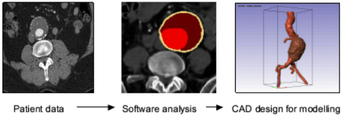
uses for Realtime?

- Initially started out as a gimic, used for teaching and open days

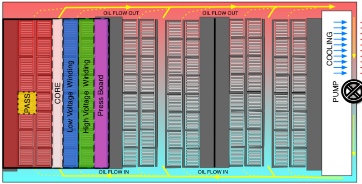




- Attracting increased attention in some areas
    - training for medical surgical proceedure
- for complex industrial applications where ‘intuition’ is missing



mimit MANCHESTER
INTEGRATING MEDICINE & INNOVATIVE TECHNOLOGY



Conclusions

- LBM solver implemented on Kepler hardware with some optimizations
  - 814 MLUPS peak on K20c
- IB-LB solver validated for rigid and flexible geometry
  - various flow physics investigations underway
  - context of EU project PELSKIN
  - BBSRC project on 'protein manufacturability'
- Realtime version of the solver available
  - with various visualization options
  - exploring potential applications

References I

Astorino, M., Sagredo, J. B., and Quarteroni, A. (2011).
A modular lattice boltzmann solver for gpu computing processors.

Bhatnagar, P., Gross, E., and Krook, M. (1954).
A model for collision processes in gases. i: small amplitude processes in charged and neutral one-component system.
Physical Review, 94:511–525.

Favier, J., Dauptain, A., Basso, D., and Bottaro, A. (2009).
Passive separation control using a self-adaptive hairy coating.
Journal of Fluid Mechanics, 627:451–483.

Favier, J., Revell, A., and Pinelli, A. (2013).
A lattice boltzmann - immersed boundary method to simulate the fluid interaction with moving and slender flexible objects.
HAL, hal(00822044).

Frisch, U., B Hasslacher, B., and Pomeau, Y. (1986).
Lattice-gas automata for the navier-stokes equation.
Physical review letters.

Ghia, U., Ghia, K., and Shin, C. (1982).
High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method.
Journal of computational physics.

Grad, H. (1949).
On the kinetic theory of rarefied gases.
Communications on pure and applied mathematics.

Guo, Z., Zheng, C., and Shi, B. (2002).
forcing term lbm.
Physical review letters E, 65(4).

Hardy, J., Pomean, Y., and de Pazzis, O. (1973).
Time evolution of a two-dimensional model system.
Modelling Simul. Mater. Sci. Eng., 14:17461752.

Huang, W., Shin, S., and Sung, H. (2007).
Simulation of flexible filaments in a uniform flow by the immersed boundary method.
Journal of Computational Physics, 226(2):2206 – 2228.

References II

Jiang, B., Lin, T., and Povinelli, L. (1994).
Large-scale computation of incompressible viscous flow by least-squares finite element method.
Computer Methods in Applied Mechanics and Engineering, 114.

Koumoutsakos, P. and Shiels, D. (1996).
Simulations of the viscous flow normal to an impulsively started and uniformly accelerated flat plate.
Journal of Fluid Mechanics, 328:177–227.

Latt, J. (2013).
Introduction to lattice boltzmann method.
http://www.youtube.com/watch?v=I82uCa7SHSQ.

Nie, X.B., S. X. and Chen, H. (2009).
Lattice-boltzmann/finite- difference hybrid simulation of transonic. flow.
AIAA-Paper 2009-139.

Obrecht, C., Kuznik, F., Tourancheau, B., and Roux, J.-J. (2013).
Efficient gpu implementation of the linearly interpolated bounce-back boundary condition.
Computers & Mathematics with Applications.

Peskin, C. S. (1977).
Numerical analysis of blood flow in the heart.
Journal of Computational Physics, 25(3):220–252.

Pinelli, A., Naqavi, I., Piomelli, U., and Favier, J. (2010).
Immersed-boundary methods for general finite-difference and finite-volume navier-stokes solvers.
Journal of Computational Physics, 229(24):9073 – 9091.

Raabe, D. (2004).
Overview of the lattice boltzmann method for nano- and microscale fluid dynamics in materials science and engineering.
Modelling Simul. Mater. Sci. Eng., 12.

Revell, A., Favier, J., and Pinelli, A. (2013).
Drag reduction of flow around a cylinder with attached flexible filaments.
In *ERCOFTAC international symposium onUnsteady separation in fluid-structure interaction*.

Rinaldi, P., Dari, E., Vénere, M., and Clausse, A. (2012).
A lattice-boltzmann solver for 3d fluid simulation on gpu.
Simulation Modelling Practice and Theory, 25:163–171.

References III

Shan, X., Yuan, X.-F., and Chen, H. (2006).
Kinetic theory representation of hydrodynamics: a way beyond the navierstokes equation.
Journal of Fluid Mechanics, 550(1):413–441.

Uhlmann, M. (2005).
An immersed boundary method with direct forcing for the simulation of particulate flows.
Journal of Computational Physics, 209(2):448–476.

Volkov, V. (2010).
Better performance at lower occupancy.
In *Proceedings of the GPU Technology Conference, GTC*, volume 10.

Zou, Q., H. X. (1997).
On pressure and velocity boundary conditions for the lattice boltzmann bgk model.
Phys. Fluids, 9(6).