Hemodynamics & Large-Scale Challenges

AC290r - Simone Melchionna

National Research Council Italy & Lexma Technology LLC

April 16, 2019

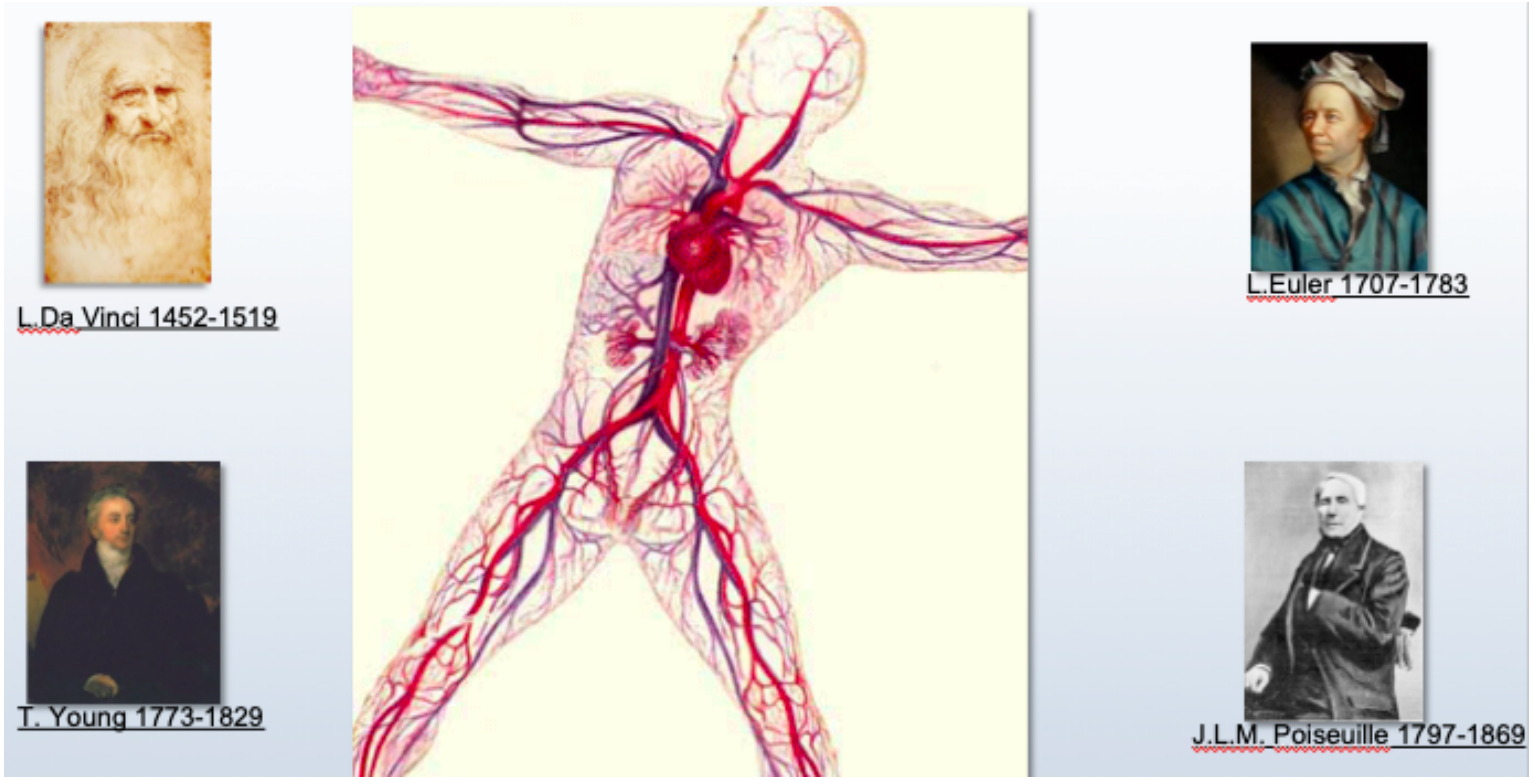
Discussion on previous GPU runs

Questions ? Comments ?

A note on Fault Tolerance computing:

- The paradigm tasks $\leftrightarrow$ domain decomposition is rigid and not fault tolerant
- Thread-based models are more flexible
- In general for tightly coupled tasks not easy to comply with hardware failures

Hemodynamics: scope and objectives



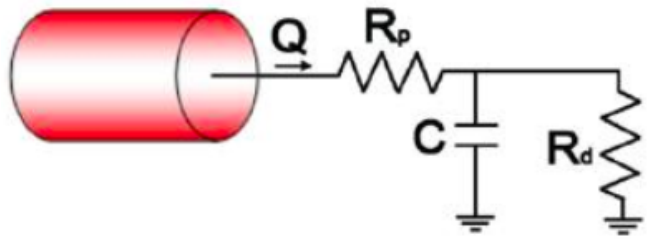
In vivo behaviour requires an holistic view for the in silico approach. Patient-specific requirement, the modeling of compex multi-scale behavior, time-to-diagnostics are critical factors.

At this time, virtual diagnostics starts being recognized as the future of precision medicine: cheaper, better and objective.

Physiological boundary conditions

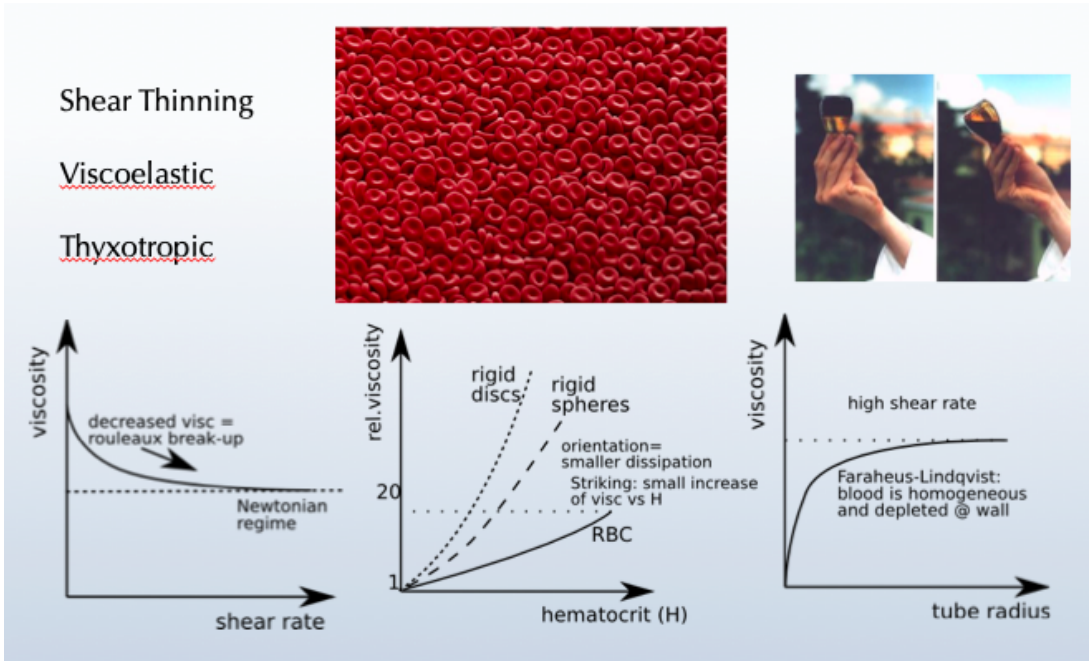
To reproduce in vivo flow requires to include the physiology feedback from downstream the outlet.
Remember that flow and electric circuits have a strict analogy:
$\Delta p \rightleftarrows \Delta V$, $Q \rightleftarrows I$, $\frac{\nu R^2}{L} \rightleftarrows R_c$, $\delta R \rightleftarrows C$, ...

A 3-elements Windkessel model includes:
R_p: resistance to the flow just downstream the outlet (proximal)
C: capacitor for vessel compliance of all the downstream vasculature
R_d: resistance for the capillaries and veins (distal)



R_p, C and R_d are tuned so that instantaneous outlet pressures match patient-specific physiology.
It's a low-pass filter, reflections from outlet are minimized.

Non-Newtonian rheology

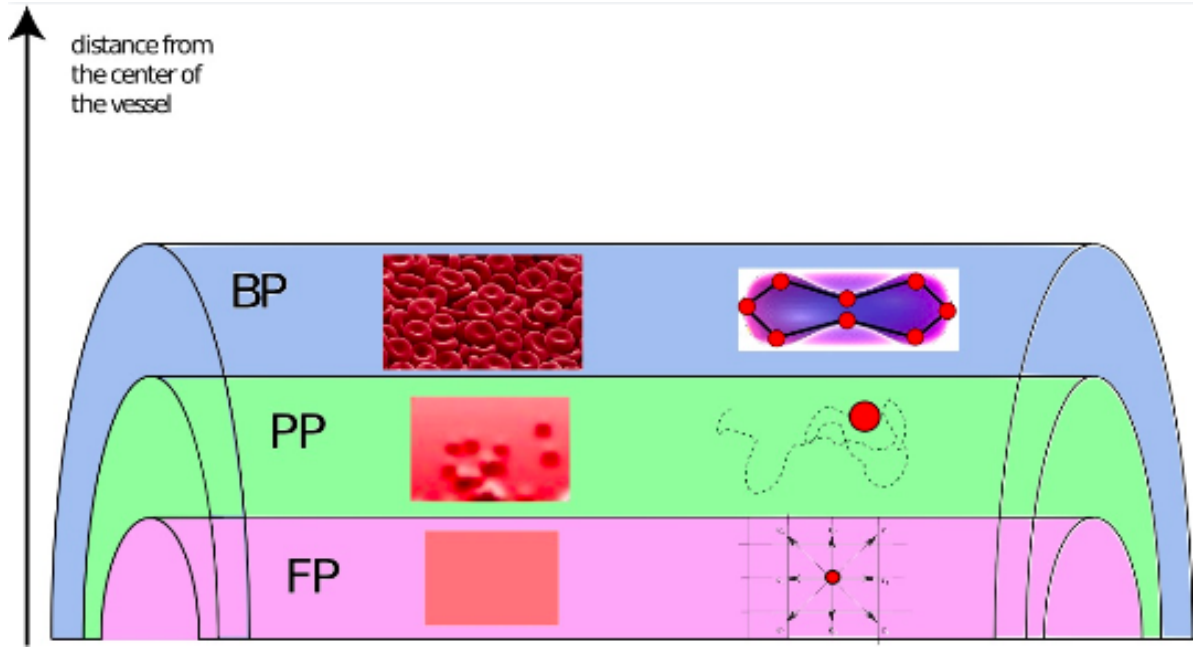


Biofluids often exhibit non-trivial rheology that can be explained based on the internal fluid structure.
Hemorheology has a key role in our functioning, such as controling the amount of oxygen delivered to capillaries and tissues (Fahraeus-Lindqvist effect).

A challenging scenario for CFD! Movie Time (bifurcation + RBC)

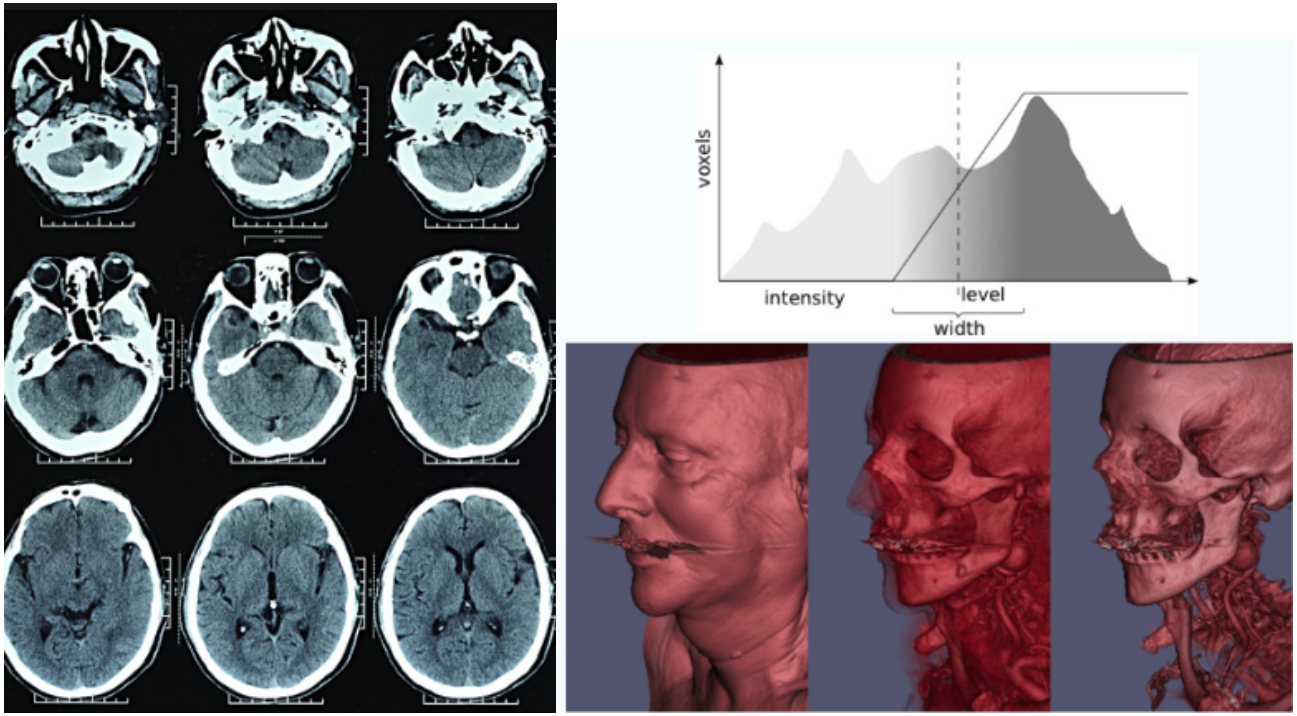
A multi-resolution scenario

In real vessels, most of action takes place near the walls, where the granularity of blood is important and reacts to the locally high shear: breakdown of rouleaux, shear thinning, skimming, delivery of nutrients, endothelial cell response, plaque build-up, ...



Away from walls, blood can be considered as Newtonian and approximated as a continuum.

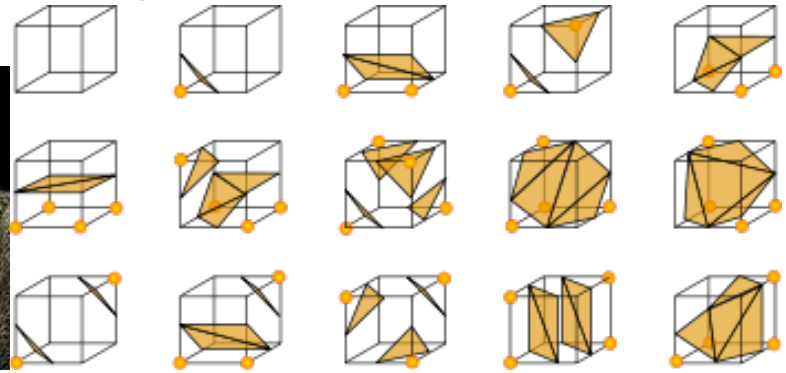
Medical Imaging: segmentation and marching cubes



Segmentation: to extract the Region of Interest by various techniques (filters, level sets, machine learning, ...)
Registration: to align two or more images of the same scene to improve signal or to determine time variations

Marching Cubes

The ability to transform 3D image (pixels in tones of grey) into a solid object can be done by defining a threshold.
To generate a set of triangles, we can then use Marching Cubes, probably the most used algorithm to generate solid models.



Based on values of the image on the cube corners, it selects one of the 15 cross-cuts of a given cube properly interpolated.
The result is a fully connected, hole-free, surface that has the same resolution of the original image.
The method is embarassing parallel and can be performed with the highest efficiency.

Recap: MUPHY usage

Example of run2.py:

```
from MagicUniverse import *
MagicBegins()

# define universe & actors
u = Universe()
s = Scale()
m = Mesh()
f = Fluid()
t = Tracker()

u.addItems([s,m,f,t])
u.setTitle('Periodic channel')
u.setNumberOfSteps(100)
u.setStateRestart(False)
u.setStateDumpFrequency(-1)
u.create()

s.set(name='MonoScale',
mesh=m, actors=[f,t])

# set various parameter
m.setRegularMesh(True)
m.setPeriodicity('100')

t.setDiagnosticFrequency(10)
t.setDataShow(density=True,velocity=True)
t.setMapDirections('zx')
t.setVtkDump(True, start=0,
frequency=10)

f.setName('Fluid')
f.setViscosity(1./6.)
f.setInletOutletMethod('closed')
f.setHomogeneousForce(1.e-4,0.,0.)

u.decorate()

for itime in u.cycle():
    u.animate()

MagicEnds()
```

Recap: MUPHY usage

Beware of parallel runs: any python operation is made by all tasks.
To operate from the master task:

```
myid = get_myproc()
if myid == 0:
    do_domething....
```

or to perform a parallel reduction (sum):

```
X = sumpara(12)
if myid==0:
    print ('parallel sum is :',X)
```

Recap: MUPHY Files

- Input

run2.py : runner
bgkflag.hdr : mesh header file
bgkflag.dat : mesh nodes
OR
bgkflag_0....N-1.dat and nodeownr.inp for user-defined partitioning on N tasks
bgfklag.ios : inlet/outlet boundary conditions

- Output

standard output to track simulation
DIRDATA_X : the folder with 1D and 2D data for density and velocity
DIRDATA_X/VTK/* : folder with files for visualization by frames (paraview)

Recap: modules and environment

module load gcc/7.1.0-fasrc01 openmpi/3.1.1-fasrc01
cuda/10.0.130-fasrc01

export MOEBIUS_ROOT=_path_the_MAGIC_folder

export PYTHONPATH=$PYTHONPATH:$MOEBIUS_ROOT/BACKEND/SHOP
export PYTHONPATH=$PYTHONPATH:$MOEBIUS_ROOT/BACKEND/SCRIPTS

git clone https://github.com/smelch000/AC290r.git

./run.py -x cpu
./run.py -x gpu
mpirun -np ... ./run.py -x cpu
mpirun -np ... ./run.py -x gpu

Working with physical units: (ideal) blood vessel

Consider a straight artery with a diameter of 5 mm, let's discretize with $\Delta x = 10^{-4} m$, knowing that blood streams at $U = 0.1$ m/s.

What is a LBM typical velocity (in LBM units)?

A value that provides stable results (e.g. next to the inlet) is $u = 0.1$ implying $\Delta t = (10^{-4} \times 10^{-1}/10^{-1})s = 100\mu s$.

If we want to cover a heart beat period ($\sim 1s$) we need to run for 10^4 steps to cover a few heart beats and get converged results. Taking for viscosity blood = water $\nu_{ph} = 10^{-6} m^2/s \rightarrow \nu = 10^{-2}$: value borderline for LBM stability.

To increase viscosity we need to increase resolution (eg $\Delta x = 5 \times 10^{-5}$m) and Δt will decrease proportionally (eg $\Delta t = 5 \times 10^{-5}$s).

Working with physical units

Memo: given Δx (grid spacing in MKS), $\mathbf{U}$ (characteristic velocity in MKS), and lattice velocity taken as a sweet spot value $u = 0.1$, the timestep is $\Delta t = \frac{\mathbf{u}}{\mathbf{U}} \Delta x$.

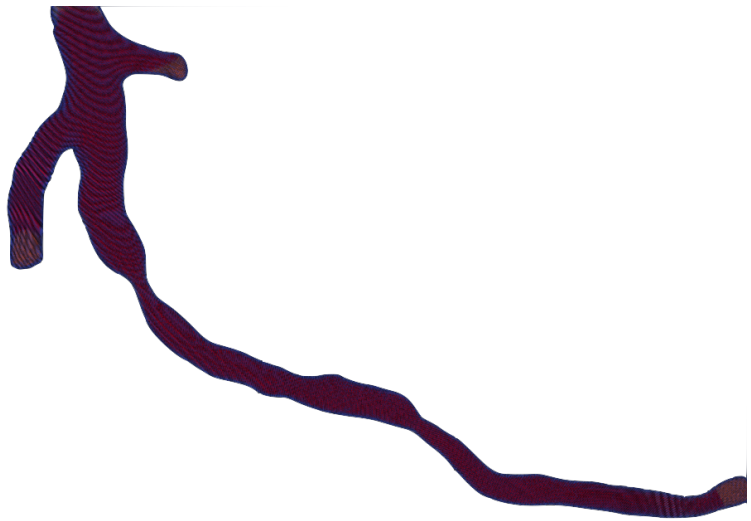
Given the physical viscosity ν_{ph}, the lattice value is $\nu = \frac{\Delta t}{\Delta x^2} \nu_{ph}$

```
from MagicUniverse import *
MagicBegins()

u = Universe()
f = Fluid()
... other actors instantiated...

...
u.setUnits('MKS')
u.setFinestResolution(1.e-3) # in MKS
u.setCharacteristicValues(Velocity=0.1, Pressure=1e+5, Mach=0.2) # in MKS
u.setSimulationTime( 20 ) # in MKS
u.create()
...actors parameters specified...
f.setViscosity( ...value in MKS... )

u.decorate()
for itime in u.cycle():
    u.animate()
```

This will print on standard output infos on units conversion.

Simulate a multibranched system & visualize

Case 106_SSUNG_WK is in the git repository and can be used to simulate a fairly complex behavior in a multi-branched artery. Note that:



- the small branches are very small (unresolved)
- we expect a strong unbalance of intrinsic resistances for each branch
- the corrugated shape of the vessel (because this is the system "from inside")

How would you impose boundary conditions on such system ?
Let's estimate the computational resources required to simulate it (10 cm long, 5 mm wide)

Inlet/Outlet specification (.ios file)

Head of file bgkflag.ios:
4
1 inlet pressure -1 0 0 1. 0. 0. 0.1 : inlet=>fluid search direction, velocity unit vector, velocity magnitude
2 outlet density 0 1 0 0. 1. 0. 1. : inlet=>fluid search direction, unused, density
3 outlet density 1 0 0 1. 0. 0. 0.97
4 outlet density 0 0 1 0. 0. 1. 0.98
....list of inlet & outlet nodes....
...

all values are expressed in code units.

Values can be overriden in python script:
f.setIOValue('inlet', 1, 0.12) # inlet/outlet, index of i/o, boundary value
which is useful to impose time-varying (eg pulsatile) conditions.

Preprocessing a stenotic vessel

Look at the ShapePainter.py script in the git repository (git clone https://github.com/smelch000/AC290r.git)

The script leverages VTK, a very useful toolkit for visualization and algorithmics, typically used by medical softwares (vtk.org).

The script uses the approach of "painting" pixels and then marching cubes.