

Interactive Educational Materials for Computational Tissue Engineering Using Jupyter Notebooks

Mojtaba Barzegari

Biomechanics Section, Department of Mechanical Engineering,
KU Leuven, Leuven, Belgium



Reproducible Educational Materials

- Reproducibility matters a lot in today's research (in computational sciences)
- What for education?
- Open source reproducible interactive educational materials
 - More fun as well as more comprehensive for students/learners
 - Easier to copy/distribute
 - Starting point for creating similar content

Jupyter Comes Into Play

- Jupyter notebooks: “killer app” for interactive computing
- Supporting content and the codes, side by side
- Eliminating the complexity of getting started
- Active and interactive classroom for faster learning
- Interactive assignments? (even auto-graded!)



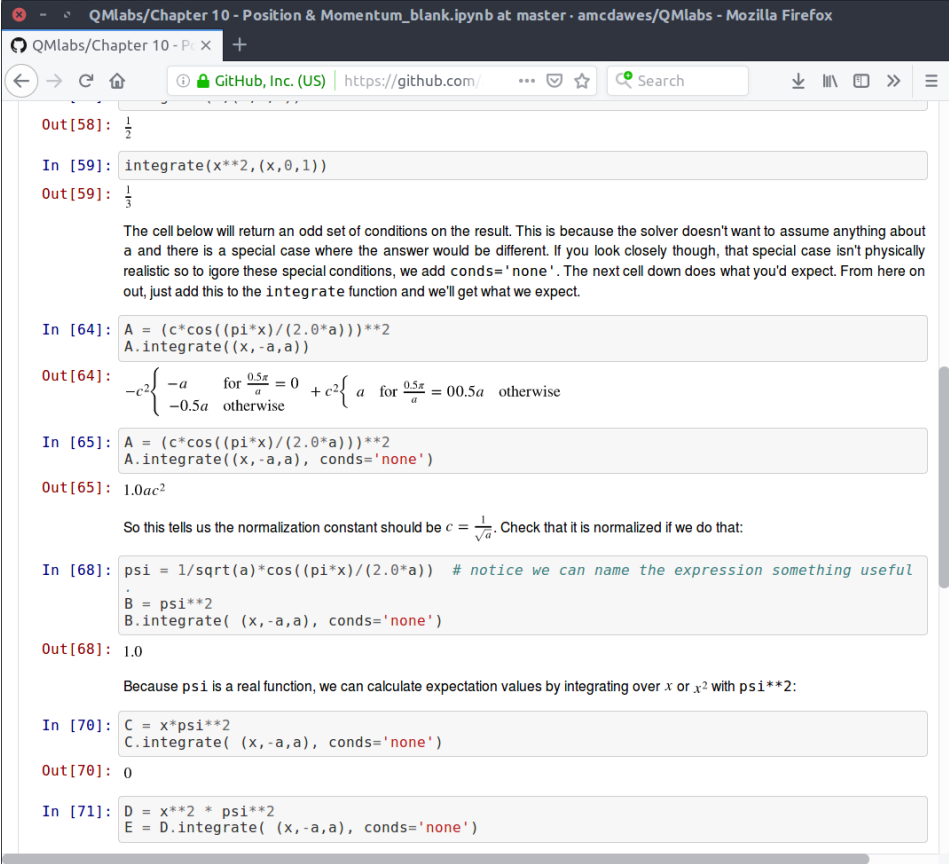
Countless Examples Already Available!

- Educational materials based on Jupyter notebooks are well adopted by open science community
(pic: UC Berkeley)



A Couple of Examples

- “Basic Physics” course
(High Point University)
- “Chemical Process Control” course
(University of Notre Dame)
- “Quantum Mechanics” course
(Pacific University)



The screenshot shows a Jupyter Notebook titled "QMLabs/Chapter 10 - Position & Momentum_blank.ipynb" in a Mozilla Firefox browser. The notebook contains several code cells with the following content:

```
Out [58]:  $\frac{1}{2}$ 

In [59]: integrate(x**2, (x, 0, 1))

Out [59]:  $\frac{1}{3}$ 

The cell below will return an odd set of conditions on the result. This is because the solver doesn't want to assume anything about a and there is a special case where the answer would be different. If you look closely though, that special case isn't physically realistic so to ignore these special conditions, we add conds='none'. The next cell down does what you'd expect. From here on out, just add this to the integrate function and we'll get what we expect.

In [64]: A = (c*cos((pi*x)/(2.0*a)))**2
          A.integrate((x, -a, a))

Out [64]: 
$$-c^2 \begin{cases} -a & \text{for } \frac{0.5\pi}{a} = 0 \\ -0.5a & \text{otherwise} \end{cases} + c^2 \begin{cases} a & \text{for } \frac{0.5\pi}{a} = 0 \\ 0.5a & \text{otherwise} \end{cases}$$


In [65]: A = (c*cos((pi*x)/(2.0*a)))**2
          A.integrate((x, -a, a), conds='none')

Out [65]:  $1.0ac^2$ 

So this tells us the normalization constant should be  $c = \frac{1}{\sqrt{a}}$ . Check that it is normalized if we do that:

In [68]: psi = 1/sqrt(a)*cos((pi*x)/(2.0*a)) # notice we can name the expression something useful
          B = psi**2
          B.integrate((x, -a, a), conds='none')

Out [68]: 1.0

Because psi is a real function, we can calculate expectation values by integrating over x or x^2 with psi**2:

In [70]: C = x*psi**2
          C.integrate((x, -a, a), conds='none')

Out [70]: 0

In [71]: D = x**2 * psi**2
          E = D.integrate((x, -a, a), conds='none')
```

An Interactive Example for Computational TE

- A set of Jupyter notebooks on computational mass transfer in TE
- Accompanied by prerequisites
- Initially developed for a course
- Freely available on GitHub

The screenshot shows the GitHub repository interface for 'mass-transport-tissue-engineering-spring2021' by user 'mbarzegary'. The repository is on the 'main' branch with 1 branch and 0 tags. It has 11 commits and was last updated on May 31. The file list includes:

File	Description	Time
img	add jupyter notebooks	7 months ago
.gitignore	add gitignore	7 months ago
0-Check-installation.ipynb	clear output for installation check	7 months ago
1-Introduction-to-Python-Programm...	add jupyter notebooks	7 months ago
2-Introduction-to-Sympy.ipynb	add transient notebook	7 months ago
3-Mass-Transfer-Cellular-Construct.ip...	add jupyter notebooks	7 months ago
4-Mass-Transfer-Krogh-Cylinder.ipynb	add jupyter notebooks	7 months ago
5-Numerical-Transient-Diffusion.ipynb	add transient notebook	7 months ago
README.md	Update README.md	6 months ago
installation_guide.pdf	initial commit	7 months ago
mymodule.py	add jupyter notebooks	7 months ago
presentation.pdf	add presentation file	6 months ago
requirements.txt	downgrade matplotlib	7 months ago

The README.md file is displayed below the file list, showing the title 'Mass Transport in Tissue Engineering, Spring 2021' and buttons to 'launch' or 'binder' the notebooks. The right sidebar contains sections for 'About' (Materials used in the sub-course "Mass transport in tissue engineering", Spring 2021), 'Releases' (No releases published), 'Packages' (No packages published), and 'Languages' (Jupyter Notebook 99.8%, Python 0.2%).

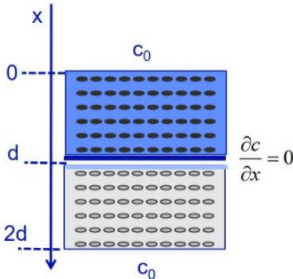


<https://github.com/mbarzegary/mass-transport-tissue-engineering-spring2021>

An Interactive Example for Computational TE

- Solved elaborated problems on modeling of mass transfer

Mass transfer in cellular construct



The problem is to solve the following equation

$$\frac{\partial c}{\partial t} = D \frac{\partial^2 c}{\partial x^2} - V_{max, cell} \rho_{cell} \quad (1)$$

For a steady state solution, we can neglect the left hand side term, so the equation will be

$$D \frac{\partial^2 c}{\partial x^2} - V_{max, cell} \rho_{cell} = 0 \quad (2)$$

We want to know the maximum depth of oxygen penetration.

We already know the value of C_1 , and by using the first boundary condition, we can obtain the value of C_2 as well. According to the first boundary condition, $c(x) = c_0$ at $x = 0$, so:

$$C_2 = c_0 \quad (9)$$

Now, by knowing the value of C_1 and C_2 and having no remained derivative in the equation, we can write the equation of the change of concentration of oxygen throughout the construct:

$$c(x) = \frac{V_{mp}}{D} \frac{x^2}{2} - \frac{V_{mp} d}{D} x + c_0 \rightarrow c(x) = \frac{V_{mp}}{D} \left(\frac{x^2}{2} - x d \right) + c_0 \quad (10)$$

It's the time to calculate the maximum depth of diffusion. It can be simply calculated by finding the value of x at which the concentration of oxygen is zero. Let's denote it by d_{max} (we can say the value of d equals d_{max} as well), and solve the derived equation:

$$x = d_{max}, c(x) = 0 \rightarrow 0 = \frac{V_{mp}}{D} \left(\frac{d_{max}^2}{2} - d_{max} d \right) + c_0 \quad (11)$$

$$\rightarrow d_{max} = \sqrt{2 c_0 \frac{D}{V_{mp}}} \quad (12)$$

Solve with Sympy

```
In [1]: from sympy import init_session
init_session(quiet=True)

In [2]: x, D, V, rho = symbols("x, D, V_m, rho", positive=True)
c = Function("c")
ode = D * c(x).diff(x, 2) - V * rho
Eq(ode, 0)


Out[2]: D * d^2 c(x) / dx^2 - V * rho = 0

In [3]: ode_sol = dsolve(ode)
ode_sol

Out[3]: c(x) = C1 + C2 * x + (V * rho * x^2) / (2 * D)
```

An Interactive Example for Computational TE

- Basic discussion on necessity and applications of numerical methods



mass-transport-tissue-engineering-spring2021 / 5-Numerical-Transient-Diffusion.ipynb

Transient two-dimensional diffusion equation

The two-dimensional diffusion equation is

$$\frac{\partial U}{\partial t} = D \left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} \right) \quad (1)$$

where D is the diffusion coefficient. A simple numerical solution on the domain of the unit square $0 \leq x < 1, 0 \leq y < 1$ approximates $U(x, y, t)$ by the discrete function $u(n)i, j$ where $x = i\Delta x$, $y = j\Delta y$ and $t = n\Delta t$. Applying finite difference approximations yields

$$\frac{u_{i,j}^{(n+1)} - u_{i,j}^{(n)}}{\Delta t} = D \left[\frac{u_{i+1,j}^{(n)} - 2u_{i,j}^{(n)} + u_{i-1,j}^{(n)}}{(\Delta x)^2} + \frac{u_{i,j+1}^{(n)} - 2u_{i,j}^{(n)} + u_{i,j-1}^{(n)}}{(\Delta y)^2} \right], \quad (2)$$

and hence the state of the system at time step $n+1$, $u_{i,j}^{(n+1)}$ may be calculated from its state at time step n , $u(n)i, j$ through the equation

$$u_{i,j}^{(n+1)} = u_{i,j}^{(n)} + D\Delta t \left[\frac{u_{i+1,j}^{(n)} - 2u_{i,j}^{(n)} + u_{i-1,j}^{(n)}}{(\Delta x)^2} + \frac{u_{i,j+1}^{(n)} - 2u_{i,j}^{(n)} + u_{i,j-1}^{(n)}}{(\Delta y)^2} \right]. \quad (3)$$

Consider the diffusion equation applied to a metal plate initially at temperature T_{cold} apart from a disc of a specified size which is at temperature T_{hot} . We suppose that the edges of the plate are held fixed at T_{cold} . The following code applies the above formula to follow the evolution of the temperature of the plate. It can be shown that the maximum time step, Δt that we can allow without the process becoming unstable is

$$\Delta t = \frac{1}{2D} \frac{(\Delta x \Delta y)^2}{(\Delta x)^2 + (\Delta y)^2}. \quad (4)$$

In the code below, each call to `do_timestep` updates the numpy array `u` from the results of the previous timestep, `u0`. The simplest approach to applying the partial difference equation is to use a Python loop:

```
fig = plt.figure()
for m in range(nsteps):
    u0, u = do_timestep(u0, u)
    if m in mfig:
        fignum += 1
        print(m, fignum)
        ax = fig.add_subplot(220 + fignum)
        im = ax.imshow(u.copy(), cmap=plt.get_cmap('hot'), vmin=Tcool, vmax=Thot)
        ax.set_axis_off()
        ax.set_title('%i.%i ms' % (m*dt*1000))
fig.subplots_adjust(right=0.85)
cbar_ax = fig.add_axes([0.9, 0.15, 0.03, 0.7])
cbar_ax.set_xlabel('$T$ / $K$', labelpad=20)
fig.colorbar(im, cax=cbar_ax)
plt.show()

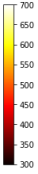
0 1
10 2
50 3
100 4
```

0.0 ms

6.3 ms

31.3 ms

62.5 ms



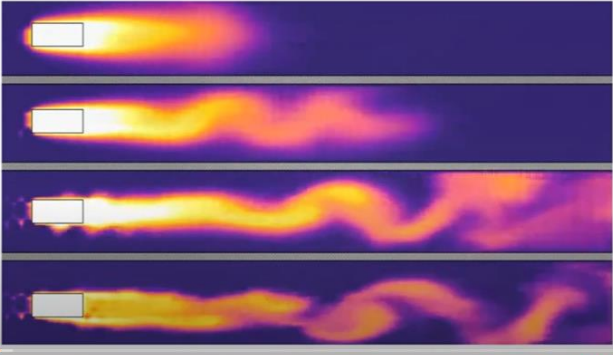
To set a common `colorbar` for the four plots we define its own `Axes`, `cbar_ax`, and make room for it with `fig.subplots_adjust`. The plots all use the same colour range, defined by `vmin` and `vmax`, so it doesn't matter which one we pass in the first argument to `fig.colorbar`.

The state of the system is plotted as an image at four different stages of its evolution.

An Interactive Example for Computational TE

- Recorded sessions and accompanying materials on YouTube

Convection Effect - Velocity

$$\frac{\partial c}{\partial t} = \nabla \cdot (D \nabla c) + \nabla \cdot (\mathbf{v}c)$$


Increasing v

Computer methods for mass transport in tissue engineering

39 views · May 31, 2021

1 0 SHARE SAVE ...

$$\frac{\partial u}{\partial t} = \nabla \cdot [D \nabla u] - \nabla \cdot (\mathbf{v}u) + f(u)$$

$$\rho c_p \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = \dot{q}_V$$

$$\frac{\partial C_{Mg}}{\partial t} = \nabla \cdot (D_{Mg}^e \nabla C_{Mg}) - k_1 C_{Mg} + k_2 C_{Film} [Cl]^2$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} - \nu \nabla^2 \mathbf{u} = -\nabla w + \mathbf{g}$$

Heat transfer in solids Mass transfer in corrosion process Navier-Stokes in fluid flow

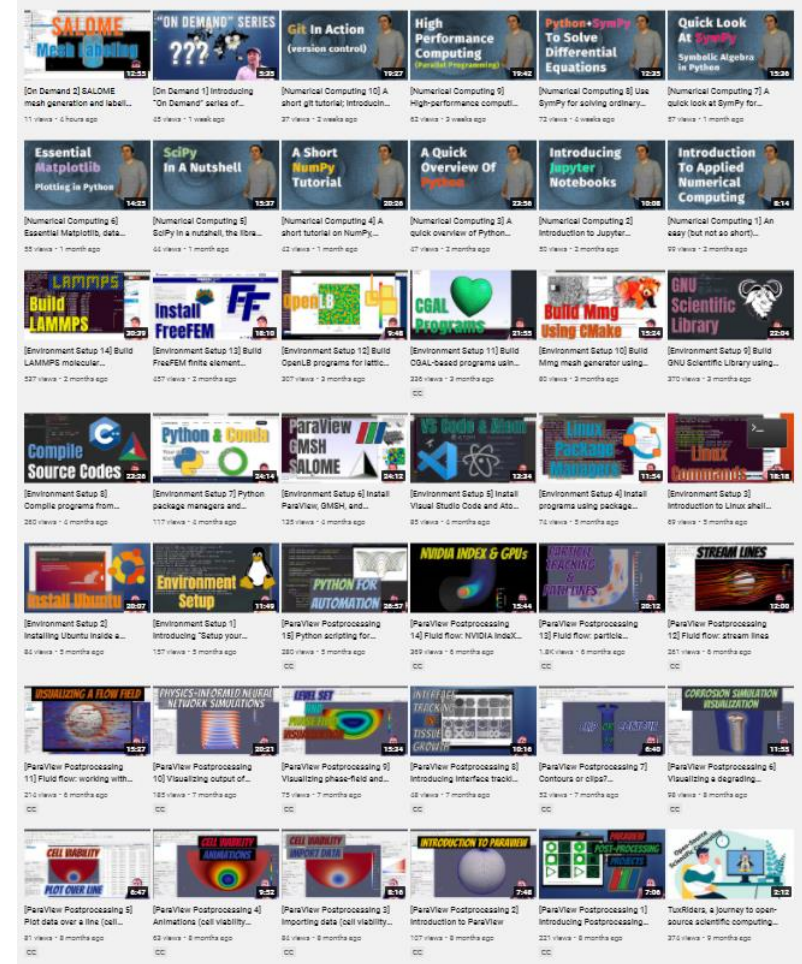
An Introduction to Reaction-Diffusion-Advection Equation

919 views · Apr 22, 2021

30 0 SHARE SAVE ...

More on Open Science

- A project to share experiences on computational modeling (mostly TE) using open-source tools
- More info: TuxRiders.com
YouTube.com/TuxRiders
- Using Jupyter for educational videos



Thank you for your attention!



<http://mbarzegary.github.io>



Mojtaba.Barzegari@kuleuven.be



[@MojBarz](https://twitter.com/MojBarz)