SIAM Conference on Computational Science and Engineering

MS78:
Teaching Computational Thinking and Practice

Prof. Lorena A. Barba

Mechanical and Aerospace Engineering

The George Washington University

Twitter: @LorenaABarba

Google "computational thinking"

Scholar About 7,060 results (0.08 sec)

[PDF] **Computational thinking**

JM Wing - Communications of the ACM, 2006 - www-cgi.cs.cmu.edu

• CT is interpreting code as data and data as code.• CT is using abstraction and decomposition in tackling a large complex task.• CT is judging a system's design for its simplicity and elegance.• CT is type checking, as a generalization of dimensional analysis. ...

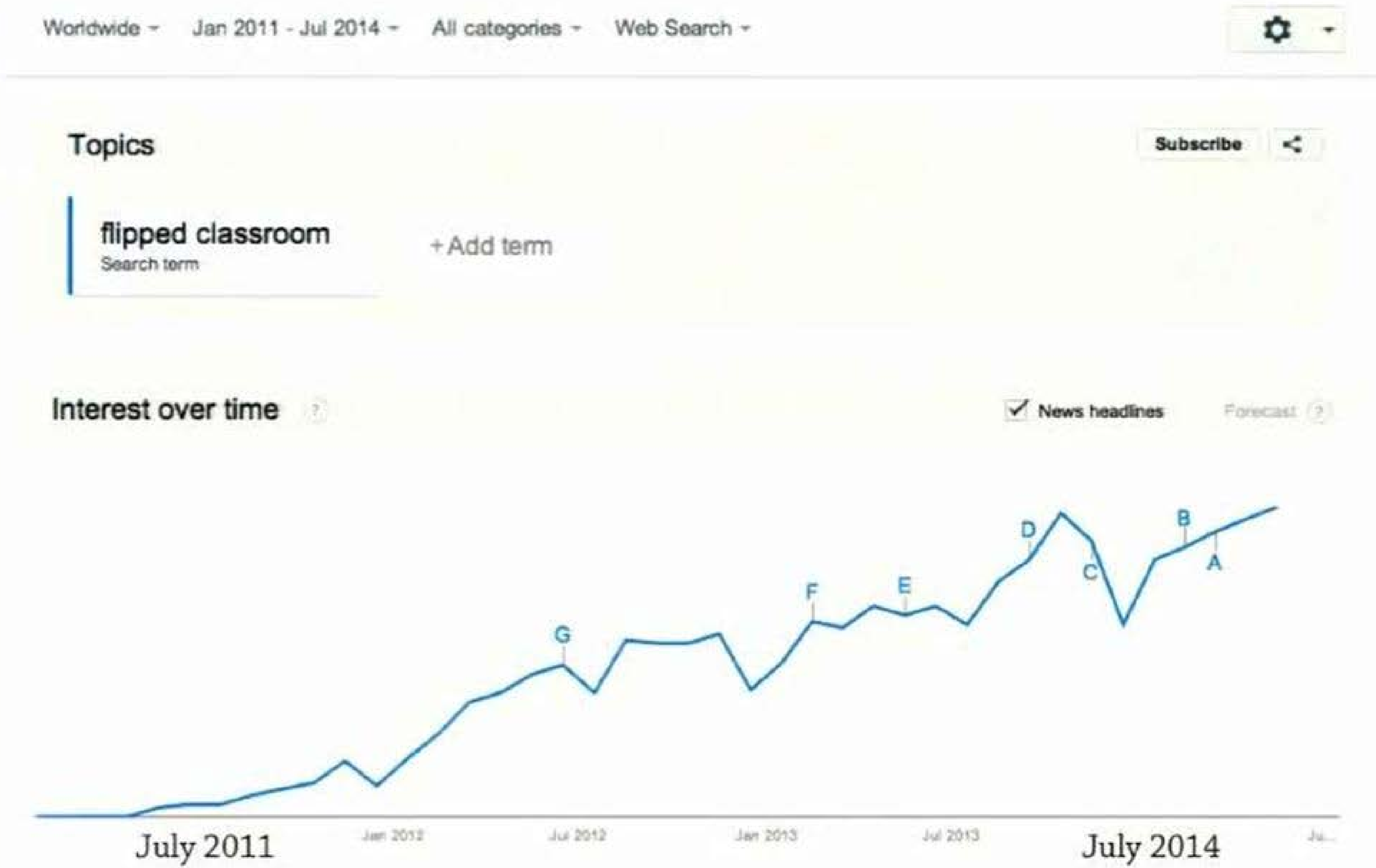
Cited by 1422 Related articles All 120 versions Import into BibTeX Save More

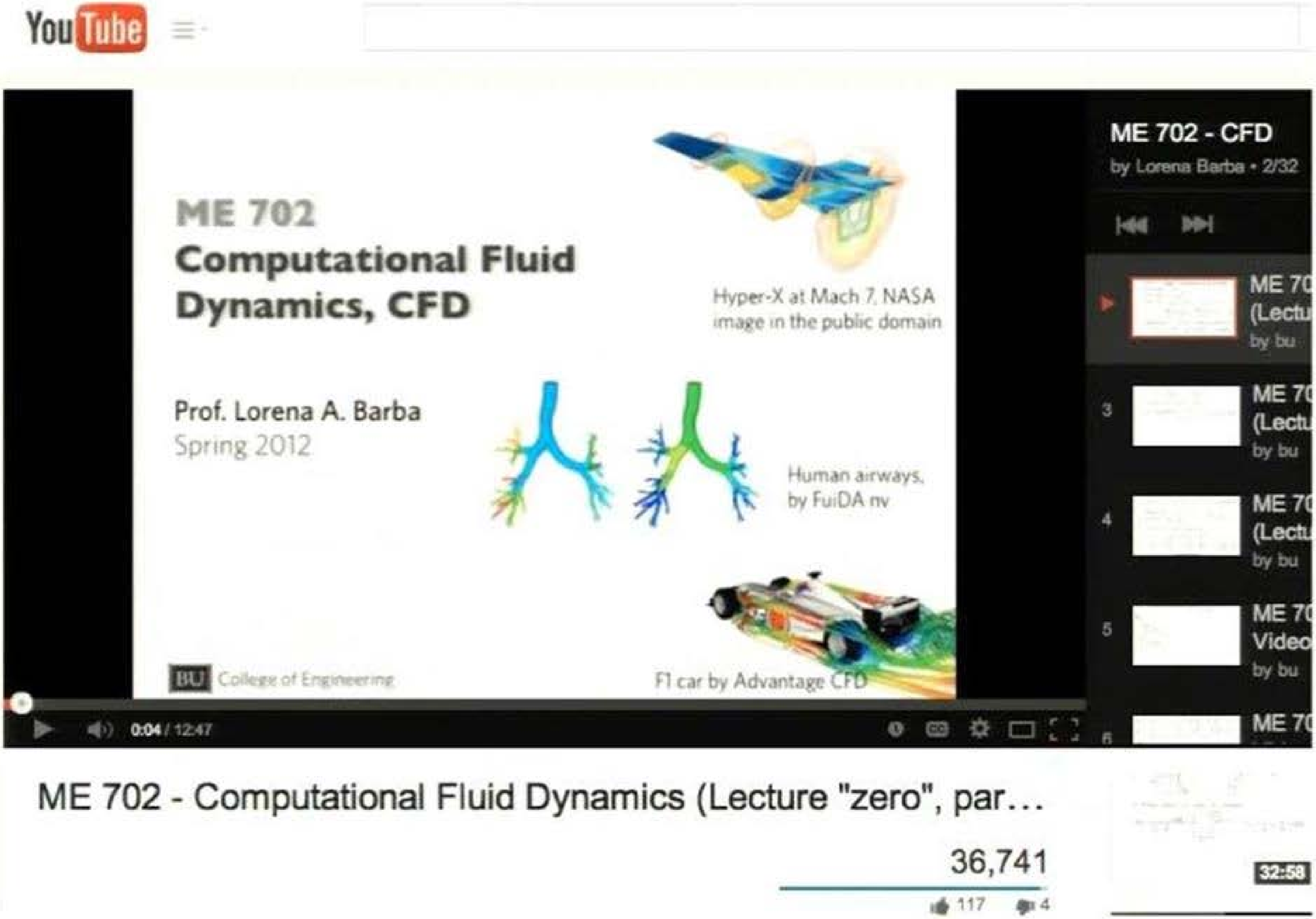
1,422


What's New
Fluid Mechanics (...
SUBSCRIBE F...
Computational Fl...
Lorena A. Barba; B
SUBSCRIBE F...
Fluid Mechanics (2010...
SUBSCRIBE F...
Information Te
SUBSCRIBE F
First-Semester Chines...
Boston University Moder...
SUBSCRIBE F...
PASI - Scientific ...
SUBSCRIBE F...
Second-Semester Chi...
Boston University Moder
SUBSCRIBE F...
Third-Semeste
SUBSCRIBE F
Top Downloads
1 Lecture 1-part 1: What is a fluid? Viscosity a... Fluid Mechanics (2010) - EN... L A Barba, Boston University 39:11
2 Welcome to the course for iTunes subscribers Computational Fluid Dynamic... Lorena A. Barba, Boston Univ... 15:35
3 CS632 - Cost Estimation Overview Information Technology Proje... Boston University 0:37
4 CS632 - Risk Management Information Technology Proje... Boston University 0:43
5 Lecture 1 movie credits Fluid Mechanics (2010) - EN... L A Barba, Boston University –
6 Lecture 4-part 1: Fluid statics, hydrostatic pr... Fluid Mechanics (2010) - EN... L A Barba, Boston University 49:46
7 Lecture 1-theory supplement: Viscosity and... Fluid Mechanics (2010) - EN... L A Barba, Boston University 15:16
8 Lecture 8: new schemes for convection, an... Computational Fluid Dynamic... Lorena A. Barba, Boston Univ... 1:22:40
9 Lecture 3: FD explicit/implicit methods: Cra... Computational Fluid Dynamic... Lorena A. Barba, Boston Univ... 1:09:21
Computational Fluid Dynamic... Lorena A. Barba, Boston Univ... –
Computational Fluid Dynamic... Lorena A. Barba, Boston Univ... 12:37

2009 to 2012

Flipped classroom — on Google Trends

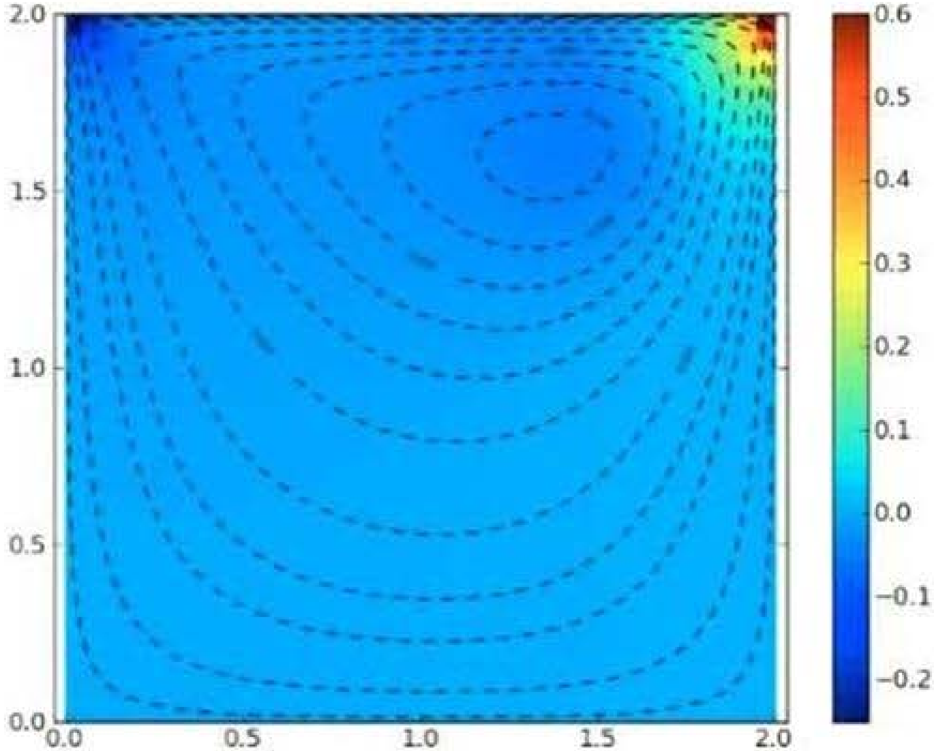




Added views > 300,000

Lorena A. **Barba group**

CFD Python



Cavity flow solution at Reynolds number of 200 with a 41x41 mesh.

Posted on 07.22.2013

Lessons

- Quick Python Intro
- Step 1
- Step 2
- CFL Condition
- Step 3
- Step 4
- Array Operations with NumPy
- Step 5
- Step 6
- Step 7
- Step 8
- Defining Function in Python
- Step 9
- Step 10
- Optimizing Loops with Numba
- Step 11
- Step 12

https://github.com/barbagroup/CFDPython

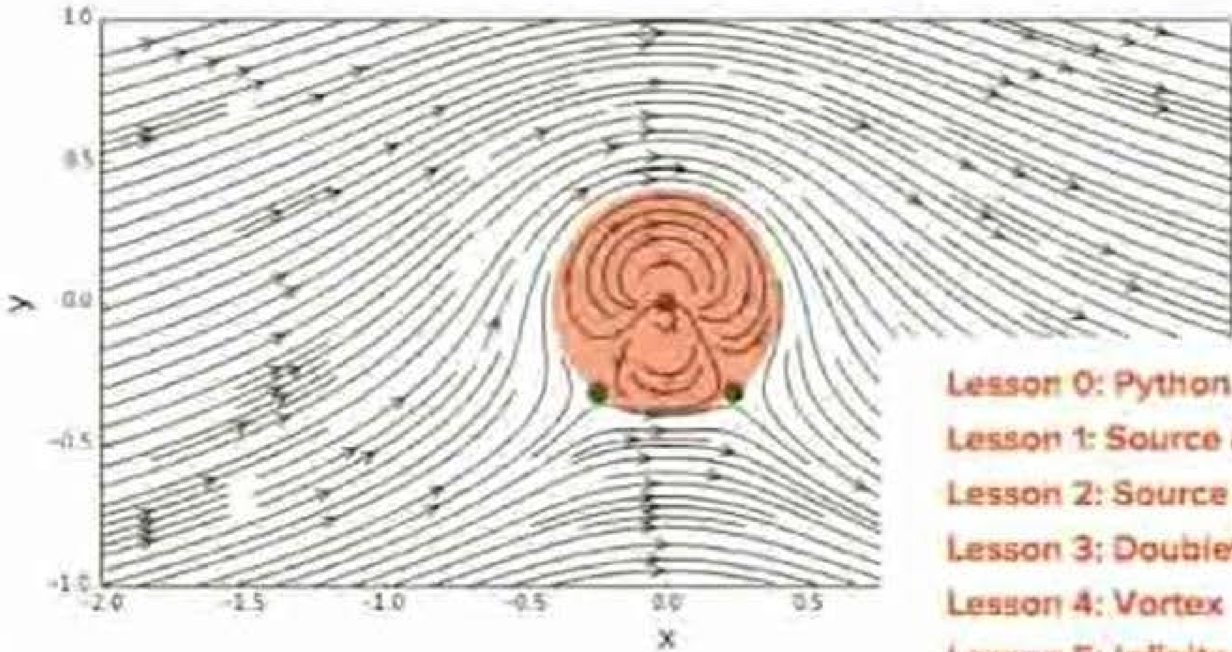
Why Python?

Language choice for beginners

Mannila & de Raadt (2006): criteria for a teaching language

- it was designed with teaching in mind
  (simple syntax, natural semantics)
- it can be used to apply physical analogies
  (provides multi-media capabilities)
- it offers a general framework
  (serving as a basis for learning other languages later)
- it promotes a new approach for teaching
  (augmented by principles, tools and libraries)

Announcing AeroPython!



Posted on 04.20.2014

- Lesson 0: Python Crash Course
- Lesson 1: Source & Sink
- Lesson 2: Source & Sink in a Freestream
- Lesson 3: Doublet
- Lesson 4: Vortex
- Lesson 5: Infinite row of vortices (student task)
- Lesson 6: Lift on a cylinder
- Lesson 7: Method of Images
- Lesson 8: Source Sheet
- Lesson 9: Flow over a cylinder with source panels
- Lesson 10: Source panel method
- Lesson 11: Source-vortex panel method

http://lorenabarba.com/blog/announcing-aeropython/

@LorenaABarba

Version 0.2 – February 2014

Lift on a cylinder

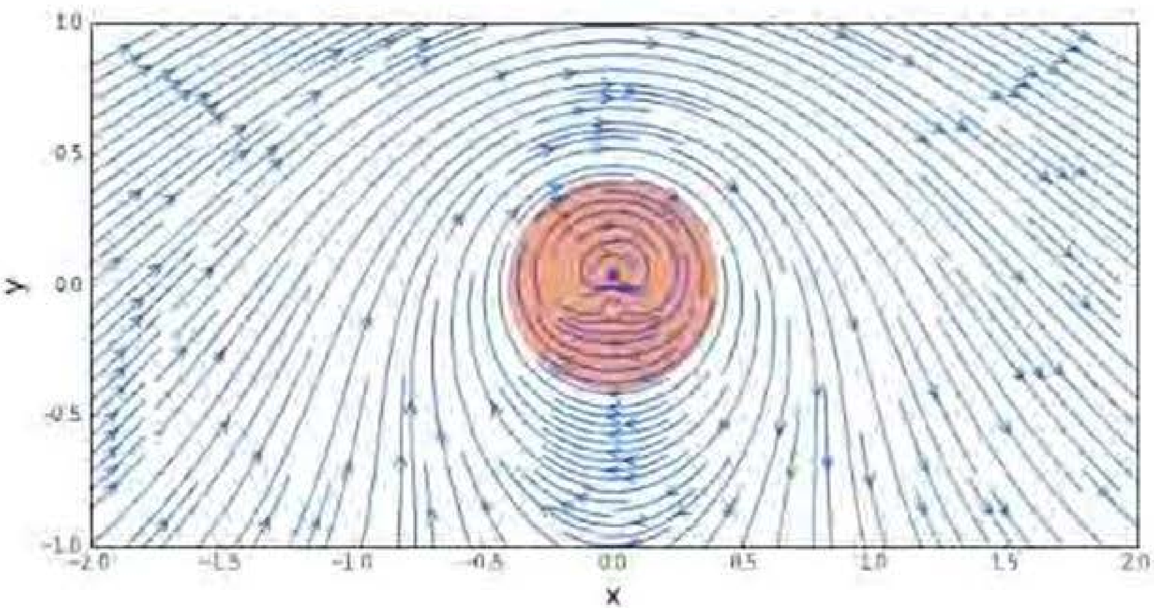
Remember when we computed uniform flow past a doublet? The stream-line pattern produced flow around a cylinder. When studying the pressure coefficient, we realized that the drag on the cylinder was exactly zero, leading to the *D'Alembert paradox*.

What about lift? Is it possible for a perfectly circular cylinder to experience lift? What if the cylinder is rotating? Have you heard about the Magnus effect?

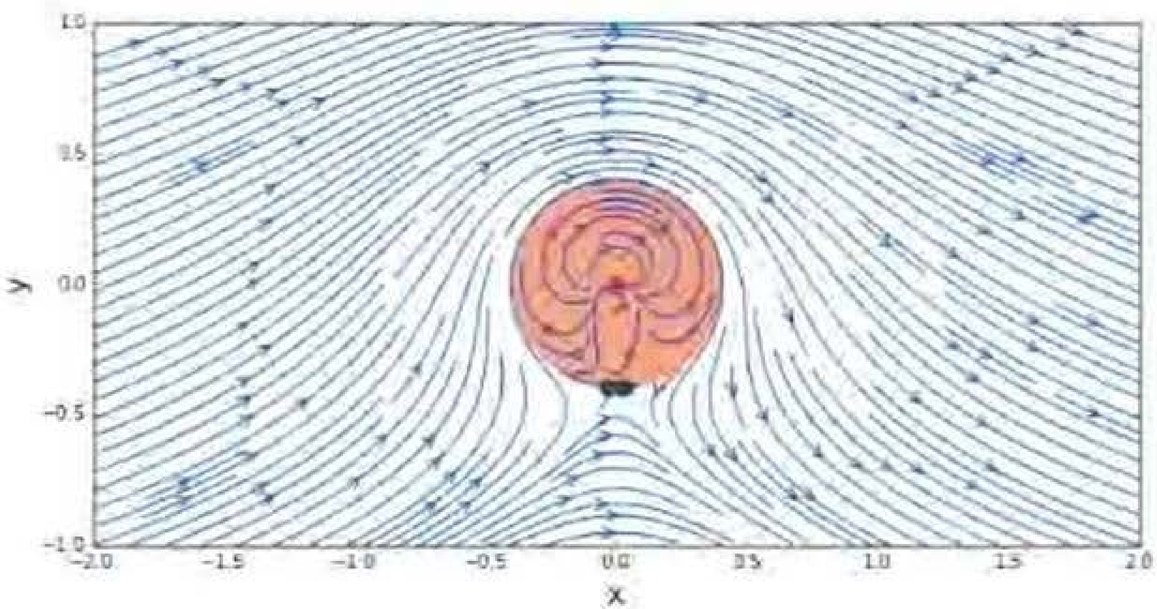
You might be surprised to learn that all we need to do is add a vortex in the center of the cylinder. Let's see how that looks.

First, we recall the equations for the flow of a doublet. In Cartesian coordinates, a doublet located at the origin has a stream function and velocity components given by

$$\psi(x,y) = -\frac{\kappa}{2\pi}\frac{y}{x^2+y^2}$$



$\gamma = 10$



$\gamma = 5$

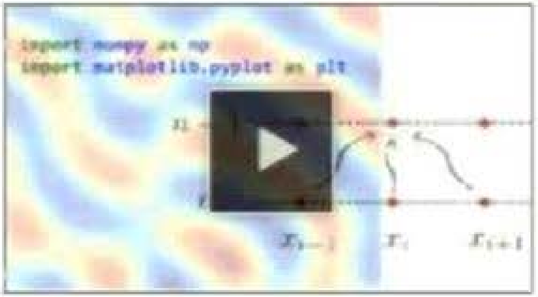
GW Online FIND COURSES

LorenaBarba

Practical Numerical Methods with Python *Engineering*

YOU ARE REGISTERED FOR THIS COURSE

VIEW COURSEWARE



VIEW ABOUT PAGE IN STUDIO

overview

Course Number	MAE6286
Classes Start	Aug 18, 2014

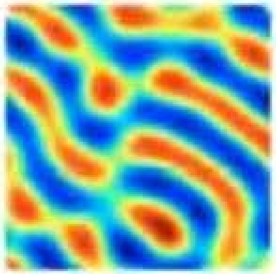
WHY TAKE THIS COURSE?

Even if this is the only numerical methods course you ever take, dedicating yourself to mastering all modules will give you a foundation from which you can build a career in scientific computing.

ABOUT THIS COURSE

This course is a collaboration between faculty at three institutions across the world: the George Washington University (Washington, DC, USA); University of Southampton (UK) and Pontifical Catholic University of Chile (Santiago, Chile).* A credit-bearing course will run at each institution at the same time as this MOOC, and students at all locations will participate in the same learning community with MOOC participants.

Search GitHub | Explore | Gist | Blog | Help



numerical-mooc

http://openedx.seas.gwu.edu/courses/GW/MAE6286/2014_fall/about

Filters ▾ | Find a repository... | + New repository

numerical-mooc

Python ★ 97 ⑂ 552

A collaborative educational initiative in computational science and engineering.

Updated a day ago

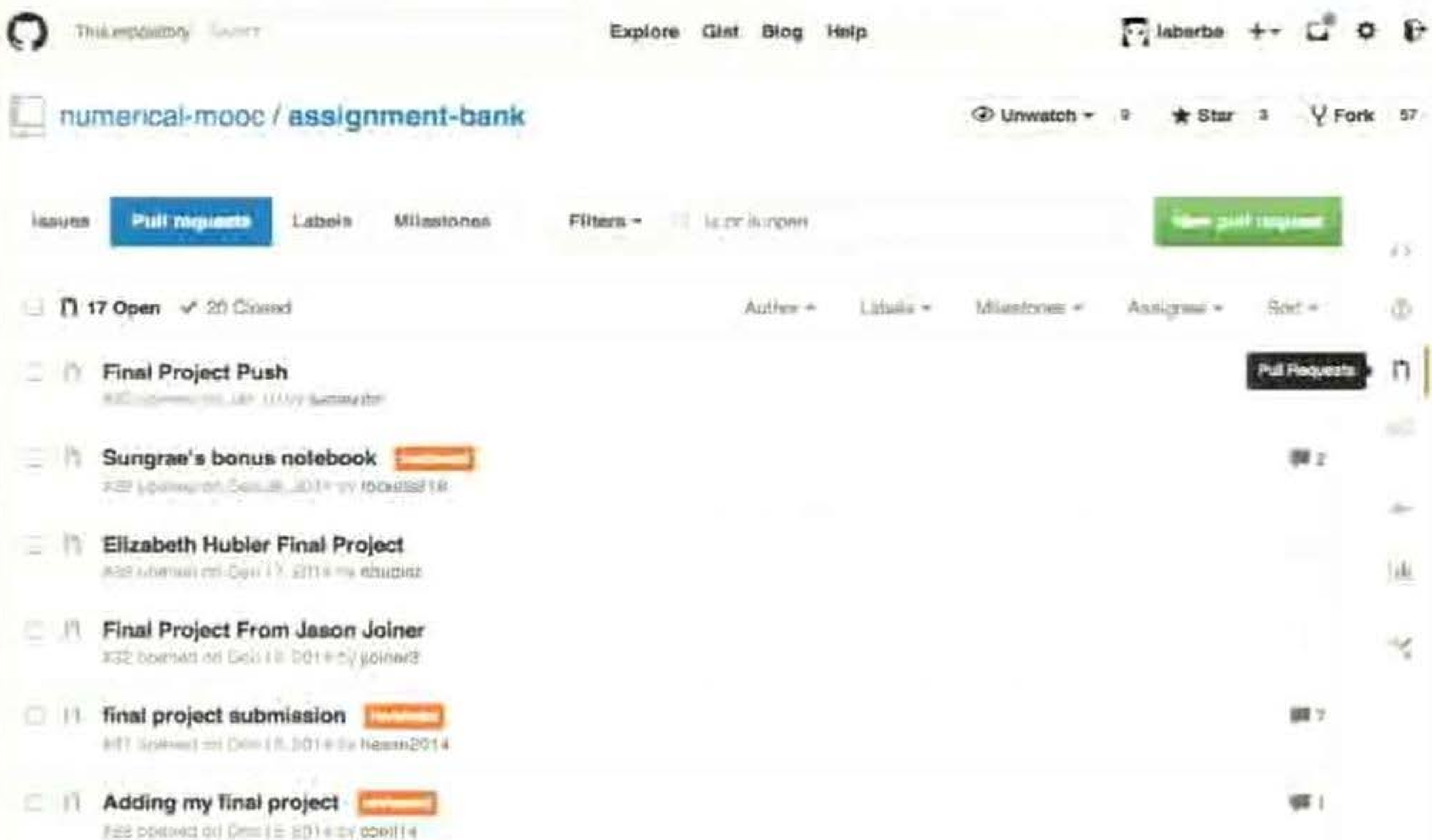
assignment-bank

Python ★ 3 ⑂ 57

Contribute alternative assignments for Numerical Methods with Python

Updated on Jan 20

Assignments by pull-request on GitHub



Module 1: The phugoid model.

1. Phugoid motion
2. Phugoid oscillation
3. Full phugoid model
4. Bonus! Second-order and multi-step methods

Module 2: Space and Time

Introduction to finite-difference solutions of PDEs

1. 1D linear and nonlinear convection
2. CFL condition
3. Diffusion equation in 1D
4. Burgers' equation

Module 3: Riding the wave

Convection problems

1. Conservation laws and the traffic-flow problem
2. Numerical schemes for hyperbolic PDEs
3. A better flux model
4. Finite volume and MUSCL methods.
5. Assignment: Sod's test problems

Module 4: Spreading out

Diffusion problems

1. Diffusion equation in 1D and boundary conditions
2. Implicit schemes in 1D and boundary conditions
3. 2D heat (diffusion) equation with explicit scheme
4. 2D heat equation with implicit scheme, and applying boundary conditions
5. Crank-Nicolson scheme and spatial & time convergence study
6. Assignment: Reaction-diffusion with the Gray-Scott model in 2D

Phugoid Oscillation

Welcome back! This is the second IPython Notebook of the series *"The phugoid model of glider flight"*, the first learning module of the course **"Practical Numerical Methods with Python."**

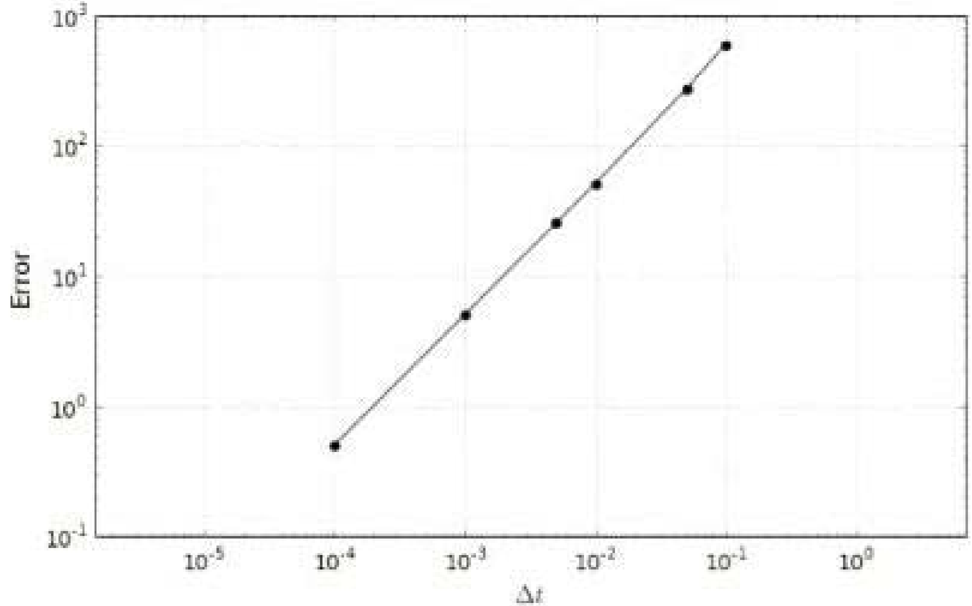
[...]

Convergence

To compare the two solutions, we need to use a **norm** of the difference, like the L_1 norm, for example.

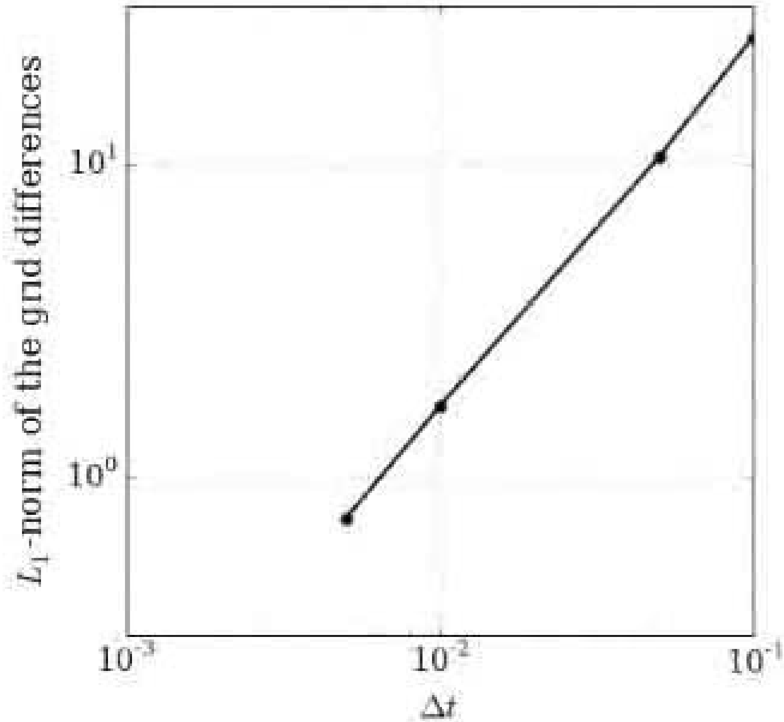
$$E = \Delta t \sum_{n=0}^{N} |z(t_n) - z_n|$$

In [12]:
```
pyplot.figure(figsize=(10, 6))
pyplot.tick_params(axis='both', labelsize=14) #increase tick font size
pyplot.grid(True)                              #turn on grid lines
pyplot.xlabel('$\Delta t$', fontsize=16)  #x label
pyplot.ylabel('Error', fontsize=16)        #y label
pyplot.loglog(dt_values, error_values, 'ko-')  #log-log plot
pyplot.axis('equal')                           #make axes scale equally;
```



This is the kind of result we like to see! As Δt shrinks (towards the left), the error gets smaller and smaller, like it should.

http://nbviewer.ipython.org/github/numerical-mooc/numerical-mooc/blob/master/lessons/01_phugoid/01_02_Phugoid_Oscillation.ipynb



Order of convergence

The order of convergence is the rate at which the numerical solution approaches the exact one as the mesh is refined. Considering that we're not comparing with an exact solution, we use 3 grid resolutions that are refined at a constant ratio r to find the *observed order of convergence* (p), which is given by:

$$p = \frac{\log\left(\frac{f_3 - f_2}{f_2 - f_1}\right)}{\log(r)} \tag{16}$$

where f_1 is the finest mesh solution, and f_3 the coarsest.