Python for HPC - Day 2

April 13, 2022

Presented by:
Nicholas A. Danes, PhD
Computational Scientist
Cyber Infrastructure & Advanced Research Computing (ITS)

Recap of Day 1

- Overview of what is HPC and HPC @ Mines
- Introduce researchers to Python, an interpreted programming language
- Overview basics of the Jupyter notebooks and the Python programming language

Today's Goals

- Introduce researchers to Python, an interpreted programming language, as an option for scientific computing
- Show overview on how to setup your own Python environment on Mines' HPC systems
- Highlight how Python can be used as an alternative (!= replacement) to MATLAB

Notable Scientific Python Codes & Libraries

- **NumPy – Scientific Computing Library**
  - **Written in C under the hood**
- **SciPy**
  - **Linear algebra interface written in Python**
  - **Provides sparse linear algebra libraries**
  - **Uses NumPy as a dependency**
- **Matplotlib**
  - **Object-oriented plotting interface**
- Pandas
  - Data science
- **Jupyter Notebooks**
  - **Web-based notebook-style interface for Python computing**
- FEniCS
  - C++/Python software suite for solving Differential Equations using the Finite Element Method
- Paraview
  - VTK-based parallel 2D/3D data analysis and visualization (supports Python scripting!)

Why consider Python over MATLAB?

1. It’s free and open source!

2. Objected-oriented first programming language

3. Easy access to non-scientific Python libraries

4. A helpful, online community!

Why to not consider Python over MATLAB?

1. Established code/workflow already written for MATLAB

2. Performance constraints for a problem

3. Availability of a solving method or library

4. License availability is not an issue

Comparing to MATLAB to Python

Command Description	MATLAB Command	Python Command

If statements	`if statement1_true`
	`  commands`
	`elseif statement2_true`
	`  commands`
	`else`
	`  commands`
	`end`
		`If statement1_true:`
		`  commands`
		`elif statement2_true:`
		`  commands`
		`else:`
		`  commands`

For loop	`for i=1:10`
	`  commands`
	`end`
		`for i in range(1,11):`
		`  commands`

Functions	`function [x,y,z] =`
	`myfunc (a,b,c)`
	`  commands`
	`end`
		`def myfunc (a,b,c):`
		`  commands`
		`  return x,y,z`

Comparing to MATLAB to NumPy/SciPy

Command Description	MATLAB Command	NumPy/SciPy Command
Create row vector	`u = [1,2,3]` `u = 1:2:10`	`u = numpy.array([1,2,3])` `u = numpy.arange(1,10,2)`
Create column vector	`v = [1;2;3]`	`v = numpy.array([[1, 2, 3]]).T`
Create a NxM matrix	`A = zeros(n,m)`	`A = numpy.zeros((n,m))`
Check matrix dimensions	`size(A)`	`A.shape`
Array slicing	`u(1:3)` `u(end)`	`u[0:3]` `u[-1]`
Solve linear system (Ax = b)	`A \ b`	`numpy.linalg.solve(A,b)`
Solve ODE IVP (non-stiff)	`[t,y] =` `ode45(fun,tspan,y0)`	`sol = scipy.integrate.solve_ivp(fun,` `t_span, y0, method='RK45')`
Solve ODE (IVP stiff)	`[t,y] =` `ode15s(fun,tspan,y0)`	`sol = scipy.integrate.solve_ivp(fun,` `t_span, y0, method='BDF')`

Further reference: https://numpy.org/devdocs/user/numpy-for-matlab-users.html

Using Python with a GUI/IDE

If you are a MATLAB user who is more comfortable with a GUI, there are several options for Python:

- Spyder
- Atom (GitHub)
- Sublime Text 3
- Jupyter Notebooks

And many more!

Getting started with Python on HPC I

- Step 0: Obtain an account on our HPC systems
  - Proposal Request by PI/Advisor to Mines help desk:
    - Request Need for HPC and how resources will be used
    - New accounts on “Wendian” available!
    - Node owners on “Mio” can add new users
- Step 1: Log into HPC system

```
$ ssh username@wendian.mines.edu
```

- Step 2: Load a python module (multiple options)

```
$ module avail python
apps/python2/2.7-anaconda-2018.12        apps/python3/3.7/anaconda-2018.12
apps/python2/2.7-intel-2018.3            apps/python3/2020.02
apps/python3/3.6-intel-2018.3
$ module load apps/python3/2020.02
$ which python
/sw/apps/python3/anaconda-2020.02/bin/python
```

Getting started with Python on HPC II

Setup your own anaconda environment for Python:

```
$ conda create -y --name my_env python=3.8 numpy scipy matplotlib
```

Activate using your environment by sourcing it:

```
$ source activate my_env
(my_env) $
```

Test Problem #1: 1D Poisson's Equation

Consider

$$-u''(x) = f(x)$$

on the domain (a, b) subject to the boundary conditions:

$$u(a) = \alpha,$$
$$u(b) = \beta$$

Discretization using Finite Difference Method

Let X define the set of grid points which discretizes the domain (a, b):

$$X_i = a + i\frac{b-a}{N}, \text{ where } i = 0,1, \dots, N$$

And N is the number of interior grid points ("cells"). We will also define the grid spacing h as:

$$h = \frac{b-a}{N}$$

Discretization using Finite Difference Method

Let U be the vector of unknowns representing the discretized solution $u(x)$ on the grid X:

$$\mathrm{U} = [u_1, u_2, \dots, u_{N-1}]$$
$$u_0 = \alpha$$
$$u_N = \beta$$

Discretizing the second derivative $u''(x)$ using a centered finite difference on the interior grid points of X:

$$u''(x_j) \approx \frac{u_{j-1} - 2u_j + u_{j+1}}{h^2}, \quad j = 1,2, \dots, N-1$$

Discretized 1D Poisson Equation

For a given index i, the discretized 1D Poisson equation can be written as:

$$-\frac{u_{j-1}-2u_j+u_{j+1}}{h^2} = f(X_j), \quad j = 1,2,\ldots,N-1$$

The boundary conditions can be accounted for by moving them to right hand side of the equation corrwhen $j = 0$ and $j = N$ respectively:

$$\frac{2u_1-u_2}{h^2} = f(X_1) + \frac{\alpha}{h^2}, \quad j = 1$$

$$\frac{u_{N-2}+2u_{N-1}}{h^2} = f(X_{N-1}) + \frac{\beta}{h^2}, \quad j = N-1$$

Discretized 1D Poisson Equation

In summary, we can write this discretized 1D Poisson equation as:

$$AU = F$$

where

$$\mathrm{A} = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & \cdots & \cdots & 0 \\ -1 & 2 & -1 & \cdots & \ddots \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & \ddots & 2 & -1 \end{pmatrix} \quad \mathrm{U} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix} \quad \mathrm{F} = \begin{pmatrix} f(X_1) + \alpha/h^2 \\ f(X_2) \\ \vdots \\ f(X_{N-2}) \\ f(X_{N-1}) + \beta/h^2 \end{pmatrix}$$

Live Demo

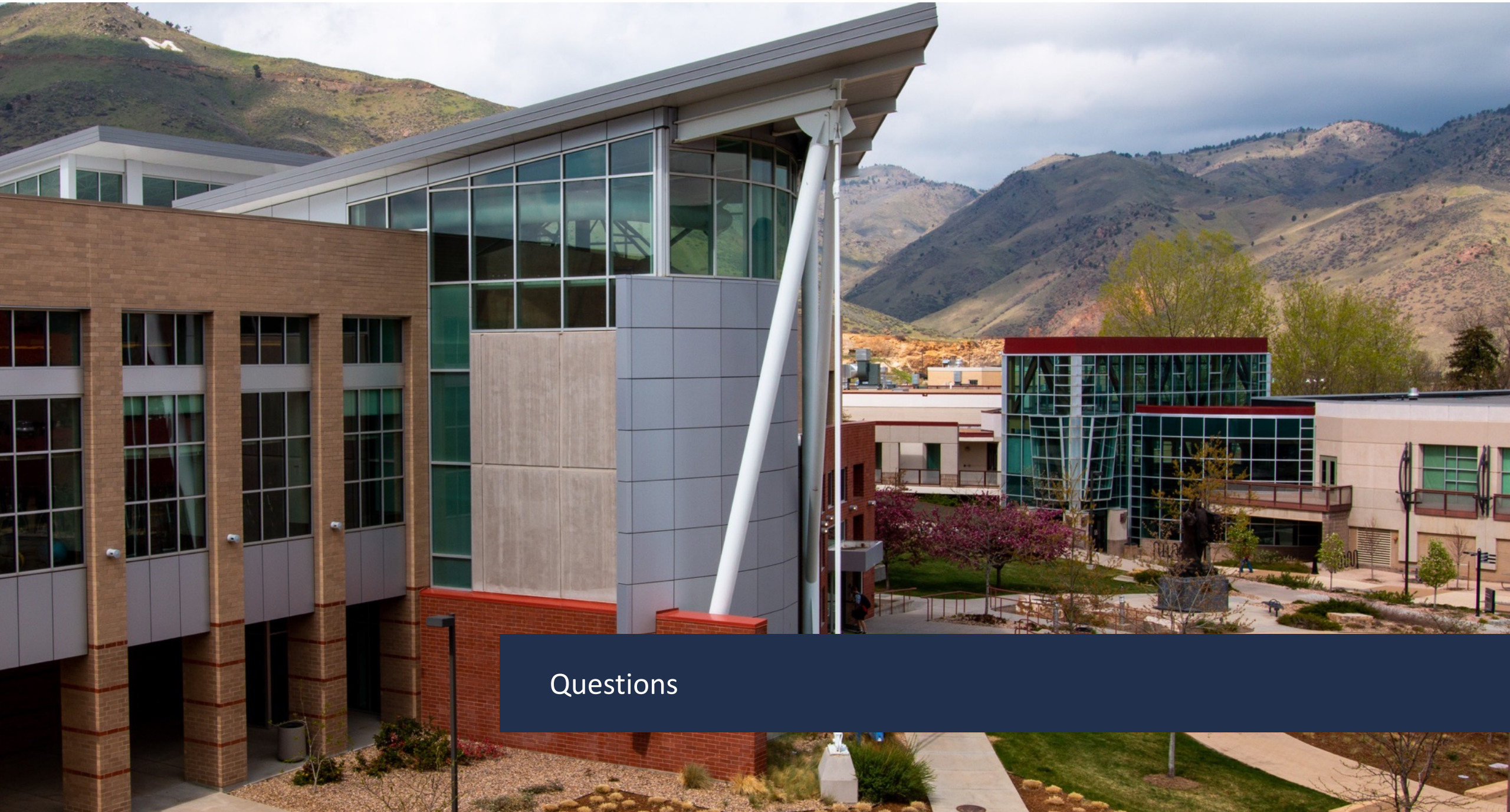
- Let’s solve this equation only using NumPy!
- Plot results using matplotlib
- Submit a job to SLURM (HPC Job Scheduler)
- Open and run script using Jupyter Notebooks on HPC!

Further Resources

- Mines CIARC HPC Website:
  - https://ciarc.mines.edu/hpc
  - Pages are under construction!
- For HPC-related questions:
  - Submit a ticket to the help desk!
  - https://helpcenter.mines.edu/TDClient/1946/Portal/Requests/ServiceCatalog?CategoryID=11036
- More References:
  - https://realpython.com/matlab-vs-python/
  - https://matplotlib.org/3.1.1/tutorials/index.html

Day 3 Plans

- Optimizing Python Performance
  - Scipy
  - Numba
  - Cython
- Intro to Parallel Programming using MPI
  - Mpi4py
  - Petsc4py



Questions