Tutorial 01

MECHTRON 3X03
(Scientific Computation)

Scientific Computation

- We will learn techniques for solving problems from the physical sciences with computers (scientific computation).
- We will cover examples of such techniques during tutorials.
- Why is this important? Some mathematical computations cannot be cleanly translated into computer software because of:
  - resource and hardware constraints (e.g., limited memory and computation time);
  - complexity of algorithms that carry out advanced mathematical operations; and
  - challenges arising from discrete (i.e., digital) representations of continuous (i.e., real) numbers.
- In the next slide we can see an example of this limitation.

Example: Euclidean norm of a vector

- You can simply find the Euclidean norm (or length/magnitude) of a vector using the following formula: $\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$
- However, this formula is not well-defined for finite memory computations.
  - The IEEE 754 (a binary floating-point number system) maximum exponent is 127.
  - If a floating-point vector contains a number with exponent value of 64 (for example), then it can overflow!
    - fl((2^64)^2) = fl(2^128) = +Infinity. Same with fl(-2^127) = -Infinity.
- To fix this, we can set $M = \max(|x_1|, |x_2|, \ldots, |x_n|)$
- Then, we compute as follows: $M \cdot \sqrt{\left(\frac{x_1}{M}\right)^2 + \left(\frac{x_2}{M}\right)^2 + \ldots + \left(\frac{x_n}{M}\right)^2}$

This scales all floating-point numbers down to the interval [-1,1] which prevents an overflow when squared, then rescales by M after the square root.

Learning Objectives (taken from syllabus)

- summarize the advantages and disadvantages of floating-point representations of real numbers;
- detect and fix issues in floating-point computations (e.g., overflow, cancellations, and roundoff errors);
- derive error bounds on computational results;
- select and implement an appropriate solution method for a variety of linear systems, interpolation
- problems, unconstrained continuous optimization problems, and ordinary differential equations;
- describe and implement various numerical integration schemes;
- apply linear algebraic concepts to solving and characterizing linear least squares problems;
- perform simple complexity analysis on numerical algorithms;
- explain the importance of eigenvalues and eigenvectors in numerical methods;
- analyze the convergence and stability of numerical routines; and
- **write Julia programs implementing numerical methods**

Tutorial format

- Tutorials start with a code demonstration, with the exception of this tutorial. We will use Jupyter Notebooks with a Julia kernel to demonstrate certain computations.
- Students can feel free to ask the TAs relevant questions during the tutorial. Any remaining questions can be answered through email (read syllabus)
- Pass-fail tutorial quizzes will be conducted in-class weekly for **5% of final grade**. Collaboration in groups of up to three is permitted.
  - Tutorial quizzes should take 15-30 minutes to complete, and they are due near the end of the tutorial.
  - If you miss a tutorial, the quiz grade weight will be added into the final exam without MSAF.
  - Assignments, midterm, and final exam will remain **individual assessments**.

Technical Requirements

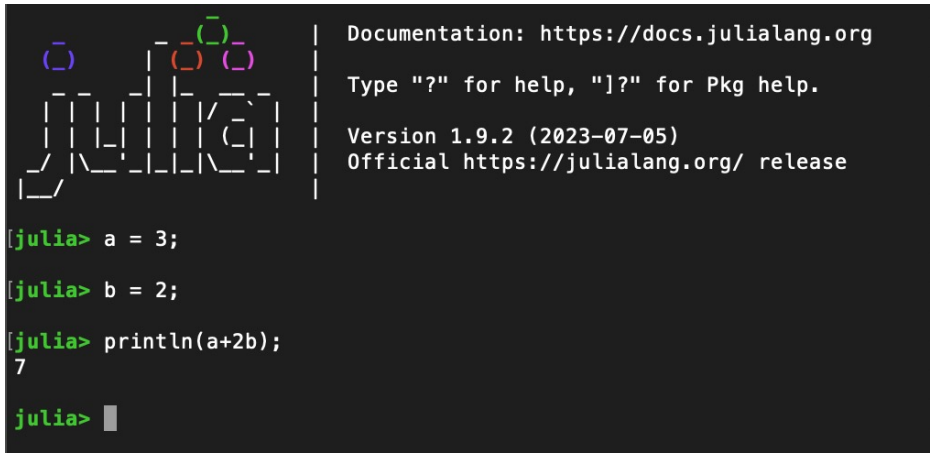
- Laptop
- Text Editor (preferably Visual Studio Code)
- JupyterHub and/or Jupyter Notebook
- Julia language support

Requirement	Windows	Apple
Operating system	Windows 10	Big Sur 11.0 (or latest)
Web browser	Firefox or Google Chrome	Firefox or Google Chrome
Screen resolution	1024 x 768	1024 x 768
Webcam	640X480	640X480
Processor	Intel Core i5 7th or 8th generation minimum	Intel Core i5 7th or 8th generation minimum or Apple M1 processor
RAM/Memory	16 GB minimum	16 GB minimum
Hard drive (Storage)	256+ GB minimum, 500 GB recommended (SSD-Solid State Disk recommended)	256+ GB minimum, 500 GB recommended (SSD-Solid State Disk recommended)
Wireless	Wi-Fi 5 (802.11ac) minimum Wi-Fi 6 (802.11ax) recommended	Wi-Fi 5 (802.11ac) minimum Wi-Fi 6 (802.11ax) recommended
Headset	USB or Bluetooth with built-in mic	USB or Bluetooth with built-in mic

https://uts.mcmaster.ca/technology-resources-for-mcmaster-students/#tab-content-device-recommendations

Julia installation

- Go on the official Julia website and download the latest release (v 1.9.3): https://julialang.org/downloads/
- Choose your Operating system and download the installer.
- Run the installer.
- If prompted, add Julia to the Path.
- Open the Julia app from the Start Bar/Launchpad or by writing in the Command Prompt/Terminal “julia”.
- To exit Julia, use the command “exit()”.



VSCode Installation

- Feel free to use any Text Editor you want (Sublime Text, Atom, Vim, Notepad++, etc.)
- Download the Visual Studio Code installer by going to https://code.visualstudio.com
- Run installer with admin privileges.
- Agree with license agreement.
- Install with default settings.

Julia Extension Installation

- Click on the Extensions tab to see available extensions.
- Search for extensions containing "julia"
- Install the Julia extension. A restart may be required
- Create a workspace to start off.
- Now create a file and name it HelloWorld.jl (jl is the file extension for Julia)
- Now you can either print "Hello World" by writing print("Hello World") or println("Hello World")

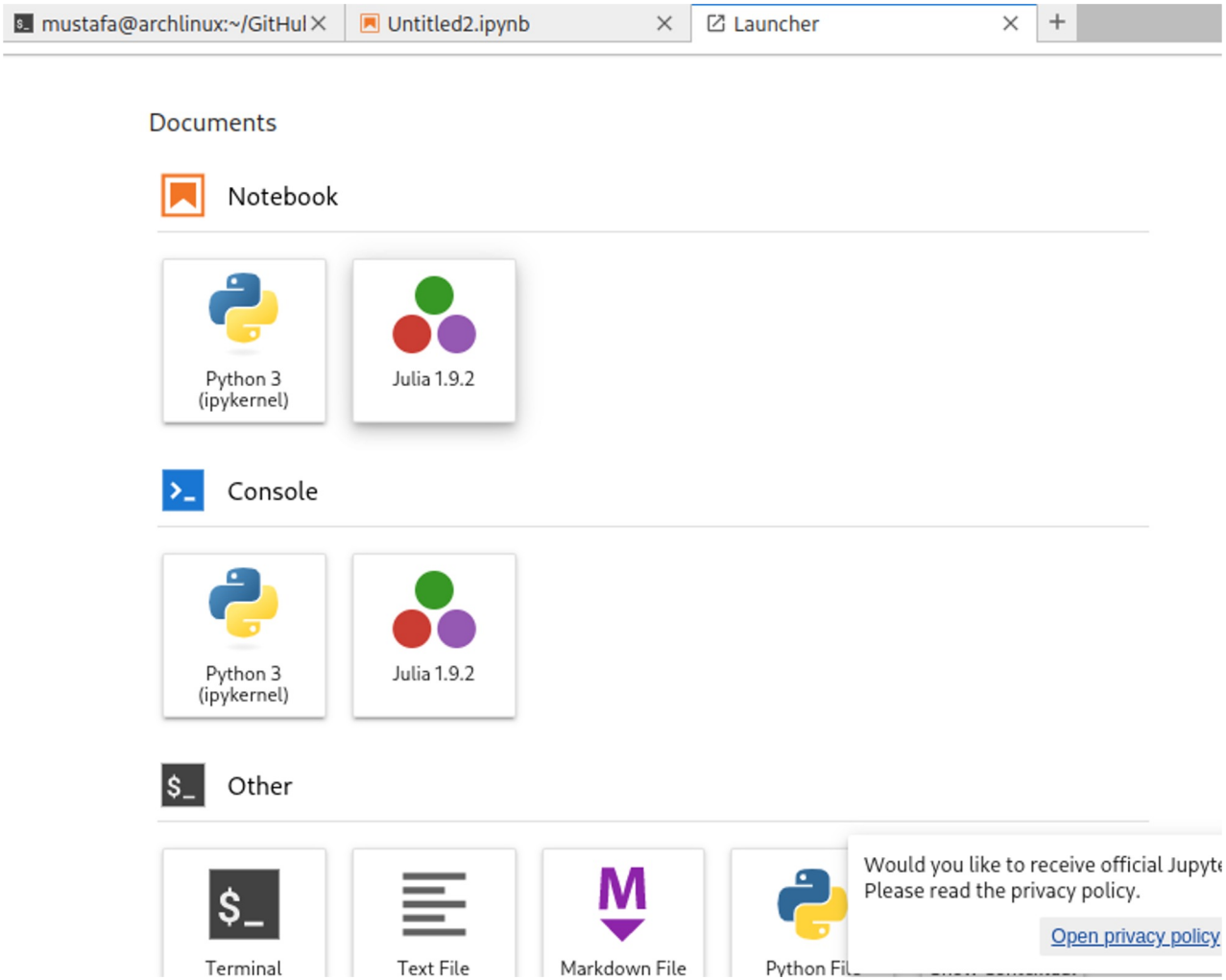
Jupyter Installation

- Anaconda is a scientific computing distribution for Python and R. Installing Anaconda should also install Jupyter.
- Install Anaconda here: https://www.anaconda.com/download
- If you want to avoid installing Anaconda, it's possible to install just Jupyter using the following link: https://jupyter.org/install

IJulia Installation

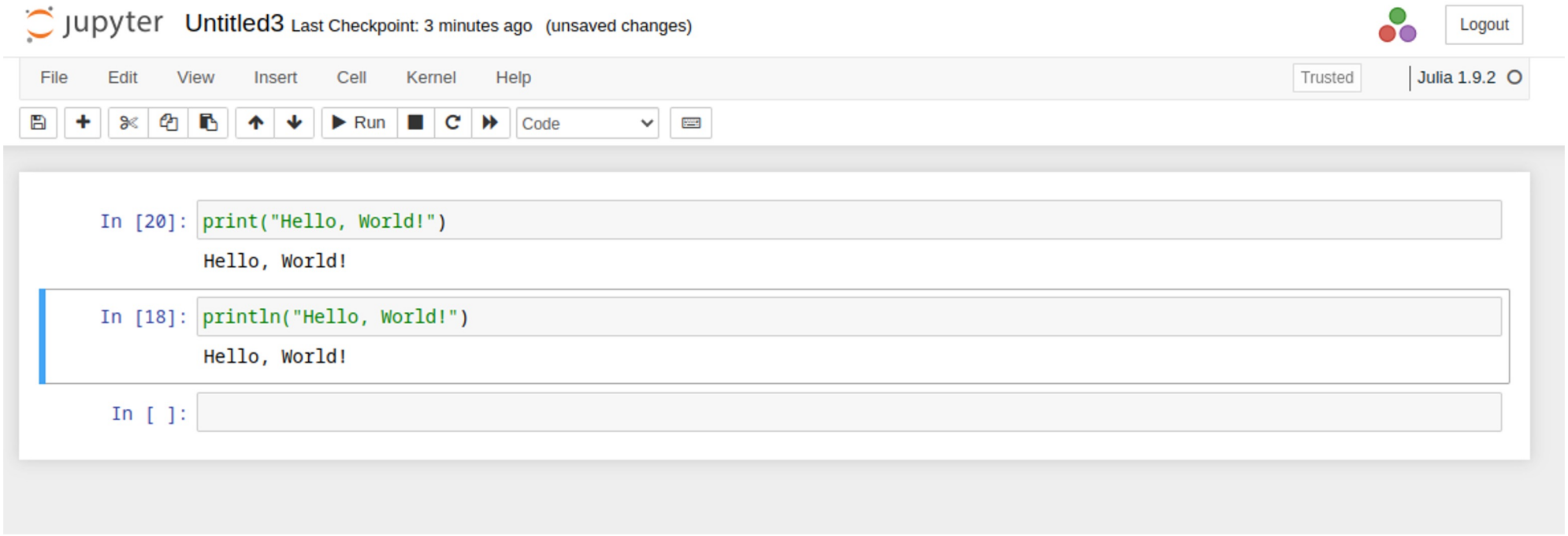
- IJulia is the Julia kernel for Jupyter.
- After installing Julia, run the following commands in your terminal:
  - julia — (opens Julia)
  - using Pkg — (loads Pkg)
  - Pkg.add("IJulia")
- An alternative using VSCode is the following:
  - Write "]" to enter package mode.
  - Write "add IJulia"
  - Backspace to exit package mode.
- After installing IJulia you should be able to see Julia as an available kernel in Jupyter Notebook.

JupyterHub


mustafa@archlinux:~/GitHul
Untitled2.ipynb
Launcher
Documents
Notebook
Python 3 (ipykernel)
Julia 1.9.2
Console
Python 3 (ipykernel)
Julia 1.9.2
Other
Terminal
Text File
Markdown File
Would you like to receive official Jupyte
Please read the privacy policy.
Open privacy policy

Jupyter Notebook

Jupyter Notebook Example



Contact Information

- Prof. Matthew Giamou: giamoum@mcmaster.ca
- Teaching Assistants
  - Federico Formica (grad TA): formicaf@mcmaster.ca
  - Mustafa Abdulameer (undergrad TA): abdulm55@mcmaster.ca

Questions?