CS207: Systems Development for Computational Science

https://harvard-iacs.github.io/2019-CS207/

Instructor: David Sondak
TFs: Lindsey Brown, Feiyu Chen, Aditya Karan, Bhaven Patel

Harvard University
Institute for Applied Computational Science

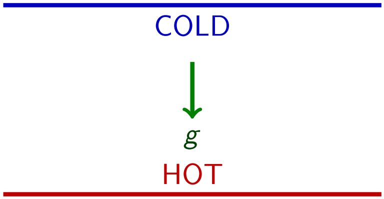
9/3/2019

Motivation: Thermal Convection and the Geodynamo

Thermal convection drives most fluid flows in the universe

Motivation: Thermal Convection and the Geodynamo

Thermal convection drives most fluid flows in the universe

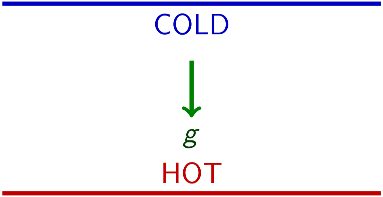


Cold fluid falls, hot fluid rises

▸ Plate Tectonics Video

Motivation: Thermal Convection and the Geodynamo

Thermal convection drives most fluid flows in the universe



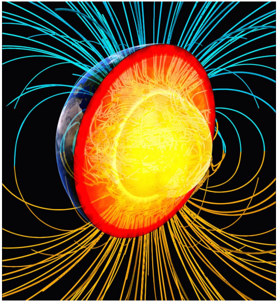
COLD

g

HOT

Cold fluid falls, hot fluid rises

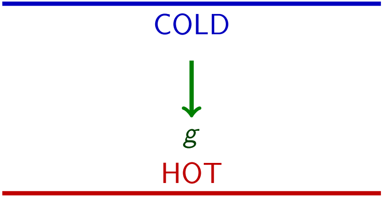
▸ Plate Tectonics Video



DESY

Motivation: Thermal Convection and the Geodynamo

Thermal convection drives most fluid flows in the universe



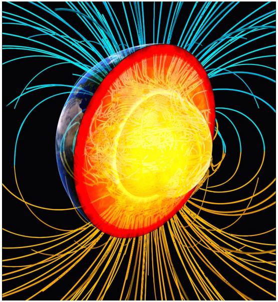
COLD

g

HOT

Cold fluid falls, hot fluid rises

▸ Plate Tectonics Video

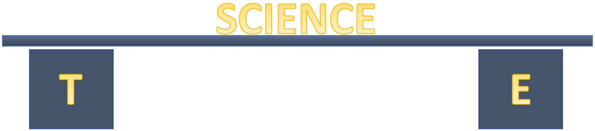


DESY

$$\boxed{\frac{\partial T}{\partial \mathrm{t}} + \nabla \cdot (\mathbf{u}\,T) = k\nabla^2 T}$$

- Ignoring $\nabla \cdot (\mathbf{u}\,T)$ gives the usual heat conduction equation!

Motivation: The Pillars of Science



Motivation: The Pillars of Science

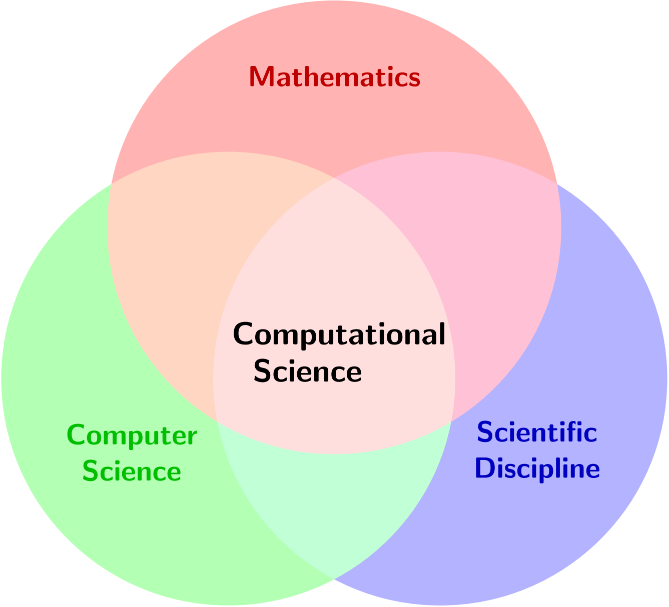
SCIENCE

T

C

E

Computational Science



Why take this class?

- Scientific software is complex
- Your code needs to be:
  - Reuseable
  - Portable
  - Robust
- Must go beyond “scripting”

Why take this class?

- Scientific software is complex
- Your code needs to be:
  - Reuseable
  - Portable
  - Robust
- Must go beyond “scripting”

Why take this class?

- Scientific software is complex
- Your code needs to be:
  - Reuseable
  - Portable
  - Robust
- Must go beyond “scripting”

CS207 Objectives

To give students who may not have a traditional computer science background the knowledge and tools to develop and maintain effective software for computational science applications.

Why take this class?

- Scientific software is complex
- Your code needs to be:
  - Reuseable
  - Portable
  - Robust
- Must go beyond “scripting”

CS207 Objectives

To give students who may not have a traditional computer science background the knowledge and tools to develop and maintain effective software for computational science applications.

Who should take this class?

- Any kind of scientist is welcome to take this class!
- This course is computer science for people who aren't computer scientists:
  - Data scientists
  - Biologists
  - Chemists
  - Engineers
  - Physicists
  - Mathematicians
  - Economists
  - ⋮
- It is also for computer scientists who want to develop scientific software
- CS207 is for students who need to know effective and modern software practices for their career

Sample Topics

A few selected topics to be covered:

- Unix and Linux
- Version control
- Python
- Software documentation
- Software testing
- Object-oriented programming
- Data structures
- Databases

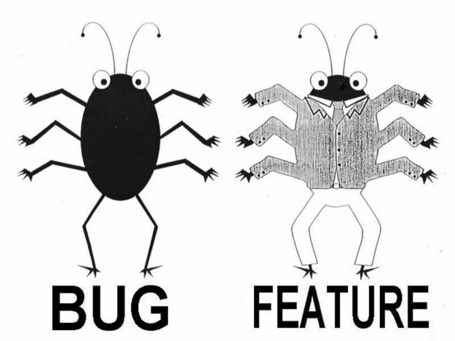
Sample Topics

A few selected topics to be covered:

- Unix and Linux
- Version control
- Python
- Software documentation
- Software testing
- Object-oriented programming
- Data structures
- Databases

Other potential topics
(not guaranteed):

- Debuggers and debugging
- Build systems (Makefiles, autotools, ...)
- Compiled languages



Course Structure

- CS207 is an application-driven course
- Two, 75 minute lectures per week
- Lectures centered around group programming exercises
- Programming assignments for homework
- Primary deliverable is a software development project
- All course content hosted on GitHub

Course Website:

https://harvard-iacs.github.io/2019-CS207/

Course Project: Overview

- You will work in groups of 3 to 4 people (assigned by teaching staff)
- You will add to your library throughout the semester
- The project consists of two milestones
- For the final project, you will add a non-trivial feature to your library
- A portion of your grade will come from peer-assessment
- Exact details on website

Course Project: The Topic

Automatic differentiation

Course Project: The Topic

Automatic differentiation

What is Automatic Differentiation?

Course Project: The Topic

Automatic differentiation

What is Automatic Differentiation?

- A way to evaluate derivatives of functions and computer programs

Course Project: The Topic

Automatic differentiation

What is Automatic Differentiation?

- A way to evaluate derivatives of functions and computer programs
- Computes derivatives to *machine precision*!

Course Project: The Topic

Automatic differentiation

What is Automatic Differentiation?

- A way to evaluate derivatives of functions and computer programs
- Computes derivatives to *machine precision*!
- Can be very efficient and accurate

Course Project: The Topic

Automatic differentiation

What is Automatic Differentiation?

- A way to evaluate derivatives of functions and computer programs
- Computes derivatives to *machine precision*!
- Can be very efficient and accurate
- Also known as “algorithmic differentiation”

Course Project: The Topic

Automatic differentiation

What is Automatic Differentiation?

- A way to evaluate derivatives of functions and computer programs
- Computes derivatives to *machine precision*!
- Can be very efficient and accurate
- Also known as “algorithmic differentiation”

We will have four lectures on automatic differentiation this semester to cover the main points.

Why Automatic Differentiation?

- Encapsulates many ideas in software design
  - Object-oriented programming
  - Operator overloading
  - Datastructures

Why Automatic Differentiation?

- Encapsulates many ideas in software design
    - Object-oriented programming
    - Operator overloading
    - Datastructures
- Pervasive throughout science and gaining steam
    - Neural networks and backpropagation
    - Hamiltonian Monte Carlo methods
    - Full Jacobian calculations
    - Jacobian-free calculations

AD Teaser

Suppose we have a function like

$$y = \exp\left(-\sqrt{x + \cos^2(x)}\right) \sin\left(x \ln\left(1 + x^2\right)\right).$$

AD Teaser

Suppose we have a function like

$$y = \exp\left(-\sqrt{x + \cos^2(x)}\right) \sin\left(x \ln\left(1 + x^2\right)\right).$$

The symbolic derivative is

$$\begin{aligned} y' = {} & \exp\left(-\sqrt{x + \cos^2(x)}\right) \cos\left(x \ln\left(1 + x^2\right)\right) \left(\frac{2x^2}{1 + x^2} + \ln\left(1 + x^2\right)\right) \\ & - \exp\left(-\sqrt{x + \cos^2(x)}\right) \frac{1 - 2\cos(x)\sin(x)}{2\sqrt{x + \cos^2(x)}} \sin\left(x \ln\left(1 + x^2\right)\right) \end{aligned}$$

AD Teaser

Suppose we have a function like

$$y = \exp\left(-\sqrt{x + \cos^2(x)}\right) \sin\left(x \ln\left(1 + x^2\right)\right).$$

The symbolic derivative is

$$\begin{aligned} y' = {} & \exp\left(-\sqrt{x + \cos^2(x)}\right) \cos\left(x \ln\left(1 + x^2\right)\right) \left(\frac{2x^2}{1 + x^2} + \ln\left(1 + x^2\right)\right) \\ & - \exp\left(-\sqrt{x + \cos^2(x)}\right) \frac{1 - 2\cos(x)\sin(x)}{2\sqrt{x + \cos^2(x)}} \sin\left(x \ln\left(1 + x^2\right)\right) \end{aligned}$$

And that's only the first derivative!

Demo

Next Steps

Go to

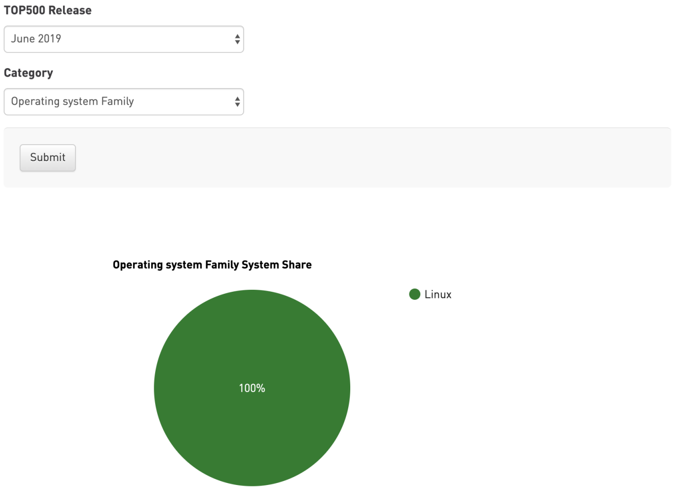
https://harvard-iacs.github.io/2019-CS207/lectures/lecture0/.

Unix and Linux

Portions of this lecture taken from the lecture notes of Dr. Chris Simmons.

Why Unix / Linux?

https://www.top500.org/lists/2019/06/



TOP500 Release: June 2019

Category: Operating system Family

https://www.top500.org/statistics/list/

What is Unix?

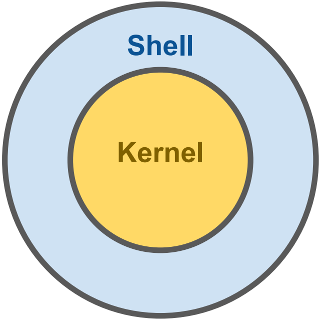
- Unix is a multi-user, preemtive, multitasking, operating system
- It provides several facilities:
  - Management of hardward resources
  - Directories and file systems
  - Loading, execution, and suspension of programs
- There are many versions of Unix:
  - Solaris
  - AIX
  - BSD
  - Linux (*not* unix, but pretty close)
  - ⋮

What is Linux?

- Linux is a clone of Unix
  - Written by Linus Torvalds
- First version dates to September 1991
- Linux has been further developed by people around the world
- Developed under the GNU General Public License
  - Source code for Linux is freely available

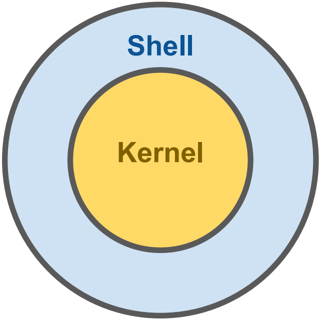
How Does Unix Work?

- Unix has a *kernel* and one or more *shells*
- The kernel is the core of the OS
- It receives tasks from the shell and executes them
- Users interact with the shell!



How Does Unix Work?

- Everything in Unix is a *process* or a *file*
- A process
  - Is an executing program (has a unique PID)
  - May be short or run indefinitely
- A file
  - Is a collection of data
  - Created by users
- The Unix kernel is reponsible for organizing processes and interacting with files



The Shell

- The Unix interface is called the shell
- The shell basically does four things repeatedly:
  - Display prompt
  - Read command
  - Process command
  - Execute command

How to Interact with Unix

- The user interacts with Unix via a shell
- Different kinds of shells
  - Graphical, e.g. X-Windows
  - Text-based (command-line), e.g. `bash` and `tcsh`
- To remotely access a shell session, use `ssh` (secure shell)

Some Common Unix Terminology

- Unix has the notion of *accounts*, which include:
  - a username/password
  - userid/groupid
  - home directory
  - a shell preference
- userids are called *UIDs*
- Unix has the notion of *groups*:
  - A Unix group can share files and active processes
  - Each account is assigned a primary group
  - The `groupid` corresponds to this primary group
- `groupids` are called *GIDs*

Unix Files and Directories

- A file is a basic unit of storage
- Every file must have a name
- Unix is case-sensitive
- A directory is a special kind of file
  - Directories hold information about other files
- We often think of a directory as a container that holds other files
  - e.g. folders for Mac or Windows users

Comments on the Unix Filesystem

- The filesystem is a hierarchical system of files and directories
- The top level in the heirarchy is called the `root`
- The full *pathname* of a file includes the filename and all directories up to the root
  - e.g. `/Users/dsondak/Teaching/Harvard/CS207/2019-CS207/`
- Absolute and relative pathnames:
  - Absolute pathnames start at the root
  - Relative pathnames are specified in relation to the current working directory
    - e.g. `Harvard/CS207/2019-CS207/`

Special Directory Names

- There is a special relative pathname for the current working directory
  - .
  - Just a *dot*
- There is a special relative pathname for the parent directory
  - ..
  - Pronounced *dot-dot*
- There is a special symbol for the home directory
  - ~
  - Just a tilde

These commands will become second nature to you.

Next Steps

Go to

https://harvard-iacs.github.io/2019-CS207/lectures/lecture0/.