Introduction to Computational Science and Basics of Computer Programming
(CDSC 601)

Addisu G. Semie, PhD
Asst. Prof.,Computational Data Science Program,
Addis Ababa University
Email: addisu.semie@aau.edu.et

Objectives

- Introduce the core meaning of computational science
- Introduce concepts of Operating Systems, Linux operating systems, Compilers and Editors
- Introduce Principles of Programming using FORTRAN programming language

The course grade will be calculated as follows:

Final exam: 50%;

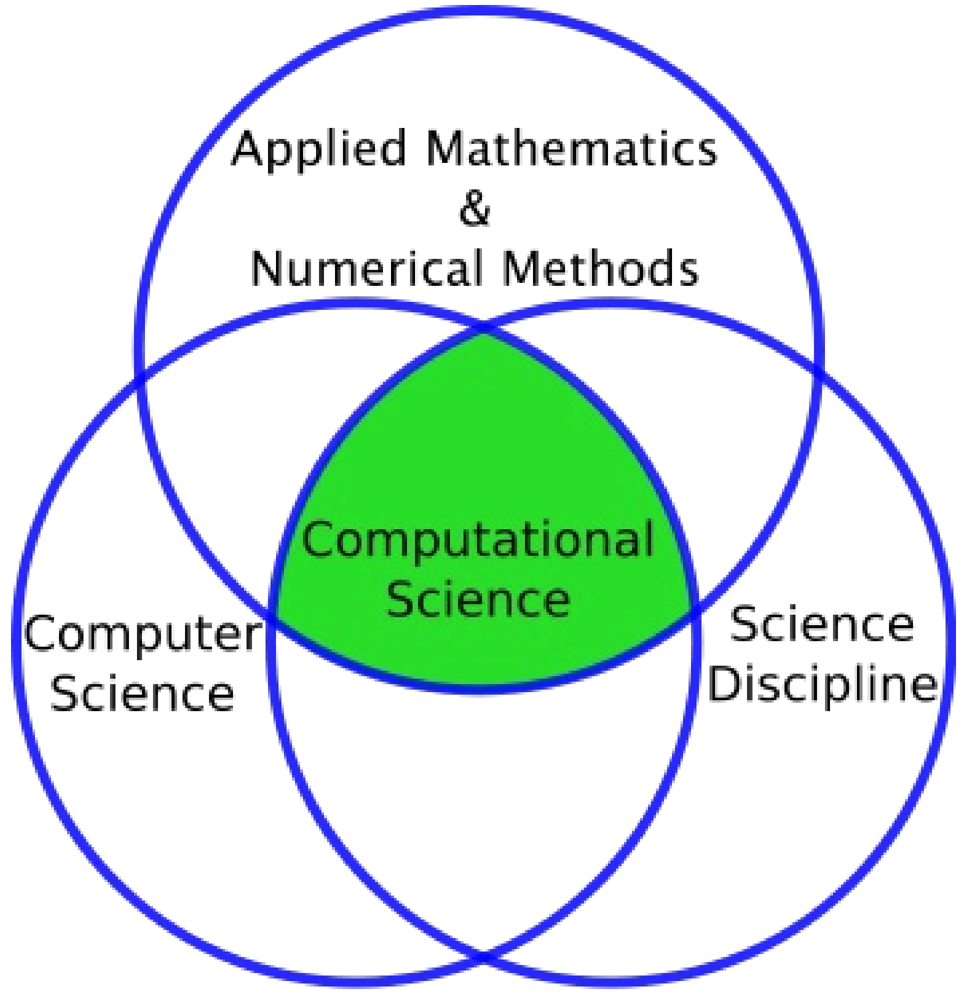
Test: 20%;

Assignments: 20%;

Class activity: 10%

What is computational Science?

- Computational science is the application of computational and numerical techniques to solve large and complex problems.
- Working in computational science requires a strong background in a variety of areas including computer programming, mathematics, and a background in one or more scientific fields.
- This Creates a new discipline that goes beyond the elements of any single area of study



Computational Science

- The complexity of the equations we use to describe nature sometimes make it impossible to do predictions just using pencil and paper calculations.

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) + \frac{\partial}{\partial z}(\rho w) = 0$$

- Computers allow us to make numerical approximations of these equations.

Computational Science

- Takes advantage of not only the improvements in computer hardware, but probably more importantly, the improvements in computer algorithms and mathematical techniques.
- Allows us to do things that were previously too difficult to do due to the complexity of the mathematics, the large number of calculations involved, or a combination of the two

Computational Science versus Computer Science.

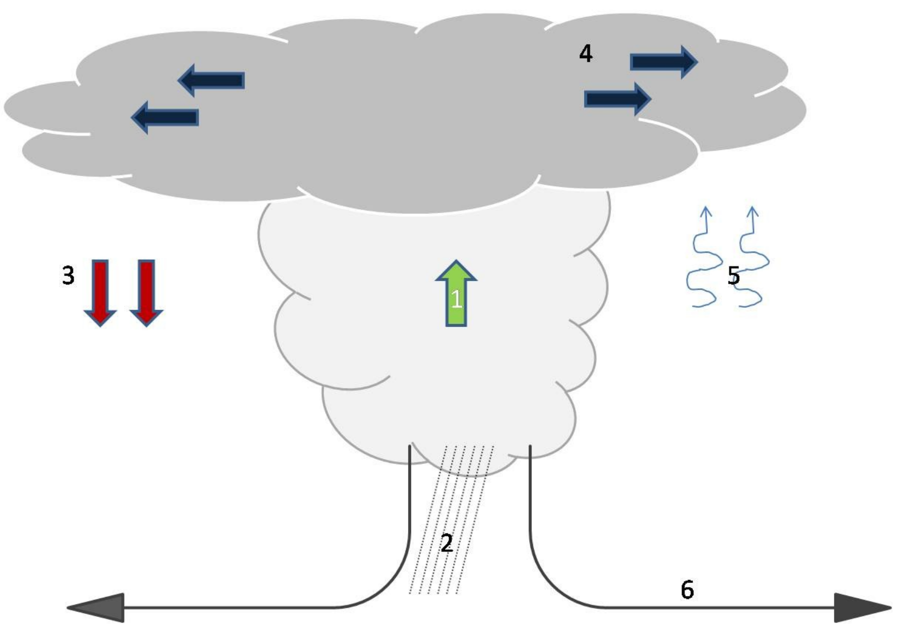
- Computational science primarily involves the use of computation to answer scientific questions.
- Computer science primarily involves the engineering of software and hardware systems and the theory of computation.

Why Computing for Scientists?

- Using computation to solve science problems is as important as
  - Using math to solve science problems
  - Using a microscope or a telescope to solve science problems

Why Computing for Scientists?

- Stimulates insight and understanding
- Sometimes leads to new theories, suggests new experiments
- Used to test new theories



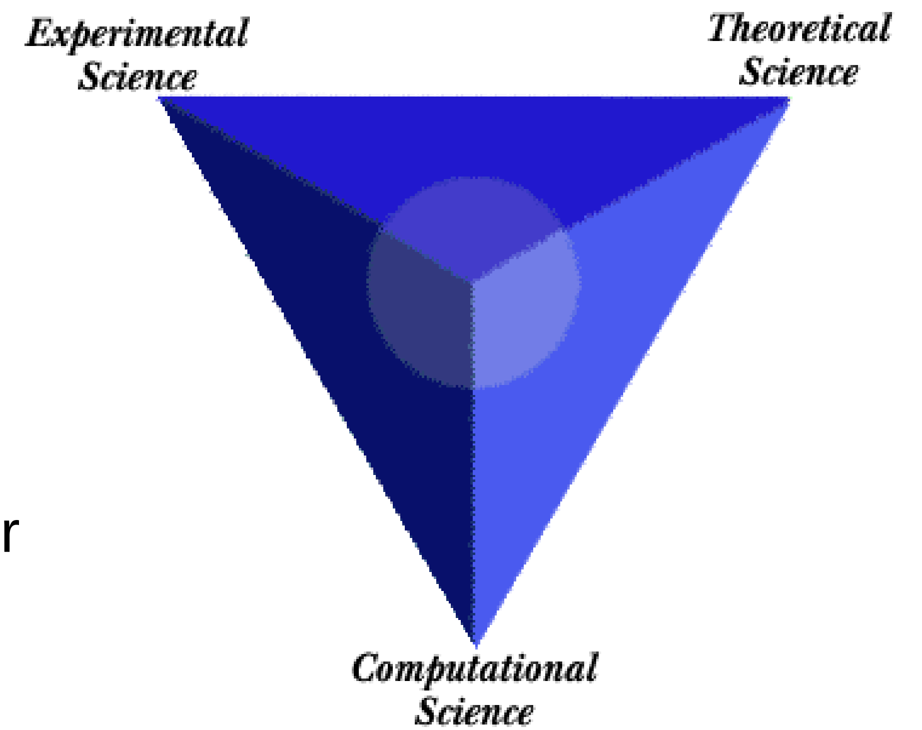
ISS004E12656

Application of Computational Science

- Spread of disease
- Weather prediction
- Analysis of space shuttle disaster
- Air quality model
- Creation of a galaxy simulation
- Potential for earthquake damage
- Modeling heart for treatment
- Others….

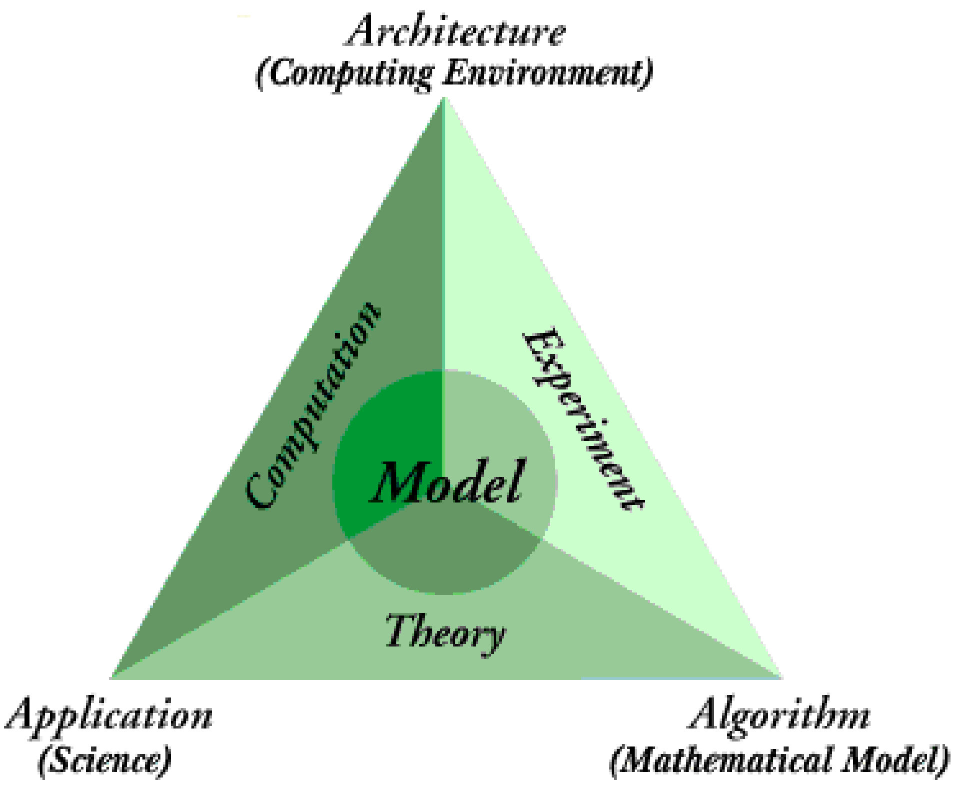
Science investigation

- Theoretical
  - Rich in concepts
  - Can develop only a few analytical methods
- Experimental
  - Heavy costs
  - Specific arrangements
- Numerical simulation
  - Complementary choice
  - Requires high computational power



Science investigation

- Computational science is a scientific endeavor (application) that is supported by the concepts and skills of mathematics (algorithms) and computer science (architecture).
- Central to any computational science problem is the science itself



Requirement to Solve a Computational Science Problem

- A Scientific Model (A description of the system)
- A Mathematical Model (A translation of the description to a set of mathematical equations)
- Computation (Solving the mathematical equations with a computer)
- Science Analysis and Interpretation

Description of the system

- Projectile motion

  A projectile is an object that rises and falls under the influence of gravity, and projectile motion is the height of that object as a function of time.

- Factors that influence the height of the projectile include:
  - the height from which the object dropped or thrown
  - upward/downward velocity
  - the pull of gravity on the object.

Description of the system

- Example problem:

  A projectile is fired vertically from the edge of a bridge at an initial velocity of 40 meters/sec. The bridge is at a height of 60 meters above the surface of water. We need to calculate:

  1. The height (relative to the bridge) reached by the projectile after 2 sec.
  2. The time taken for the projectile to reach the maximum height.
  3. The maximum height (relative to the bridge) reached by the projectile.
  4. The time taken for the projectile to hit the surface of water.

Mathematical Model

- The general formula for projectile motion is given by:

  $y(t) = –0.5*gt^2 + v_0t + y_0$,

  where $y(t)$ = height (at time t)

  t = time in seconds

  g = acceleration due to gravity

  v_0 = initial velocity

  y_0 = initial height

Computation

- Write a computer program to calculate and plot separately the height of the projectile and the velocity of the projectile for t from 0 till it hits water

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import host_subplot

v0    = 40
y0    = 60
t     = np.linspace(0,9.5,96)
y0_t  = y0*np.ones((np.size(t)))
y_t   = -0.5*9.81*t**2 + v0*t + y0
vel   = -9.81*t+v0;
t_max = np.max(y_t);
y_max = -0.5*9.81*(t_max)**2 + v0*(t_max) + y0

fig, ax = plt.subplots()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.plot(t,y_t, 'r',linewidth=2.5)
ax.plot(t,y0_t, 'black',linewidth=2.5)
ax.set_xlabel('Time (Sec)')
ax.set_ylabel('Height (Meters)')
plt.savefig('projectile.png')
```

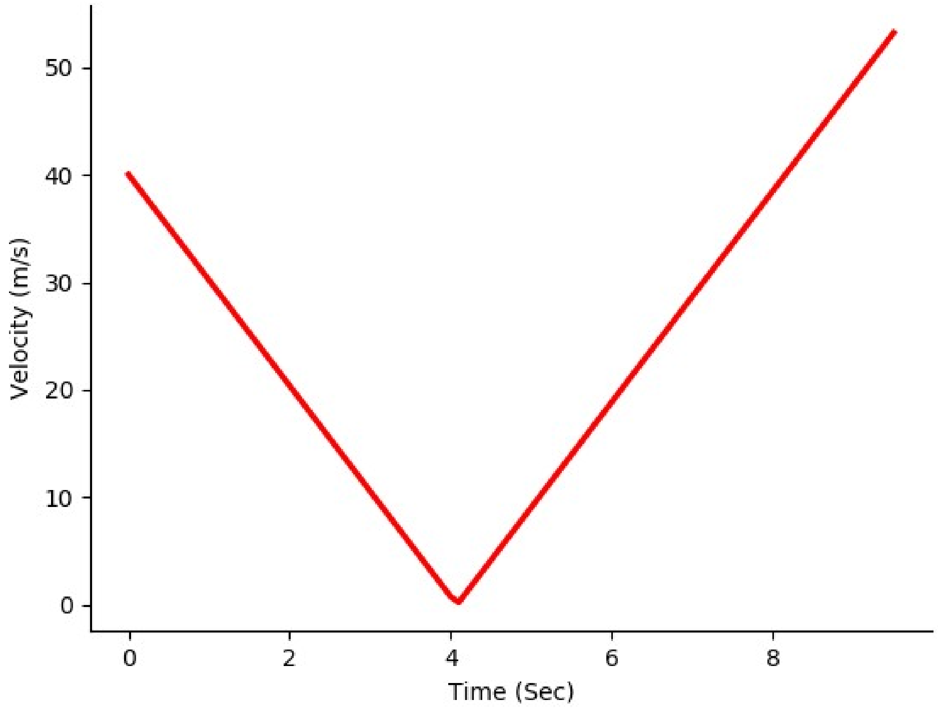
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import host_subplot

v0    = 40
y0    = 60
t     = np.linspace(0,9.5,96)
y0_t  = y0*np.ones((np.size(t)))
y_t   = -0.5*9.81*t**2 + v0*t + y0
vel   = -9.81*t+v0;
t_max = np.max(y_t);
y_max = -0.5*9.81*(t_max)**2 + v0*(t_max) + y0

fig, ax = plt.subplots()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.plot(t,np.abs(vel), 'r',linewidth=2.5)
ax.set_xlabel('Time (Sec)')
ax.set_ylabel('Velocity (m/s)')
plt.savefig('velocity.png')
```

Science Analysis and Interpretation

- Plots of height and velocity of the projectile



Assignment 1

Prepare a short report/presentation that explains a simple physical problem by applying the following procedures:

1. Description of the system
2. Translation of the description to a set of mathematical equations
3. Solving the mathematical equations using a FORTRAN computer language
4. Science Analysis and Interpretation

Specialization areas of Computational Science

- Computational Biology
- Computational Chemistry
- Computational Control Theory
- Computational Earth Science
- Computational Mechanics and Dynamics Systems
- Computational Operational Research
- Data Science
- High performance computing

Computational Biology

- Understand the principles and some methods of genomics, gene expression, and proteomics that aid precision medicine, modern plant, and animal breeding.
- Emphasis on problem-solving in biology and bioinformatics, especially through algorithms.
- Modeling and simulation of biological networks
- Computational sequence analysis
- Algorithms for reconstructing phylogenies

Computational Chemistry

- Simulation of chemical processes
- Many-electron wave functions, and matrix elements
- Hartree-Fock molecular orbital theory
- Density functional theory.

Computational Control Theory

- Basic facts of Linear time-invariant systems (LTID) Systems, Stability, Reachability and Controllability, Feedback Control, Observability, Observers, Transfer matrix, and Realization.

Computational Earth Science

- Environmental Modeling
- Computational Hydrogeology
- Geo-statistics
- Advanced Spatial Analysis and Modeling
- Geochemical Modeling
- Palaeo-stress Analysis in Earth Sciences

Computational Mechanics and Dynamics Systems Courses

- Finite Element Methods
- Computational Fluid Mechanics and Heat Transfer
- Qualitative and Numerical Approaches to Differential equations
- Boundary Integral Methods
- Computational Epidemiology
- Nonlinear Dynamics and Chaos

Computational Operational Research

- Application of Linear models: transportation Problem, transshipment problem, assignment problem, Inventory problem; data envelopment analysis; Integer Programming
- Network Optimization
- Discrete Optimization
- Decision Theory