Numerical Computing using Python

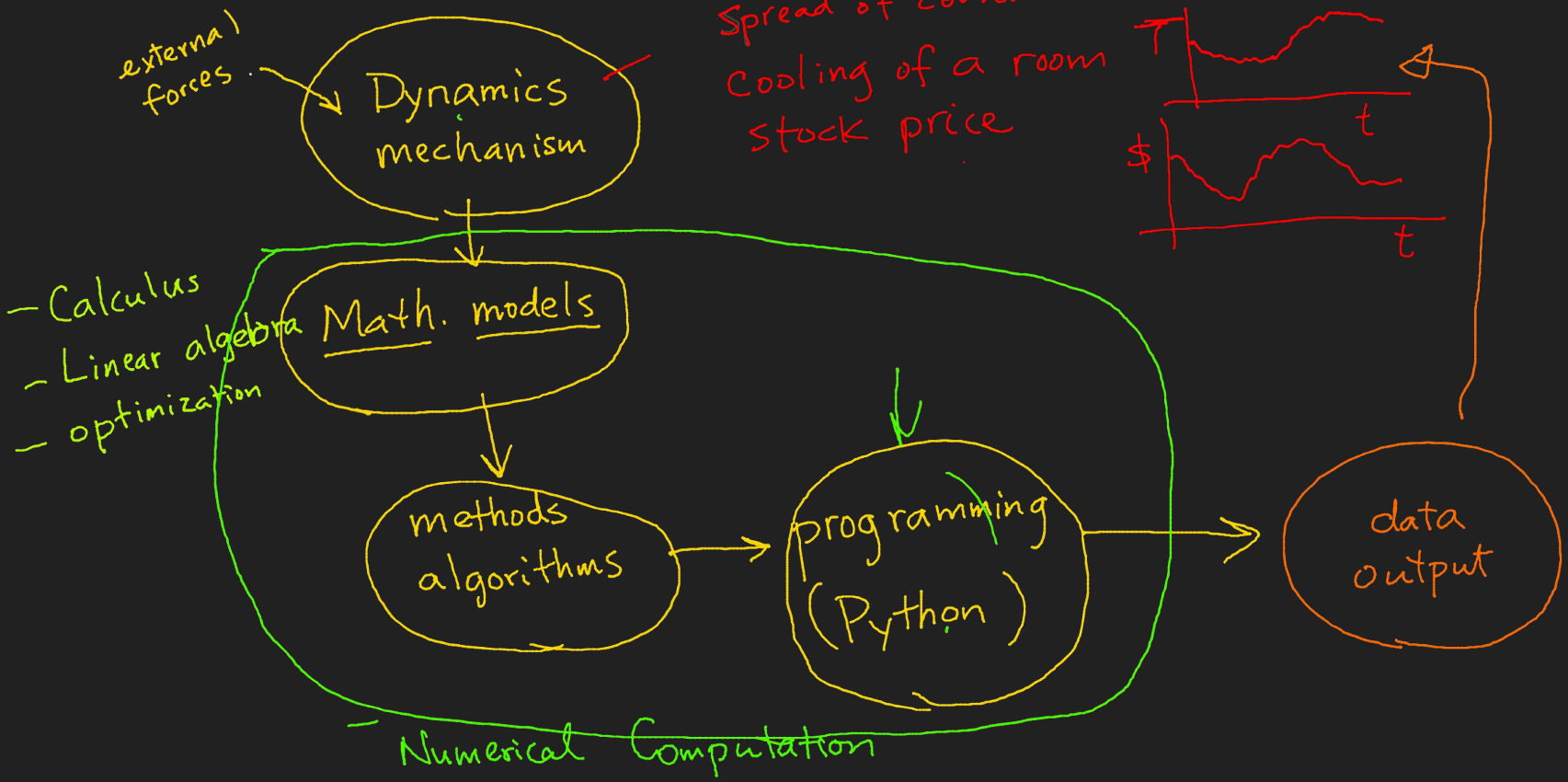
Lecture 2

S.C. Huang

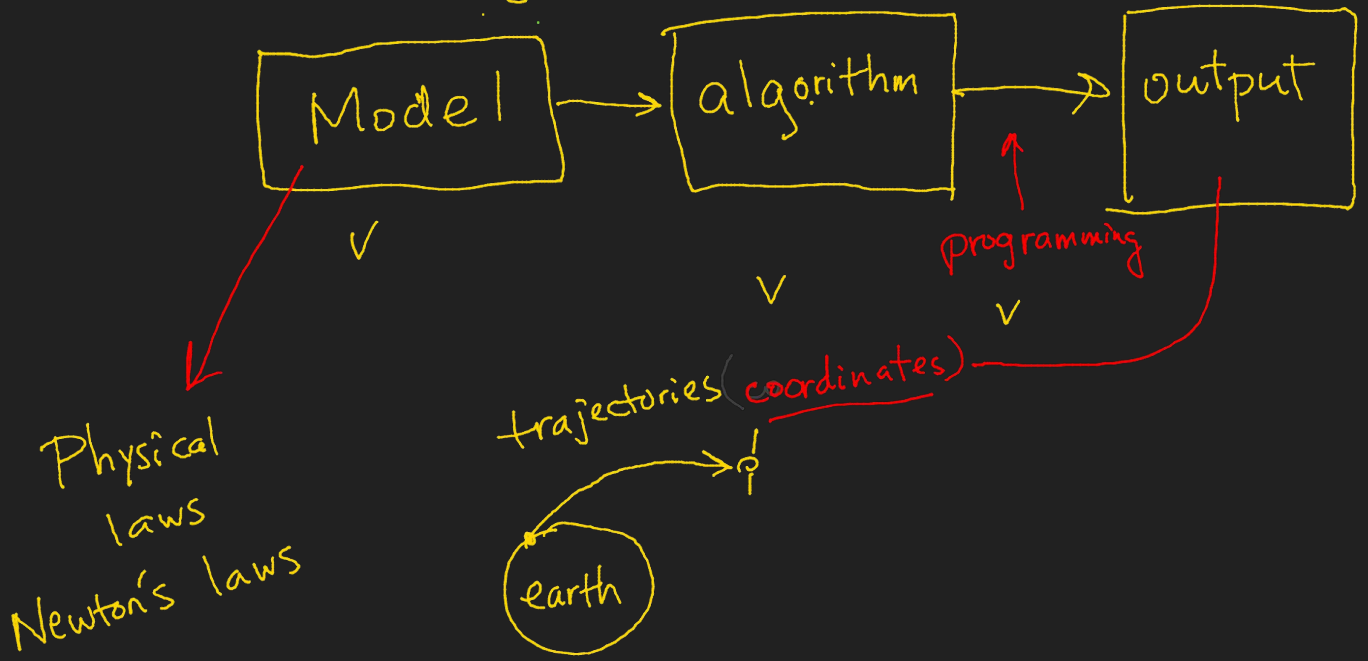
7pm 2020-07-02

Numerical computing: the big picture

Computation



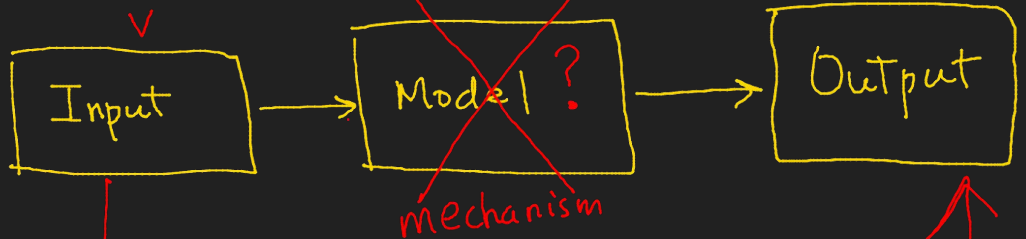
Scenario 1 : know the model
want: generate output

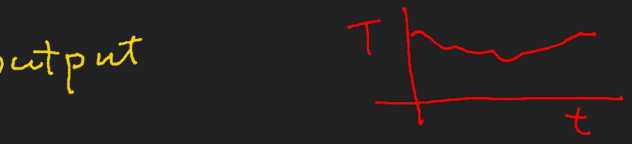


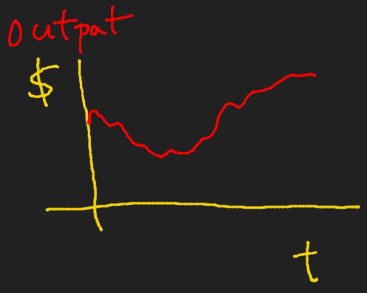
Scenario 2

Know: Input & output

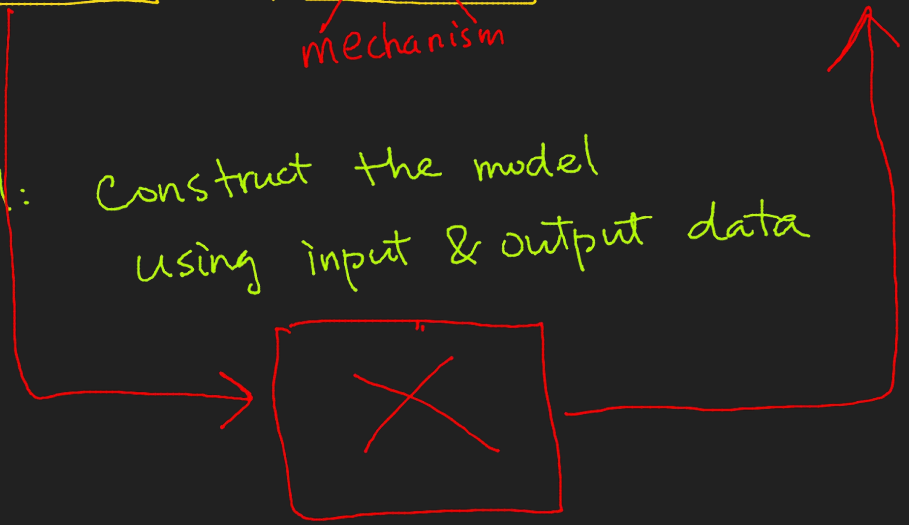
model ?







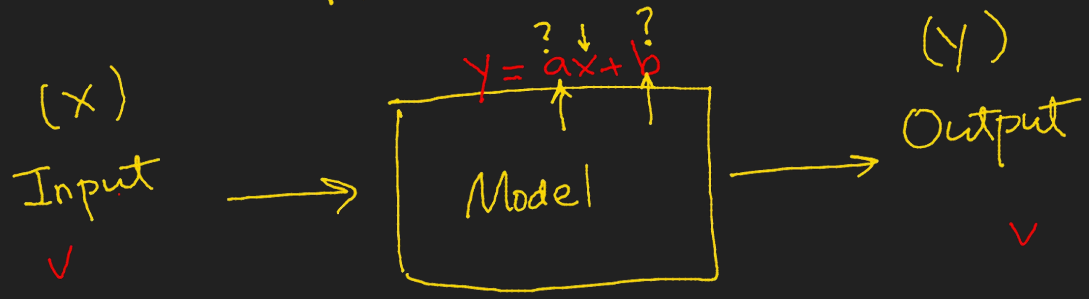
Goal: Construct the model using input & output data



Scenario 3

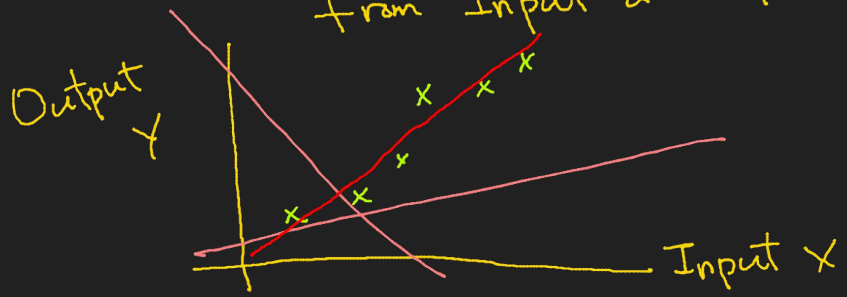
Know part of the model

Have Input & output



Goal: Find model parameters from Input & output

X, Y —find→ a, b



Disease Transmission Modeling (Example)

- S – # of susceptible population
- I – infected population
- R – recovered population

N = total pop. = $S + I + R$

rate of change

$$\frac{dS}{dt} \sim -S \cdot \frac{I}{N}$$

$$\frac{dI}{dt} \sim +S\frac{I}{N} - I$$

$$\frac{dR}{dt} \sim +I$$

infection rate

recovered rate

$$\frac{dS}{dt} = -\beta \frac{S \cdot I}{N}$$

$$\frac{dI}{dt} = +\beta \frac{S \cdot I}{N} - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

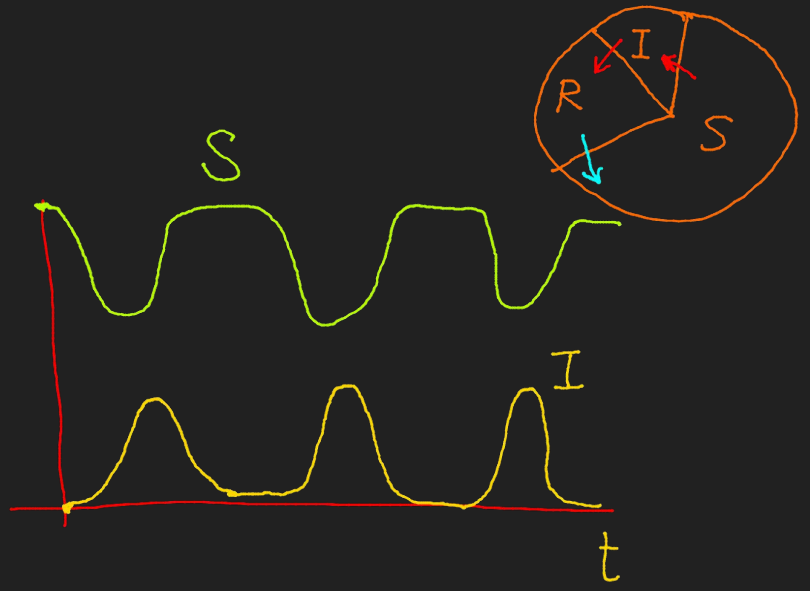
β, γ: model parameters

Finite immunity (eg. flu)

$$\frac{dS}{dt} = -\beta \frac{SI}{N} + \xi R$$

$$\frac{dI}{dt} = +\beta \frac{SI}{N} - \gamma I$$

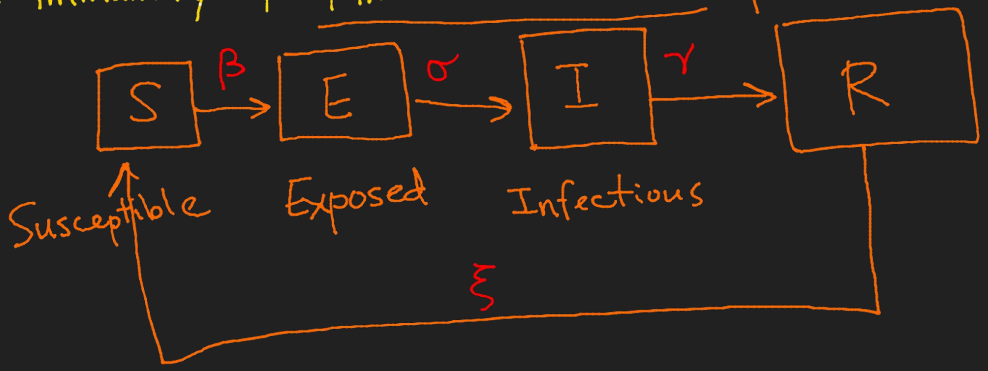
$$\frac{dR}{dt} = +\gamma I - \xi R$$



β, γ

ξ – immunity factor

Finite immunity + Finite incubation period

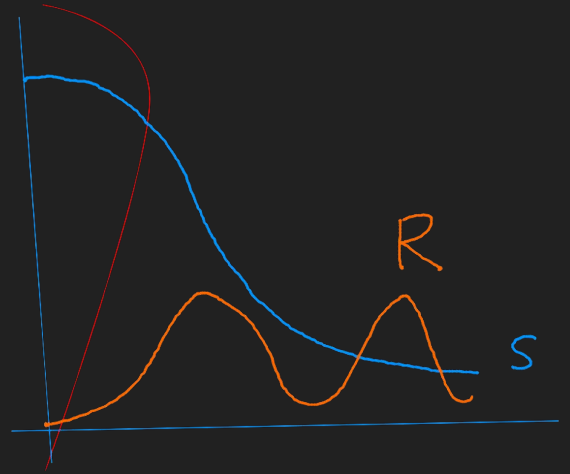


$$\frac{dS}{dt} = -\beta \frac{SI}{N} + \xi R$$

$$\frac{dE}{dt} = +\beta \frac{SI}{N} - \sigma E$$

$$\frac{dI}{dt} = -\gamma I + \sigma E$$

$$\frac{dR}{dt} = +\gamma I - \xi R$$



Python vs other solutions

- C, C++, Fortran
  - Fast
  - but more difficult (no interactivity, low-level etc)
- Matlab
  - easy to use, friendly environment, commercial support, syntax designed for linear algebra
  - base language not strong, restrictive
- Julia
  - Easy connect to Python or C
  - still young (but worth watching)
- Python
  - Very mature, clear code
  - Many use beyond scientific computing

Python ecosystem

- Python is a generic modern computing language
- Core language: flow control, data types, data collections (list, dict, etc)
- Modules from standard library: string, file, network, etc.
- A large number of specialized modules: web, scientific computing, data analysis
- Development tools: testing, documentation generation etc.

Main Python numerical modules

- numpy http://www.numpy.org/
  - powerful numerical arrays that support a variety of fast operations
- scipy http://www.scipy.org/
  - numerical routines (many are built on top of numpy)
  - optimization, regression, interpolation etc.
- Matplotlib https://matplotlib.org/
  - 2D plotting in publication quality
  - simple 3D

Specialized modules

- pandas
  - data analysis
- statsmodels, seaborn
  - statistics
- scikit-image
  - image processing
- scikit-learn
  - machine learning
- pytorch, tensorflow, etc.
- Many many others

Interactive Python environments

- ipython https://ipython.org/
- Jupyter https://jupyter.org/
  - "notebook" (text + code + plot) in web browser
  - Jupyter notebook, Jupyter lab
- Python IDEs
  - PyCharm, Spyder, and several others
  - https://wiki.python.org/moin/IntegratedDevelopmentEnvironments