# An Introductory Lecture to Mathematical Modeling

郭方健
richardkwo@gmail.com
http://richardkwo.net

Yingcai Experimental School, 09'

March 21, 2013

# My personal experience

- Experience with contests:
  - 2011 Summer: training in school
  - 2011 CUMCM (China Undergraduate Mathematical Contest in Modeling) (国赛): 2nd prize in Sichuan Province
  - 2012 MCM/ICM (Mathematical/ Interdisciplinary Contest in Modeling) (美国赛): Outstanding Winner for ICM
- What I learned
  - *Do not* prepare for the MM Contest, *do* prepare for **scientific research** instead!
  - Many less important points that I will talk later... :)

# What is a mathematical model?

Okay, some quotes from *Wikipedia*:

- A mathematical model is a description of a system using mathematical concepts and language.
- The process of developing a mathematical model is termed mathematical modeling.
- A model may help to explain a system and to study the effects of different components, and to make predictions about behavior.
- Mathematical models are used in
  - the natural sciences (such as physics, biology, earth science, meteorology)
  - engineering disciplines (e.g. computer science, artificial intelligence)
  - the social sciences (such as economics, psychology, sociology and political science)

# What is a mathematical model?

In my humble words...

**A small note**

*A mathematical model roughly means the* ***theory*** *(i.e.* ***formalism****) for a problem.*

Examples:

- The interaction and motion of physical entities — *Newtonian mechanics*
- Signals and systems — *What you have learned: LTI systems, response functions, ...*

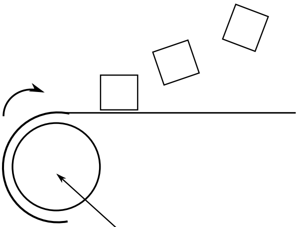
# What is a mathematical model?

To summarize:

- **Mapping** real world entities/mechanisms/phenomenon to *mathematical abstractions*.
  (e.g. interaction $\rightarrow F(t)$ in Newton's theory)
- Constructing and *developing* the model with **mathematical theories**.
  - Reasoning in pure math: *Definitions* $\rightarrow$ *Theorems* $\rightarrow$ *More theorems...*
  - Discovery by other means, especially *simulation*: **computer simulation** of math models
- **Solving** real-world problems within the *theoretical framework* that you developed.
  (proposing a solution that proves to **work** based on your theory)

# An illustrative example

Too abstract? Let's see an example.




- Mapping: $m, P, v, \mu, \Delta t, \cdots$
- Theory: Newtonian mechanics
- Solution: A **minimum** input power so that the boxes will not collide into each other?
  - Strong math: I can *analytically* solve it!
  - Weaker math: I do not know, but I can run a simulation to see how big $P$ should be.

# What makes a good mathematical model?

**Question:** What makes a *good* mathematical model?

Here I summarize some essential **principles** in my opinion.

- Reasonable: reasonable mapping between the real and the math
- Formal: formal way of developing theories:
  - Mathematically sound/rigorous.
  - New insight (**Do not** just restate the question in math without any new *insight*)
- Generalizable: working on a general *framework* rather than too specific
  - Can I solve the problem if the occasion changes?
  - What else can I do/learn with this model?
- Simplicity: choose the simplest one that can solve your problem!
  - Occam's razor
  - **Do not** be obsessed with fancy stuff.
  - "Simple enough, but no simpler" — Einstein
- Verifiability: how do you know your model is not wrong?
  - Does your solution solve your problem?
  - Compliant with *data*.
  - Other supportive theories/conclusions.

# Common pitfalls

You are probably **doing it wrong** if you have any of these feelings:

- ☐ Our model is not a "math model" at all!
  — You did not find the right math tool to address the problem.
  — You did not well define the stuff mathematically (dirty mapping).
- ☐ Our model and solution are unrelated!
  Or: we only have an algorithm. Can we call it a model?
  — Your math model must explain *why* you propose the solution and *why* your solution works.
- ☐ We built our model but still cannot solve the problem!
  — Model too complex or you do not know how to solve.
- ☐ We have one model for Question 1 and another for Question 2!
  — Model too specific, not generalizable.
- ☐ We have a perfect model, so what?!
  — Do you understand/clarify the problem that you want to solve?

# Common pitfalls

- Our model can explain everything!
  — Then can anything prove it wrong? If not, it is still **not verifiable**!
- Our model is just a linear combination of several answers!
  — This is the easiest but dirtiest way of *model selection*.

# What model should I use?

Consider

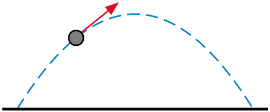
- What real-world problem do we have?
- What do we have in our theoretical arsenal?
- Does the model capture the most important **mechanism** of the problem? (white/black boxes)
- Can use the model to **solve** the problem?

Then, I will briefly several categories that may be useful for your future modeling tasks.

# Differential/Difference Equations

This category is often called as **Dynamical Systems**.

- **Continuous time**
  - Ordinary differential equations (ODE).
    e.g. Newtonian mechanics



    **Solution**: analytical, numerical
  - Partial differential equations (PDE).
    e.g. heat, fluid, electrodynamics, etc.
    **Solution**: Only several simplest can be analytically solved.
    And you may need to know some numerical solvers (c.f. MATLAB's PDE toolbox).
- **Discrete time**: usually recursively defined relations.
  e.g. Fibonacci numbers $F_n = F_{n-1} + F_{n-2}, F_0 = 1, F_1 = 1$.
  **Solution**: usually easier to solve with *characteristic equations*, *transforms*, etc. Also easy for simulation.

# Differential/Difference Equations

Concerns with *dynamical systems*

- Initial conditions
- Stability

# Probabilistic models

Probabilistic models are something I want to **stress** here.
They are **definitely overlooked** in our undergraduate training compared to their role in modern science and engineering.

### A small note

*Probabilistic models are common in problems with **abundant data**, **classification**, **prediction**, etc.*

# Probabilistic models

Something you may want to learn afterwards:

- Stochastic processes:
  - Markov Chains, Hidden Markov Model (HMM)
    e.g. application in disease/information spreading
  - Poisson process, branching process e.g. queueing, population growth
- Machine/Statistical learning & Pattern recognition
  - Supervised learning (logistic regression, neural networks, etc.)
    e.g. hand-writing recognition and other common classification tasks
    e.g. ICM 2012: *identifying criminals in a social network*
  - Unsupervised learning
    e.g. clustering, dimensionality reduction

Concerns with *probabilistic models*:

- Parameter estimation from data (training/learning)
  - MLE or MAP? (frequencist or Bayesian)
  - Optimization
    e.g. gradient descent and other optimization methods (ref. optimization books and MATLAB's opt toolbox)

# Programming/Planning

This is also called *optimization* and itoperational research.
Some math you may want to learn:

- Linear programming, integer programming and their solvers
- Basic non-linear programming
- Optimization methods
    - With gradient and even Hessian
    - Gradient-free methods (ref. MATLAB)
    - General **heuristic searching algorithms** (if too hard to solve)
      e.g. simulated annealing, genetic algorithms

Concerns with *Programming/Planning*:

- How to formulate your decision variables?
- How to define the target/reward function?
- Solver available or we have to do with a heuristic search?
  Then how large is your search space?

# Programming and software usage

Of course, there are *many other* interesting mathematical theories/methods that are useful for modeling. Anyway, I must pause here.

Let me talk a little about softwares.

- *C/C++*: generally, we **do not** use it for **taking contests**. But they are powerful when you care for performance.
- *MATLAB*: we use **a lot**.
  **Strength**: Linear algebra, numerical simulation
- *Python*: **I** use **a lot**.
  **Strength**: Scientific computation/simulation (*NumPy, SciPy*), network analysis (*NetworkX*), plotting (*MatPlotLib*).
- *Mathematica*: **I** sometimes use for an analytical solution.
  **Strength**: It is **without rival** for solving something **analytically**.
- *R*: **I** love to use it more often for dealing with data. **Strength**: Doing statistics smoothly with **data**, beautiful plotting.
- *LaTeX*: **Our team** loves to use it for professional typesetting.
  **Strength**: Document produced with publishable quality and academic flavor.

# Thank you

*Thanks a lot, any question?*

- Reach me via `richardkwo@gmail.com`
- Feel free to download the slides from my personal website `http://richardkwo.net`
- You can also find our solution summary and outstanding winner paper for ICM 2012 on my website. :)