Computational Modeling

*CPS 111 Supplementary Lecture Notes*
Carlo Tomasi – Duke University

Spring 2007

**Warning.** These notes are being written as the course progresses, and are provided only when there is insufficient coverage of a topic in the textbook. As errors are discovered, old notes will be corrected and the new version will be posted online.
1 General Remarks

Practicalities

Course mechanics. Information on course mechanics and other aspects of this class are posted on the web page http://www.cs.duke.edu/education/courses/spring07/cps111/. Please read this page thoroughly now, and then frequently during the semester for announcements.

Project. The sub-page on project guidelines is still empty. This is because we will negotiate the scope, level, and format of the projects over the first few weeks of class based on your background and interest. Homework assignments will guide you through the various steps of the project in the second half of the course.

Recitation. Recitation sessions on Fridays are mandatory. They start on January 19, so there will be no recitation on January 12.

Reading Assignment. Please read pages 1-18 of the textbook. If you do not understand something, make an effort to work through an example. If you still fail to understand, that’s OK, but do try hard. What counts in a first pass is your struggle with the material. Please ask questions in class.

Homework 1. The first homework assignment is being handed out together with this note and is due in class on Wednesday, January 17. This assignment is mostly for my own understanding of your interests and background. Because of this it will only be graded pass/fail (that is, you will receive either 0 or 100 points for your work). A passing grade requires a non-perfunctory\(^1\) answer to each question.

The Nature of Computational Modeling

Most sciences today are quantitative. As the phenomena to be understood become more complex, qualitative descriptions soon lose their predictive power. For instance, the weather here tomorrow depends on a large number of factors such as the distribution of pressure, temperature, and humidity in the atmosphere, ocean currents, ozone concentration, seasonal astronomical variables, and so forth. Even the answer to a question as simple as “will the average temperature increase or decrease between today and tomorrow?” can be 100 percent wrong if the importance of each of these and other factors is not correctly assessed in a quantitative way. Quantitative modeling is common even in the social sciences, where aspects related to human behavior and psychology are more difficult to quantify.

The phenomena that are modeled in science are so diverse that it would seem doubtful that they share any commonality. Indeed, a large part of modeling depends on the specifics of the

---

\(^1\)Perfunctory: characterized by routine or superficiality, or lacking in interest or enthusiasm (Merriam-Webster).
domain. To model weather one must understand physics, and to model commodity prices one must understand how markets work. In this sense, there is no such thing as “generic modeling.”

Yet in all mathematical modeling efforts one identifies a set of quantities that can be measured, called the input, and a set of quantities of interest, the output, that in some sense depend on the input. A collection of mathematical expressions, called a model, relate output to input, perhaps with the aid of some additional internal quantities. So in an abstract sense every mathematical model is a system that takes input quantities and transforms them into output quantities through mathematics.

For instance, input quantities in weather prediction may be the values of relevant variables now, and output quantities may be the values of the same variables tomorrow. For weather, the model is usually in the form of a set of partial differential equations\(^2\) that describe the rates of change of the variables over time and space.

Inputs to a commodity price predictor may be the average price of crude oil and average daily temperature in the winter, and the output may be an estimate of average heating oil prices in the same season. The system may be a function chosen out of some predefined class, say a neural net.\(^3\)

The description of a system is not necessarily mathematical. For example, one could provide a behavioral or otherwise qualitative explanation for why a heating oil retailer may decrease oil price when the average daily temperature increases. Similarly, weather could be explained in pictures or cause-effect relationships. However, in this course we are interested mainly in mathematical models, so the word “mathematical” will be usually left implicit. In summary,

A model or system is a mathematical transformation between inputs and outputs.

What is input and what is output is in the eye of the beholder. In the examples above, some quantities look more like “causes” (temperature today or average crude oil prices) and others look more like “effects.” Causes are usually modeled as inputs, and effects as outputs. The choice, however, is not always obvious. For instance, we could try to predict atmospheric pressure from other “input” quantities (temperature, humidity, currents). Pressure would then be an output. If we have pressure data and we are interested in estimating temperature instead, pressure becomes an input, and temperature an output.

One of the most important modeling decisions is to define inputs, outputs, and other quantities that are relevant to the problem.

Time may or may not be explicitly part of a model. In the weather example, time is crucial, and quantities are defined as functions\(^4\) of time (and space, and possibly other variables). In the

\(^2\)Do not worry now about the exact meaning of “partial differential equation.” We will touch (lightly) upon this concept later in the course.

\(^3\)Again, no need to worry about neural nets.

\(^4\)You do know what a function is. Please review if you forgot. The notion is revisited later on in this lecture under “types of model.”
oil price example, time seems to be there, because we talk about average daily prices, or average prices during winter. However, time would have an explicit role in the model only if we were interested in describing, say, the delay between a price fluctuation for crude oil and the resulting fluctuation for heating oil. Otherwise, there is no need to represent time anywhere explicitly in the system. In fact, inputs and outputs for our original price example are probably not even functions, but just numerical values (e.g., $58 per barrel).

Among all types of variable (temperature, price, time, pressure), time has a special role, mainly because it “goes on forever” and "is irreversible.” When time is involved, the equations in the model describe rates of change, either between discrete time instants, or instantaneously, at a given point in time. Think of the velocity of a car as the instantaneous rate of change of the car’s position, and the price variation of one share of a certain stock between consecutive Wall Street closings as a discrete change.

A model whose inputs and outputs are functions of time is called a dynamic system. The system is discrete-time if time proceeds in distinct clock ticks, and continuous-time if time flows without quanta.

Often one just says “discrete” and “continuous” rather than “discrete-time” and “continuous-time”, but this can be misleading, as quantities in the model other than time can be continuous or discrete.

Another important set of variables denote positions in space. For instance, temperature in the atmosphere is a function of time and usually two or three spatial coordinates: two (perhaps longitude and latitude) if we describe temperature on the ground or at a fixed altitude, three if we are also interested in temperatures at different altitudes. Space is mathematically less peculiar than time (for instance, there is no notion of “irreversibility”), but has nonetheless a special place in modeling because it relates to the notion of shape that people are so often interested in.

Challenges of Modeling

The first challenge in modeling is to understand the problem, list the relevant quantities, and understand how they interact. This stage of model design is entirely in the realm of the application domain: meteorology for weather predictions, economics for price estimation, and so forth.

However, the main criterion to follow in this stage applies to all domains, and is discussed next.

Model Complexity

Make the model as simple as possible, but no simpler.

Consider for instance modeling a pendulum, i.e., a bob hanging from a rod that oscillates around a pivot. The action of gravity on the bob seems crucial, as without it the pendulum will
not move. Do we need to consider air drag, which will slow down motion relative to what would happen *in vacuo*?

The answer depends on what we use the model for. As we add more and more variables, the mathematics will become more and more complex, and will obscure our understanding of the interrelationships among quantities. In addition, more quantities to describe require more data: if we do consider air drag, we need to know its quantitative values and parameters it depends on.

Equations have parameters, and parameters need to be estimated from data. The more parameters, the more data.

Also, more equations to solve provide more opportunities for all the types of problems that arise when implementing solutions to them on a computer. Not only do the problems increase in number: they interact, and it becomes harder to understand how they do so, and therefore fix the problem.

Computational difficulties escalate with the complexity of a model.

Parameter estimation and proper handling of computational issues are very serious concerns. However, the most important motivation for simple model is perhaps conceptual, rather than practical:

Simple models provide deeper insight than complex ones. This is true even when simplicity results from approximation.

For instance, a simple model of a pendulum will include only gravity, allow only for small oscillations, neglect the mass of the rod, and ignore air drag and other effects. With all these approximations, the model will predict that the time it takes for the pendulum to swing left to right and back again is

\[ T = 2\pi \sqrt{\frac{l}{g}} \text{ seconds} \]

where \( l \) is the length of the pendulum’s rod in meters, and \( g \) is gravitational acceleration in meters per second square. This statement is clear. If I want the pendulum to swing twice as slowly, I need to make the rod four times long. If I bring the pendulum to the moon, where gravity is one sixth of what it is on Earth, the pendulum will swing about \( \sqrt{6} \approx 2.4 \) times more slowly. If I make the rod four times longer and I bring the pendulum to the moon, the effect of the combination is multiplicative.
Allowing for large oscillations makes the formula more complex:

\[ T = -2\pi \sqrt{\frac{l}{g}} \ln \cos(\theta_0/2) \] seconds

where \( \theta_0 \) is the angle of release. The basic dependence on \( l \) and \( g \) is still discernible here, so the added complexity is harmless. If on the other hand we include the mass \( m \) of the rod compared to the mass \( M \) of a spherical bob of radius \( R \), the new formula is

\[ T = 2\pi \sqrt{\frac{m l^2/3 + 2/5MR^2 + M(l + R)^2}{g[m + M(l + R)]}} \] seconds .

This is the period of a so-called compound physical pendulum, still for small oscillations and no air drag. Making simple predictions from this formula is much harder. The formula may be more accurate, but we paid for accuracy with insight.

Is the greater accuracy worth it? Figure 1 plots the pendulum period in seconds for the simple (solid) and the compound physical (dotted) pendulum for different lengths of the rod between half a meter and two meters. The periods are between about 1.4 and 3 seconds, and the maximum discrepancy between the two periods is less than 10 milliseconds. Whether this difference is important or not depends on the application. If we are trying to build a grandfather’s clock, the answer is definitely yes. If we are trying to understand the overall relationship between rod length and pendulum, even in the initial stages of design for a grandfather’s clock, the answer is a resounding no!

At some point, formulas become so complex that they are even hard to write. For instance, adding air drag makes the pendulum a damped oscillator, and if one considers large oscillations with air drag there isn’t even a period to talk about, as consecutive swings take slightly different amounts of time. It still makes sense to talk about an approximate period, but the model has become too accurate to reveal this important parameter directly.

When this is the case, one usually abandons formulas and resorts to simulations. In the pendulum case, the simulation would eventually produce a plot of the angular displacement of the pendulum’s rod over time. However, it is important to observe that simpler models are usually better than complex ones at least when accuracy is not the primary goal. Even when accuracy is the primary goal, added complexity must be weighed very carefully, because it leads to systems that become harder to both understand and implement. A fascinating example of a system where the full complexity is unavoidable is the Global Positioning System. Please consult the Wikipedia at http://en.wikipedia.org/wiki/Global_Positioning_System for details.

**Types of Model**

Once the key quantities of a model are identified, one has to come up with the equations that relate them to each other. A daunting array of mathematical concepts and computational techniques are
available to the modeler, but options can be narrowed down effectively by considering what type of problem is to be solved.

One way to put some order in the notion of a problem “type” is to note that a “quantity” can be generally viewed as a function. To review, a function is a mapping from a set of numbers called the domain to another set of numbers called the codomain, with the following property: a function associates exactly one element of the codomain to each element in the domain.

It is fairly obvious that the displacement $\theta$ of the pendulum discussed earlier can be seen as a function of time, $\theta = f(t)$. However, even a single number can be viewed as a function, in a pinch. For instance, the number -7.2 is a map from the set $\{0\}$ (or any other singleton) to the reals. This map has a single value, $f(0) = -7.2$.

If this sounds contrived, that’s because it is. However, this simple notational trick let’s us unify the discussion of model types. We can then categorize a model depending on the type of functions that it describes:

- Discrete or continuous domain
- Discrete or continuous codomain
- Zero, one, or multi-dimensional domain
- Scalar or vector codomain
- Deterministic or stochastic
- Free or constrained
These options give rise to \(2 \times 3 \times 2 \times 2 \times 2 \times 2 = 96\) possible combinations.

For instance, atmospheric temperature may be viewed as a function \(T(x, y, z, t)\), where \(x\) and \(y\) are latitude and longitude, \(z\) is elevation, and \(t\) is time. The codomain is typically continuous, since the values of \(T\) are not discretized. Whether the domain, that is the set in which \(x, y, z,\) and \(t\) live, is continuous or not depends on how we want to look at it. Space \((x, y, z)\) and time \((t)\) are inherently continuous, but we may only be interested in temperatures at noon and at a fixed set of weather stations. In that case, we would think of the domain of \(T\) as a discrete set. Hybrid choices are possible as well, with, say, discrete time and continuous space. The domain is four-dimensional (four variables \(x, y, z,\) and \(t\)), and the codomain is scalar: The temperature is a single number, as opposed to a vector. If we were measuring wind velocity, that would be two numbers (for instance, speed and a direction angle), \(i.e.,\) a vector. If we attempt to capture every aspect of weather by measurement, we would write deterministic equations for the temperature model. This means that if I know the temperatures and all other relevant factors at time \(t\) everywhere, my equations would let me predict the temperatures at time \(t'\), and if the prediction is made twice from the same starting conditions the same result obtains twice. A stochastic (or random) model would instead make a statement of the following flavor: “If I start from these conditions at time \(t\), then at time \(t'\) I will observe a temperature \(T_1\) with probability \(p_1\), or a temperature \(T_2\) with probability \(p_2\), or . . . .” Finally, any model that describes temperature would have to incorporate some constraint that ensures that no temperature ever occurs that is below absolute zero. So the constraint would be

\[
T \geq 0^\circ K \quad \text{or} \quad T \geq -273.15^\circ C \quad \text{or} \quad T \geq -459.67^\circ F
\]

depending on whether temperatures are measured in Kelvin, Celsius, or Fahrenheit degrees. If no such constraint is imposed, the problems is said to be free or unconstrained.

Surprisingly, almost every combination of choices in the list above corresponds to a different set of mathematical techniques. For instance, a free, stochastic problem with continuous domain and codomain, multi-dimensional domain, and scalar codomain would be ruled by a set of stochastic partial differential equations. These equations would become deterministic if no probabilities are involved; ordinary, as opposed to partial, if the domain is in one-dimensional; and they would become difference, or recurrence, equations, rather than differential, if the domain is discrete instead. For each of these changes, you would typically have to take a different college course to learn the corresponding technique in some detail.

Unfortunately, there is no fixed recipe for coming up with the appropriate equations for a given modeling problem. However, the taxonomy above lets you at least look up the proper books. The rest is physics, chemistry, economics, or whatever the relevant domain knowledge is.

**Computation**

The inputs to the computation are nearly always approximate (that is, wrong!). For instance, most data come from measurements, and these are made with physical devices of limited capabilities. In addition to these inherent errors, additional errors occur when a mathematical computation is implemented on a computer. Real numbers then become essentially integers (even floating-point
numbers are more like integers than reals in some sense), and one needs to round or truncate numerical values. These are numerical errors.

It makes little sense to measure errors by their absolute values: is 1 cm a large error or a small one? If we are measuring the length of the body of an ant, 1 cm is unacceptably large. If we are measuring the diameter of the Earth, 1 cm would be an astonishingly small error. So instead errors are usually measured in relative terms. If a numerical value that is supposed to be equal to \( x \) is instead equal to \( x + \Delta x \), where \( \Delta x \) is the absolute error, then \( x \) is said to be affected by the relative error

\[
\frac{|\Delta x|}{|x|}
\]

where the bars denote the absolute value of a number. If an ant is about 5 mm = 0.5 cm in length (depends on the particular species), then a 1 cm absolute error results in a relative error of \( 1/0.5 = 2 \) (or 200 percent). The radius of the Earth is approximately 6,400 km = 640,000,000 cm and a 1 cm absolute error corresponds to a relative error of \( 1/640,000,000 \approx 1.5 \times 10^{-9} \), or one part in one and a half billion.

Typically, inherent errors in the data are much larger than numerical ones, so it would seem that the latter can be ignored. However, both types of errors tend to propagate through computations, and often to amplify as a result. Whether this propagation results into disaster depends on both the problem itself and the computational algorithm used to solve it. Problem-dependent amplification of errors is called sensitivity, and computation-dependent amplification is called instability.

As an example of sensitivity, consider solving the system of equations

\[
\begin{align*}
0.0001x &= 1 \\
x + y &= 9999
\end{align*}
\]

where \( a = 1 \) and \( b = 9999 \) are “data” (say, the result of some measurement). The solution \((x, y)\) is easy to find by substitution:

\[
x = 10000 \quad \text{and} \quad y = 9999 - 10000 = -1 .
\]

This is a sensitive problem. For instance, if we perturb the data \( a = 1 \) by just one percent, and replace it with \( a' = 1.01 \), and leave \( b = 9999 \) unaltered, we obtain

\[
x = 10100 \quad \text{and} \quad y = 9999 - 10100 = -101 .
\]

While the relative change in the result \( x \) is modest,

\[
\frac{|10100 - 10000|}{|10000|} = 0.01
\]

or 1 percent, just as the change in the data, the relative change in \( y \) is much greater:

\[
\frac{|-101 - (-1)|}{|-1|} = 100
\]
or 10,000 percent! Sensitivity here is a property of the problem itself. We made no approximations in the computation, so the computation is not to blame.

The following system, on the other hand, is relatively insensitive to errors in the data $a = 1$ and $b = 3$:

\[
\begin{align*}
0.0001x + y &= 1 \\
x &= 3.
\end{align*}
\]

Again, substitution yields the solution

\[
x = 3 \quad \text{and} \quad y = 0.9997 .
\]

There are no dangerous multipliers here. For instance, replacing $a = 1$ with $a' = 0.99$ yields a new solution

\[
x = 3 \quad \text{and} \quad y = 0.9897 .
\]

The relative error in $x$ is zero, and that in $y$ is

\[
\frac{|0.9897 - 0.9997|}{|0.9997|} \approx 0.01 .
\]

just one percent.

So it is important to look for possible sources of sensitivity in a modeling problem. Even if the problem is insensitive, however, numerical inaccuracies can be exacerbated by an unstable algorithm. Consider for instance the linear system

\[
\begin{align*}
0.001x + y &= 1 \\
x + 2y &= 3.
\end{align*}
\]

This is by itself only moderately sensitive to errors in the data. A systematic algorithm for solving this and larger systems on a computer is Gaussian elimination, which you may have seen in high school: Multiply the first equation by 1000, and then subtract the result from the second equation. This will cancel $x$ from the second equation:

\[
\begin{align*}
0.001x + y &= 1 \quad \text{times } 1000 \text{ yields } x + 1000y &= 1000
\end{align*}
\]

and subtracting this from $x + 2y = 3$ yields

\[
-998y = -997 .
\]

A new system equivalent to the original one is then

\[
\begin{align*}
0.001x + y &= 1 \\
998y &= 997
\end{align*}
\]

(the two minus signs in the second equation can be harmlessly canceled).
From here we can solve for \( y \),

\[
y = \frac{997}{998} ,
\]

and replace this result into the first equation to obtain an equation in \( x \) alone:

\[
0.001x + \frac{997}{998} = 1
\]

which then can be solved for \( x \):

\[
x = 1000(1 - \frac{997}{998}) = 1000(998 - 997)/998 = 1000/998 .
\]

In summary, \( x \) is a bit more than 1, and \( y \) is just shy of 1:

\[
x = \frac{1000}{998} = 1.002004 \ldots \approx 1 \quad \text{and} \quad y = \frac{997}{998} = 0.998997 \ldots \approx 1 .
\]

Suppose now, however, that our computer can store only two decimal digits and an exponent of 10 for each number. For instance, 0.001 is stored as \( .10 \times 10^{-2} \), and 1 is stored as \( .10 \times 10^1 \). This is the usual floating point notation used in scientific calculations, just with a smaller number of digits than usual to make the illustration simpler. The number \( \frac{997}{998} = 0.998997 \ldots \) would need infinitely many digits, and is instead rounded to \( .99 \times 10^0 \).

Gaussian elimination on this floating-point computer would represent the transformed system

\[
\begin{align*}
0.001x + y &= 1 \\
998y &= 997
\end{align*}
\]

as follows

\[
\begin{align*}
.10 \times 10^{-2} x + .10 \times 10^1 y &= .10 \times 10^1 \\
.99 \times 10^3 y &= .99 \times 10^3
\end{align*}
\]

which is equivalent to

\[
\begin{align*}
0.001x + y &= 1 \\
990y &= 990 .
\end{align*}
\]

The discrepancy in the coefficients of the second equation is relatively harmless, since our choice of using two decimal digits in the computer representation means that we do not care about the third. The solution \( y \) is about right, within two decimal digits:

\[
y = 1 \quad \text{is close to} \quad 0.998997 \ldots \approx 1 .
\]

However, replacing this slightly approximated solution for \( y \) into the first equation yields a badly approximated solution for \( x \). The first equation becomes the floating point equivalent of

\[
0.001x + 1 = 1 ,
\]
which has (exact) solution \( x = 0 \):

\[
x = 0 \quad \text{is not close to} \quad 1.002004 \ldots \approx 1 .
\]

In this case, the errors come from a poor algorithm, not from the problem itself.

A more stable Gaussian elimination algorithm would do \textit{pivoting}. In the example, this amounts to switching the two equations:

\[
\begin{align*}
x + 2y &= 3 \\
0.001x + y &= 1
\end{align*}
\]

Amazingly, this simple change does the job. Gaussian elimination multiplies the first equation by 0.001 and subtracts it from the second to obtain the following equivalent system:

\[
\begin{align*}
x + 2y &= 3 \\
0.998y &= 0.997
\end{align*}
\]

which rounds to

\[
\begin{align*}
x + 2y &= 3 \\
0.99y &= 0.99
\end{align*}
\]

The solution for \( y \) is then \( y = 1 \), which replaced into the first equation yields

\[
x + 2 = 3 \quad \text{or} \quad x = 1 .
\]

The approximate solution \( x = 1 \) and \( y = 1 \) is very close to the exact solution \( x = 1.002004 \ldots \) and \( y = 0.998997 \ldots \).

In summary, Gaussian elimination is unstable, and Gaussian elimination with pivoting is stable. In general, a good solution to a modeling problem requires an insensitive formulation of the problem \textit{and} a stable algorithm.

\section*{A User's View of Mathematics}

It should be apparent from the previous discussion that mathematical skills are increasingly important in all sciences. For some, mathematics is a lifetime occupation. For us in this course, and often in our careers, mathematics is a tool needed to understand phenomena in different domains. Just as it is possible to use a word processor without knowing how one works, the argument goes, it ought to be possible to use a mathematical model, typically also in the form of a software package, without knowing mathematics.

To some extent, this is the case. For instance, today’s weather models are so complex that even expert meteorologists often use them, sometimes even in academic research, without knowing what equations are at work under the hood. They understand meteorology (with different degrees
of mathematical sophistication) and they have read the software’s user interface manual, and they can use the package effectively without knowing what a partial differential equation is.

However, the word-processor analogy is not entirely correct. When we use a word processor we may not know programming, but we do know the language we use the word processor to write in. Mathematics is the language of models, so we may have to know at least a little of it. At the very least, we need to know what difficulties may arise at the mathematical level if we want to be able to trust the model’s outputs. How do we know that the numbers that come out of it are not all garbage? If the model fails miserably and the software crashes, we may not be happy, but at least we know that a problem has occurred. It is much worse when the model keeps crunching away but produces poor results, as we may not realize this, and make bad predictions.

In addition, we may have to build the models ourselves if they are not already available. Plausibly, this requires much more detailed knowledge of mathematics. For simple phenomena, the models may be simple, and we may be able to come up with working code in a few days of work. This is a thrilling experience, and the insights that come from the effort of putting the model together are worth the trouble, even if the model is never translated into software. When writing a model, we cannot leave assumptions untold or quantities poorly defined. Mathematics demands clarity.

The most important role of a model in research is to force a clear statement of the relationships of interest.

For more complex problems, we may be able to adapt someone else’s formulas or even code. In that case, we need to understand at least enough mathematics to be able to glue different pieces together in a sensible way.

In brief, understanding of mathematics vis-à-vis domain knowledge in modeling is not a black-and-white issue: We need to understand some mathematics most of the time, but we can achieve quite a bit with relatively little math knowledge if we adopt a pragmatic attitude, a user’s view of mathematics. Here are some of its aspects.

**Build on simplicity.** Try to understand issues in a simple context, but in detail. It is often possible to transfer the resulting knowledge to more complex domains. The details of the transfer may be unclear, but if you read about the same issue in an unfamiliar context you will often be able to understand what is going on.

**Take a black-box view.** For any computation, you need to understand (precisely) what goes in, what comes out, and what could possibly fail to work. You definitely want to be able to detect failures when they occur. Fixing them would be great, but not always possible. Knowing exactly how the computation works is often unnecessary.

**Separate the concepts from the technicalities.** Mathematical statements and their proofs are sometimes complex, because they need to be water tight. However, if a statement is important, there must be some central core of truth to it that can be expressed more simply. For most purposes, this simple core is all you need to understand, as long as you know that someone
else has handled all of the details: the details must be known, but not by you. It may be very hard to separate the core truth from the technicalities. Looking at different books and papers for one that does it for you may save you some effort, but not always.

Other aspects of the user’s view of mathematics are more subtle, and will hopefully emerge during this course.

The earlier discussion of sensitivity and stability provides an example of the user’s view of mathematics. Understanding these issues on a $2 \times 2$ linear system is an instance of building on simplicity: The problem is small, so it can be worked out thoroughly. However, it is important to understand all details of it to internalize the main concepts and understand them well.

For larger linear systems, we can take a black-box view: we may not know exactly how to solve a large linear system. However, linear solvers exist. What goes in is a set of numbers that represent the coefficients of the system, and what comes out is the solution $(x, y, \ldots)$. We need some way to tell if the system is sensitive. By reading up on linear systems, we find out that a number called the condition number of the system is a measure of its sensitivity: a condition number of 1 represents an insensitive problem, and the more sensitive the problem the greater the condition number.

In addition, the Matlab manual tells us that the function $\text{cond}$ returns the condition number of a set of coefficients.

So we have all we need: the small example tells us what sensitivity is all about, and we trust that $\text{cond}$ will measure sensitivity for larger problems.

When looking for a stable algorithm, we understand the core concept of pivoting: always pick first the equation with the largest leading coefficient, so that division by small numbers is avoided. Making this actually work well and efficiently requires very delicate work, but we do not need to understand the delicacies: we have separated the core concept from the technicalities. Of course, it is crucial that someone has taken care of all those technicalities for us, and cleanly wrapped them into a numerical software package.

**Pedagogical Strategy**

We will take a user’s view of mathematics in this course. To this end, we will understand models in the simplest case of discrete dynamic systems, which are free, deterministic problems with one-dimensional, discrete domain and scalar codomain. We will then extend these to stochastic versions, and to systems with vector codomain. These are flexible and general enough to cover the vast majority of modeling issues and challenges. This relatively detailed study will take about half of this course, and will require some patience, curiosity, and attention to detail.

In the second part of the course, we will work on projects and read case studies. Projects will still be of the “small” variety, so we can handle the details. For the case studies, we will make black-box leaps of faith, and examine how the issues we understand in the small affect larger modeling problems.

---

6In these notes, comments on Matlab are typeset like this paragraph.

7Really, a matrix of coefficients. We’ll see matrices a bit later.