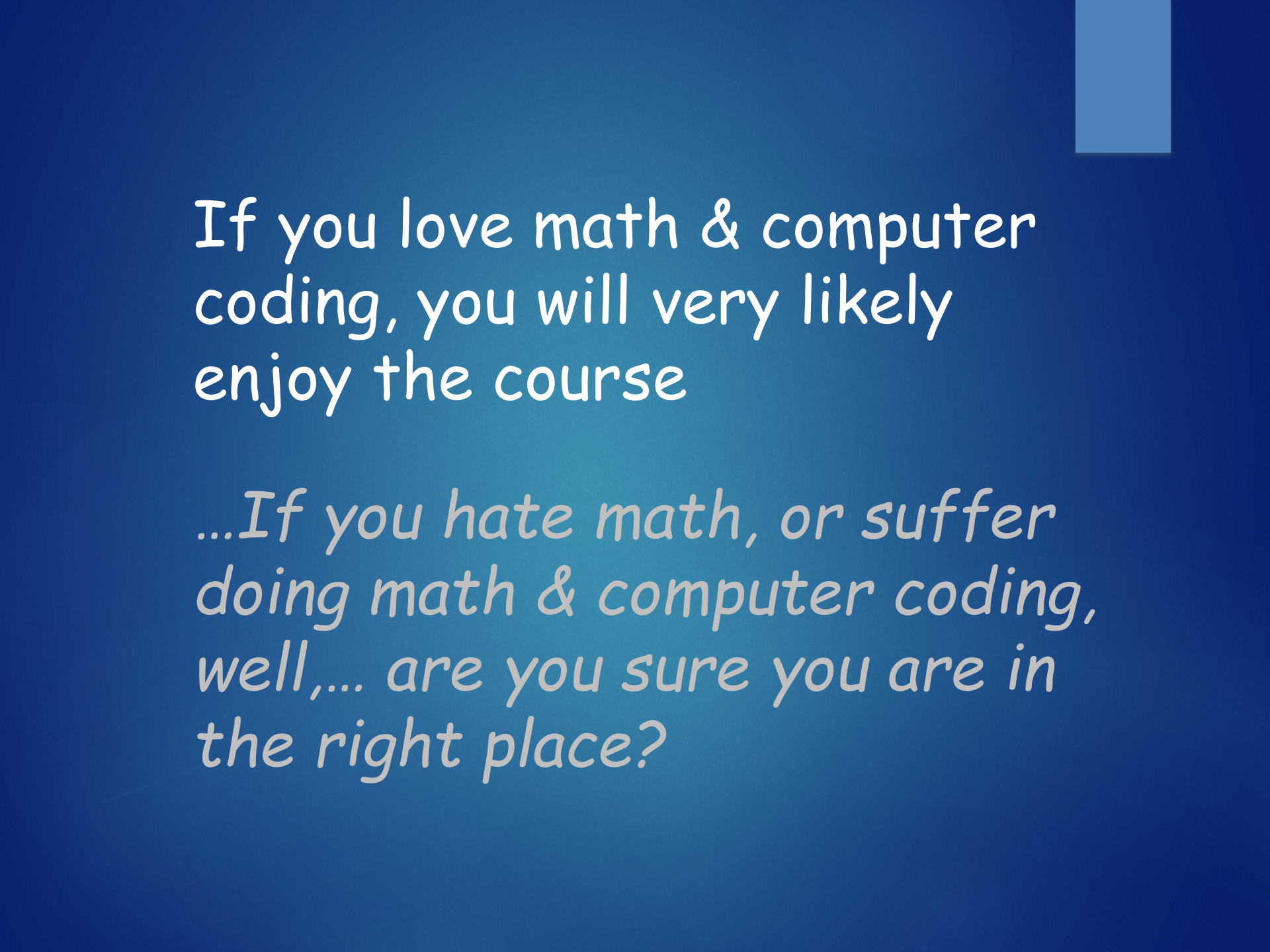


ECE3340

Numerical Methods for Electrical and
Computer Engineering

PROF. HAN Q. LE



If you love math & computer coding, you will very likely enjoy the course

...If you hate math, or suffer doing math & computer coding, well,... are you sure you are in the right place?

$\left\{ \begin{array}{l} \text{first 10-digit prime found} \\ \text{in consecutive digits of } e \end{array} \right\} .\text{com}$

CLEAR CHANNEL

An example of "numerical math"
(not pure math)

An example of **pure** (number theory) math

Prove that: $2+2=4$

Or...

Prove that there are more irrational numbers between (0 1) than all integers or rational numbers

Or...

No three integers a, b, c can satisfy the equation below for integer $n > 2$

$$a^n + b^n = c^n$$

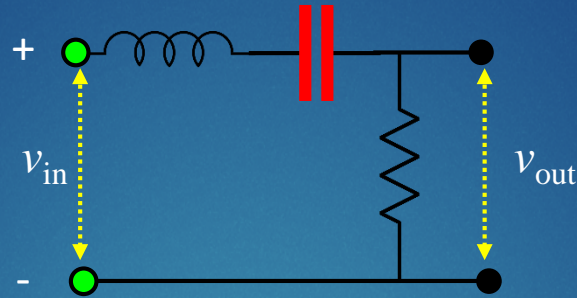


GÖDEL, ESCHER, BACH:
An Eternal Golden Braid
DOUGLAS R. HOFSTADTER

A metaphorical voyage on words and machines in the spirit of Lewis Carroll

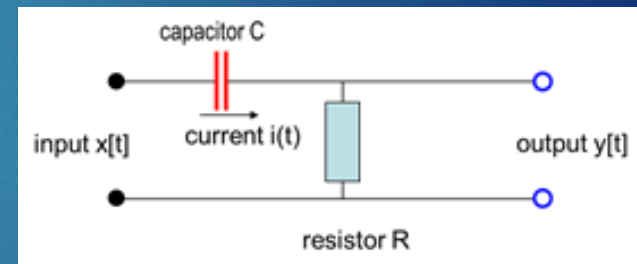
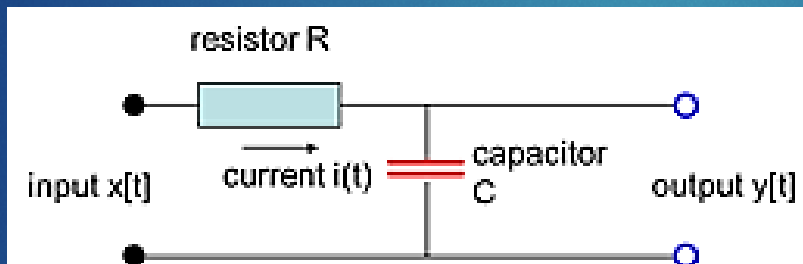
- This course is about applied math for scientific, electrical engineering applications, and also general quantitative science.
- Physical quantities, e. g. mass, velocity, voltage, current, frequency, temperature, pressure, flux, power, energy, and whatnot... are represented by real numbers, and related to each other by physical laws.
- Why numerical methods?
 - Data analysis of experiments and measurements.
 - Practical scientific/engineering problems aren't solved with abstract symbols ($v, i, \omega, P...$), but with **numerical values** for applications. We need **numerical methods** to obtain accurate & precise results.
 - Beyond practical applications, computation – especially **simulation** can give valuable insights and understanding of a problem. **Numerical methods** are needed for efficiency and accuracy.

Example



Q.1 What is a most common name for this circuit?

Examples without inductor L

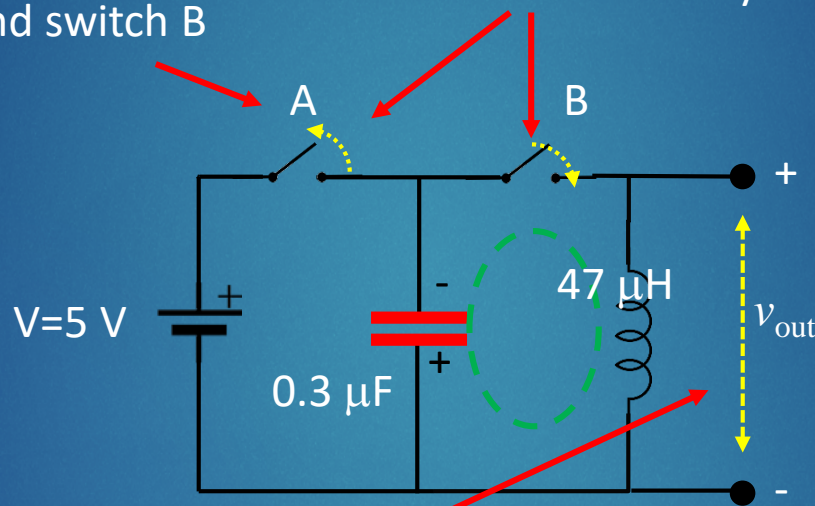


Q.2 What is a most common name for these two circuits? (extra credits if you name each one differently based on its function and not by a permutation of letters)

Consider this circuit that has no resistors, no dissipative elements (things that absorb power or energy permanently and not give back to the circuit). Assume perfect capacitor and perfect inductor.

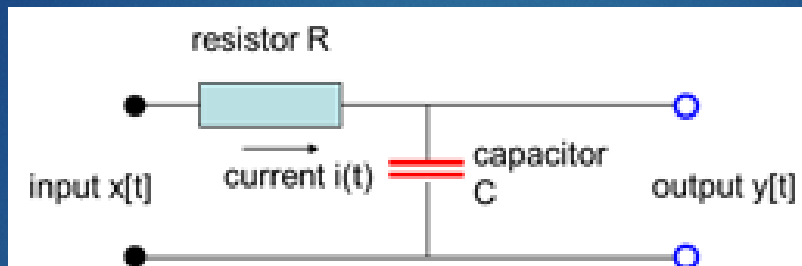
Switch A was connected for a long time $t < 0$ and switch B was open.

Then switch A is opened at time $t_A \leq 0$; simultaneously or after t_A , switch B is instantaneously closed at $t = 0$.



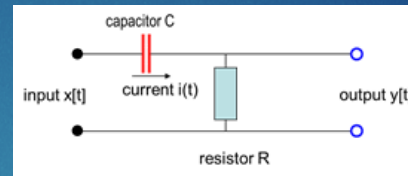
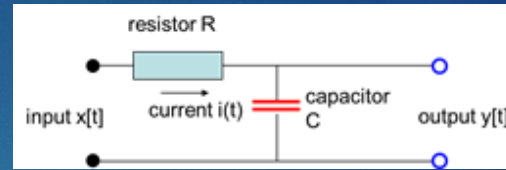
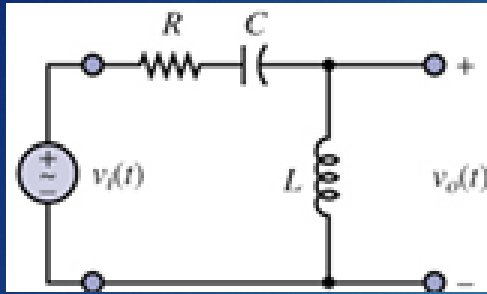
Q.3 What is the voltage v_{out} before and after the switches are activated? Sketch your guess what it looks like (only "guessing" is asked, no solving equation or anything complicated).

A similar question but for a different circuit:



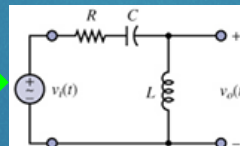
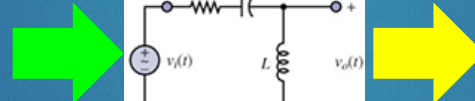
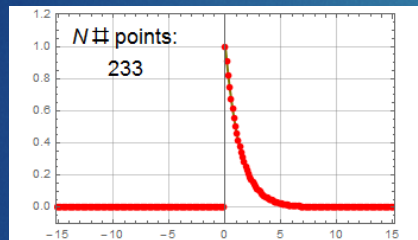
Q.4 For all time $t < 0$, input $x[t] = 1$ Volt. Suddenly at $t = 0$, the input is switched to 0 V (with zero impedance at the input). Sketch your best guess what voltage $y[t]$ looks like as a function of time for $t \geq 0$. Extra credit if you write the correct mathematical expression for $y[t]$.

Numerical calculation can give us practical results (with values for actual application) as well as insight for designing, modification, and invention!

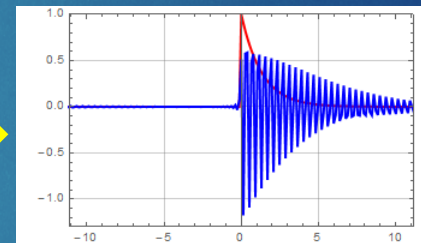


When design a circuit, we want to know how it works, what it does for the intended application. We want numerical simulation results such as this:

input



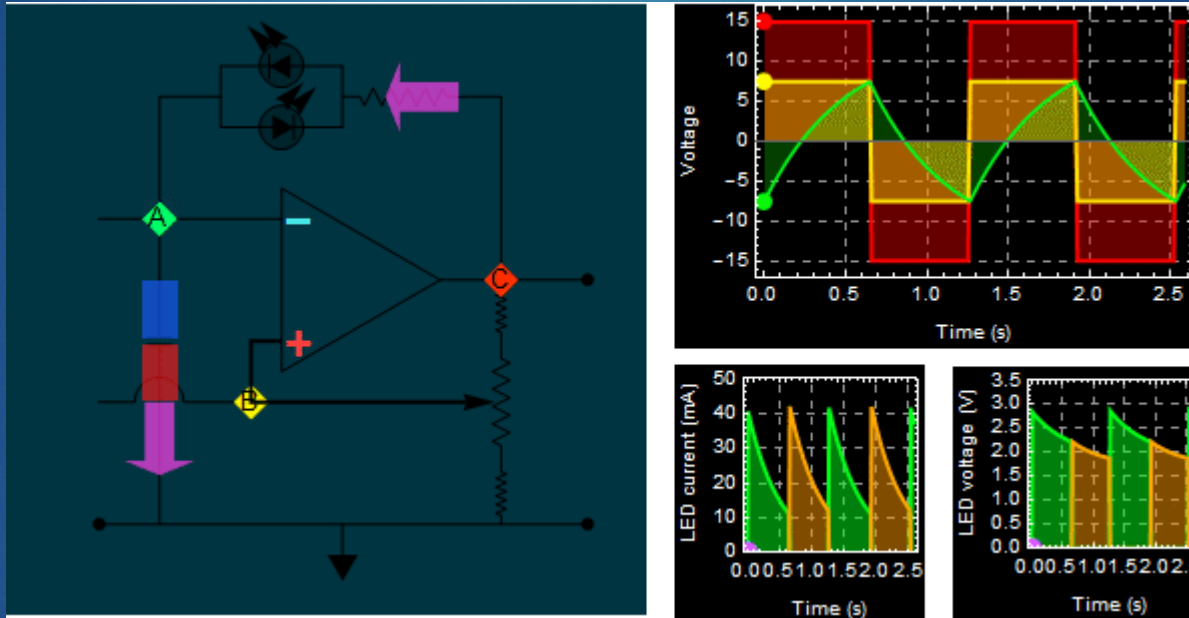
output



This is what "numerical methods" is about. The objective: learn how to use computers to apply to ECE problems.

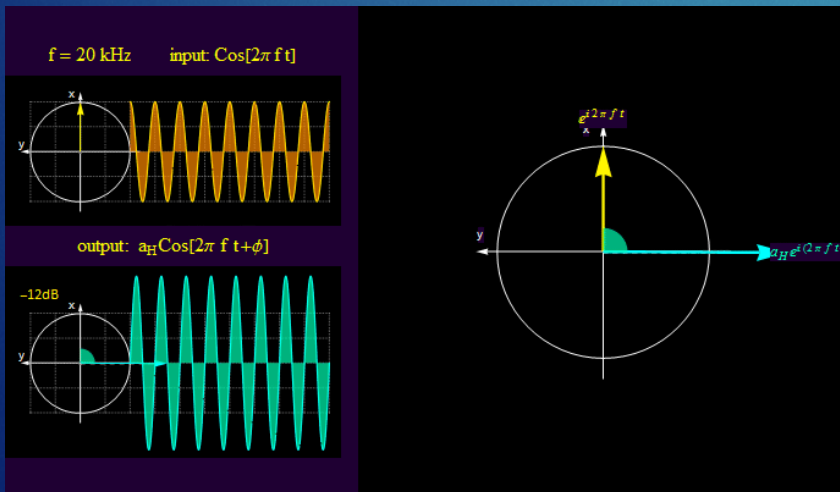
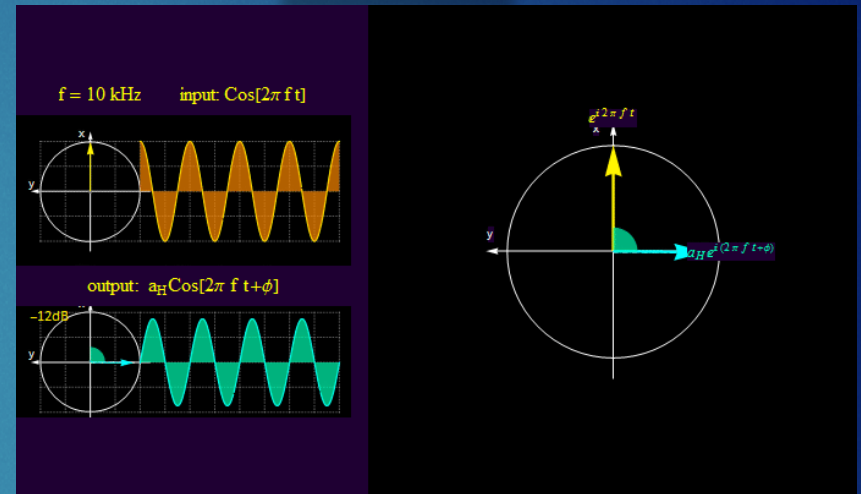
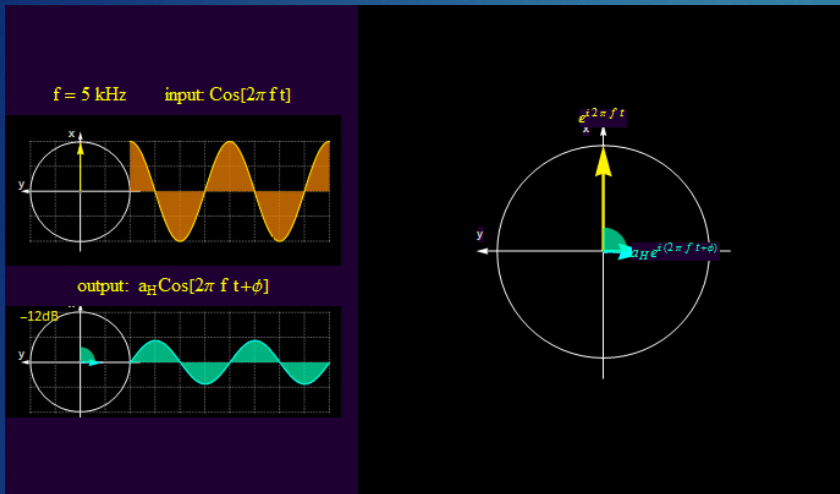
For those who have taken ECE2100 there are many examples of circuit simulation using numerical methods [on the web site](#)

Example: an op amp oscillator



For those who have taken ECE2100 there are many examples of circuit simulation using numerical methods [on the web site](#)

Example: harmonic responses of analog PID controller

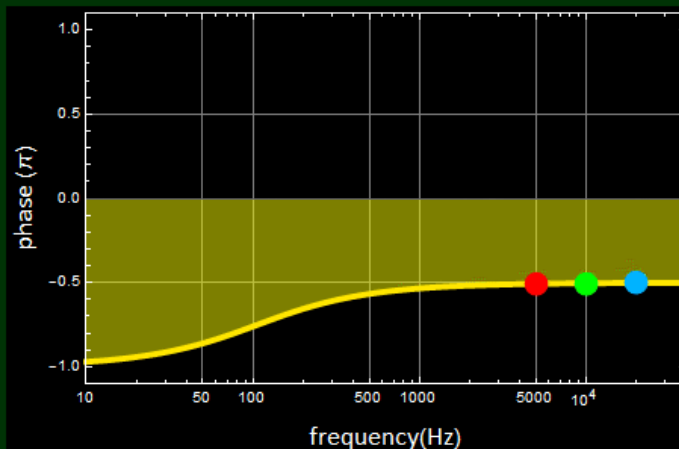
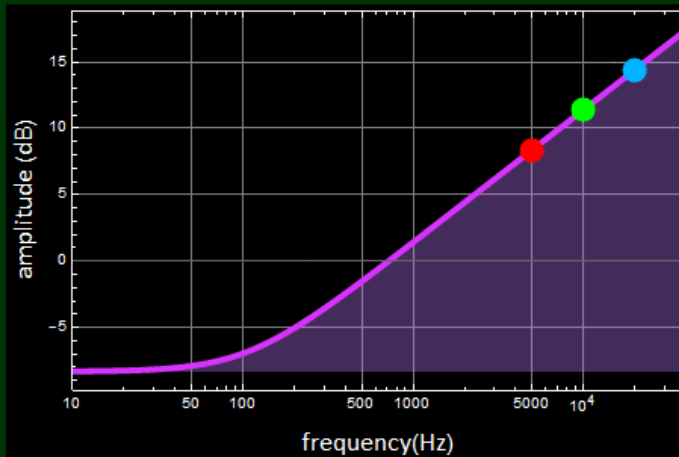


For those who have taken ECE2100 there are many examples of circuit simulation using numerical methods [on the web site](#)

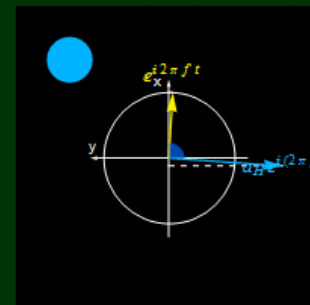
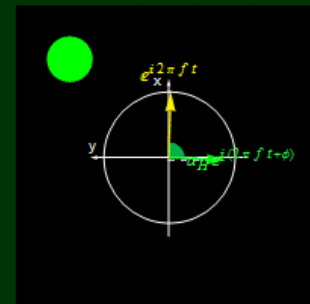
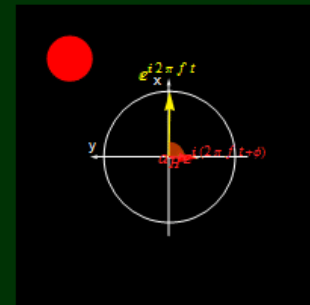
Example: Bode plots and harmonic responses



*Bode plots: amplitude (top), phase (bottom),
With Bode plots, any phasor at any frequency is determined*

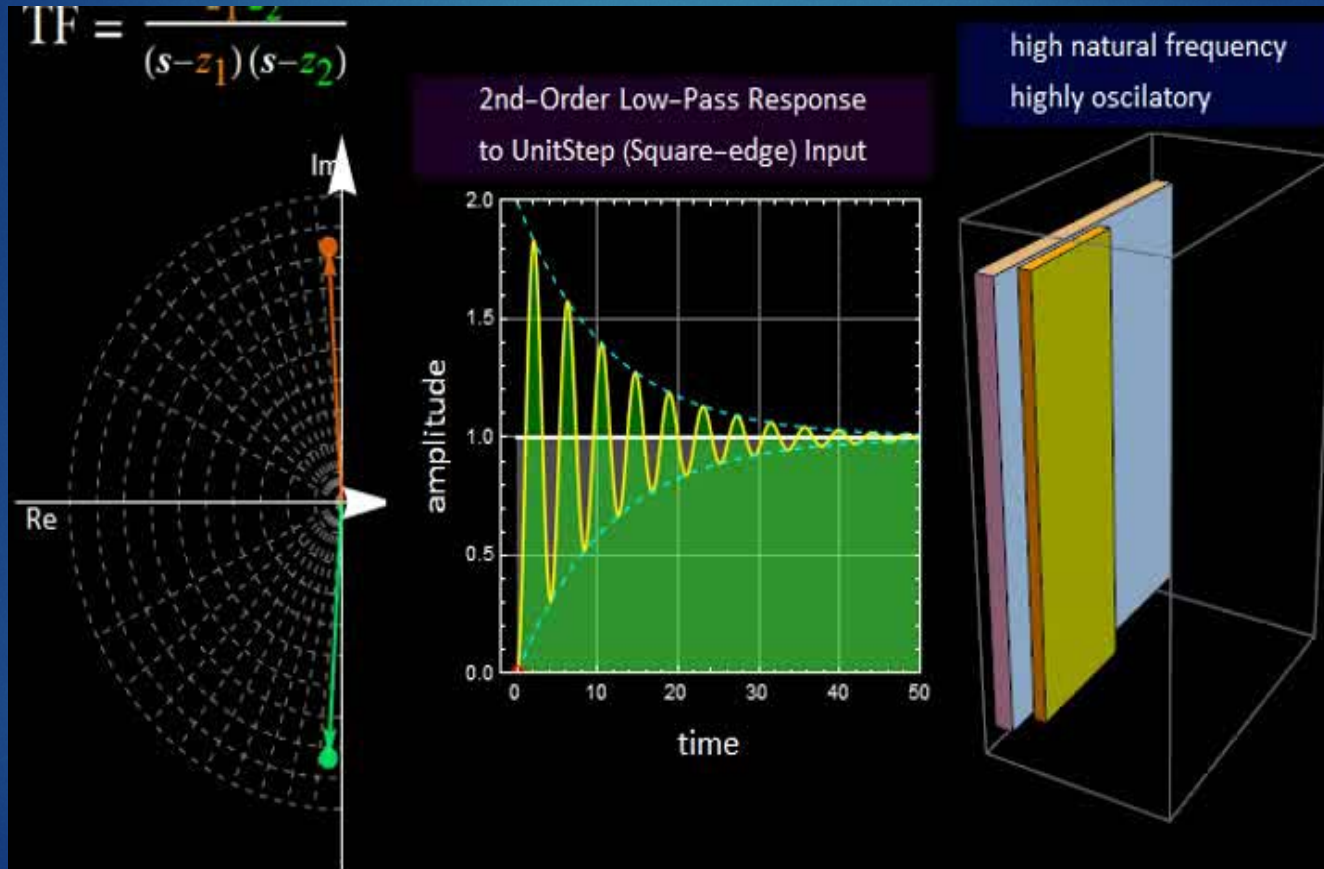


*phasors corresponding
to points on Bode plots*



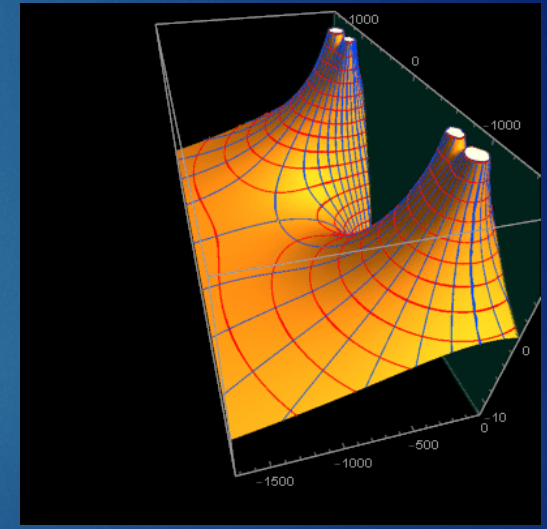
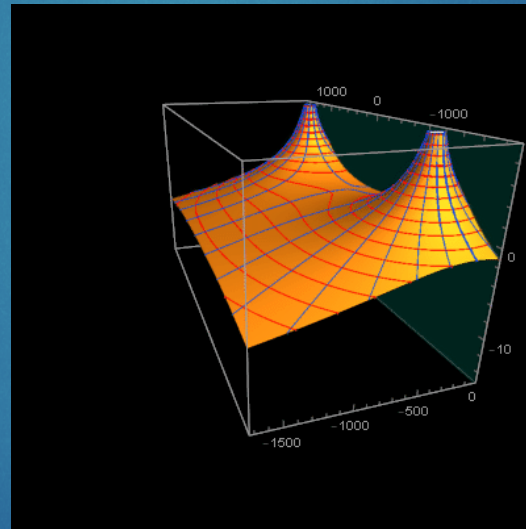
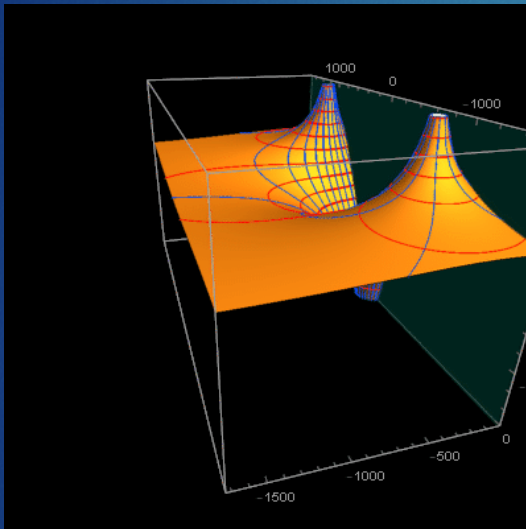
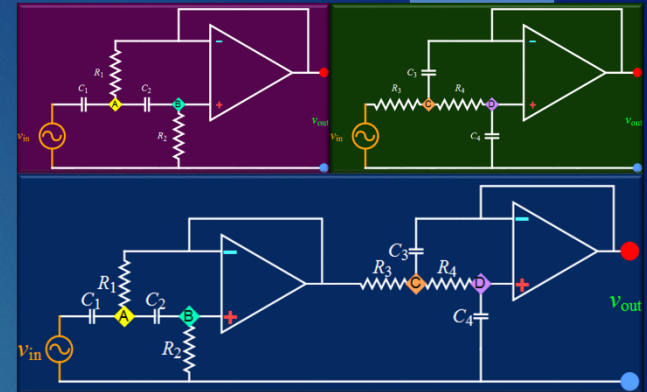
For those who have taken ECE2100 there are many examples of circuit simulation using numerical methods [on the web site](#)


Example: transient response of a second order circuit



For those who have taken ECE2100 there are many examples of circuit simulation using numerical methods [on the web site](#)

Example: circuit response functions





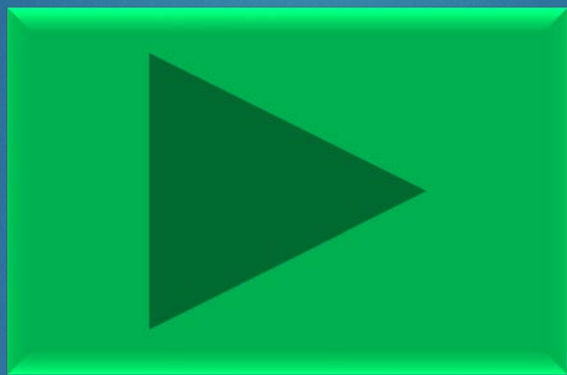
All the above calculations
and simulations are fairly
simple and straightforward
(thanks to powerful CPU and
sophisticated software).
Yes, you can do similar
things after this course.
(hard working is required)



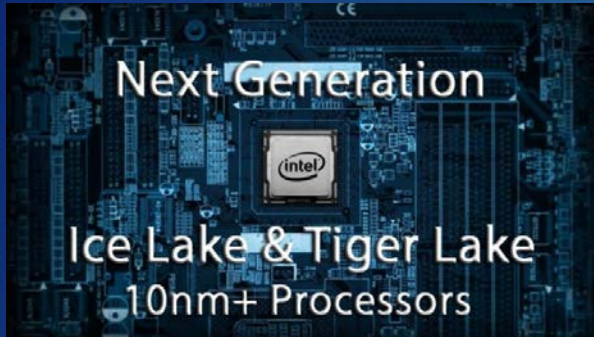
Although not scientific or engineering, this is an example of a **numerical problem involving computer and programming (coding)**.

Without a computer, it may take days, weeks for someone to do by hand, which serves no useful purpose other than knowing the company to send your resume in (if looking for a job)... ("googling" it!)

A link to history of computer



Today



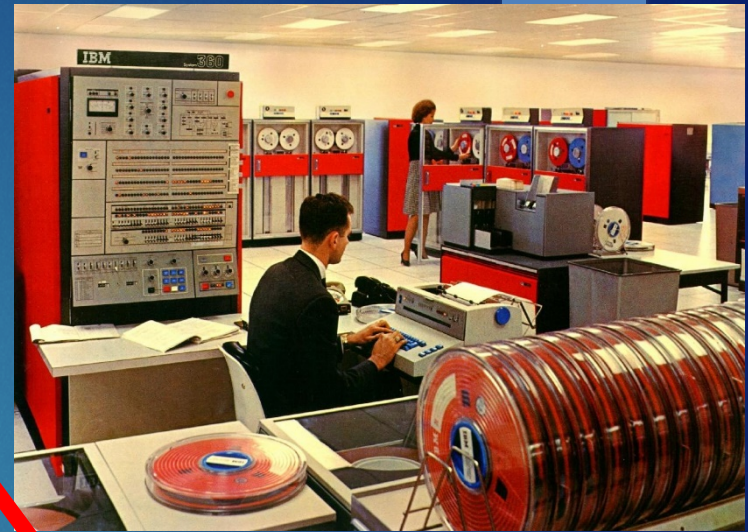
Enhanced IP Suite for 2015 Mid-range Market

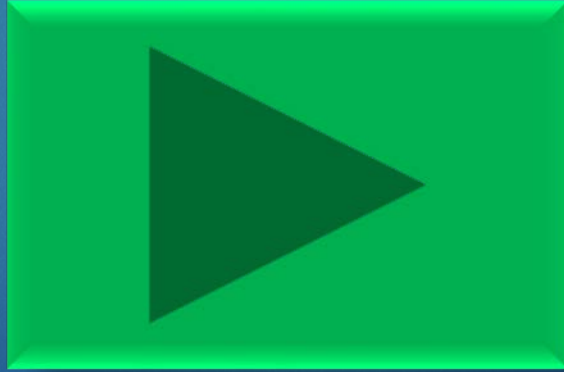
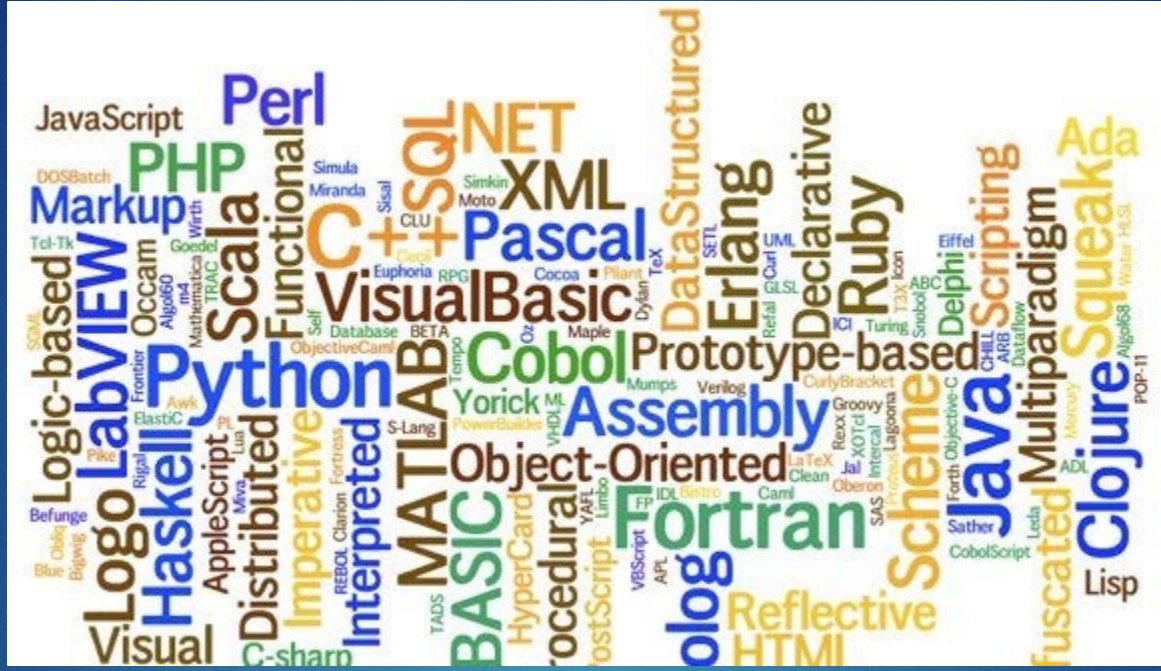
2015 System Solution for Mid-range Market

ARM® Cortex®-A17 CPU	ARMCORTEX Processor Technology	105M Mid-range tablets in 2015
ARM Mali™-T720 GPU ARM Mali™-V500 Video ARM Mali™-DP500 Display	ARMMALI Your Computing	344M Mid-range smartphones in 2015
ARM POP™ IP for CPU and GPU at 28nm	ARMARTISAN Physical IP	141M smart TVs shipping in 2015

Solutions for Every Screen

Yesteryear





Mother Tongues

Tracing the roots of computer languages through the ages

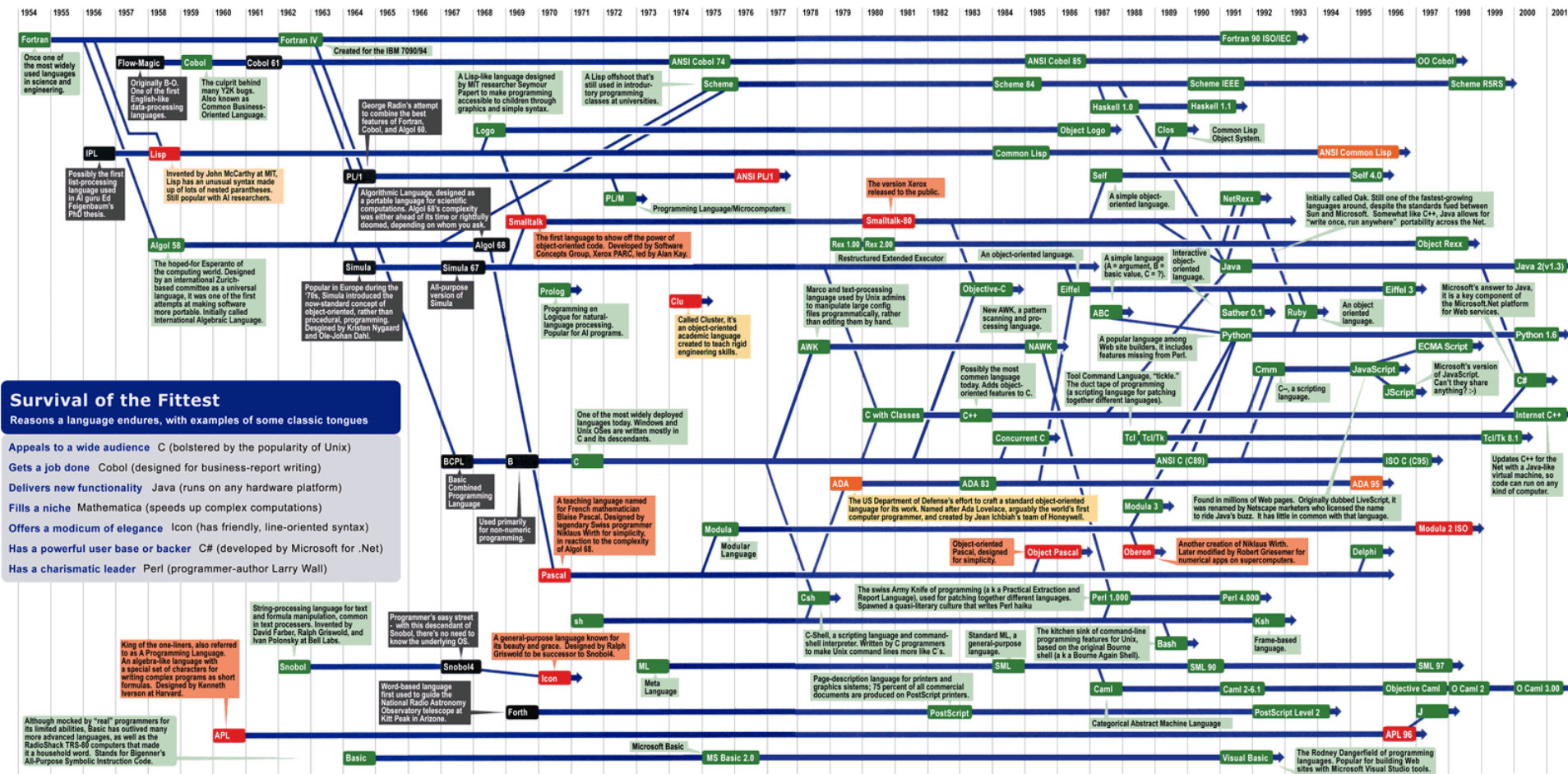
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java and other modern source codes dominate our systems, hundreds of older languages are running out of life.

An ad hoc collection of engineers-electronic lexicographers, if you will-aim to save, or at least document the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in these nearly forgotten lingua frangas. Among the most endangered are Ada, APL, B (the predecessor of C), Lsp, Oberon, Smalltalk, and Simula.

Code-raker Grady Booch, Rational Software's chief scientist, is working with the Computer History Museum in Silicon Valley to record and, in some cases, maintain languages by writing new compilers so our ever-changing hardware can grok the code. Why bother? "They tell us about the state of software practice, the minds of their inventors, and the technical, social, and economic forces that shaped history at the time," Booch explains. "They'll provide the raw material for software archaeologists, historians, and developers to learn what worked, what was brilliant, and what was an utter failure." Here's a peek at the strongest branches of programming's family tree. For a nearly exhaustive rundown, check out the Language List at [HTTP://www.informatik.uni-freiburg.de/Java/misc/lang_list.html](http://www.informatik.uni-freiburg.de/Java/misc/lang_list.html). - Michael Mendeno

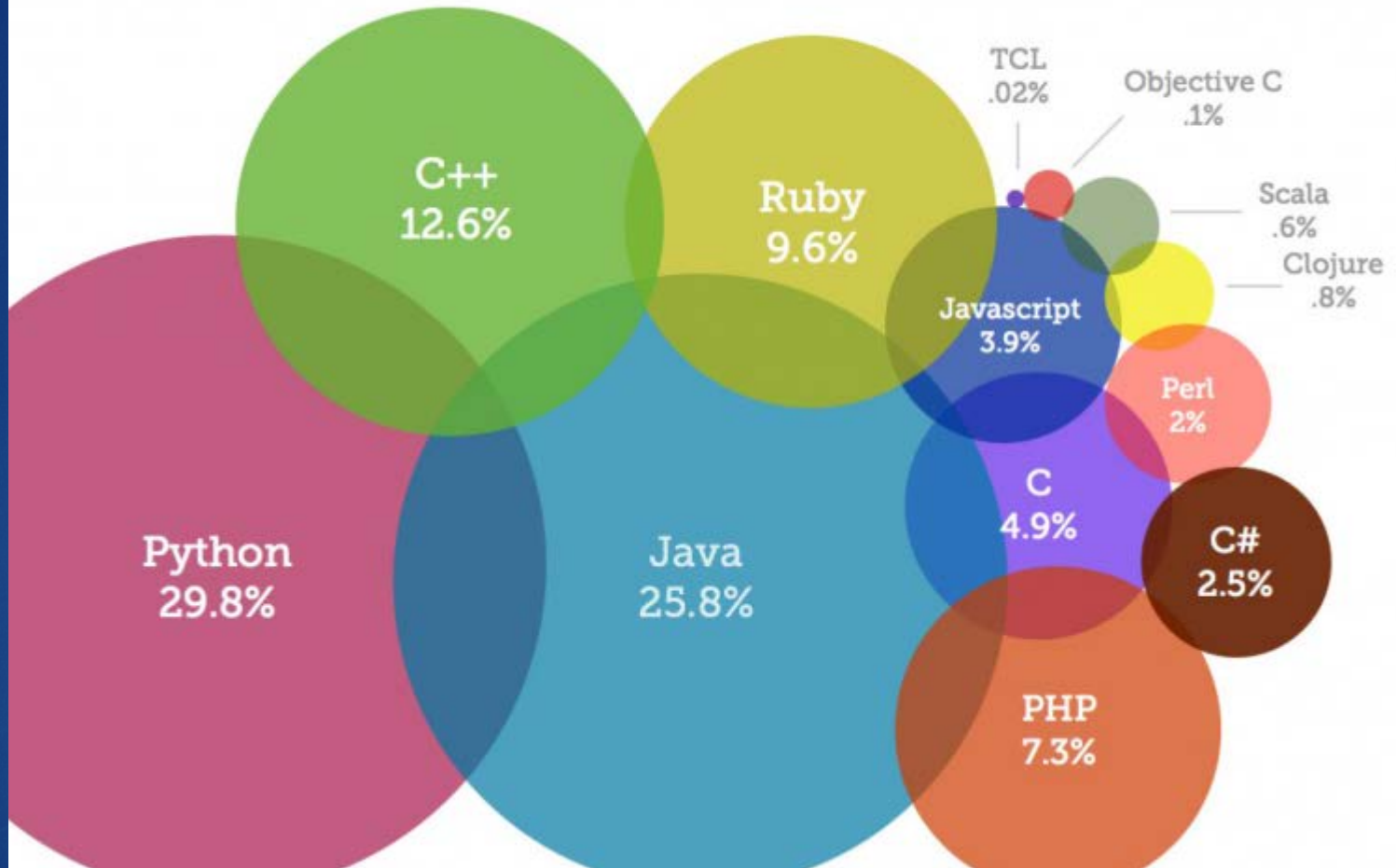
Key

- 1954 Year Introduced
- Active: thousands of users
- Protected: taught at universities; compilers available
- Endangered: usage dropping off
- Extinct: no known active users or up-to-date compilers
- Lineage continues

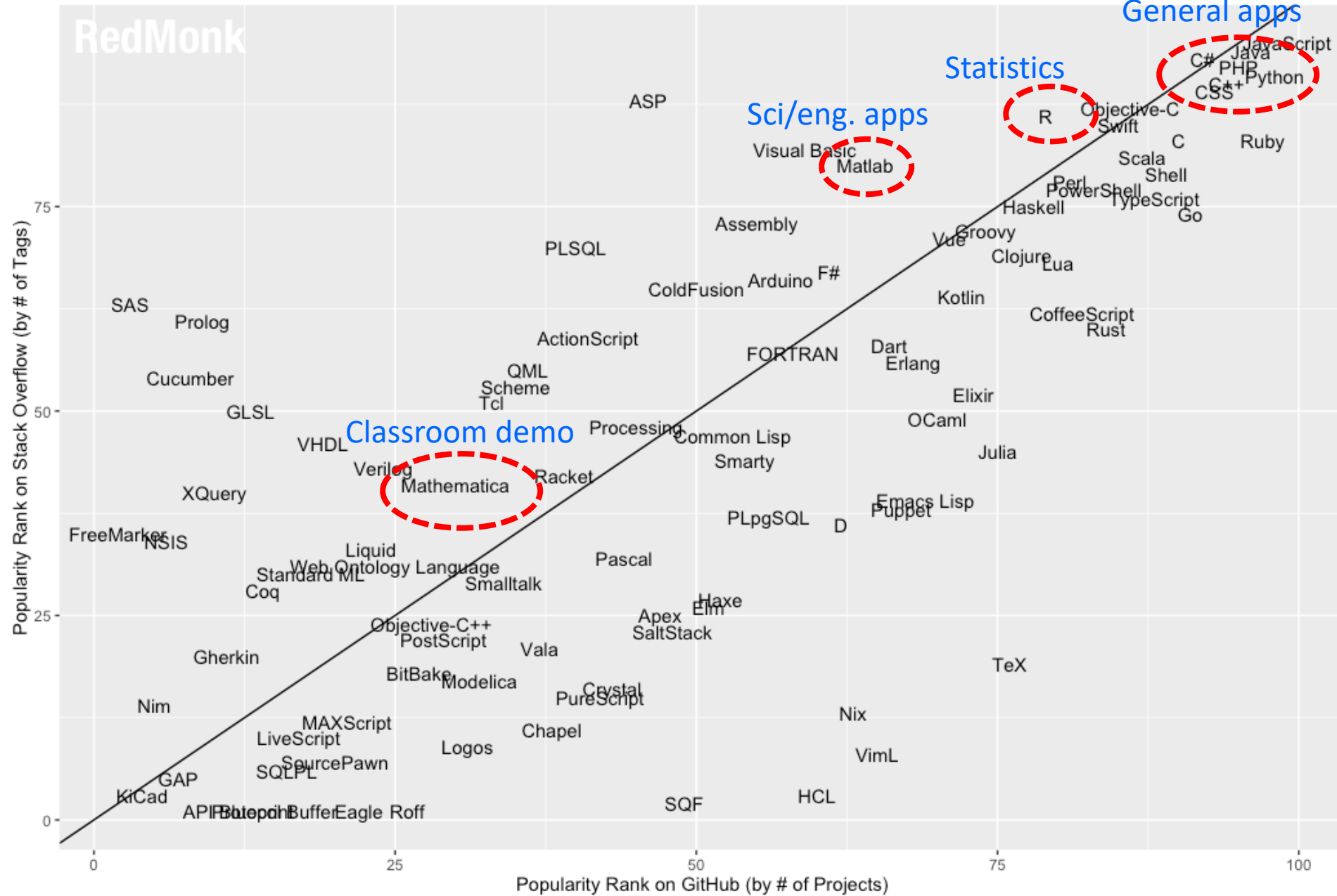


Sources: Paul Boutin; Brent Halpern, associate director of computer science at IBM Research; The Retrocomputing Museum; Todd Proebsting, senior researcher at Microsoft; Gio Wiederhold, computer scientist, Stanford University

Most Popular Coding Languages of 2018



RedMonk Q318 Programming Language Rankings

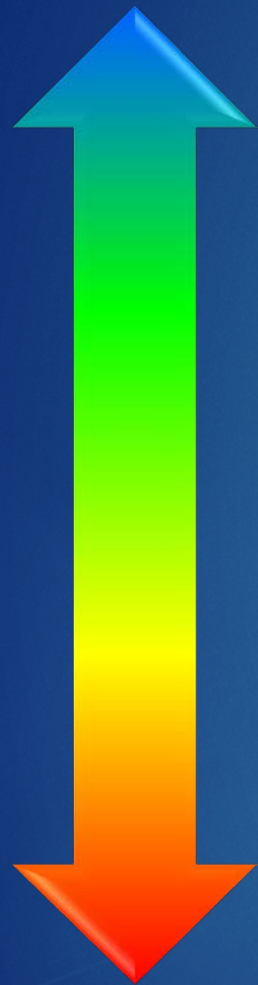




Numerical methods for scientific and electrical engineering applications

- This course is about concepts, methods, and algorithms. It is (and should be) hardware-and-software-agnostic.
- Once you master the fundamentals, you can apply what you learn on any platform & IDE of **your choice**. (e. g. Python,...)
- That said, for convenience, there must be a single common platform for course materials: Wolfram Mathematica (read the syllabus).

Higher level - specialized for scientific/engineering app.



Lower level implementation – general app. (web, games)

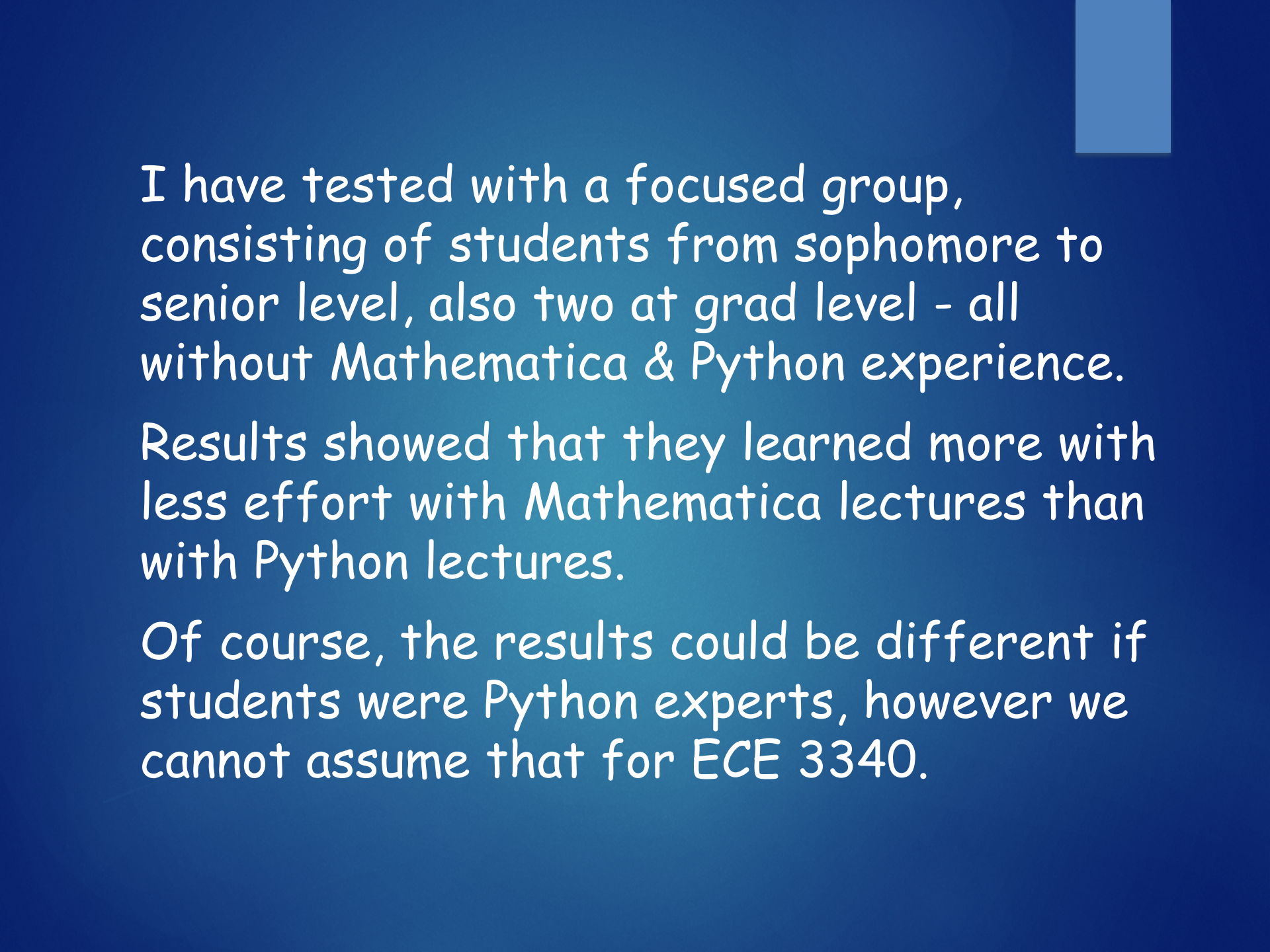
U of Houston licensed software



- This Course lectures & HW
- Easy for quick & dirty demo.
 - Convenient symbolic manipulation.

Free IDE stuffs (many...)



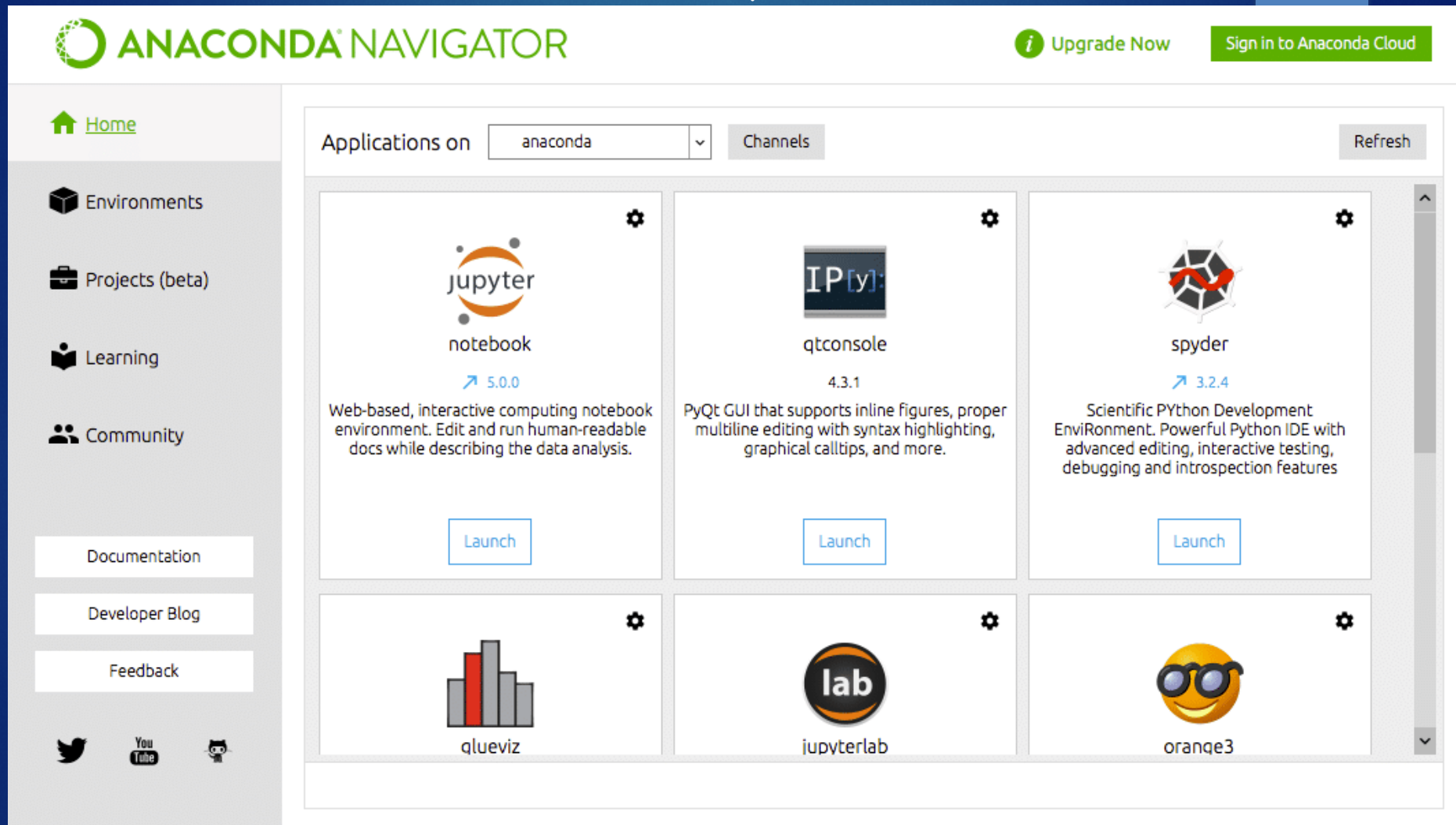


I have tested with a focused group, consisting of students from sophomore to senior level, also two at grad level - all without Mathematica & Python experience.

Results showed that they learned more with less effort with Mathematica lectures than with Python lectures.

Of course, the results could be different if students were Python experts, however we cannot assume that for ECE 3340.

For those who use Anaconda, jupyter notebook is quite similar to Mathematica notebook (in concept)



Mathematica has a huge list of built-in functions

Course material formats for download

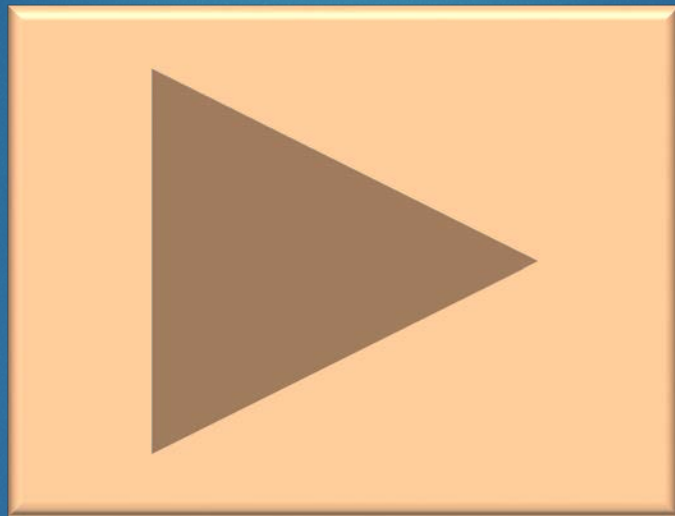
Main



Others



A tutorial in Mathematica



About of this course

- ▶ Review and learn some essential mathematics that have widespread applications in scientific/engineering problems.
- ▶ The three most important things of the course are application, application, and application. “*It's all about APPs!*”
- ▶ Computer usage is of course the essential vehicle of learning. *Bring your computer to do work in class as often as possible.*

What this course is **not**

- ▶ Learning computer programming. *That's for other courses, e. g. ECE 3331. Practice makes perfect: whether you learned it in 3331 or you may already be an APP developer since HS, you are strongly encouraged to hone your programming skill by practicing what you learned. (Will help you to review/refresh your memory as much as possible).*

If you love **math AND computer programming**, you will very likely enjoy the course.

- The more coding you do, the better you will be at it
- Reinforcing your knowledge of electrical engineering (in this case, we pick circuit to be the example topic for numerical math)

Math knowledge at MATH 3321 level and proficiency in computer programming at ECE 3331 level or better are prerequisite.

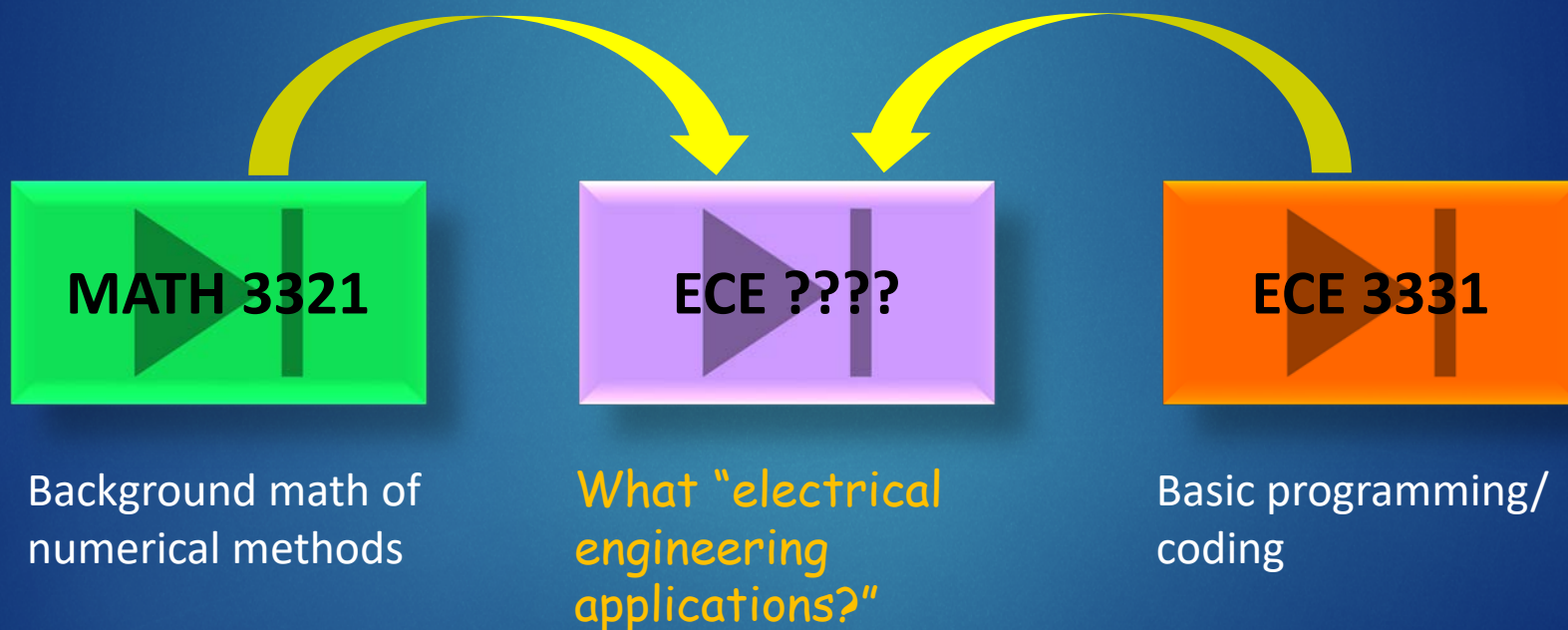
ECE 3340 - Numerical Methods for Electrical and Computer Engineers

Credit Hours: 3.0

Lecture Contact Hours: 3 Lab Contact Hours: 0

Prerequisite: ECE 3331 and MATH 3321.

Basic linear algebra and numerical methods with electrical engineering applications. Emphasis on use of computer-based solution techniques.



We will not learn "numerical methods" in theory and abstract, but by specific examples and applications in electrical engineering.

Expected work in this course

Weight

- ▶ **Active lecture and class participation:** you will participate to contribute to lecture and discussion in class

- ▶ **Classwork is more important than HW: Bring paper, pens and your computer to do work in class.** Turn in at the end of the class for credit.

40-60%

▶ **Most important: take ownership of your own education: contribute to your own "lecture."**

- ▶ **Homework:** First, try to do on your own. Then get help and collaborate if needed (peer learning). Submit on time and follow policy on late HW.

20-30%

▶ **A good combination of personal best effort and/or peer interactive learning can help you a great deal.**

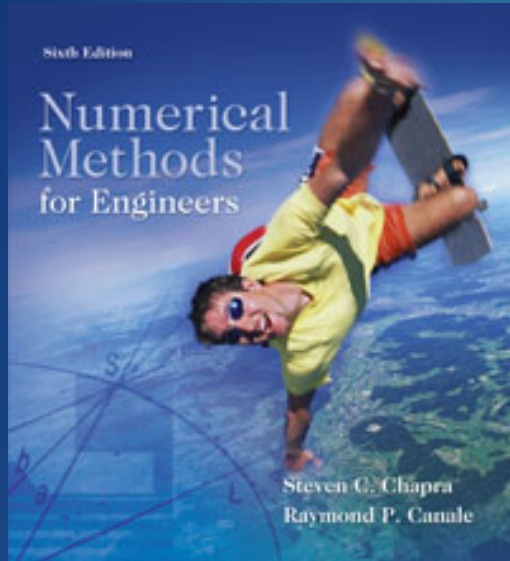
- ▶ **Tests/exams:** to be determined. No need for them if the other two categories are "good enough."

25-35%

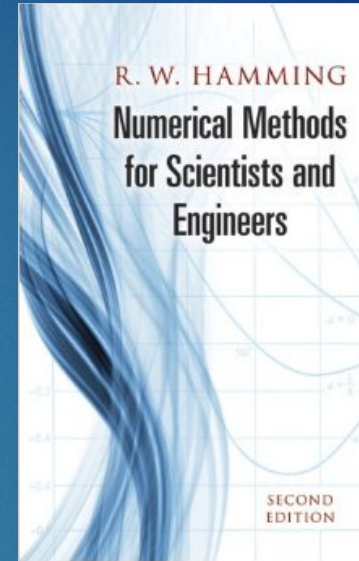
- ▶ **Final personalized project:** to be determined.

10-15%

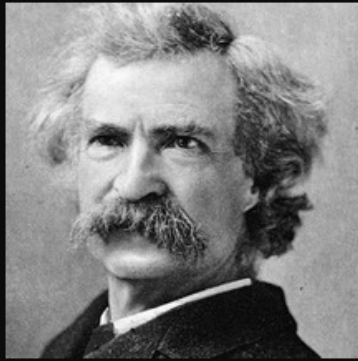
Two **reference** textbooks



Very easy to read
and practical (used
in ECE 2331)



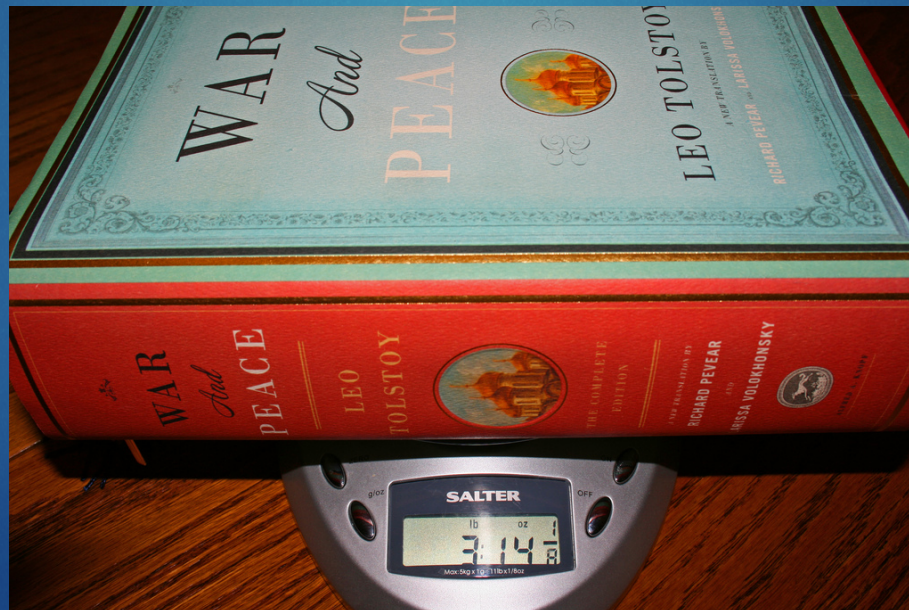
This book is a classic!
And like many other
classics...

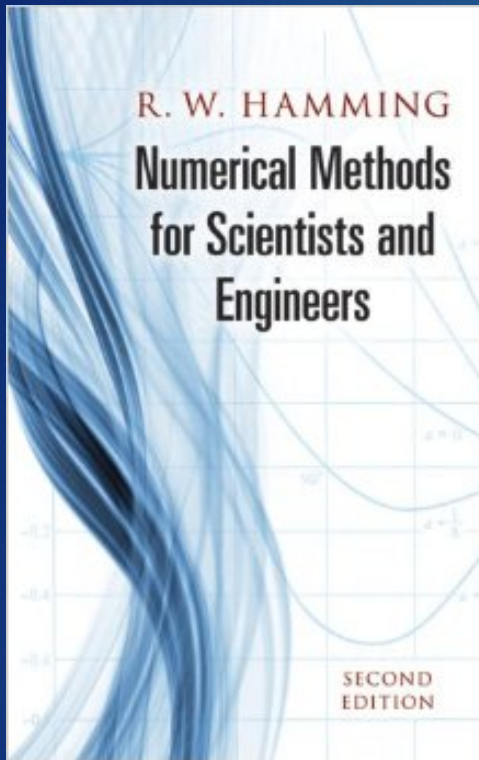


A classic is something that everybody wants to have read and nobody wants to read.

~ Mark Twain

AZ QUOTES





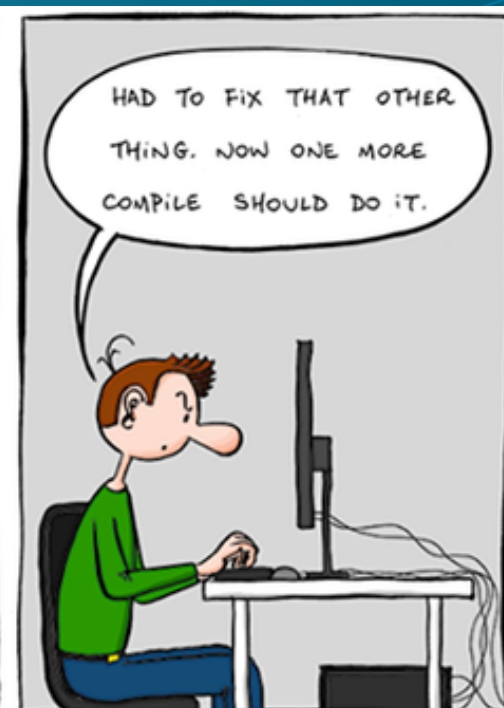
- A beautiful work – a lifetime of wisdom from the author, R. W. Hamming.
- It's hard to absorb the wisdom of a lifetime in a short semester. (you wouldn't need taking this course if you read and understand the book)
- We will use it as a reference textbook:
 - Much of things in the book are already used and implemented in commercially available or free software.
 - Some are still relevant, some are less crucial because of increasing computing power and more sophisticated (intelligent) software.

In practice

- ▶ Rarely, if ever - do practicing scientists/engineers have to write own codes for utility apps such as find roots, integration, differential eqs., FDTD, FEM, special functions, linear/nonlinear systems,... and many others.
- ▶ There are huge libraries for those things in commercial or free software, developed by dedicated professionals using well established or latest algorithm development, debugged, optimized for CPU speed, memory... Why re-invent the wheel?
- ▶ Actually, in a professional world, **we would insist any user to use established commercial software - don't write your own except for own research**: higher risk of bugs and errors. *If you need a car, would you rather buy one or drive a contraption built by your colleagues?*
- ▶ This course helps you become an intelligent, informed user of scientific/engineering software to be a professional electrical engineer - but not to develop you into a professional software developers: this is the job for computer science major at PhD level.

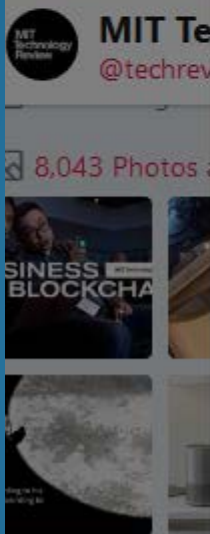
A note about Computer Programming

*or how not to be an
obsolete human model
in the HAL 9000 world*





Home Moments Search Twitter Have an account



MIT Technology Review 
@techreview

[Follow](#) 

When robots are your colleagues, which human skills will still matter? Today we will be discussing the future of work and business with some of the world's leading experts. 🤖

🌟🤖 Join the conversation by following [#emtechnext](#).

ECE 3331, Programming Applications in Electr & Comptr Engr

	Topics
1	Introduction to Computers and Programming
2	Basic Elements of the C Programming Language
3	Input and Output
4	Variables, operators, control flow
5	Functions and program structure
6	One-Dimensional Arrays
7	Multidimensional Arrays
8	Storage classes and type qualifiers
9	More input/output
10	Structures, unions
11	Enumerated types
12	Tables and syntax

What expected is the acquiring of concepts and skills of computer programming in general, which are not necessarily limited to specific knowledge of C, C++, or C#,...

Programming concepts such as flowchart, object-oriented structure, procedural process are not language-dependent.

5. Linear Algebra

- 5.1 Introduction
- 5.2 Systems of Linear Equations; Some Geometry
- 5.3 Solving Systems of Linear Equations, Part I
- 5.4 Solving Systems of Linear Equations, Part II

Exam 2

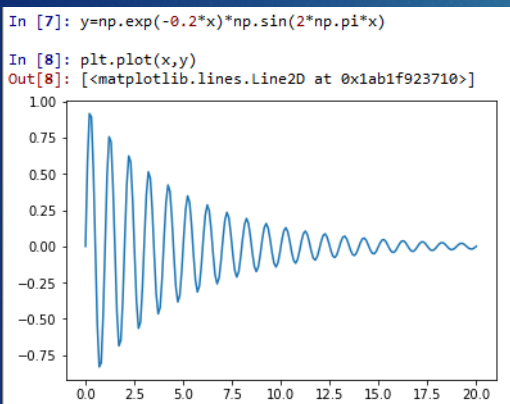
- 5.5 Matrices and Vectors
- 5.6 Square Matrices; Inverse of a Matrix, Determinants
- 5.7 Vectors; Linear Dependence and Linear Independence
- 5.8 Eigenvalues and Eigenvectors

Pre-req for 3340, ideally

	#units	price\$/unit	store A	store B	store C	store D
juice	3	1.55	1.45	1.65	1.4	
eggs	4	1.95	2.4	2	2.2	
fruits	12	0.85	0.8	0.7	0.8	
vegetable	8	1.35	1	1	1.1	
milk	2	2.55	2.25	2.55	3.1	
cereals	6	2.7	3.35	3.05	3.45	
coffee	1	10.85	7.5	8.45	8.5	
tea	2	4.2	4.35	3.75	3.95	
ice cream	3	7.35	6.75	6.75	4.95	
napkins	5	1.1	1.2	1.2	1.35	
foils	2	3.5	3.75	3.6	3.75	
storage bi	10	0.8	0.7	0.85	0.7	
toothpast	4	1.6	1.6	1.55	1.6	
shampoo	3	3.55	2.6	2.9	2.7	
detergent	3	9	8.55	8.55	7.1	

`value = pricedataT . quantity;`

(array data, linear algebra)



“hello world”

$A = \pi r^2$
 $A = \text{len} * \text{wid}$

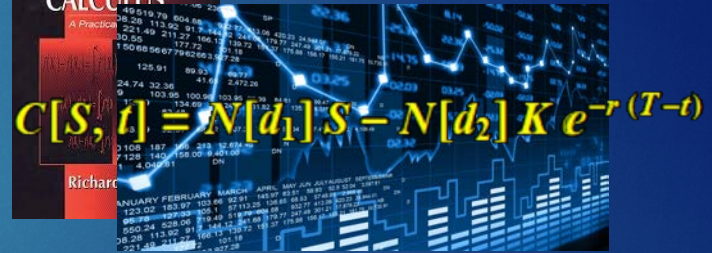
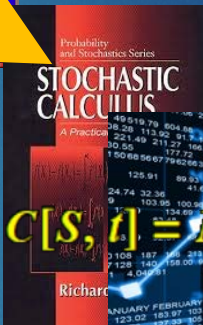
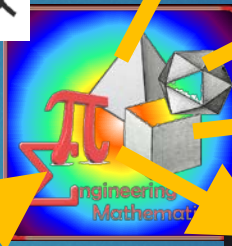
```
In [15]: def myfunc(name):
...:     if name == 'drew':
...:         print('oh, drew, I love you')
...:     else:
...:         if name == 'justin':
...:             print('justin, go away!')
...:         else:
...:             print('who are you?')

In [16]: nm='drew';

In [17]: myfunc(nm)
oh, drew, I love you
```

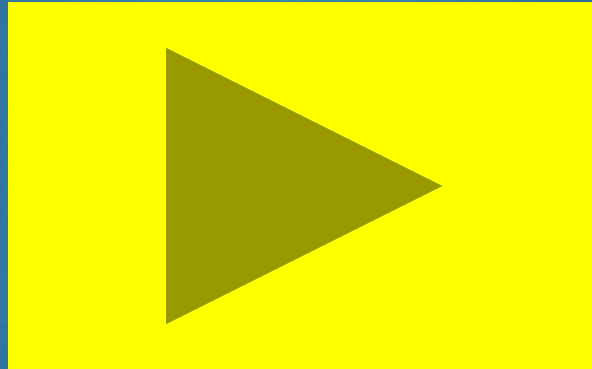
```
In [21]: class mysignalprocessing:
...:     {def etc...
...:         def etc...
...:         def etc...
...:     }
```

symbolic manip,
anonym func



$$C[S, t] = N[d_1] S - N[d_2] K e^{-r(T-t)}$$

Learning a language



An example of how one can quickly adapt...

List

Python	Mathematica
[a,b,c]	{a,b,c}
[5, 'string',1.4]	{anything, all objects, multi-media...}
len()	Length[]
["list" , ["nested list item",a,b] , ...]	Same. Use Depth[] to see nest level.
indexing: my_list[n]	mylist[[n]]
negative index	same
list.append()	AppendTo[]... Prepend, PrependTo
list.remove()	Drop[,n] ... (* see details *)
del my_list[n]	see Drop[] above
insert()	Insert[]
clear()	<i>not</i> Clear[]
my_list=['p','y','t','h','o','n']; my_list[0:2]= ['m','a','r','a']	Join[{"m","a","r","a"} ,Drop[mylist,2]]
extend()	Join[]
sorted()	Sort[]
reverse	Reverse[]
list=[function(x) for x in range(n)]	mylist=Table[...,{x,n}]
range(n)	Range[n] , but starting from 1 not zero and end with n, not n-1.
x=['a',2,'b',4.2] for u in x: if isinstance(u,numbers.Number): print(u**2) else: print("not a number")	Do[If[NumberQ[u], Print[u^2], Print[u, " is not a number"]] , {u, x}]
min(), max()	Min[], Max[], MinMax[]
sum()	Total[]
all(), any()	AllTrue[], AnyTrue[]

An example of how one can quickly adapt...

<code>all(), any()</code>	<code>AllTrue[], AnyTrue[]</code>
<code>u=range(5); v=range(3); [f(x,y) for x in u for y in v] [[f(x,y) for x in u] for y in v]</code>	<code>Table[x y, {x,0,4}, {y,0,2}]</code> output is 5 x 3 matrix. <code>Flatten[]</code> to get Python 1-D list.
<code>np.ndarray.flatten</code>	<code>Flatten[]</code>
<code>list.index(), enumerate()</code>	<code>Position[]</code>
<code>list1+list2</code>	<code>Join[list1, list2]</code>
<code>list(map(operator.add, list1, list2)) np.add(list1, list2)</code>	<code>List1+List2 MapThread[Plus, {list1, list2}]</code>
<code>list*n</code>	<code>Flatten[ConstantArray[list, n], 1]</code> or <code>Flatten[Join[Table[list, n], 1]</code>
<code>[x*const for x in a]</code>	<code>a*constant</code>
<code>x in list</code>	<code>MemberQ[]</code>
	<code>Select[]</code>
<code>myset={a,b,c}</code>	set in Mathematica is an ordered List
<code>union(), difference(), intersection</code>	<code>Union[], Complement[], Intersection[]</code>
<code>//is// isdisjoint()</code>	<code>//Q// DisjointQ, IntersectingQ</code>
<code>isinstance(),</code>	<code>...Q... StringQ, NumberQ, ListQ, FileExistsQ</code>

<code>np.array() np.ndarray()</code>	<code>Table[]</code>
<code>np.random.randint(n1,n2,size=(j,k)) np.random.rand()</code>	<code>RandomInteger[{n1,n2}, {j,k}] RandomReal[]</code>
<code>np.random.normal(mu,s,n)</code>	<code>RandomVariate[NormalDistribution[<i>mu</i>,<i>s</i>], <i>n</i>]</code> see Mathematica docum for full list of distributions
<code>np.dot(a,b)</code>	<code>a . b</code>
<code>np.matmul(m,a) m @ a</code>	<code>m . a a . m</code>
<code>my_array.shape()</code>	<code>Dimensions[]</code>
<code>np.full()</code>	<code>ConstantArray[]</code>
<code>my_array[i,k:m]</code>	<code>my_array[i,k;;m] [[1]] Take[my_array[[i]], {k,m}]</code>
<code>m.T np.matrix.transpose()</code>	<code>Transpose[]</code>
<code>np.linalg.inv()</code>	<code>Inverse[]</code>



Jupyter
notebook
example

Know what you don't know – prepare for changes.

Be consistent with formatting.

Be consistent with naming conventions.

Push into production only when you are confident.

Global variables]

Recover gracefully.

Assume default formats.

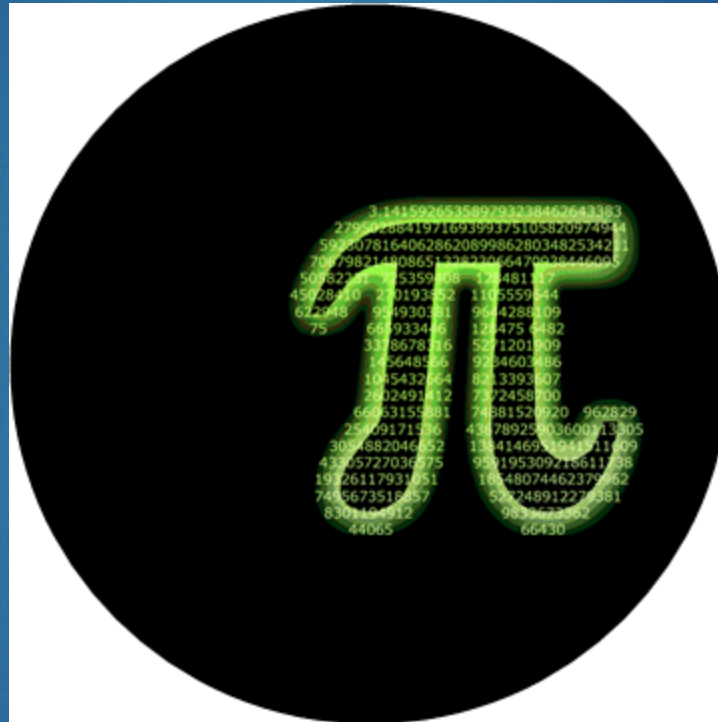
Practice, practice, practice

Provide useful error messages.

Add comment to your code – explain *what* and *why*

Some math fun example

Check your answer with this APP



If you haven't got a license for Mathematica, download free CDF player and you can run this

