Examples, Examples, Examples

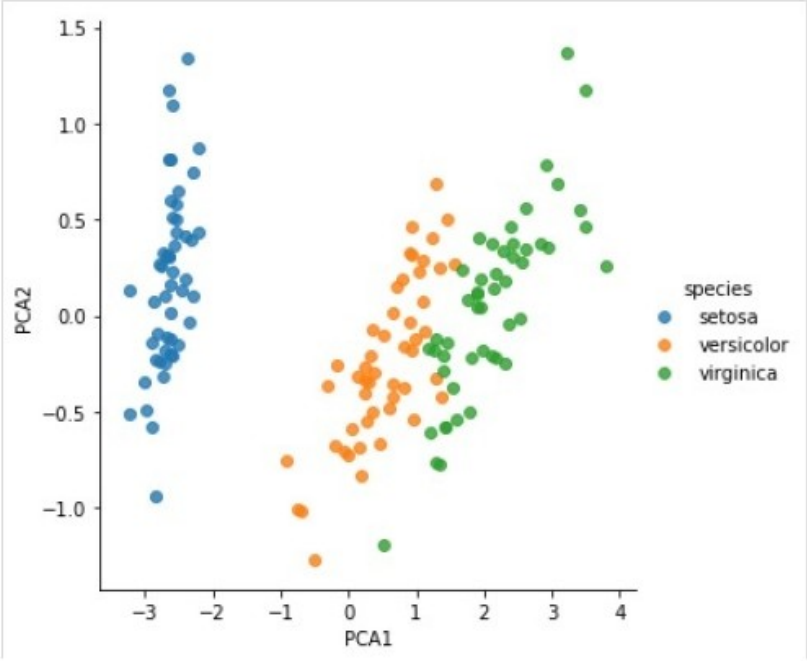
- In this set of slides
  - 6: Iris Classification
  - 7: Logic Puzzle
  - 8: Matched Differential Equation
  - Digging into the code to see the sheaves

Iris Classification

- This will be a quick pass-through to emphasize a couple of points.
- Mainly: Sometimes you can get sheaves to do a job, but that doesn't mean you'll do it well.
- If other tools are available and are better, use them!
- Just because you're using sheaves, you won't magically have a good model.

A classic classification problem

- https://www.tutorialspoint.com/scikit_learn/index.htm
- pip install scikit-learn *# use old name to install*
- from sklearn.datasets import load_iris



Data

- 150 Observation rows
  - Randomly split into training and testing data sets (in the ratio of your choice) using a specific random seed if reproducibility is needed.

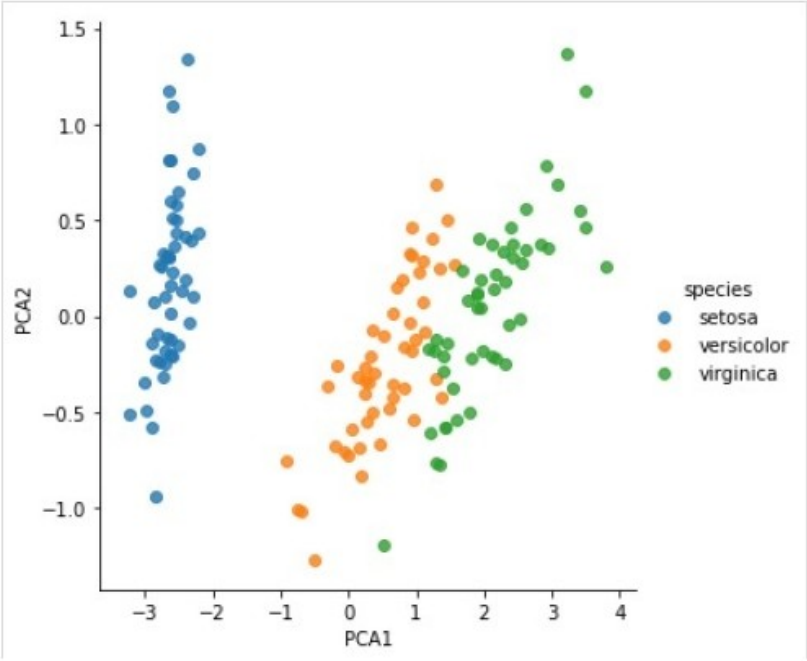
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Three training data sheaves

- One sheaf per species, with CR. (deepcopy)
- “Star” shaped sheaves, just finding averages
- One at a time, introduce a test observation into the three sheaves. Conclude that the sheaf with the smallest increase in CR is most likely where the test observation belongs.

Results

- About 10% failure in classifications.
- The sklearn method had about 2% error rate.
- This approach to classification was too simplistic. It's surprising that it worked at all!



Logic Puzzle

- Verbal matching riddles or “Rook” problems.
- You have to match *n* items with *n* attributes.
- Sample Grid:

	A	B	C	D
p	0	1	0	0
q	0	0	1	0
r	0	0	0	1
s	1	0	0	0

Logic Puzzle

- Partly solved “manually” to arrive at conditions

	Red	Yellow	Green	White
Alice-f	1			
Bob-m			1	
Chris-m				
Dillon-f				

Logic Puzzle

- Fully solved manually

	Red	Yellow	Green	White
Alice-f	1	0	0	0
Bob-m	0	0	1	0
Chris-m	0	1	0	0
Dillon-f	0	0	0	1

This took some effort!

- Early attempts had negative values and sometimes even values over 1.

[1.00001231e+00 -1.31589982e-01 -7.89579474e-02 2.10520613e-01
 -1.31578294e-01 4.48318049e-06 9.99982029e-01 1.31600688e-01
 -7.89648655e-02 1.00002204e+00 -1.05248430e-01 1.84204375e-01
  2.10513940e-01 1.31590171e-01 1.84205064e-01 4.73693199e-01]

Some conditions

- Force all rows and columns to sum to 1
- Force all values to be non-negative
- But I was still getting things like that last row of values “splitting up” the sum of 1.
- Getting “Dillon” with the “white” shirt was really difficult

Finally

- shf.GetCell('M').SetBounds([(0,1)]*16)
- This constraint forced the problem to converge to:

[9.99987476e-01 4.19151026e-06 1.44965167e-17 1.04653633e-05
0.00000000e+00 4.72662509e-05 9.99967076e-01 7.38538561e-06
1.00464822e-05 9.99944452e-01 2.62113751e-05 3.70911265e-05
7.88571982e-17 2.15261044e-05 2.34201361e-05 9.99946740e-01]

If you want to practice:

For this exercise, dispense with the storyline.

You have a 5x5 binary matrix whose rows and columns must sum to 1.

The cells are arranged such that only 7 have special significance:

```
_ _ A _ _
B _ _ _ _
_ C _ _ _
_ _ _ D E
_ _ _ F G
```

A must be 1.

G must be 0.

There is a (B OR C) rule.

There are XOR rules: (D XOR F), (D XOR E), and (E XOR G)

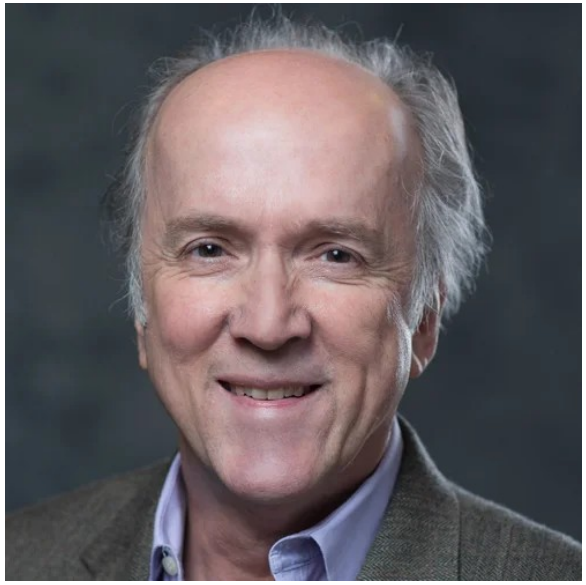
Note: Postion A is 2 in python (not 3) etc... so B is 5 and G is 24.

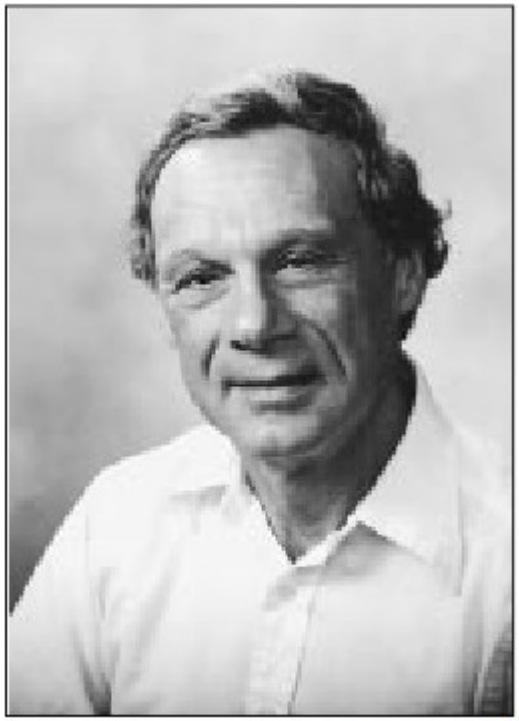
For XOR, I use the rule that the elements must sum to 1 but have a product of zero.

For the regular OR rule, I use min(X+Y, 1) and expect that value to come out to "1".

Matched Differential Equation

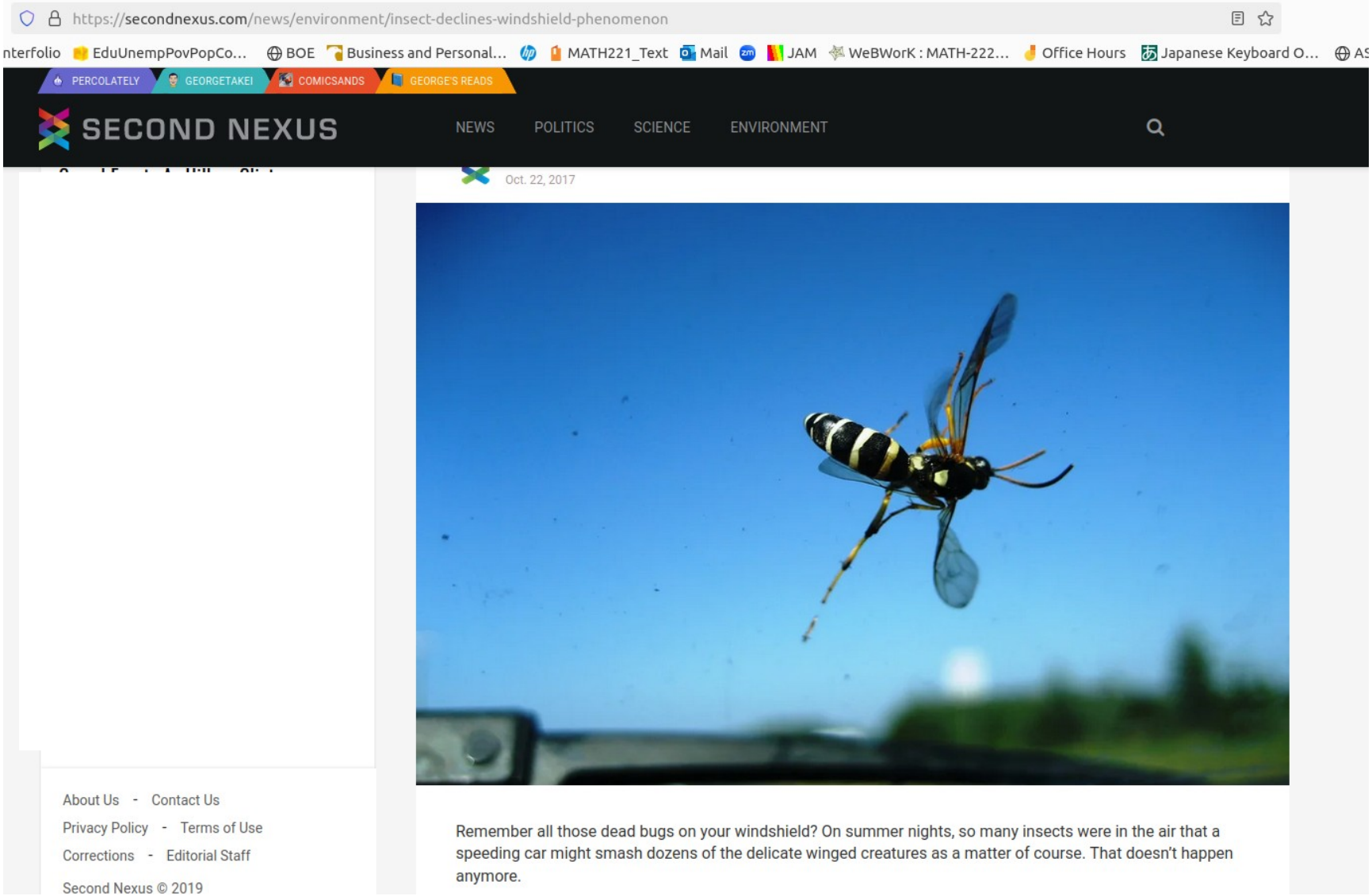
- Shout out to our Alma Mater, RPI
  - and Julian Cole and Mark Holmes
  - asymptotic expansions





Julian D. Cole

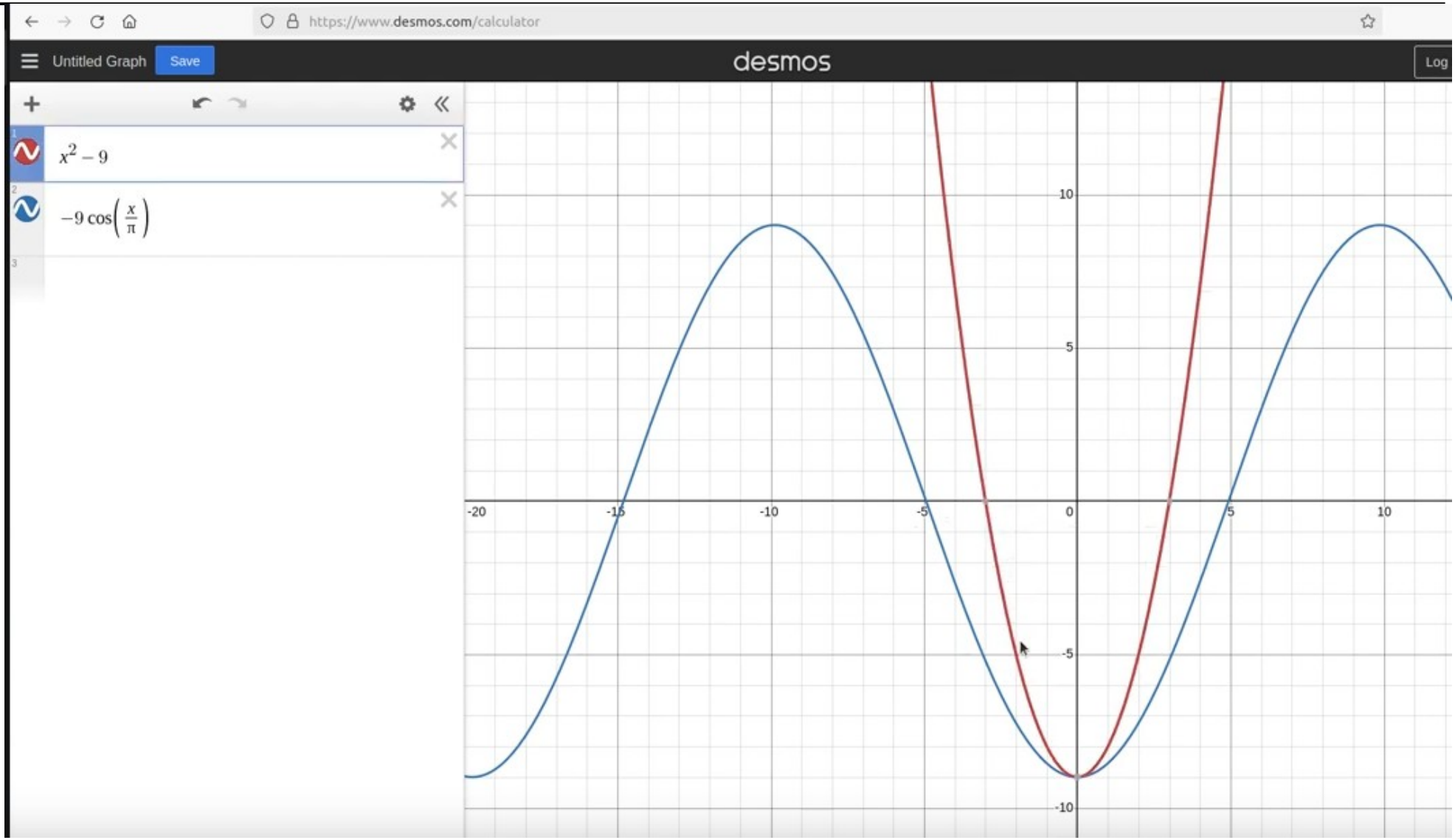
Boundary Layer Example

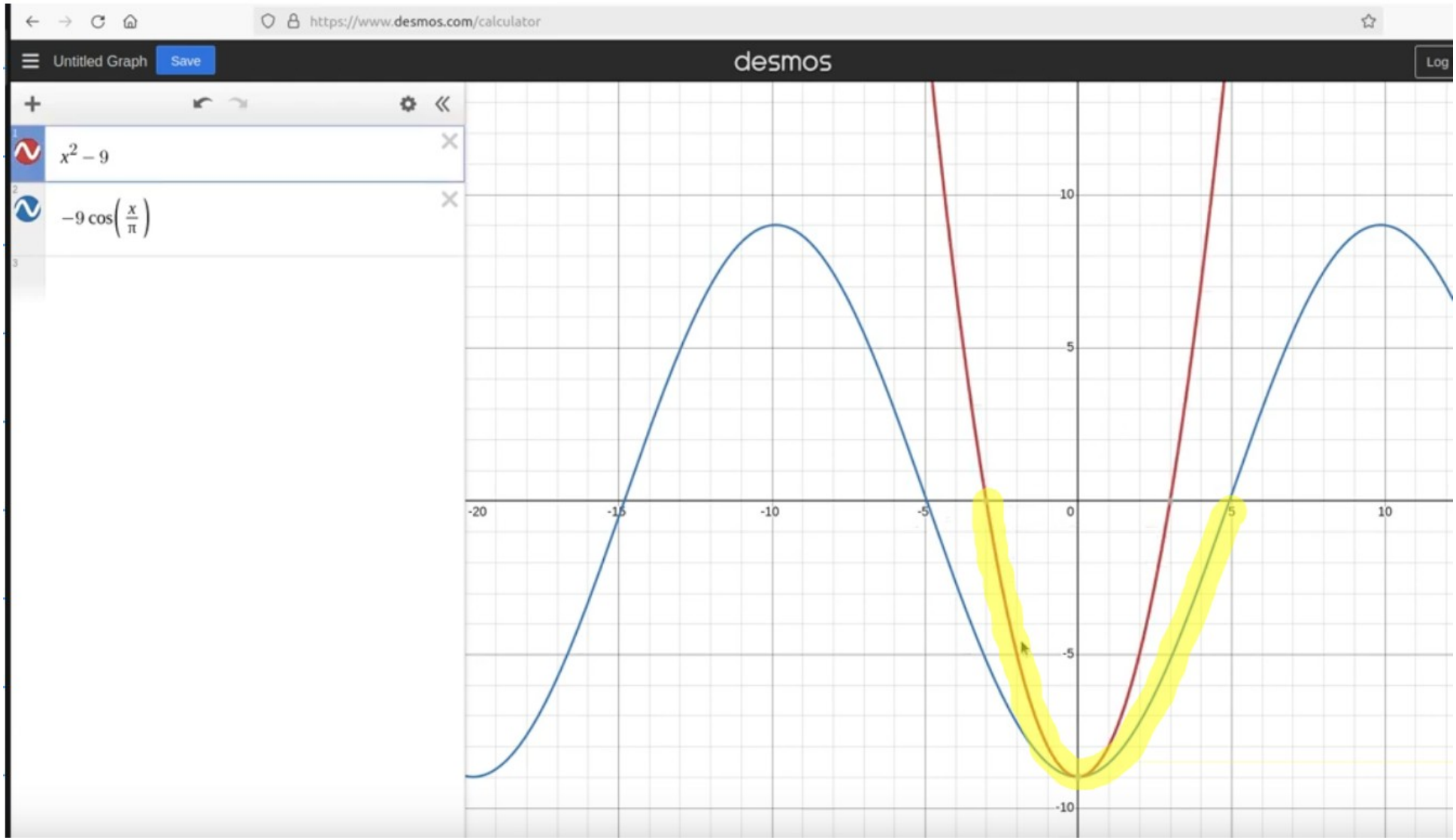


Numerical Differential Equation

- Non-uniform step sizes easily implemented
- Matching in the middle
- Conditions on the ends

Start with solution to toy problem




https://www.desmos.com/calculator
Untitled Graph
Save
desmos
Log
$x^2 - 9$
$-9\cos\left(\frac{x}{\pi}\right)$
10
5
-5
-10
-20
-15
-10
-5
0
5
10

Problem Setup

- From -3 to 0

  y'=2x
- Must match at 0
- Zero at -3
- Use 12 steps: -0.25

- From 0 to $\frac{\pi^2}{2}$

  y'= $\frac{9}{\pi}\sin\left(\frac{x}{\pi}\right)$
- Must match at 0
- Zero at $\frac{\pi^2}{2}$
- Use 16 steps: $\frac{\pi^2}{32}$

The cells to the right

Do not optimize end

Data on cells:
(x,y,y')

$h = \pi^2/32$

Except end:
(x,y) only

$$\lambda w: \left(w_0 + h,\ w_1 + w_2 \cdot h,\ \frac{9}{\pi} \sin\left(\frac{w_0 + 1.5h}{\pi}\right)\right)$$

The cells to the left

Do not optimize end

$x = -3$

Except end:
(x,y) only

$h_2 = -0.25$

Data on cells:
(x,y,y')

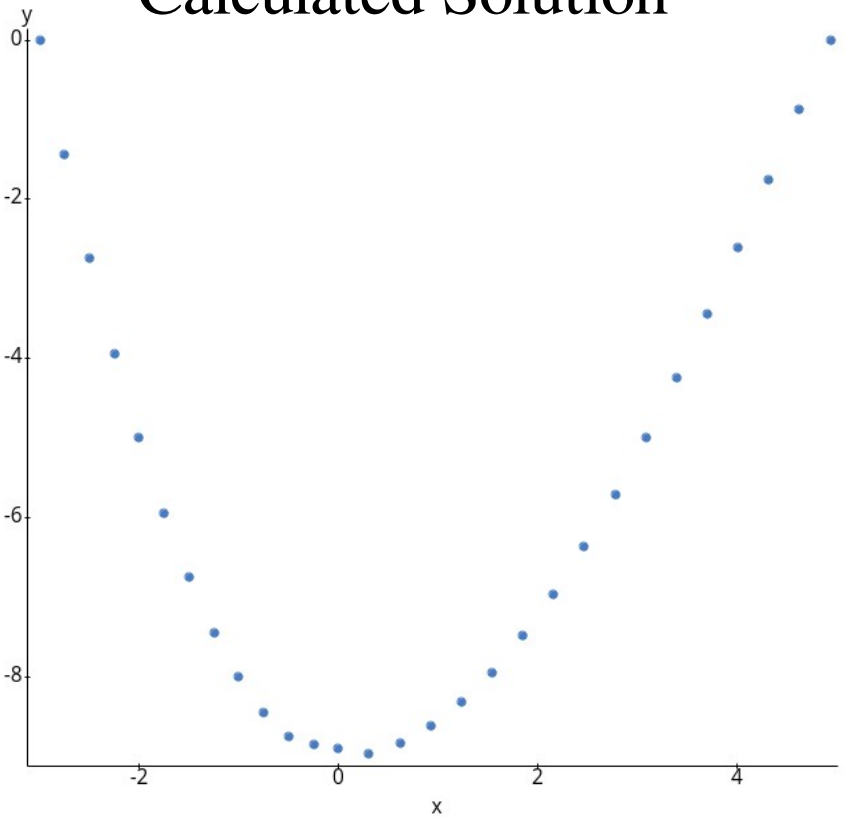
$$\lambda w: \left(w_0 + h_2,\ w_1 + w_2 \cdot h_2,\ 2w_0 + 1.5h_2\right)$$

Of course...

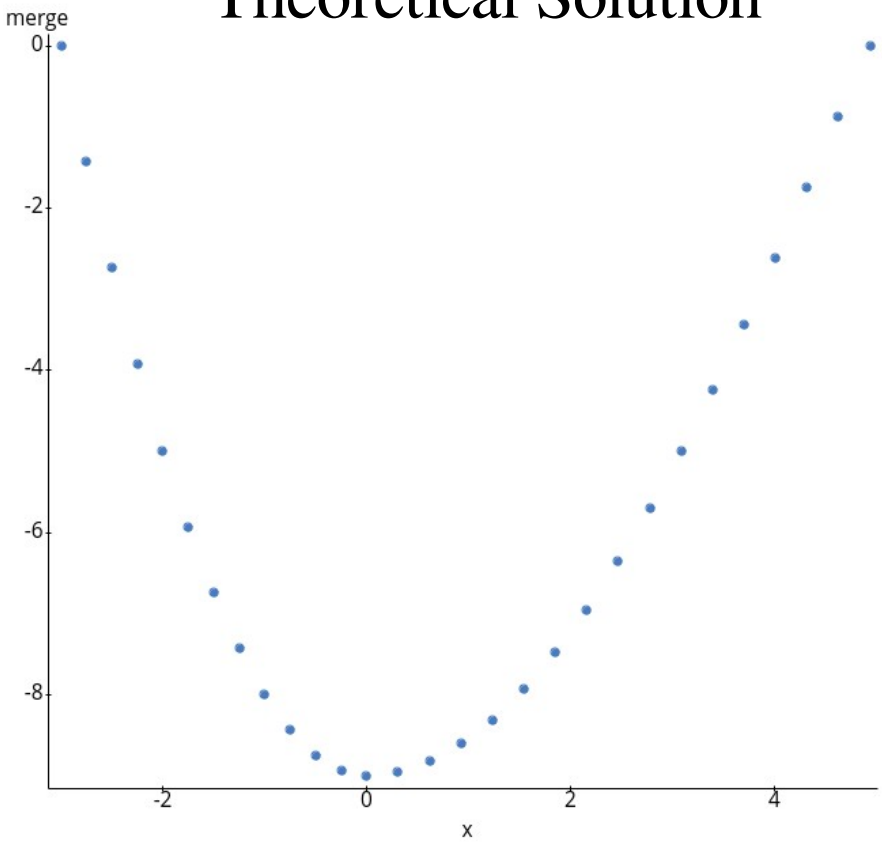
- Use a script to generate all these nodes and cofaces, or you will take forever and make errors!

Results

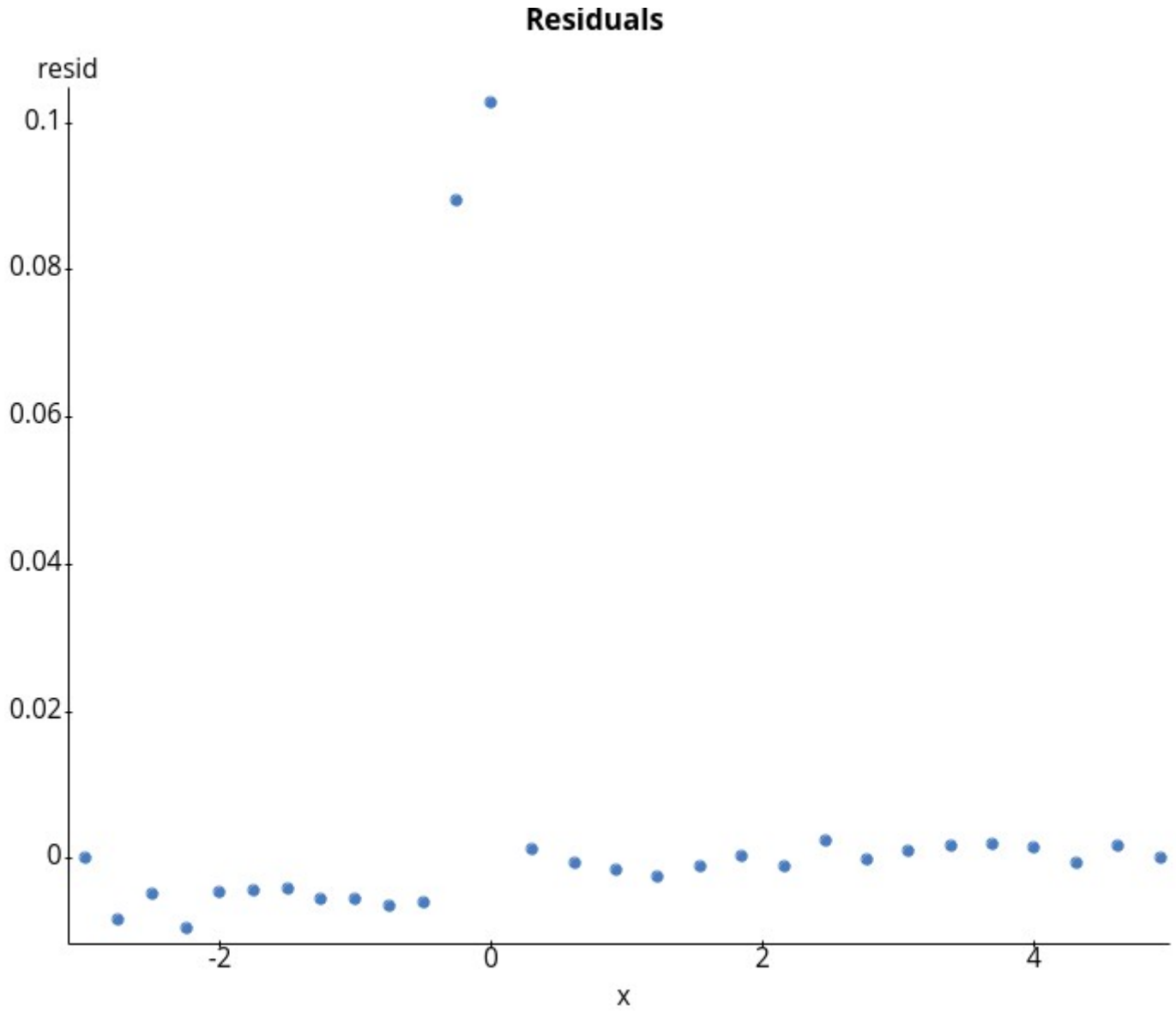
Calculated Solution



Theoretical Solution



Error is worst at join

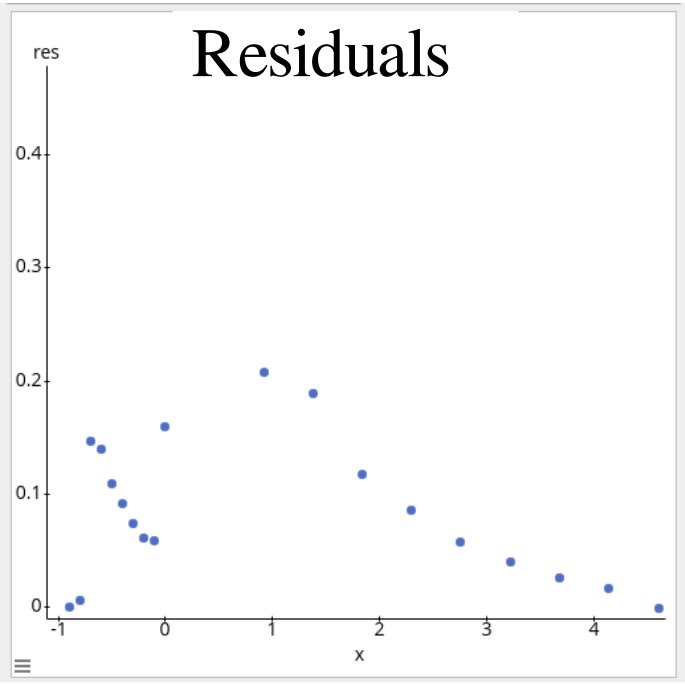


A problem for you!

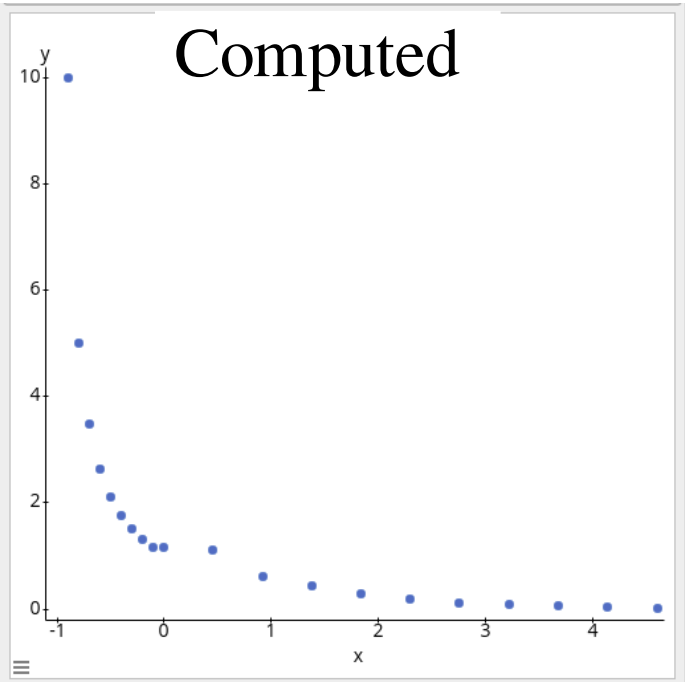
```
Problem Statement:
Fit a simple differential equation model like in the example.
Notes: x**2 is how we square x in Python.  Note that x^2 is a bit-operation!!! math.exp(x) is e^x  math.log(100) is ln(100)

The center point : x=0, y=1, y'=-1  (Fix this initially only for debug, then allow to optimize on this.)
The right side of the problem:
  Fixed at (log(100), 0.01)  #NOTE: math.log(100) is ln(100) which is about 4.6 but if importing all of "math", I type log(100)
  10 steps with h=log(100)/10  (All internal steps should be free to optimize.)
  Differential Equation to use: y' = -y
  KEY: exp(-x)
The left side of the problem:
  Fixed at (-0.90, 10)
  9 steps with h2=-0.1 (Internal steps should be free to optimize.)
  Differential Equation to use: y' = -y**2
  KEY: 1/(x+1)
```

(Analysis of Results you should get.)

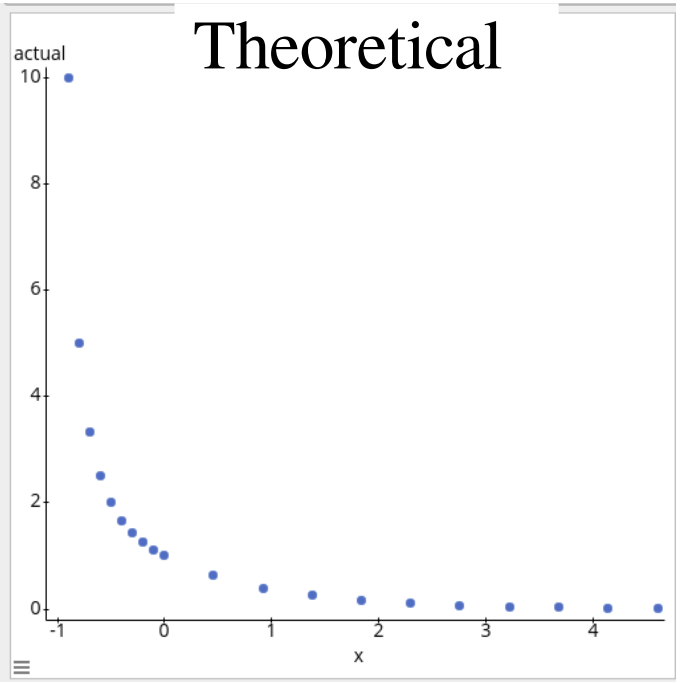
Residuals



Computed



Theoretical



Now, just some practical tips...

- What can I coax out of the code?
  - Mostly everything except the lambdas. (Python's fault!)
- shf.GetCellIndexList() *#Native to PySheaf*
  - Lists the node names
- shf.nodes() *# From networkx*
  - Also lists the node names
- shf.GetCell(*node*).mExtendedAssignments
  - Shows you the work done within MaximallyExtend
  - The deeper in you are from the top, the more you get!

More quick tips...

- shf.GetCoface(*node1*,*node2*).mEdgeMethod

  <function <lambda> at 0x7930193481f0>

  You can't see it, but can use it and know if it's there.

- shf.GetCoface(*node1*,*node2*).mInputTagType

  You can see the tag you selected for this input type

- shf.edges, shf.out_degree, shf.in_degree
  - All these are from *networkx*
- Documentation will list all functions both written and inherited. This is just a quick guide.

Main concepts in these slides:

- Sheaves are not magical. You can still have a poor model using sheaves.
- Sheaves can sometimes get you a decent answer quickly, nonetheless. (Possibly better than the currently top option!)
- Use scripts to generate parts of the sheaf. (In fact, you don't need to generate any parts you don't access!)
- When we find ourselves needing to use Real values rather than Boolean, for Boolean-valued problems, this is part of a more general theme. In order to optimize, we need a smooth (not discrete) distance metric so we can "slide" smoothly between "states". If we can provide such a "ramp" system, we can still solve the problem numerically.
- SetBounds sometimes helps us constrain difficult problems.
- Looking at what you're building can help sometimes