Computational Linear Algebra: Module 1  
Maya Shende  
Due: January 31st, 2018

Exercise 1

* We ran the demos and they worked.

Exercise 2

* I solved this equation by hand, and found that x1 = 1.5 and x2 = 2
* We verified this solution by running the code
* Algorithm: For 2 equations, solve the first equation for x1, then plug this into the second equation to solve for x2. Then plug x2 back into the first equation to solve for x1.
* General Algorithm: For n equations, solve the first equation for x1, plug this into the remaining equation to get n-1 new equations. Then, solve the second equation for x2 and plug this into the remaining n-2 new equations. Repeat this process until you have xn = some number. Then plug each variable back into the above equation to solve for the remaining n-1 variables.

Exercise 3

* For these equation, I solved the first equation for x1 and plugged this into the second and third equations. Then, I solved the second equation for x2 and plugged this into the third equation. By doing this, I found that x3 = **plug in value there**.
* I verified this result by running the code.

Exercise 4

* variableCoeffs: 2D array
* rightHandSide: 1D array containing the b vector from Ax = b

Exercise 5

* When I apply my algorithm to this set of equations, I find that 15=10, which is not true. So, I can conclude that there is no solution to this system of equations.

Exercise 6

* There are more unknowns than equations, so there are multiple solutions. In these problems, the equations act as constraints on the unknowns, so since there are less constraints, there are more possibilities for the values of the unknowns.

Exercise 7

* In this case, there is a solution: x1 = 1.5 and x2 = 2. However, for such systems where there are more equations than unknowns, we can guarantee a unique solution.

Exercise 8

* Here we see that there is not a solution. This is because the last equation is written with more precision, so the solution we got in Exercise 7 is not exact here.

Exercise 9

* Want to handle any number of equations
* Want to handle any number of unknowns
* Want to handle situations where there can be multiple solutions, or situations where there is no solution.

Exercise 10

* There is one parameter for the ellipse
* The minimum number of points needed to draw an ellipse is 5 points.

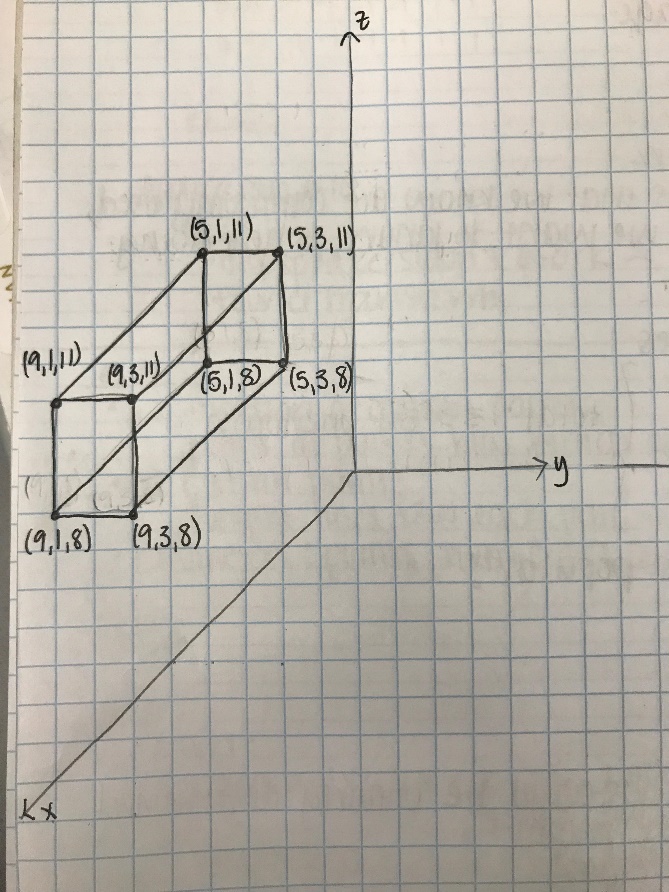
Exercise 11

|  |  |  |
| --- | --- | --- |
|  | Advantages | Disadvantages |
| A | * Easy to store * Easily generalizable | * Computationally expensive * Memory expensive |
| B | * Exact * Real-time * Less computationally expensive | * Can’t generalize; have to write equations for each curve explicitly * Maybe there isn’t an easy mathematical formula |

Exercise 12

* A: location of control points, pairs of points between which there are lines
* B: control points, tangent lines between each pair of control points, points to use for the curve
* Parameters: points, numCurvePoints

Exercise 13

 This cube looks like it is lying in the negative-y side of the xz-plane, when in reality it is in the positive xz-plane. This is where eye-coordinate matters. In the 2D world, it looks like the mapping is as follows:

(9,1,8) 🡪 (0,0)

(9,3,8) 🡪 (2,0)

(9,1,11) 🡪 (0,3)

(9,3,11) 🡪 (2,3)

(5,1,8) 🡪 (4,4)

(5,3,8) 🡪 (6,4)

(5,1,11) 🡪 (4,7)

(5,3,11) 🡪 (6,7)

Exercise 14

* The eye coordinate determines where it “looks” like the cube is positioned

Exercise 15

* The size of the image is 20x20 (400) pixels
* The size of the png file is 90 bytes
* I don’t know if this is lossy or lossless compressions, but I can see that the png file is compressed because the number of bytes is less than it should be given the size of the image (in pixels)

Exercise 16

* Demo1:
  + Original data: Vector, length=8: ( 0.000 1.000 2.000 3.000 4.000 5.000 6.000 7.000)
  + Inverted Data: Vector, length=8: (   -0.052 1.037 2.059 2.820 4.281 4.680 6.279 6.838)
* Demo2:

original image: Matrix, numRows=8 numCols=8:

0.505 1.237 2.630 3.728 4.415 5.843 6.703 7.402

8.863 9.316   10.930   11.609   12.326   13.220   14.732   15.405

   16.878   17.411   18.228   19.638   20.566   21.329   22.844   23.014

   24.004   25.152   26.833   27.416   28.003   29.160   30.971   31.453

   32.486   33.013   34.448   35.521   36.758   37.753   38.988   39.745

   40.907   41.913   42.774   43.089   44.964   45.588   46.815   47.878

   48.008   49.765   50.175   51.522   52.139   53.467   54.950   55.634

   56.877   57.280   58.844   59.738   60.540   61.257   62.416   63.626

Decompressed image: Matrix, numRows=8 numCols=8:

0.453 1.298 2.604 3.709 4.474 5.763 6.778 7.357

8.897 9.279   10.934   11.647   12.253   13.310   14.652   15.452

  16.948   17.317   18.309   19.575   20.610   21.302   22.857   23.008

   23.803   25.411   26.645   27.511   27.997   29.107   31.043   31.405

   32.795   32.619   34.723   35.402   36.730   37.877   38.844   39.839

   40.555   42.359   42.467   43.212   45.012   45.429   46.994   47.762

   48.314   49.378   50.440   51.419   52.091   53.611   54.788   55.738

   56.700   57.504   58.691   59.796   60.570   61.171   62.511   63.565

* Demo 3:

original image: Matrix, numRows=8 numCols=8:

0.505 1.237 2.630 3.728 4.415 5.843 6.703 7.402

8.863 9.316   10.930   11.609   12.326   13.220   14.732   15.405

   16.878   17.411   18.228   19.638   20.566   21.329   22.844   23.014

   24.004   25.152   26.833   27.416   28.003   29.160   30.971   31.453

   32.486   33.013   34.448   35.521   36.758   37.753   38.988   39.745

   40.907   41.913   42.774   43.089   44.964   45.588   46.815   47.878

   48.008   49.765   50.175   51.522   52.139   53.467   54.950   55.634

   56.877   57.280   58.844   59.738   60.540   61.257   62.416   63.626

Decompressed image: Matrix, numRows=8 numCols=8:

0.591 1.207 2.665 3.644 4.442 5.599 6.932 7.382

8.794 9.416   10.726   11.629   12.433   13.465   14.716   15.224

   16.778   17.344   18.639   19.514   20.445   21.378   22.593   23.218

   24.030   25.326   26.436   27.470   28.048   29.337   30.911   31.435

   32.499   33.039   34.502   35.454   36.741   37.637   38.962   39.876

   40.964   41.578   42.789   43.635   44.828   45.590   46.811   47.736

   47.994   49.625   50.483   51.500   52.088   53.318   54.974   55.678

   56.870   57.556   58.619   59.412   60.691   61.322   62.513   63.592

It looks like as you go through the demos, the compression becomes less and less lossy.

Exercise 17

* Query image 0: image 3
* Query image 1: image 7

Exercise 18

* All eigen images have a little bit of everything from the collection of images; ghost-like images are eigenvectors

Exercise 19

* One advantage of this approach is that it means that webpages that are more “mentioned” elsewhere will have more incoming edges. However, a webpage that is considered very important as a stand-alone page may not have a lot of links to it, thus causing it to have very few incoming edges and thus lowering its “importance”.

Exercise 20

* Node 6 has 3 incoming edges, which is the most number of incoming edges
* Node 2 looks like it has the highest visit probability

Exercise 21

* This is different because node 4 doesn’t have an edge to 6 or 5, so 4 doesn’t go anywhere.

Exercise 22

* linMagic.pageRankViaEigenValues(…)
* linMagic.pageRankViaPowerMethod(…)

Exercise 23

* The **poodle** was developed as a water retriever, and the distinctive clipping of its heavy coat was initiated to increase the animal's efficiency in the water. ... An elegant-looking dog, often ranked as one of the most intelligent of all breeds, the **poodle** has been bred in three size varieties—standard, miniature, and toy.
* **Labrador retriever**, breed of sporting dog that originated in Newfoundland and was brought to England by fishermen about 1800. It is an outstanding gun dog, consistently dominating field trials. Standing 21.5 to 24.5 inches (55 to 62 cm) and weighing 55 to 80 pounds (25 to 36 kg), it is more solidly built than other retrievers and has shorter legs. Distinctive features include its otterlike tail, thick at the base and tapered toward the end, and its short, dense coat of black, brown (“chocolate”), or yellow. The Labrador retriever is characteristically rugged, even-tempered, and gentle. In England it has been used in military and police work, as a rescue dog, and as a guide dog for the blind. An ideal family pet, the Labrador retriever became in the 1990s the most popular dog breed in the United States.
* These two pieces of text are both broadly about dogs, or breeds of dogs, however a a simple keyword text on both would show that the first is about poodles and the second is about Labrador retrievers.

Exercise 24

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0 | 1 | 1 | 1 | .5 | -.5 | -1 | -1 |
| 1 | 1 | 1 | 1 | .5 | -.5 | --1 | -1 |
| 2 | 1 | 1 | 1 | .5 | -.5 | -1 | -1 |
| 3 | .5 | .5 | .5 | 1 | .5 | -1 | -1 |
| 4 | -.5 | -.5 | -.5 | .5 | 1 | -1 | -1 |
| 5 | -1 | -1 | -1 | -1 | -1 | 1 | 1 |
| 6 | -1 | -1 | -1 | -1 | -1 | 1 | 1 |

Exercise 25

* LSA demo doesn’t work
* Stop words are words that we know are commonly used, like “the”, which we want to ignore when doing text analysis.

Exercise 26

* 33/54 ratings are unknown
* Movie 1 is the most popular
* A movie with less ratings could be unfairly determined as the most popular movie if the few ratings that is has a really high.
* Users 4 and 5 may be similar, as well as users 7 and 8

Exercise 27

* The input is the known ratings and the output is the unknown ratings.

Exercise 28

* We compiled and executed this, but neither of us have working laptop speakers, so we listened to groups around us to hear the signals, filtered and unfiltered.

Exercise 29

* There are about 3,270,000 pdf documents with the word “matrix”. About 1/3 of these are scientific publications.