AMATH 352: Problem set 1

Niall Mangan

October 5, 2016

Read the Instructions: There is a longer Matlab part an short written part for this assignment. Upload your written part (scanned in or LaTeX) as pdf to Canvas. Submit the MATLAB portion to scorelator. You will need to upload a MATLAB .m script file that produces all the .dat files indicated.
Suppress Matlab output using ';' at the end of a line of code for submissions to scorelator. You should start out coding with out the ';' to make sure the output is what you expect, but add before you submit.

1 Matlab Tutorial on Vectors and Matrices

Write one script (.m file) for all of part 1. Yes this will be a long file, you can use `%%` to subdivide into sections. For each problem save the indicated values in the .dat file using save() command.

```
save('filename.dat', 'variable_name', '-ascii')
save filename.dat variable_name -ascii
```

It is good practice to comment your code so that it is easy to read. To make comments, any text to the right of % will not be executed by Matlab. If you have not programed in Matlab before I strongly recommend watching the Matlab tutorial videos posted on the class website.
Google is your friend for learning syntax!

1.1 Operations: vectors and scalars

If you want to perform an operation on each element of a vector, you will have to use the element wise operations for some operations (.$*$, ./, .$\hat{}$). If a is a scaler, you do not need to use the point wise operations: $a * x$, x/a. The operations +,- work for either scaler or element wise ($x + y$ and $a + x$). One of the most common errors you will get is when you try to do a scalar operation when you mean a element wise operation. (1 pt each)

(a) Define a vector $x1 \in [-1, 0]$ with 10 points. Use the `linspace(start, end, number_points)` function.
Save $x1$ as A11a.dat.

(b) Define a vector $x2 \in [0.1, 1]$ with spacing 0.1 between each point. Use the x = (start point):(increment):(end point) notation.
Save $x2$ as A11b.dat.

(c) concatenate (ie combine into 1 vector) `x = [x1 x2]`
Save `x` as A11c.dat

(d) calculate the transpose of x, `xtrans = x';`.
Save `xtrans` as A11d.dat

(e) calculate $y1 = a * x$ for a = 100
Save `y1` as A11e.dat

(f) calculate $y2 = x * x$
Save `y2` as A11f.dat

(g) calculate $y3 = x^4 + ax^3 + bx^2 + c$, for $a = 4,\ b = -3,\ c = 1$
Save `y3` as A11g.dat

1.2 Built in Functions

Write a script to do the following : Using x from the previous problem calculate vectors, $y4,\ y5$, and $y6$ in Matlab. Define coefficients a and b before defining the y-vectors. Use Google or Matlab help to find built in functions for exponents, sines, cosines, and logarithms. (1 pt each)

(a) $y4 = a + e^{bx}$, for $a = 2,\ b = 0.5$
save `y4` as A12a.dat

(b) $y5 = \sin(ax) + x\cos(bx)$, for $a = \pi/4,\ b = \pi/2$
save `y5` as A12b.dat

(c) $y6 = \log(\sqrt{\frac{ax+1}{x^3+b}})$ for $a = 0.2,\ b = 5$
save `y6` as A12c.dat

1.3 Matrix operations

Let the following be defined:

$$A = \begin{bmatrix} 1 & 1 \\ -2 & 1 \end{bmatrix},\ B = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix},\ C = \begin{bmatrix} 2 & 0 & -5 \\ 0 & 1 & -1 \end{bmatrix}\ D = \begin{bmatrix} 0 & 2 \\ 2 & 3 \\ -1 & 1 \end{bmatrix}$$

$$x = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ y = \begin{bmatrix} 0 \\ 1 \end{bmatrix},\ z = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$$

Try calculating (a-l) in Matlab. Save a data file for a-l as A13a.dat A13b.dat, ect. You will need to give the result a variable name in order to save it (Ma, Mb, ect). If the calculation fails, comment out the line of code (use % in front

of the code) and set the output to zero in the .dat file for that problem. (1 pt each)

For example you if (d) did not work you would have

```
% 1.3.d
% Md = A*x;
Md = 0;
save('A13d.dat','Md', -ascii)
```

For multiplication you should use matrix multiplication not element-by-element multiplication. (`*`, not `.*`)

(a) $\mathbf{A} + \mathbf{B}$

(b) $2\mathbf{A} - \mathbf{D}$

(c) $3\mathbf{x} - 4\mathbf{y}$

(d) $\mathbf{A}\mathbf{x}$

(e) $\mathbf{B}(\mathbf{x} - \mathbf{y})$

(f) $\mathbf{C}\mathbf{y}$

(g) $\mathbf{D}\mathbf{x}$

(h) $\mathbf{D}\mathbf{x} + \mathbf{y}$

(i) $\mathbf{A}\mathbf{B}$

(j) $\mathbf{B}\mathbf{C}$

(k) $\mathbf{A}\mathbf{C}$

(l) $\mathbf{C}\mathbf{D}$

1.4 Matrix Manipulation

In this section you will formulate and manipulate matrices using index selection. For example, if you have a matrix `M` you can select sub sets of the rows by indexing `M(row1:row2,:)`. The colon (:) selects all columns. `end` references the last index in a row or column. (1 pt each)

(a) Use the `zeros(m,n)` and `ones(m,n)` commands to create a matrix. There are multiple ways to do this (indexing a column, concatenation).
$M1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
save `M1` as A14a.dat

(b) Using indexing in M1 to make this matrix:
$M2 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 5 & 5 & 1 \\ 0 & 0 & 2 & -1 \end{bmatrix}$
save `M2` as A14b.dat

(c) Use indexing to sub-select the 2X2 matrix
$M3 = \begin{bmatrix} 5 & 1 \\ 2 & -1 \end{bmatrix}$
save `M3` as A14c.dat

(d) Use indexing to switch the first and second row vectors and create this matrix:
$M4 = \begin{bmatrix} 1 & 5 & 5 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & -1 \end{bmatrix}$
save `M4` as A14d.dat

1.5 Loops and logic

The main advantage of Matlab is that it performs vector-wise or matrix calculations much faster than loops, which are faster in languages such as C++ and Python. Still, loops and logic are important parts of programming. Here you will use the `for ... end` and `if ... elseif ... else ... end` syntax. There are other ways to create these vectors, but use the loops and logic functions for practice. (1 pt each)

(a) Write a `for` loop that creates a column vector (note the transpose)
$v1 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}^T$
save `v1` as A15a.dat

(b) Augment your code, using `if ... elseif ... else ... end` syntax and using the logical operator `<` and `>=` to create the column vector
$v2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}^T$
save `v2` as A15b.dat

1.6 Profit from Roundoff Error Accumulation

In the 1999 movie Office Space, a character creates a program that takes fractions of cents that are truncated in a bank's transactions and deposits them into his own account. This is not a new idea, and hackers who have actually attempted it have been arrested. In this exercise, we will simulate the program to determine how long it would take to become a millionaire.

Assume we have access to 50,000 bank accounts. Initially, we can take the account balances to be uniformly distributed between, say \$100 adn \$100,000. The annual interest rate on the accounts is 5%, and interest is compounded daily and added to the accounts, except that fractions of a cent are truncated.

These will be deposited into an illegal account that initially has balance \$0. (Problem from Greenbaum & Chartier Numerical Methods)

Write a MATLAB program that simulates the Office Space scenario. You can set up the initial accounts with the commands

```
accounts = 100+(100000-100)*rand(50000,1);
% Sets up a vector of 50,000 accounts with balances between $100 and $100,000
accounts = floor(100*accounts)/100;
% truncate each account to the nearest penny from initial balances
```

Write a MATLAB program that increases the accounts by (5/365%) interest in each day (ie next day value = $(1 + 0.05/365) * \text{value}$), truncating each account to the nearest penny, and placing the truncated amount into an account called the illegal account (define as `illegal = 0;` at the beginning). You will need to calculate the rounded values for each account and then sum them (use `sum(vector)` command) Assume the illegal account can hold fractions (i.e. do not truncate this accounts values). Let the illegal account also accrue interest, and assume it does so before the fractional cents are added each day. (1 point each)

Use a `for` loop to calculate the value in your account over 10000 days. Use `if elseif else end` syntax to save `.dat` files for the following days. You will need to use the `==` logical operator to compare the day index.

(a) 10 days
save `illegal` as A16a.dat

(b) 100 days
save `illegal` as A16b.dat

(c) 1000 days
save `illegal` as A16c.dat

(d) 10000 days
save `illegal` as A16d.dat

2 Written Section: Linearity

In class, I showed you the definition of linearity for functions in one dimension. (8 pts total)

The function $f : U \to V$ is a linear function if
$\forall u_1, u_2, \in U \ f(u_1 + u_2) = f(u_1) + f(u_2)$
and
$\forall u \in U$ and $\alpha \in R \ f(\alpha u) = \alpha f(u)$

2.1 Consider the polynomial function $f_1(x, a) = ax^3$.

(a) Is f linear in x? Show why it is or is not. Make sure to include both parts of the 2-part definition. (2 pt)

(b) When fitting some data to this fuction, we have the values x^3 for each data point. In this case we want to determine the coefficient a. Is $f(x, a) = ax^3$ linear in a? Show why it is or is not. Make sure to include both parts of the 2-part definition. (2 pts)

2.2 Consider the function $f_2(x, a, b) = ae^{bx}$.

(a) For which of x, a, and b is f_2 linear in? (2 pts)

(b) Set $f_2(x, a, b) = y$. Transform $y = ae^{bx}$ into a linear problem in the other variables (2 pts)

(c) In the transformed system, is the transformed function linear in all variables? (1 pt)