

MATH 313 Spring 2009, Homework 3, Computer-based problems

As I mentioned in the first Matlab assignment, in Matlab, almost everything is a matrix or a vector. In that assignment, though, we didn't really work much with matrices. In this assignment we do!

Matrices can be loaded into Matlab in a variety of ways. The easiest is to simply list the entries with semicolons to separate rows:

```
>> theta=pi/4;
>> a=[cos(theta) sin(theta); -sin(theta) cos(theta)]
```

If you want to know how big a matrix is, use the "size" command:

```
>> size(a)
```

will tell you that a is 2x2. For small matrices, this isn't very useful, but for larger ones it's handy. If you want a matrix with random entries, for instance a 7x4 matrix, use

```
>> b=rand(7,4)
```

To get a new random matrix of the same size, you can use

```
>> c=rand(size(b))
```

There are other useful matrix-making commands, such as "zeros", "ones", "eye", and "meshgrid". Read about them! (Also, Matlab can read almost any data file format you can imagine, like JPEG images! Look into the documentation or ask me if you're interested...)

In the previous assignment, we used the elementwise operations to avoid loops. Now that you know more about matrices, we can use matrix operators to do useful things to matrices. For instance, we can multiply two random matrices

```
>> rand(5,3)*rand(3,4)
```

In Matlab, we have easy access to gaussian elimination. It's essentially matrix division, so Matlab represents it by "\". In particular, if you want solve $Ax = b$, we'd execute

```
>> x=A\b
```

Imagine that we're writing $A^{-1}b$ as $\frac{b}{A}$, and you'll remember to use the backslash. (If you wanted to write bA^{-1} for b a row vector, you can use

```
x=b/A
```

which result in x being a row vector...)

Matlab also provides a way to calculate matrix inverses using the "inv" command. As we discussed in class, matrix inversion is not (numerically) the same as gaussian elimination. In this problem, we examine this a little (courtesy of the MATLAB documentation). Type the following into Matlab:

```
>> n = 500;
>> Q = orth(randn(n,n));
>> d = logspace(0,-10,n);
>> A = Q*diag(d)*Q';
>> x = randn(n,1);
>> b = A*x;
```

Now, you've got a matrix A (yes, it's invertible) with a random vector x , and b , their product. The task is to recover an approximation of x ... Obviously, $x = A^{-1}b$, so try

```
>> y = inv(A)*b;
```

Now, using gaussian elimination, we should execute

```
>> y2=A\b;
```

Now these two aren't very different, but compare the error

```
err = norm(y-x)
```

with

```
err2 = norm(y2-x)
```

It's not so different... But the residuals

```
res = norm(A*y-b)
```

and

```
res2 = norm(A*y2-b)
```

differ significantly!

Matlab provides the ability to define new functions, which is an important way to manage complicated programs. A function has a number of inputs and a number of outputs... Like scripts, functions are stored in plain text files with the extension ".m". However, the first line of a function file must begin with the word "function" and then an input/output specification. For instance, the following could be stored in "helix.m"

```
function [x,y,z]=helix(t)
% Compute points on a parametric curve (a helix)
% Input: t = matrix or vector of parameter values to evaluate
% Output: x,y,z = coordinates of points computed

x=cos(t);
y=sin(t);
z=t;
```

Notice that the function name in the top line must match the filename! If this file is stored in Matlab's current directory, you can call it:

```
>> [x,y,z]=helix(0:0.1:3*pi);
>> plot3(x,y,z);
```

Problem 1 Write a function to do something interesting. For instance, you could write a function that computes the number of bus rides I can take, given what's in my pocket: pennies, nickels, dimes, quarters, and SEPTA tokens...

(Discrete differentiation and integration.) Suppose v is a column vector, which might represent samples of data. If we wanted to approximate a running integral of $v = \int_0^t v(x)dt$, we might compute a cumulative sum of v . (There's a Matlab command to do this, "cumsum", but we won't use it for this problem.) This can be computed by multiplying v with a lower-triangular matrix of ones:

$$(1) \quad C = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots \\ 1 & 1 & 0 & 0 & \dots \\ & & \dots & & \\ 1 & 1 & 1 & 1 & \dots \end{pmatrix}$$

Here's how to make such a matrix in Matlab:

```
>> [cols,rows]=meshgrid(1:length(v),1:length(v));
>> C=1.0*(rows>=cols);
```

Which says make C a matrix with entries=1 when the row number is greater than or equal to the column number. It's a square matrix that's as wide as v .

Problem 2 What matrix D would you multiply v by to compute a vector which contains the differences between adjacent elements? Write a Matlab function to construct D given v (or the size of v).

Problem 3 The Fundamental Theorem of calculus states that differentiation and integration are inverse operations. An analogous statement in the discrete context is that D and C are inverses of each other.

Part 3a Is this true for the matrix D you computed? Why or why not?

Part 3b What is the inverse of C and how is it different?

Hint: For a few random vectors v , you should compute $(DC - I)v$ (I is the identity matrix) to tell what your matrix C is doing.

Remember to print out your commands in a script, and print it out, along any functions or figures. Hand these in with your problems from the book!