MATLAB for MAPH 3071 Lab 3

1. Matrices

MATLAB is especially designed to handle matrices. For example if we wanted to store the matrix

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

in MATLAB we would enter the elements row by row with the following syntax,

» A=[0 1 1;1 1 0;1 0 1]

i.e. each row is separated by a semi-colon and each element in a row is separated by a space. Therefore the column vector

$$b = \begin{pmatrix} 5 \\ 3 \\ 4 \end{pmatrix}$$

has only one element in each row and is entered into MATLAB by

» b=[5;3;4]

We could now try to solve the set of three simultaneous equations given by Ax=b. In class we have seen how to do this by Gaussian elimination . We could write an M-file which would do this for us and which would contain many **for** loops. Fortunately MATLAB has preprogrammed Gaussian elimination and it is given by the backslash operator '\'. Therefore,

»x=A\b

will find the solution to our set of simultaneous equations and print the solution out on the screen.

Now have a look at my attempt to write a quick m-file to do the same thing. You will find it in the class library (L: drive). The m-file is called ***Gauss.m****. This is a very simplistic attempt – I did not event program in any pivoting !.*

Other matrix operations are illustrated by the following examples. Input the new matrix A,

»A=[1 2 3;3 4 5;7 8 0]

Then, the transpose of A can be obtained by

»B=A'

B =

1	4	7
2	5	8
3	6	0

and the product of A and B is,

»C=A*B

C =

14	32	23
32	77	68
23	68	113

The determinant of A is given by

»det(A)

and the eigenvalues of A can be found by typing

»eig(A)

You can even use MATLAB to do what you are warned in the lectures that you should never do - calculate an inverse !!

»inv(A)

To reference subsets of a matrix use the following syntax;

Given a matrix of dimension N X M (i.e. N rows and M columns), you may select reference an individual element by using parentheses (),

e.g. to get the element of the 3rd row and 2nd column in the matrix C above use;

» C (3,2)

ans =

68

You may also reference a whole row or column by using the colon (**:**) operator. For example to reference the first row in the matrix C above use;

» C (1,:)
ans =

14 32 23

or to reference the middle column in C,

» C(:,2)

ans =

32
77
68

You can also reference subsets of a row or column, e.g. to reference the first 2 elements of the 2nd row of the matix C, use;

» c(1,1:2)

ans =

14 32

You will find the use of these referencing features crucial to write programs for solving matrix problems in this course. You can also exchange actual numbers in the referencing statement for values of i in for loops etc. For example;

» for i = 1:3
v(i) = c(2,i)
end

v =

32

v =

32 77

v =

32 77 68

2. Iterative methods for solving non-linear equations

You have covered three methods of solving systems of linear equations in lectures;

1. Gaussian elimination
2. Jacobi method
3. Gauss-Seidel method

I have given you one example of a simple program to perform Gaussian elimination in the class library (see above). You may use the in built '\' operator in MATLAB to perform Gaussian elimination rather than attempt to write your own (if you feel you can – certainly have a go !). However you will now be required to write you own m-file to perform the other two iterative methods.

Here is an example of solving a 4 by 4 system of linear equations using the Jacobi method. The code is annotated so I will not explain further. This program uses a matrix formulation and therefore involves computing a matrix inverse. This is generally a bad idea, however in this case we are dealing with a diagonal matrix and therefore we know we can compute the inverse as the inverse of the diagonal elements with only the usual (i.e. the same as scalar) problems of finite computer arithmetic.

L:\jacobi.m

```
% jacobi - Jacobi iteration performed on a matrix A and vector b.
% ***This program does NOT pivot !!!! ***
% Assumes well conditioned matrix with n equation to solve n unknowns

clear;help jacobi;

A=[5 1 1 -2;
   4 9 2 1;
   3 2 6 0;
   1 3 2 7];            % matrix A

A

b=[0;
   8;
   7;
   3];                  % vector b
s=size(A);              % gets no of rows & columns in A
rows=s(1);              % number of rows in A
cols=s(2);              % number of columns in A

S=zeros(size(A));
for i=1:cols;
   S(i,i)=A(i,i);
end

S

T=S-A

x=zeros(size(b))
xnew=zeros(size(b));

for it=1:10
```

```
   for i=1:cols
      xnew(i)=b(i);
      for j=1:rows
         xnew(i)=xnew(i)+T(i,j)*x(j);
      end
      xnew(i)=xnew(i)/S(i,i);
   end

x=xnew;
disp('iteration ')
it
x
pause
end;
```

This program is available on the class library. You should now write you own to solve the same system of linear equations using the Gauss-Seidel method. Compare both iterative methods with the direct method from Gaussian elimination.

END
