MATLAB for MAPH 3071 Lab 4

Interpolation

There are a number of commands in MATLAB that will help you in programming interpolation problems.

1. POLYFIT(x,y,n)

The polyfit command will actually deliver a vector with the coefficients of a polynomial of degree n which fits the vector p(x)=y in the least squares sense. Note that this is the same as Lagrangian polynomials where n is one less than the number of x's (i.e. the n+1 points from your lectures).

This command is useful for checking your programmed answers – but is not acceptable to use it if asked to write your own m file to do the job !

Example: from page 76 of your lecture notes:

```
» x=[-2 -1 1 3];
» y=[16 1 0 -2];
» polyfit(x,y,3)
ans =
   -0.9917    2.8500    0.4917   -2.3500
```

Where the langangian cubic polynomial is:

$$-0.9917x^3 + 2.85x^2 + 0.4917x - 2.35$$

2. CONV(x,y)

This is the convolution command is MATLAB which is the same as polynomial multiplication where x and y are vectors containing the coefficients of two polynomials.

For example: from the same example, it is necessary to find the fundamental polynomial for x(1) (i.e.) the cubic polynomial l_k such that $l_k(x_j)=1$ when j=k and zero for $j \neq k$
Example:

$$L_0(x_0) = 16 \cdot \frac{(x+1)(x-1)(x-3)}{(-2+1)(-2+1)(-2+3)}$$

The constant multiplier and the denominator are fairly easily computed as they are real numbers in the x,y vectors. However the numerator does present a new challenge in MATLAB as we are dealing with symbolic x's.

This is the type of multiplication that the CONV command will deal with. What we really want is MATLAB to multiply out the coefficients of the polynomial correctly.

For example, the correct answer for the multiplication (x - 1)(x + 2) = x^2 + x - 2 . If we just look at the coefficients of this quadratic polynomial, this bit of elementary algebra looks like this, (1 -1) (1 2) = (1 1 -2).

Lets try this using the CONV(x,y) command in MATLAB.

```
» x = [1 -1];
» y = [1  2];
» conv(x,y)

ans =

   1    1   -2
```

Which is the answer that we are looking for.
Here is a MATLAB program to find the numerator for 1st required fundamental polynomial for the example above i.e. the solution to (x +1) (x - 1) (x -3)

This Example in on the class library as L:\eg_conv.m

```
% Sample program of how the CONV(x,y) command works
% to perform polynomial multiplication

clear;help eg_conv;

format compact;

x=[-2 -1 1 3];

y=[16 1 0 -2];

poly0=[1];  % polynomial for x(0) initialised as 1
  for j=1:4;
     if j ~= 1;    % reads if j is not = 1;
        poly0=conv(poly0,[1 -x(j)]); % see below for sequence here
        display('result of this parse = ');
        poly0
        display('press any key to continue');
        pause;
     end;
  end;
```

```
%%%% Here is what it is doing for the 4 parses %%%
%
% Parse (1): j=1 therefore does nothing
% Parse (2): j=2
%            sets poly0 => [1 1]
% Parse (3): j=3
%            convolves [1 1][1 -1]
%            sets poly0 => [1 0 -1]
% Parse (4): j=4
%            convolves [1 0 -1][1 -3]
%            sets poly0 => [1 -3  -1  3]
% end
```

Use the CONV command to calculate the lagrange part of question 5 on problem sheet 3. Remember you will have to get all the fundamental polynomials and add them together to give the lagrange interpolating polynomial.

3. Splines

There is a command is MATLAB that will fit a cubic spline to a set of data. This command takes the form

» yy = **spline(x,y,xx)**

Where x,y are the given data vectors and xx is the range across which you wish to interpolate. For plotting purposes set xx to a sequence across you x range (say of 100 values or so). The returned vector yy are the corresponding interpolated values for xx using a natural cubic spline. Plot the points x,y and then plot the line xx,yy.

Here is an example program which is on the class library.

L:\eg_spline

```
% eg_spline.m - a demo of the 'spline'command
%
clear; help eg_spline;

x=[1 2 4 5];
y=[0 0 2 2];

m=length(x);
xleft = min(x);
xright = max(x);
mp = 100*m;       % plot the curve for mp points
dx = (xright - xleft)/(mp - 1);
xs = xleft:dx:xright;
ys = spline(x,y,xs);

plot(x,y,'o',xs,ys)
return;
```

However useful, writing your own m-file to return the matrix of splines between points takes more thought and a bit of programming !

Here is an example of a program that returns coefficients of the cubic splines as rows in a matrix. The example is taken from page 94 of your lecture notes.

L:\splinep.m

```
% A program to find the natural cubic spline to fit a set of n=4 x,y
pairs

clear;help splinep;
x=[1 2 4 5];
y=[0 0 2 2];

n=length(x);

% need to find A(x), B(x), C(x) and D(x)

% (1) A(x) and C(x)

for j=1:n-1;
   A(j,1:4)=[0 0 -1 x(j+1)]/(x(j+1)-x(j));
   B(j,1:4)=[0 0 1 -x(j)]/(x(j+1)-x(j));
   D1=[1];
   C1=[1];
   for i=1:3
      C1=conv(C1,[A(j,3) A(j,4)]);
      D1=conv(D1,[B(j,3) B(j,4)]);
   end
   C(j,1:4)=(C1-A(j,1:4))*((x(j+1)-x(j))^2)/6;
   D(j,1:4)=(D1-B(j,1:4))*((x(j+1)-x(j))^2)/6;
   % Have A(x), B(x), C(x), D(x) need to get  yi''  y''i+1

   AA=zeros(n,n);
   BB=(zeros(size(x)))';

   for i=2:(n-1)
         if i-1~=1
          AA(i,i-1)=(x(i)-x(i-1));
       end
       AA(i,i)=2*(x(i+1)-x(i-1));
       if i+1 ~=n
          AA(i,i+1)=(x(i+1)-x(i));
       end
       BB(i)= 6*(((y(i+1)-y(i))/(x(i+1)-x(i)))-((y(i)-y(i-1))/(x(i)-
x(i-1))));
   end

   %  Use MATLABs in built gaussian elimination to solve for unknowns
   ypp=zeros(size(x));
   ypp(2:n-1)=AA(2:n-1,2:n-1)\BB(2:n-1);

end

   % add all together to get s(x)
for j=1:n-1
   S(j,:) = A(j,:)*y(j) + B(j,:)*y(j+1)+ C(j,:)*ypp(j) +
D(j,:)*ypp(j+1);
end

disp(' Each row of the matrix s expresses the 3 polynomials as cubic
coefficients');

S
```

The approach taken in this program is to split the problem into parts and then add them together.

1. An algorithm to calculate $A_j(x)$, B_j (x), C_j (x) and D_j (x) using the **conv** command for C and D.

2. An algorithm to find y_j'' a the solution to a set of linear equations using MATLABs built in function A\b.

3. find S_j (x) using all of the above and storing the resulting 3 cubic spline functions in a matrix which is displayed on screen.

Study what's going on in this program and then write your own m-file to answer the cubic spline part of question 5 on problem sheet 3.

END