Numerical Methods ITGS219

Lecture: Interpolation and Extrapolation

Newton Forward and backward Polynomials

By: Zahra A. Elashaal

5. Interpolation and Extrapolation

Introduction

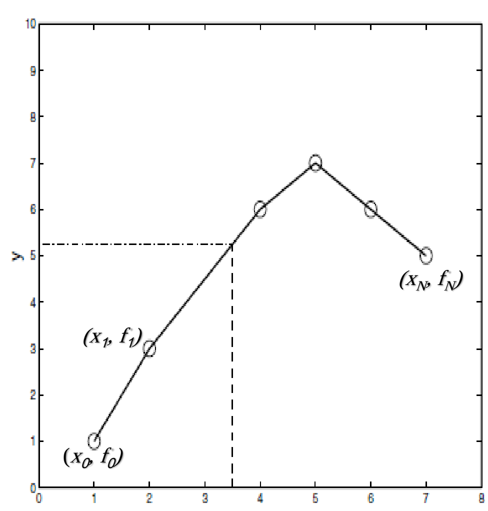
We shall now consider the problem where we know the values of a function at a certain set of predetermined points, $(x_0, f_0), (x_1, f_1), \cdots$ *through to* (x_N, f_N).

The question to be answered in this chapter is:

- What values does the function take at intermediate values of x (which is called **interpolation**), ?
- *or alternatively* what values does the function take external to this range (which is called **extrapolation**).?

Interpolation is the process of finding the value of $f(x)$ corresponding to any untabulated value of x between x_0 and x_n.

Extrapolation is the process of finding the value of $f(x)$ for some value of x outside the given range $[x_0, x_n]$.



Saving and Reading Data with MATLAB

To save all the variables which are currently in use:

```
% Save all current variables in a file
% session_vars.mat
save session_vars
```

They can be reloaded using the command

```
% Loads the variables stored in the
% file session_vars.mat
load session_vars
```

If we wish to save only certain variables we can list them

```
% Save the variables a & a1 to the
% file session_vars.mat
save session_vars a a1
% Save all variables starting with b
% to session_vars.mat
save session_vars b*
% Append all variables starting with ca
% to the file session_vars.mat
save session_vars ca* -append
```

```
a = ones(3); a1 = ones(4);
save 'session_vals.dat' a a1 -ascii
```

This creates a file whose contents looks like:

```
1.0000000e+000 1.0000000e+000 1.0000000e+000
1.0000000e+000 1.0000000e+000 1.0000000e+000
1.0000000e+000 1.0000000e+000 1.0000000e+000
1.0000000e+000 1.0000000e+000 1.0000000e+000 1.0000000e+000
1.0000000e+000 1.0000000e+000 1.0000000e+000 1.0000000e+000
1.0000000e+000 1.0000000e+000 1.0000000e+000 1.0000000e+000
1.0000000e+000 1.0000000e+000 1.0000000e+000 1.0000000e+000
```

The problem with this technique is we cannot read this back into MATLAB directly using load (it complains there are not the same number of pieces of data in each line).

Saving and Reading Data with MATLAB

The problem here is that we need to have knowledge about the data in the file to understand the form we wish to read it in.

For example consider a file containing the data

```
1.0000000e+000 2.0000000e+000 3.0000000e+000 4.0000000e+000
1.0000000e+000 4.0000000e+000 9.0000000e+000 1.6000000e+001
```

this was created using the code:

```
x = 1:4; y = x.^2;
save 'bob1.dat' x y -ascii
```

we can load this using:

```
load 'bob1.dat'
```

We can now extract the data using:

a = bob1(1,:); b = bob1(2,:); clear bob1.

These commands give us the first row and the second row in the row vectors a and b; finally we clear the array bob1 since we have extracted the requisite data.

The commands fopen, fprintf, fscanf and fclose are very powerful and as you might expect quite complicated to use:

```
x = 0:.4:2;
y = [x; exp(x).*cos(2*x)];
fid = fopen('data.dat','w');
fprintf(fid,'%6.2f %12.8f \n',y);
fclose(fid);
```

'r' read
'w' write (create if necessary)
'a' append (create if necessary)
'r+' read and write (do not create)
'w+' truncate or create for read and write
'a+' read and append (create if necessary)
'W' write without automatic flushing
'A' append without automatic flushing

```
fprintf(fid,'%6.2e %12.8e \n',y)
```

The output file called data.dat will contains:

```
0.00  1.00000000
0.40  1.03936428
0.80  -0.06498473
1.20  -2.44823335
1.60  -4.94458639
2.00  -4.82980938
```

Interpolation and Extrapolation (Which Points to Use?)

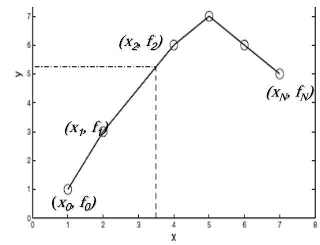
We discuss the problem of which points to use for the process, in general we apply the following method:

Pick those closest!

- This can be done by hand, or can be automated. For this purpose we use the routine **findrange.m:**

```
Run on commands
>> x=[1,2,3,4,5,6,7,8];
>> z=4.5;
>> N=3;
>> [top, bot] = findrange(x, z, N)
      top =  3
      bot =  6
>> x=[1,2,3,4,5,6,7,8];
>> z=4.5;
>> N=1;
>> [top, bot] = findrange(x, z, N)
      top =  4
      bot =  5
```

```
function [ibot, itop] = findrange(x, z, N)
if mod(N+1,2)~=0
   disp('Must use odd N')
    break
end
points = (N+1)/2; % Half number of points in stencil
[ii] = find(x>z);
if isempty(ii)
   i = length(x);
else
   i = ii(1);
end
itop = i+points-1;
ibot = i-points;
if itop>length(x)
   itop = length(x);
   ibot = length(x)-2*points+1;
elseif ibot<1
    itop = 2*points;
    ibot = 1;
end
```



Newton Forward Differences Polynomials

Newton Forward Differences

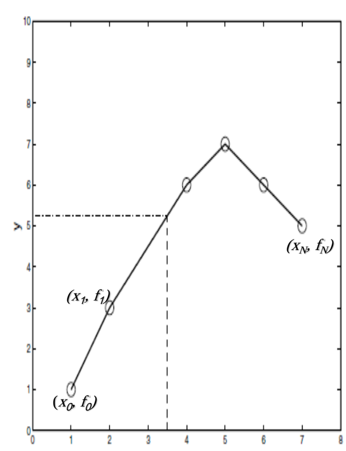
We shall now discuss the operation of fitting an ***N^{th} order polynomial*** through ***$N+1$ points*** and remark again this will yield a unique answer. We consider

- the set of data points: $(x_0, f_0), (x_1, f_1), \cdots, (x_N, f_N)$. *We now introduce the* difference operator Δ *such that*

$$\Delta f_0 = f_1 - f_0 \text{ or in general } \Delta f_j = f_{j+1} - f_j.$$

- These are called *forward differences*, since points forward of the current value are used (there are also *backward differences* and *central differences*, which we shall meet in our discussion of the solution of differential equations).
- We consider the composition of the operator whereby

$$\begin{aligned}\Delta^2 f_0 = \Delta(\Delta f_0) = \Delta(f_1 - f_0) &= \Delta f_1 - \Delta f_0 \\ &= (f_2 - f_1) - (f_1 - f_0) \\ &= f_2 - 2f_1 + f_0.\end{aligned}$$



Newton Forward Differences Cont.

Of course we can now proceed to define $\Delta^n f_0$ *in an* ***iterative manner***.

- We introduce the polynomial

$$f(x) = f_0 + (x - x_0)\frac{\Delta f_0}{h} + \frac{(x - x_0)(x - x_1)}{2!}\frac{\Delta^2 f_0}{h^2} + \cdots \quad ,$$

- which will ultimately terminate after *N terms. Alternatively this can be written* as:

$$f(x) = f_0 + \sum_{n=1}^{N-1} \frac{\Delta^n f_0}{h^n n!} \prod_{j=0}^{n-1} (x - x_j). \qquad \textbf{(5.1)}$$

- We consider the data values to be ***equally spaced***, so the value of Δx_j is $h\ \forall j$.
- This analysis can be extended to irregularly spaced points, but we shall not attempt that here.

Newton Forward Differences Cont.

We are now able to construct the polynomial (5.1) for a set of points.

Example 5.1 Let us consider the points (0, 1), (1, 3) and (2, 4).

- Here we have three points and as such we would expect to obtain a quadratic.
- We use a tabular form, which gives:

x	f	Δf	$\Delta^2 f$
0	1	2	-1
1	3	1	
2	4		

$$f(x) = f_0 + (x - x_0)\frac{\Delta f_0}{h} + \frac{(x - x_0)(x - x_1)}{2!}\frac{\Delta^2 f_0}{h^2} + \cdots \quad ,$$

Thus we have

$$f(x) = 1 + (x - 0)\frac{2}{1} + \frac{(x - 0)(x - 1)}{2!}\frac{(-1)}{1^2},$$

$$f(x) = 1 + 2x - \frac{x(x - 1)}{2},$$

where $h = 1$, $f_0 = 1$, $\Delta f_0 = 2$ *and* $\Delta^2 f_0 = -1$ *(reading from the top row of the table).*

Newton forward Methods

Forward difference table

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$
x_0	y_0					
		Δy_0				
x_1 $(= x_0 + h)$	y_1		$\Delta^2 y_0$			
		Δy_1		$\Delta^3 y_0$		
x_2 $(= x_0 + 2h)$	y_2		$\Delta^2 y_1$		$\Delta^4 y_0$	
		Δy_2		$\Delta^3 y_1$		$\Delta^5 y_0$
x_3 $= (x_0 + 3h)$	y_3		$\Delta^2 y_2$		$\Delta^4 y_1$	
		Δy_3		$\Delta^3 y_2$		
x_4 $= (x_0 + 4h)$	y_4		$\Delta^2 y_3$			
		Δy_4				
x_5 $= (x_0 + 5h)$	y_5					

Forward Differences: The differences $y_1 - y_0$, $y_2 - y_1$, $y_3 - y_2$, ……, $y_n - y_{n-1}$ when denoted by dy_0, dy_1, dy_2, ……, dy_{n-1} are respectively, called the first forward differences. Thus, the first forward differences are:

$$\boldsymbol{\Delta y_i = y_{i+1} - y_i}$$

These differences are of the first order, the effect of advanced differences of the second order can be defined as:

$$\Delta^2 \boldsymbol{y_i} = \Delta(\Delta \boldsymbol{y_i})$$

So that: $\Delta^2 yi = \Delta(\boldsymbol{y_{i+1}} - \boldsymbol{y_i}) = \Delta \boldsymbol{y_{i+1}} - \Delta \boldsymbol{y_i}$

if we have xi points on equal spaces where; $h = x_{i+1} - x_i$

So that the next polynomial :

$$P(x) = y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{\Delta^2 y_0}{2!\,h^2}(x - x_0)(x - x_1) + \dots + \frac{\Delta^n y_0}{n!\,h^n}(x - x_0)(x - x_1)\dots(x - x_{n-1})$$

satisfy the following $P(x_0) = y_0,\ P(x_1) = y_1\ \dots\ P(x_n) = y_n$ and the polynomial will pass through the points

$(x_i, y_i)\ ;\ i = 0,1,2,\dots\dots n$ which means its satisfy the characteristics of the interpolation

Newton forward Methods

Example: find the cubic polynomial for the next points:

x	0	1	2	3
y	3	3	7	21

x starts from zero and the steps between x values are constants $h=1$, so that, we can use Newton forward method to find the polynomial.

i	x_i	y_i	Δyi	$\Delta^2 y_i$	$\Delta^3 y_i$
0	**0**	**3**	**0**	**4**	**6**
1	1	3	4	10	-
2	2	7	14	-	-
3	3	21	-	-	-

So that:

$$P(x) = y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{\Delta^2 y_0}{2!\,h^2}(x - x_0)(x - x_1) + \dots + \frac{\Delta^n y_0}{n!\,h^n}(x - x_0)(x - x_1)\dots(x - x_{n-1})$$

$$P(x) = y_0 + \Delta y_0(x - x_0) + \frac{\Delta^2 y_0}{2!}(x - x_0)(x - x_1) + \frac{\Delta^3 y_0}{3!}(x - x_0)(x - x_1)(x - x_2)$$

$$P(x) = 3 + \frac{4}{2!}x(x-1) + \frac{6}{3!}x(x-1)(x-2)$$

$$P(x) = 3 + 2x(x-1) + x(x-1)(x-2)$$

Newton backward Methods

Backward Differences: The differences $y_1 - y_0$, $y_2 - y_1$, ……, $y_n - y_{n-1}$ when denoted by dy_1, dy_2, ……, d_{yn}, respectively, are called first backward difference. Thus, the first backward differences are :

$$\nabla y i = y_i - y_{i-1}$$

The Newton forward method is used when the required values (x, y) are located at the beginning of the table of differences,
but when the required points are located at the end of the table of differences, a polynomial of degree n is used called Newton's backward equation as follows:

Backward difference table

x	y	∇y	$\nabla^2 y$	$\nabla^3 y$	$\nabla^4 y$	$\nabla^5 y$
x_0	y_0					
		∇y_1				
x_1 $(= x_0 + h)$	y_1		$\nabla^2 y_2$			
		∇y_2		$\nabla^3 y_3$		
x_2 $(= x_0 + 2h)$	y_2		$\nabla^2 y_3$		$\nabla^4 y_4$	
		∇y_3		$\nabla^3 y_4$		$\nabla^5 y_5$
x_3 $(= x_0 + 3h)$	y_3		$\nabla^2 y_4$		$\nabla^4 y_5$	
		∇y_4		$\nabla^3 y_5$		
x_4 $(= x_0 + 4h)$	y_4		$\nabla^2 y_5$			
		∇y_5				
x_5 $(= x_0 + 5h)$	y_5					

$$P(x) = y_n + \frac{\nabla y_n}{h}(x - x_n) + \frac{\nabla^2 y_n}{2!\,h^2}(x - x_n)(x - x_{n-1}) + \ldots + \frac{\nabla^n y_n}{n!\,h^n}(x - x_n)(x - x_{n-1}) \ldots (x - x_1)$$

Newton backward Methods

Example: In the next table for the function e^x use Newton backward method of interpolating to calculate $e^{2.00}$:

x	0.1	0.6	1.1	1.6	2.1
$y = e^x$	1.1052	1.8221	3.0042	4.953	8.1662

Where we have x=2.0, x_n=2.1, h=0.5

i	x_i	$y_i = e^2$	∇y_i	$\nabla^2 y_i$	$\nabla^3 y_i$	$\nabla^4 y_i$
0	0.1	1.1052				
1	0.6	1.8221	0.7169			
2	1.1	3.0042	1.1821	0.4652		
3	1.6	4.9530	1.9488	0.7667	0.3015	
4	**2.1**	**8.1662**	**3.2132**	**1.2644**	**0.4977**	**0.1962**

$$P(x) = y_n + \frac{\nabla y_n}{h}(x - x_n) + \frac{\nabla^2 y_n}{2!\,h^2}(x - x_n)(x - x_{n-1}) + \ldots + \frac{\nabla^n y_n}{n!\,h^n}(x - x_n)(x - x_{n-1}) \ldots (x - x_1)$$

$e^{2.0} =$

$$e^{2.0} = 8.1662 - 2 * 3.2132\,(0.1) - 2 * 1.2644(0.1)(0.4) + \frac{4 * 0.4977}{3}(0.1)(0.4)(0.9) + \frac{2 * 0.1962}{3}(0.1)(0.4)(0.9)(1.4\Big)$$

$$e^{2.0} = 7.4224 + 0.0239 + 0.0066$$

$\therefore e^{2.0} \cong 7.4529$ (to 4 dp) The correct value for $e^2 = 7.3891$ to 4 dp

Newton forward and backward Methods

The following table shows the completion path for the advanced and lagging differences in the table of differences for the fifth degree equation n = 5.

x_{-5}	y_{-5}						
x_{-4}	y_{-4}						
x_{-3}	y_{-3}						Backward
						$\nabla^5 y_0$	
x_{-2}	y_{-2}				$\nabla^4 y_0$		
				$\nabla^3 y_0$			
x_{-1}	y_{-1}		$\nabla^2 y_0$				
		∇y_0					
x_0	y_0						
		Δy_0					
x_1	y_1		$\Delta^2 y_0$				
				$\Delta^3 y_0$			
x_2	y_2				$\Delta^4 y_0$		
						$\Delta^5 y_0$	
x_3	y_3						Forward
x_4	y_4						
x_5	y_5						

Any Question?