Numerical Methods ITGS219

Lecture: Interpolation and Extrapolation

Lagrange Polynomials

By: Zahra A. Elashaal

Lagrange Polynomials

- We remark that various conclusions can be drawn from the data by using the forward differences;
- We now construct a polynomial which goes through a set of points which **are not** necessarily **evenly spaced**. Let us consider the polynomial

$$f(z) = \frac{(z-x_2)(z-x_3)(z-x_4)}{(x_1-x_2)(x_1-x_3)(x_1-x_4)}f_1 + \frac{(z-x_1)(z-x_3)(z-x_4)}{(x_2-x_1)(x_2-x_3)(x_2-x_4)}f_2 + \frac{(z-x_1)(z-x_2)(z-x_4)}{(x_3-x_1)(x_3-x_2)(x_3-x_4)}f_3 + \frac{(z-x_1)(z-x_2)(z-x_3)}{(x_4-x_1)(x_4-x_2)(x_4-x_3)}f_4$$

- It is worth pausing at this point and checking that this curve goes through each of the points (x_j, f_j).
- For example for $j = 3$, we set $z = x_3$ and only the third term is non-zero and we have

$$f(x_3) = \frac{(x_3-x_1)(x_3-x_2)(x_3-x_4)}{(x_3-x_1)(x_3-x_2)(x_3-x_4)}f_3 = 1 \times f_3 = f_3.$$

- Hence the value of the polynomial at $z = x_3$ is f_3 (as we would hope).
- This is an example of a **Lagrange polynomial;** which could have equally been written as

$$f(z) = \sum_{i=1}^{4} f_i \prod_{\substack{j=1 \\ j\neq i}}^{4} \frac{z-x_j}{x_i-x_j}$$

Lagrange Polynomials ... Cont.

Example 1: The accompanying table gives the velocity, of a moving body, at various times. Estimate the velocity at $t = 7$ s.

Time, t, s	1	2	3	8
Velocity, v, m/s	2	4.1	6.4	36.5

$$f(z) = \sum_{i=1}^{4} f_i \prod_{\substack{j=1 \\ j\neq i}}^{4} \frac{z - x_j}{x_i - x_j}.$$

Solution:

Since h is different, we use Lagrange interpolation polynomial.

$$v(t) = \frac{(t-t_1)(t-t_2)\dots(t-t_n)}{(t_o-t_1)(t_o-t_2)\dots(t_o-t_n)} v(t_o) + \frac{(t-t_o)(t-t_2)\dots(t-t_n)}{(t_1-t_o)(t_1-t_2)\dots(t_1-t_n)} v(t_1) + \dots$$

$$v(t) = \frac{(t-2)(t-3)(t-8)}{(1-2)(1-3)(1-8)}(2) + \frac{(t-1)(t-3)(t-8)}{(2-1)(2-3)(2-8)}(4.1) + \frac{(t-1)(t-2)(t-8)}{(3-1)(3-2)(3-8)}(6.4) + \\ + \frac{(t-1)(t-2)(t-3)}{(8-1)(8-2)(8-3)}(36.5).$$

$$v(7) = \frac{(7-2)(7-3)(7-8)}{(1-2)(1-3)(1-8)}(2) + \frac{(7-1)(7-3)(7-8)}{(2-1)(2-3)(2-8)}(4.1) + \\ + \frac{(7-1)(7-2)(7-8)}{(3-1)(3-2)(3-8)}(6.4) + \frac{(7-1)(7-2)(7-3)}{(8-1)(8-2)(8-3)}(36.5) = 26.5 \text{ m/s}.$$

Lagrange Polynomials ... Cont.

Example 2: The ratio of the work done in a project, as a function of time, is found as below. Estimate this ratio at $t = 2$ month.

Time, t, (month)	3	4	5
Work, W, (%)	5	14	37

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

Solution: Since $h = 1 \Rightarrow$ We can use the particular Gregory-Newton interpolation formula directly without rescaling.

$t_o \neq 0 \Rightarrow$ Shifting is required.

t	$t_{shifted}$	W	ΔW	$\Delta^2 W$
3	0	5	9	14
4	1	14	23	
5	2	37		

$$W(t) = W(0) + t\Delta W_o + \frac{t(t-1)}{2!}\Delta^2 W_o + \dots\dots \Rightarrow W(t) = 5 + t.(9) + \frac{t(t-1)}{2!}.(14)$$

$\therefore \quad W(t) = 5 + 9t + 7t(t-1).$

At $t_{old} = 2 \Rightarrow t_{new} = 2 - 3 = -1,$

$W(t_{new}) = 5 + 9(-1) + 7[-1(-1-1)] = 10\%$ Not O.k. .

Lagrange Polynomials ... Cont.

Example 2: The ratio of the work done in a project, as a function of time, is found as below. Estimate this ratio at $t = 2$ month.

Time, t, (month)	3	4	5
Work, W, (%)	5	14	37

Solution cont.:

$$W(t_{new}) = 5 + 9(-1) + 7[-1(-1-1)] = 10\% \qquad \text{Not O.k. .}$$

If a function cannot be well approximated by a polynomial, a useful device can be adopted by plotting a (log – log) graph. This reduces a large variety of functions to essentially straight lines or to smooth curves which are easy to interpolate.

$\therefore$ Use a (log – log) graph ,

$t^* = \ln t$	1.099	1.386	1.609
$W^* = \ln W$	1.609	2.639	3.611

Now, since h is different, we use Lagrange interpolation polynomial.

$$W^*(t^*) = \frac{(t^* - t^*_1)(t^* - t^*_2)\ldots(t^* - t^*_n)}{(t^*_o - t^*_1)(t^*_o - t^*_2)\ldots(t^*_o - t^*_n)} W^*(t^*_o) + \frac{(t^* - t^*_o)(t^* - t^*_2)\ldots(t^* - t^*_n)}{(t^*_1 - t^*_o)(t^*_1 - t^*_2)\ldots(t^*_1 - t^*_n)} W^*(t^*_1) + \ldots$$

Lagrange Polynomials ... Cont.

Example 2: The ratio of the work done in a project, as a function of time, is found as below. Estimate this ratio at $t = 2$ month.

Time, t, (month)	3	4	5
Work, W, (%)	5	14	37

$t^* = \ln t$	1.099	1.386	1.609
$W^* = \ln W$	1.609	2.639	3.611

Solution cont.:

Now, since h is different, we use Lagrange interpolation polynomial.

$$W^*(t^*) = \frac{(t^* - t^*_1)(t^* - t^*_2)\ldots(t^* - t^*_n)}{(t^*_o - t^*_1)(t^*_o - t^*_2)\ldots(t^*_o - t^*_n)} W^*(t^*_o) + \frac{(t^* - t^*_o)(t^* - t^*_2)\ldots(t^* - t^*_n)}{(t^*_1 - t^*_o)(t^*_1 - t^*_2)\ldots(t^*_1 - t^*_n)} W^*(t^*_1) + \ldots$$

$$W^*(t^*) = \frac{(t^* - 1.386)(t^* - 1.609)}{(1.099 - 1.386)(1.099 - 1.609)}(1.609) + \frac{(t^* - 1.099)(t^* - 1.609)}{(1.386 - 1.099)(1.386 - 1.609)}(2.639) + \frac{(t^* - 1.099)(t^* - 1.386)}{(1.609 - 1.099)(1.609 - 1.386)}(3.611).$$

At $t = 2 \quad \Rightarrow \quad t^* = \ln 2 = 0.693,$

$$W^*(t^*) = \frac{(0.693 - 1.386)(0.693 - 1.609)}{(1.099 - 1.386)(1.099 - 1.609)}(1.609) + \frac{(0.693 - 1.099)(0.693 - 1.609)}{(1.386 - 1.099)(1.386 - 1.609)}(2.639) + \frac{(0.693 - 1.099)(0.693 - 1.386)}{(1.609 - 1.099)(1.609 - 1.386)}(3.611) = 0.576664.$$

But $W^* = \ln W \Rightarrow W = e^{W^*} = e^{0.576664} = 1.78 \qquad \therefore \; W(2) = 1.78\%$

Newton Forward Differences and Lagrange Polynomials ... Cont.

This is a convenient way to write out the cubic we require (as it is relatively easily extended to higher-order cases) and in order to evaluate it we can use

- the MATLAB code:

$$f(z) = \sum_{i=1}^{4} f_i \prod_{\substack{j=1\\ j\neq i}}^{4} \frac{z - x_j}{x_i - x_j}.$$

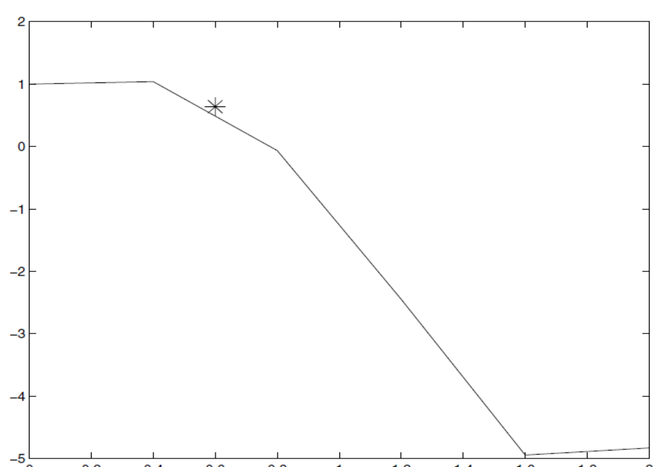
```
ip = 1:4;
f_z = 0.0;
for i = 1:4
   k = find(ip ~= i);
   prod = f(ip(i));
   for j = k
      prod = prod * (z-x(ip(j))) / (x(ip(i))-x(ip(j)));
   end
   f_z = f_z + prod;
end
```

This code has been written in this way so it could be extended to as many points as we want.

For *z = 0.6 we find a value of f_z=0.6386, which is shown on the figure as* an asterisk:

Newton Forward Differences and Lagrange Polynomials ... Cont.

- For *z = 0.6 we find a value of f_z=0.6386, which is shown on the figure as* an asterisk:



Where the file called **data.dat** contains:

x	*f(x)*
0.00	1.00000000
0.40	1.03936428
0.80	-0.06498473
1.20	-2.44823335
1.60	-4.94458639
2.00	-4.82980938

Lagrange Polynomials ... Cont.

- As mentioned we can write this expression as a summation of products and this is easily extended to *N points as*

$$f(z) \approx \sum_{i=1}^{N} f_i \prod_{\substack{j=1 \\ j \neq i}}^{N} \frac{z - x_j}{x_i - x_j}.$$

- Consider the code:

```
function [value] = poly_int(z, N)
global x f
imax = length(x);
if mod(N, 2) ~= 0
    disp(' N should be even ')
    break
elseif N >= imax;
     disp('Too many points used')
      break
end
M = N/2;
[ibottom, itop] = findrange(x, z, N);
ip = ibottom : itop;
il = 1:N;
f_z = 0.0;
for ii = 1:N
     k = find(il ~= ii);
     prod = f(ip(ii));
     for j = k
      prod = prod * (z-x(ip(j))) / (x(ip(ii))-x(ip(j)));
     end
    f _z = f_z + prod;
end
value = f_z;
```

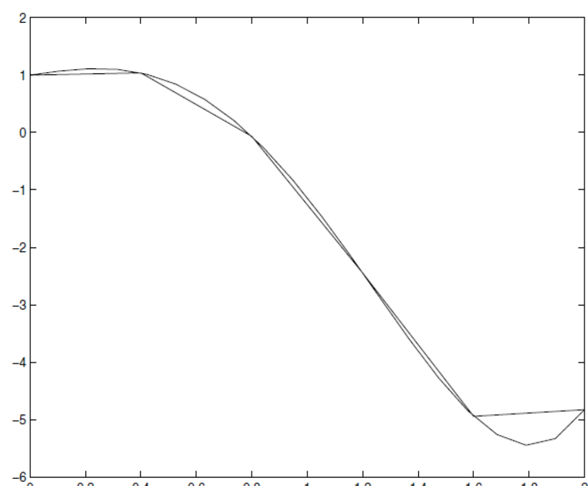
Newton Forward Differences and Lagrange Polynomials ... Cont.

- And now test the code using

```
global x f
load data.dat
x = data(:, 1);
f = data(:, 2);
N = length(x);
xmin = x(1);  xmax = x(N);
xtest = linspace(xmin, xmax, 20);
for ii = 1:20
    [ftest(ii)] = poly_int(xtest(ii), N);
end
plot(x, f, 'r', xtest, ftest, 'b')
```

This gives the next plot

As you can see using **six points** works quite well at fitting the data. Although as we will see in subsequent examples using high-order polynomials can lead to *significant errors*.

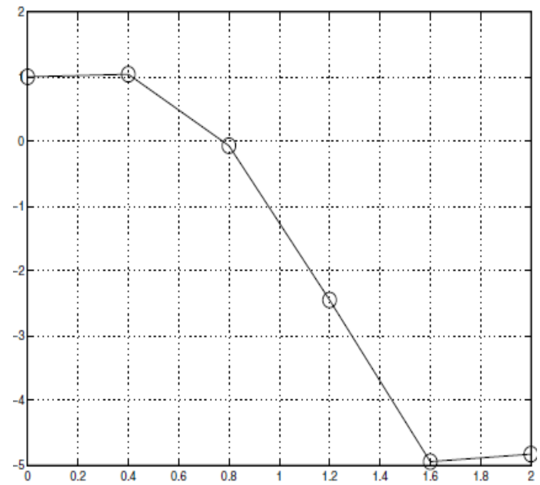


5.4.1 Linear Interpolation/Extrapolation

We consider the simplest case wherein we have two points (x1, y1) and (x2, y2). The straight line through these is given by

$$y = \frac{x - x_2}{x_1 - x_2} y_1 + \frac{x - x_1}{x_2 - x_1} y_2,$$

which here we have constructed in a Lagrange polynomial style. We note that the answer is independent of the process here and provided **x1 ≠ x2** we will always get a straight line. This can be continued for **quadratics** and **higher order** functions. This is a plot of the data we shall use for this discussion:



5.4.1 Linear Interpolation/Extrapolation

This plot was obtained using the commands:

```
clear all
load 'data.dat'
plot(data(:,1),data(:,2),'o','MarkerSize',12)
hold on
plot(data(:,1),data(:,2))
hold off
grid on
print -dps2 data.ps
```

Final command print the results to a Postscript file so that it can be included in another document (for instance this text) or sent to a printer. There are many options for this command, for instance we could use

print -djpeg90 data.jpg   to generate a JPEG file.

As mentioned above for convenience we can extract the data from the array data using:

```
x = data(:,1);
f = data(:,2);
clear data
```

Where the file called **data.dat** 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

5.5 Calculating Interpolated and Extrapolated Values

How Matlab can be used to determine the interpolating polynomial for a set of points?

We shall presume that we have the requisite number of points to perform this operation, that is ***two points*** for a ***line***, ***three*** for a ***quadratic*** and ***four*** for a ***cubic***, etc.

We shall make use of the command ***polyfit***. The syntax for this command is ***polyfit(x, y, N),*** where the points are defined in *x* and *y*, and *N* is the order of the interpolating polynomial.

This gives the curve we want provide the number of points represented in x, y is ***N+1***.

Let us consider the interpolation of data points using a **straight line**.

Example 5.2 We seek to find the value of the function at x = 4.5 where the data points are (1,−3), (3, 4), (5, 5), (7,−8), (9,−3) and (11, 0), using linear interpolation.

```
x = [1  3  5  7  9 11];
y = [-3 4  5 -8 -3  0];
xi = 4.5;
r = 2:3; % xi lies between the 2nd and 3rd point of x.
p = polyfit(x(r), y(r), 1);
yi = polyval(p, xi)
```

This gives p=[0.5000 2.5000] (representing the line ***y = x/2 + 5/2***) and y_i=4.75. We note that here we have determined *the range r* by hand but we could employ the function *findrange*

5.5 Calculating Interpolated and Extrapolated Values

Example 5.3 Using the data in the previous example now calculate the value of the interpolating polynomial at x = 4.5 using cubic interpolation.

```
x = [1 3 5 7 9 11];
y = [-3 4 5 -8 -3 0];
xi = 4.5;
[ibot, itop] = findrange(x,y,3);
r = ibot:itop;
p = polyfit(x(r), y(r), 3);
yi = polyval(p,xi)
```

We note that this is quite different to the answer given by linear interpolation and one might argue that linear interpolation is better here. This of course depends on the underlying function.

This gives *p=[-0.1667 0.75   2.667 -6.25]* and $y_i=5.75$.

Any Question?