Newton Form of the Interpolation Polynomial – MATH 4650
Fall 2006

Kawai

(#1) The name of my procedure is `Coef`.

You should create your horizontal arrays, x and y:

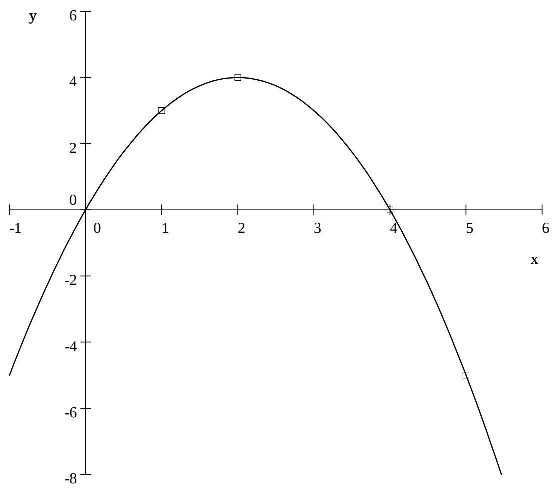
`x = [1 2 4 5]`

`y = [3 4 0 -5]`

(#2) Notice what I have done.

I generated these values from the parabola

$$y = f(x) = -x^2 + 4x.$$



x	y
1	3
2	4
4	0
5	-5

(#3) Since I have 4 data points, I will obtain a polynomial of degree $n = 3$.

This is one less than the number of data points.

In the Command Window, we type:

`a = Coef(3, x, y)`

This will return a horizontal array with $(n+1)$ elements and copy it into the matrix variable "a".

Please read the MATLAB code. Note that we must make some important adjustments to the array indices because MATLAB does not allow non-positive indices!

(#4) The output is:

`a = [3 1 -1 0]`

The Newton form of the interpolation polynomial must be:

$$\begin{aligned} f(x) &= 3 + 1(x-1) + (-1)(x-1)(x-2) + 0(x-1)(x-2)(x-4) \\ &= -x^2 + 4x. \checkmark \end{aligned}$$

Since I took 4 points from the quadratic function, I expected the last coefficient to be zero. I was not disappointed.

(#5) Now we can use the other function, `Eval`.

Assuming that we have already run `Coef`,

we can now evaluate the interpolation polynomial at particular value of t.

In the Command Window, we type:

```
Eval(3, x, a, 3)
```

The first number is $n = 3$ since the degree of the polynomial is 3.

We offer it the same horizontal arrays, x and a, from the previous step.

The last input is the value of t.

The output is:

```
The interpolated value at t = 3.0000 is 3.0000000.
```

(#6) Remember that you can call `Coef` and `Eval` from another program to create new and interesting interpolation polynomials.