LAGRANGE INTERPOLATION IN MATLAB

Introduction

We are given a set of k points of the form (x_i, y_i) and we'll want to fit a curve through these points.

We will find the lowest degree polynomial that will best fit to these points.

There are two ways of obtaining the lagrange polynomials:

Finding the Lagrangian Interpolate

Method No. 1

$$l_i(x) = \prod_{j=0, j\neq 0}^{n} \frac{(x - x_j)}{(x_i - x_j)}$$

$$L(x) = \sum_{i=0}^{n} y_i l_i(x)$$

see https://www.youtube.com/watch?app=desktop&v=bzp_q7NDdd4&ab_channel=Dr.WillWood for a good explanation

Method No. 2

$$L(x) = \sum_{i=0}^{n} a_i \cdot x^i$$

$$y_0 = \sum_{i=0}^{n} a_i \cdot x_0^i$$

$$y_0 = \sum_{i=0}^{n} a_i \cdot x_1^i$$

$$y_0 = \sum_{i=0}^{n} a_i \cdot x_2^i$$

.

.

.

$$y_0 = \sum_{i=0}^{n} a_i \cdot x_n^i$$

We obtain a coeff' matrix of the form shown below:

$$A = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}$$

We then populate M and Y:

$$M = \begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & x_1^3 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & x_2^3 & \cdots & x_2^n \\ \vdots & \vdots & & & & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & \cdots & x_n^n \end{bmatrix} Y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

We then calculate the values of the coeff's using the formula $A = M^{-1} \cdot Y$ and write L(x).

see https://people.clas.ufl.edu/kees/files/LagrangePolynomials.pdf

for a good explanation

The link between Method No. 1 and Method No. 2 is the Vandermonde determinant:

The main property of a square Vandermonde matrix

$$V = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \ldots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^{n-1} \end{bmatrix}$$

is that its determinant has the simple form

$$\det(V) = \prod_{1<i<j<n} (x_j - x_i).$$

[via en.wikipedia.org]

The MATLAB Code for determining the coeff's

```
%INPUT OF DATA POINTS
n = input("Number of points known:");
```

```
for i = 1:n
    x(i) = input("value of x:");
    y(i) = input("value of y:");
end

%CREATE FREE TERM MATRIX
for i = 1:n
    Y(i,1) = y(i);
end

%CREATE VANDERMONDE MATRIX

for i = 1:n
    for j = 1:n
        M(i,j) = x(i) .^ (j - 1);
    end
end

%DETERMINE INVERSE OF VANDERMONDE MATRIX
Minv = inv(M);

%DETERMINE COEFF'S MATRIX
A = Minv * Y;
```