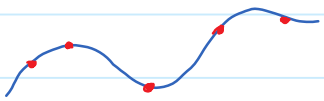
Lecture 18

Wednesday, February 19, 2020

Given n data points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where $x_1, x_2, \ldots, x_n$ are distinct, there is only one polynomial of degree $\leq n-1$ that fits these points. This observation can be proved as follows.



Let P and Q be two polynomial of degree $\leq n-1$ that fit the data points. We want to show that $P(x) = Q(x)$ for all $x \in \mathbb{R}$.

Put $R(x) = P(x) - Q(x)$. We see that the degree of R is at most $n-1$. However, R has n distinct roots $x_1, x_2, \ldots, x_n$. This is possible only if R is the constant zero function. Therefore, $P(x) - Q(x) = 0$ for all $x \in \mathbb{R}$.

Although the interpolation polynomial of $(x_1, y_1), \ldots, (x_n, y_n)$ is unique, there are many ways to find it. The most natural method is to solve a system of n equations

$$\begin{cases} P(x_1) = y_1, \\ P(x_2) = y_2, \\ \cdots \\ P(x_n) = y_n \end{cases}$$

for n unknowns (the coefficients of P). This method, however, is not computationally effective. Lagrange's formula is an alternative. We need to form n basis polynomials $L_1, L_2, \ldots, L_n$ such that

$$L_i(x) = \begin{cases} 1 & \text{if } x = x_i \\ 0 & \text{if } x = x_j \quad (j \neq i) \end{cases}$$

The formula of L_i is

$$L_i(x) = \frac{(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_1)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}$$

Then P is obtained as a linear combination of $L_1, L_2, \ldots, L_n$.

$$P(x) = y_1 L_1(x) + y_2 L_2(x) + \cdots + y_n L_n(x)$$

Ex: Find a polynomial that fits the points $(1,1), (2,1), (3,2), (0,-1)$.

In this problem, $x_1 = 1, y_1 = 1$,

$x_2 = 2, y_2 = 1$,

$x_3 = 3, y_3 = 2$,

$x_4 = 0, y_4 = -1$.

Then
$$L_1(x) = \frac{(x-2)(x-3)(x-0)}{(1-2)(1-3)(1-0)}$$

$$L_2(x) = \frac{(x-1)(x-3)(x-0)}{(2-1)(2-3)(2-0)}$$

$$L_3(x) = \frac{(x-1)(x-2)(x-0)}{(3-1)(3-2)(3-0)}$$

$$L_4(x) = \frac{(x-1)(x-2)(x-3)}{(0-1)(0-2)(0-3)}$$

Then $P(x) = L_1(x) + L_2(x) + 2L_3(x) - L_4(x) = \frac{1}{2}x^3 - \frac{5}{2}x^2 + 4x - 1.$

* How to write a program on Matlab that does the following?

- Input: array $x = [x_1\ x_2 \ldots x_n]$,
  $y = [y_1\ y_2 \ldots y_n]$.
- Output: the interpolation polynomial $P(t)$ that fits the points $(x_1, y_1), \ldots, (x_n, y_n)$.

We declare 't' as a symbolic variable by writing

```
syms t
```

We need a 'for' loop to compute each Lagrange basis polynomial $L_1, L_2, \ldots, L_n$. Recall

$$L_i(t) = \frac{(t-x_1)\ldots(t-x_{i-1})(t-x_{i+1})\ldots(t-x_n)}{(x_i-x_1)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}$$

What is annoying about constructing L_i is the skipping of x_i. We can avoid the skipping by re-indexing $x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$. We index them as $z_1, z_2, \ldots, z_{n-1}$. This is done in Matlab by

```
z = x;
z(i) = [],        (removing the i'th entry from z).
```

After the re-indexing, we can rewrite L_i as

$$L_i(t) = \frac{(t-z_1)\cdots(t-z_{n-1})}{(x_i-z_1)\cdots(x_i-z_{n-1})} = \frac{t-z_1}{x_i-z_1}\cdots\frac{t-z_{n-1}}{x_i-z_{n-1}}.$$

Then we can easily construct L_i using a 'for' loop.

```
for j = 1:n-1
    L_i = L_i * (t - z(j))/(x(i) - z(j));
end
```

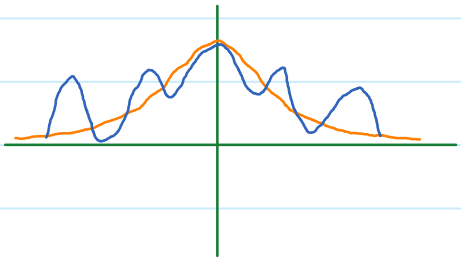
See the full code on the course website.

Ex: On the graph of the sine function, let us pick 11 points with x-cordinates equal to $0, 1, 2, \ldots, 10$. In this case,

```
x = 0:1:10;
y = sin(x);
```

We can plot the polynomial that fits these 11 points on the same graph as the sine function. We see that the polynomial approximates quite well the sine function.

With functions such that $\frac{1}{1+5x^2}$, the approximation is not very well. There are large oscilations near the endpoints. This is known as Runge phenomenon. We will see later (when analysizing the error of approximation by interpolations) that the reason for this phenomenon is that the higher dericvatives of $\frac{1}{1+5x^2}$ is large.



Runge phenomenon doesn't manifest with the sine function because

any higher derivatives of $f(x) = \sin x$ is still bounded between -1 and 1.