Lagrange & Newton interpolation

In this section, we shall study the polynomial interpolation in the form of Lagrange and Newton. Given a sequence of $(n+1)$ data points and a function f, the aim is to determine an n-th degree polynomial which interpolates f at these points. We shall resort to the notion of divided differences.

Interpolation

Given $(n+1)$ points $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$, the points defined by $(x_i)_{0\le i\le n}$ are called **points of interpolation**. The points defined by $(y_i)_{0\le i\le n}$ are the **values of interpolation**. To interpolate a function f, the values of interpolation are defined as follows:

$$y_i = f(x_i), \quad \forall i = 0, \ldots, n.$$

Lagrange interpolation polynomial

The purpose here is to determine the unique polynomial of degree n, P_n which verifies

$$P_n(x_i) = f(x_i), \quad \forall i = 0, \ldots, n.$$

The polynomial which meets this equality is Lagrange interpolation polynomial

$$P_n(x)=\sum_{j=0}^{n} l_j(x) f(x_j)$$

where the l_j 's are polynomials of degree n forming a basis of P_n

$$l_j(x)=\prod_{i=0, i\neq j}^{n} \frac{x-x_i}{x_j-x_i}=\frac{x-x_0}{x_j-x_0}\cdots\frac{x-x_{j-1}}{x_j-x_{j-1}}\,\frac{x-x_{j+1}}{x_j-x_{j+1}}\cdots\frac{x-x_n}{x_j-x_n}$$

Properties of Lagrange interpolation polynomial and Lagrange basis

They are the l_j polynomials which verify the following property:

$$l_j(x_i)=\delta_{ji}=\begin{cases}1 & i=j\\ 0 & i\neq j\end{cases}, \quad \forall i=0,\ldots,n.$$

They form a basis of the vector space P_n of polynomials of degree at most equal to n

$$\sum_{j=0}^{n} \alpha_j l_j(x)=0$$

By setting: $x = x_i$, we obtain:

$$\sum_{j=\cdot}^{n} \alpha_j l_j(x_i)=\sum_{j=\cdot}^{n} \alpha_j \delta_{ji}=\cdot \quad \Rightarrow \alpha_i=\cdot$$

The set $(l_j)_{0\le j\le n}$ is linearly independent and consists of $n+1$ vectors. It is thus a basis of P_n. Finally, we can easily see that:

$$P_n(x_i)=\sum_{j=\cdot}^{n} l_j(x_i) f(x_i)=\sum_{j=\cdot}^{n} \delta_{ji} f(x_i)=f(x_i)$$

Example: computing Lagrange interpolation polynomials

Given a set of three data points {(0, 1), (2, 5), (4, 17)}, we shall determine the Lagrange interpolation polynomial of degree 2 which passes through these points.

First, we compute l_0, l_1 and l_2:

$$l_0(x)=\frac{(x-2)(x-4)}{8},\ l_1(x)=-\frac{x(x-4)}{4},\ l_2(x)=\frac{x(x-2)}{8}$$

Lagrange interpolation polynomial is:

$$P_n = l_0(x) + 5l_1(x) + 17l_2(x) = 1 + x^2$$

Scilab: computing Lagrange interpolation polynomial

The Scilab function `lagrange.sci` determines Lagrange interpolation polynomial. X encompasses the points of interpolation and Y the values of interpolation. P is the Lagrange interpolation polynomial.

lagrange.sci

```
function[P]=lagrange(X,Y) //X nodes,Y values;P is the numerical Lagrange
polynomial interpolation
n=length(X); // n is the number of nodes. (n-1) is the degree
x=poly(0,"x");P=0;
for i=1:n, L=1;
  for j=[1:i-1,i+1:n] L=L*(x-X(j))/(X(i)-X(j));end
  P=P+L*Y(i);
end
endfunction

-->X=[0;2;4]; Y=[1;5;17]; P=lagrange(X,Y)
 P = 1 + x^2
```

Such polynomials are not convenient, since numerically, it is difficult to deduce l_{j+1} from l_j. For this reason, we introduce Newton's interpolation polynomial.

Newton's interpolation polynomial and Newton's basis properties

The polynomials of Newton's basis, e_j, are defined by:

$$e_j(x)=\prod_{i=0}^{j-1}(x-x_i)=(x-x_0)(x-x_1)\cdots(x-x_{j-1}),\quad j=1,\dots,n.$$

with the following convention:

$$e_0=1$$

Moreover

$$\begin{aligned}
e_1 &= (x - x_0)\\
e_2 &= (x - x_0)(x - x_1)\\
e_3 &= (x - x_0)(x - x_1)(x - x_2)\\
&\vdots\\
e_n &= (x - x_0)(x - x_1)\cdots(x - x_{n-1})
\end{aligned}$$

The set of polynomials $(e_j)_{0\le j\le n}$ (Newton's basis) are a basis of P_n, the space of polynomials of degree at most equal to n. Indeed, they constitute an echelon-degree set of $(n + 1)$ polynomials.

Newton's interpolation polynomial of degree n related to the subdivision $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$ is:

$$P_n(x) = \sum_{j=0}^{n} \alpha_j e_j(x) = \alpha_0 + \alpha_1(x - x_0) + \alpha_2(x - x_0)(x - x_1) + \ldots + \alpha_n(x - x_0)(x - x_1)\cdots(x - x_{n-1})$$

where

$$P_n(x_i) = f(x_i), \quad \forall i = 0, \ldots, n.$$

We shall see how to determine the coefficients $(\alpha_j)_{0 \le j \le n}$ in the following section entitled the **divided differences**.

Divided differences

Newton's interpolation polynomial of degree n, $P_n(x)$, evaluated at x_0, gives:

$$P_n(x_0) = \sum_{j=0}^{n} \alpha_j e_j(x_0) = \alpha_0 = f(x_0) = f[x_0]$$

Generally speaking, we write:

$$f[x_i] = f(x_i), \quad \forall i = 0, \ldots, n$$

$f[x_0]$ is called a zero-order **divided difference**.

Newton's interpolation polynomial of degree n, $P_n(x)$, evaluated at x_1, gives:

$$P_n(x_1) = \sum_{j=0}^{n} \alpha_j e_j(x_1) = \alpha_0 + \alpha_1(x_1 - x_0) = f[x_0] + \alpha_1(x_1 - x_0) = f[x_1]$$

Hence

$$\alpha_1 = \frac{f[x_1] - f[x_0]}{x_1 - x_0} = f[x_0, x_1]$$

$f[x_1, x_0]$ is called 1^{st} **-order divided differenc**e.

Newton's interpolation polynomial of degree n, $P_n(x)$, evaluated at x_2, gives:

$$\begin{aligned} P_n(x_2) &= \sum_{j=0}^{n} \alpha_j e_j(x_2) \\ &= \alpha_0 + \alpha_1(x_2 - x_0) + \alpha_2(x_2 - x_0)(x_2 - x_1) \\ &= f[x_0] + f[x_0, x_1](x_2 - x_0) + \alpha_2(x_2 - x_0)(x_2 - x_1) \\ &= f[x_2] \end{aligned}$$

Therefore:

$$\begin{aligned} \alpha_2(x_2 - x_0)(x_2 - x_1) &= f[x_2] - f[x_0] - f[x_0, x_1](x_2 - x_0) \\ \alpha_2 &= \frac{f[x_2] - f[x_0] - f[x_0, x_1](x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} \\ \alpha_2 &= \frac{f[x_2] - f[x_0]}{(x_2 - x_0)(x_2 - x_1)} - \frac{f[x_0, x_1]}{x_2 - x_1} \\ \alpha_2 &= \frac{f[x_0, x_2] - f[x_0, x_1]}{x_2 - x_1} \end{aligned}$$

The following form is generally preferred:

$$
\begin{aligned}
\alpha_2(x_2-x_0)(x_2-x_1) &= f[x_2]-f[x_0]-f[x_0,x_1](x_2-x_0)\\
\alpha_2(x_2-x_0)(x_2-x_1) &= f[x_2]-f[x_0]-f[x_0,x_1](x_2-x_0)-f[x_1]+f[x_1]\\
\alpha_2(x_2-x_0)(x_2-x_1) &= f[x_2]-f[x_1]+f[x_1]-f[x_0]-f[x_0,x_1](x_2-x_0)\\
\alpha_2(x_2-x_0)(x_2-x_1) &= f[x_2]-f[x_1]+(x_1-x_0)f[x_0,x_1]-f[x_0,x_1](x_2-x_0)\\
\alpha_2(x_2-x_0)(x_2-x_1) &= f[x_2]-f[x_1]+(x_1-x_2)f[x_0,x_1]\\
\alpha_2(x_2-x_0) &= \frac{f[x_2]-f[x_1]}{x_2-x_1}-f[x_0,x_1]\\
\alpha_2(x_2-x_0) &= f[x_1,x_2]-f[x_0,x_1]
\end{aligned}
$$

Hence

$$\alpha_2=\frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}=f[x_0,x_1,x_2]$$

$f[x_0, x_1, x_2]$ is called 2nd-**order divided difference**. By recurrence, we obtain:

$$\alpha_k=\frac{f[x_1,\ldots,x_k]-f[x_0,\ldots,x_{k-1}]}{x_k-x_0}=f[x_0,\ldots,x_k]$$

$f[x_0, \ldots, x_k]$ is thus called a k^{th}-**order divided difference**. In practice, when we want to determine the 3rd-order divided difference $f[x_0, x_1, x_2, x_3]$ for instance, we need the following quantities

$$
\begin{array}{lllll}
x_0 & f[x_0] & & & \\
x_1 & f[x_1] & f[x_0,x_1] & & \\
x_2 & f[x_2] & f[x_1,x_2] & f[x_0,x_1,x_2] & \\
x_3 & f[x_3] & f[x_2,x_3] & f[x_1,x_2,x_3] & f[x_0,x_1,x_2,x_3]
\end{array}
$$

Hence

$$f[x_0,x_1,x_2,x_3]=\frac{f[x_1,x_2,x_3]-f[x_0,x_1,x_2]}{x_3-x_0}$$

Properties. Let $E = \{0, 1, \ldots, n\}$ and σ be a permutation of $\mathcal{G}(E)$. Then

$$f[x_{\sigma(0)}, \ldots, x_{\sigma(n)}] = \mathrm{f}[x_0, \ldots, x_n]$$

Newton's interpolation polynomial of degree *n*

Newton's interpolation polynomial of degree n is obtained via the successive divided differences:

$$P_n(x)=f[x_0]+\sum_{j=1}^{n} f[x_0,\ldots,x_j]e_j(x)$$

An example of computing Newton's interpolation polynomial

Given a set of 3 data points {(0, 1), (2, 5), (4, 17)}, we shall determine Newton's interpolation polynomial of degree 2 which passes through these points.

$$
\begin{array}{llll}
x_0=0 & f[x_0]=1 & & \\
x_1=2 & f[x_1]=5 & f[x_0,x_1]=\dfrac{5-1}{2-0}=2 & \\
x_2=4 & f[x_2]=17 & f[x_1,x_2]=\dfrac{17-5}{4-2}=6 & f[x_0,x_1,x_2]=\dfrac{6-2}{4-0}=1
\end{array}
$$

Consequently:

$$P_2(x) = f[x_0] + f[x_0, x_1]x + f[x_0, x_1, x_2]x(x - 2) = 1 + 2x + x(x - 2) = 1 + x^2$$

Scilab: computing Newton's interpolation polynomial

Scilab function `newton.sci` determines Newton's interpolation polynomial. *X* contains the points of interpolation and *Y* the values of interpolation. *P* is Newton's interpolation polynomial computed by means of divided differences.

`newton.sci`

```
function[P]=newton(X,Y) //X nodes,Y values;P is the numerical
Newton polynomial
n=length(X); // n is the number of nodes. (n-1) is the degree
for j=2:n,
  for i=1:n-j+1,Y(i,j)=(Y(i+1,j-1)-Y(i,j-1))/(X(i+j-1)-X(i));end,
end,
x=poly(0,"x");
P=Y(1,n);
for i=2:n, P=P*(x-X(i))+Y(i,n-i+1); end
endfunction;
```

Therefore, we obtain:

```
-->X=[0;2;4]; Y=[1;5;17]; P=newton(X,Y)
 P = 1 + x^2
```