Jim Lambers
MAT 460/560
Fall Semester 2009-10
Lecture 17 Notes

These notes correspond to Section 3.2 in the text.

Divided Differences

In the previous lecture, we learned how to compute the value of an interpolating polynomial at a given point, using Neville's Method. However, the theorem that serves as the basis for Neville's Method can easily be used to compute the interpolating polynomial itself. The basic idea is to represent interpolating polynomials using the *Newton form*, which uses linear factors involving the interpolation points, instead of monomials of the form x^j.

Recall that if the interpolation points $x_0, \ldots, x_n$ are distinct, then the process of finding a polynomial that passes through the points (x_i, y_i), $i = 0, \ldots, n$, is equivalent to solving a system of linear equations $A\mathbf{x} = \mathbf{b}$ that has a unique solution. The matrix A is determined by the choice of basis for the space of polynomials of degree n or less. Each entry a_{ij} of A is the value of the jth polynomial in the basis at the point x_i.

In *Newton interpolation*, the matrix A is upper triangular, and the basis functions are defined to be the set $\{\mathcal{N}_j(x)\}_{j=0}^n$, where

$$\mathcal{N}_0(x) = 1, \quad \mathcal{N}_j(x) = \prod_{k=0}^{j-1}(x - x_k), \quad j = 1, \ldots, n.$$

The advantage of Newton interpolation is that the interpolating polynomial is easily updated as interpolation points are added, since the basis functions $\{\mathcal{N}_j(x)\}$, $j = 0, \ldots, n$, do not change from the addition of the new points.

The coefficients c_j of the Newton interpolating polynomial

$$p_n(x) = \sum_{j=0}^{n} c_j \mathcal{N}_j(x)$$

are given by

$$c_j = f[x_0, \ldots, x_j]$$

where $f[x_0, \ldots, x_j]$ denotes the *divided difference* of $x_0, \ldots, x_j$. The divided difference is defined as follows:

$$f[x_i] = y_i,$$

$$
\begin{aligned}
f[x_i, x_{i+1}] &= \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \\
f[x_i, x_{i+1}, \ldots, x_{i+k}] &= \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}.
\end{aligned}
$$

This definition implies that for each nonnegative integer j, the divided difference $f[x_0, x_1, \ldots, x_j]$ only depends on the interpolation points $x_0, x_1, \ldots, x_j$ and the value of $f(x)$ at these points. It follows that the addition of new interpolation points does not change the coefficients $c_0, \ldots, c_n$. Specifically, we have

$$
p_{n+1}(x) = p_n(x) + \frac{y_{n+1} - p_n(x_{n+1})}{\mathcal{N}_{n+1}(x_{n+1})} \mathcal{N}_{n+1}(x).
$$

This ease of updating makes Newton interpolation the most commonly used method of obtaining the interpolating polynomial.

The following result shows how the Newton interpolating polynomial bears a resemblance to a Taylor polynomial.

Theorem *Let f be n times continuously differentiable on $[a, b]$, and let $x_0, x_1, \ldots, x_n$ be distinct points in $[a, b]$. Then there exists a number $\xi \in [a, b]$ such that*

$$
f[x_0, x_1, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!}.
$$

Computing the Newton Interpolating Polynomial

We now describe in detail how to compute the coefficients $c_j = f[x_0, x_1, \ldots, x_j]$ of the Newton interpolating polynomial $p_n(x)$, and how to evaluate $p_n(x)$ efficiently using these coefficients.

The computation of the coefficients proceeds by filling in the entries of a *divided-difference table*. This is a triangular table consisting of $n + 1$ columns, where n is the degree of the interpolating polynomial to be computed. For $j = 0, 1, \ldots, n$, the jth column contains $n - j$ entries, which are the divided differences $f[x_k, x_{k+1}, \ldots, x_{k+j}]$, for $k = 0, 1, \ldots, n - j$.

We construct this table by filling in the $n + 1$ entries in column 0, which are the trivial divided differences $f[x_j] = f(x_j)$, for $j = 0, 1, \ldots, n$. Then, we use the recursive definition of the divided differences to fill in the entries of subsequent columns. Once the construction of the table is complete, we can obtain the coefficients of the Newton interpolating polynomial from the first entry in each column, which is $f[x_0, x_1, \ldots, x_j]$, for $j = 0, 1, \ldots, n$.

In a practical implementation of this algorithm, we do not need to store the entire table, because we only need the first entry in each column. Because each column has one fewer entry than the previous column, we can overwrite all of the other entries that we do not need. The following algorithm implements this idea.

Algorithm (Divided-Difference Table) Given n distinct interpolation points $x_0, x_1, \ldots, x_n$, and the values of a function $f(x)$ at these points, the following algorithm computes the coefficients $c_j = f[x_0, x_1, \ldots, x_j]$ of the Newton interpolating polynomial.

for $i = 0, 1, \ldots, n$ **do**
    $d_i = f(x_i)$
end
for $i = 1, 2, \ldots, n$ **do**
    for $j = n, n-1, \ldots, i$ **do**
        $d_j = (d_j - d_{j-1})/(x_j - x_{j-i})$
    end
end

Example We will use Newton interpolation to construct the third-degree polynomial $p_3(x)$ that fits the data

i	x_i	$f(x_i)$
0	-1	3
1	0	-4
2	1	5
3	2	-6

In other words, we must have $p_3(-1) = 3$, $p_3(0) = -4$, $p_3(1) = 5$, and $p_3(2) = -6$.

First, we construct the *divided-difference table* from this data. The divided differences in the table are computed as follows:

$$\begin{aligned}
f[x_0] &= f(x_0) \\
&= 3 \\
f[x_1] &= f(x_1) \\
&= -4 \\
f[x_2] &= f(x_2) \\
&= 5 \\
f[x_3] &= f(x_3) \\
&= -6 \\
f[x_0, x_1] &= \frac{f[x_1] - f[x_0]}{x_1 - x_0} \\
&= \frac{-4 - 3}{0 - (-1)} \\
&= -7 \\
f[x_1, x_2] &= \frac{f[x_2] - f[x_1]}{x_2 - x_1} \\
&= \frac{5 - (-4)}{1 - 0}
\end{aligned}$$

$$
\begin{aligned}
&= 9 \\
f[x_2, x_3] &= \frac{f[x_3] - f[x_2]}{x_3 - x_2} \\
&= \frac{-6 - 5}{2 - 1} \\
&= -11 \\
f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \\
&= \frac{9 - (-7)}{1 - (-1)} \\
&= 8 \\
f[x_1, x_2, x_3] &= \frac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1} \\
&= \frac{-11 - 9}{2 - 0} \\
&= -10 \\
f[x_0, x_1, x_2, x_3] &= \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0} \\
&= \frac{-10 - 8}{2 - (-1)} \\
&= -6
\end{aligned}
$$

The resulting divided-difference table is

$x_0 = -1$	$f[x_0] = 3$			
		$f[x_0, x_1] = -7$		
$x_1 = 0$	$f[x_1] = -4$		$f[x_0, x_1, x_2] = 8$	
		$f[x_1, x_2] = 9$		$f[x_0, x_1, x_2, x_3] = -6$
$x_2 = 1$	$f[x_2] = 5$		$f[x_1, x_2, x_3] = -10$	
		$f[x_2, x_3] = -11$		
$x_3 = 2$	$f[x_3] = -6$			

It follows that the interpolating polynomial $p_3(x)$ can be obtained using the *Newton divided-difference formula* as follows:

$$
\begin{aligned}
p_3(x) &= \sum_{j=0}^{3} f[x_0, \ldots, x_j] \prod_{i=0}^{j-1} (x - x_i) \\
&= f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \\
&\quad f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2) \\
&= 3 - 7(x + 1) + 8(x + 1)x - 6(x + 1)x(x - 1).
\end{aligned}
$$

We see that Newton interpolation produces an interpolating polynomial that is in the *Newton form*, with centers $x_0 = -1$, $x_1 = 0$, and $x_2 = 1$. □

Once the coefficients have been computed, we can use *nested multiplication* to evaluate the resulting interpolating polynomial, which is represented using the *Newton divided-difference formula*

$$\begin{aligned} p_n(x) &= \sum_{j=0}^{n} c_j \mathcal{N}_j(x) \\ &= \sum_{j=0}^{n} f[x_0, x_1, \ldots, x_j] \prod_{i=0}^{j-1} (x - x_i) \\ &= f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \\ &\quad f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1) \cdots (x - x_{n-1}). \end{aligned}$$

Algorithm (Nested Multiplication) Given n distinct interpolation points $x_0, x_1, \ldots, x_n$ and the coefficients $c_j = f[x_0, x_1, \ldots, x_j]$ of the Newton interpolating polynomial $p_n(x)$, the following algorithm computes $y = p_n(x)$ for a given real number x.

$y = c_n$
for $i = n - 1, n - 2, \ldots, 0$ **do**
    $y = c_i + (x - x_i)y$
end

It can be seen that this algorithm closely resembles *Horner's Method*, which is a special case of nested multiplication that works with the power form of a polynomial, whereas nested multiplication works with the more general Newton form.

Example Consider the interpolating polynomial obtained in the previous example,

$$p_3(x) = 3 - 7(x + 1) + 8(x + 1)x - 6(x + 1)x(x - 1).$$

We will use *nested multiplication* to write this polynomial in the *power form*

$$p_3(x) = b_3x^3 + b_2x^2 + b_1x + b_0.$$

This requires repeatedly applying nested multiplication to a polynomial of the form

$$p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + c_3(x - x_0)(x - x_1)(x - x_2),$$

and for each application it will perform the following steps,

$$\begin{aligned} b_3 &= c_3 \\ b_2 &= c_2 + (z - x_2)b_3 \\ b_1 &= c_1 + (z - x_1)b_2 \\ b_0 &= c_0 + (z - x_0)b_1, \end{aligned}$$

where, in this example, we will set $z = 0$ each time.

The numbers b_0, b_1, b_2 and b_3 computed by the algorithm are the coefficients of $p(x)$ in the Newton form, with the centers x_0, x_1 and x_2 changed to z, x_0 and x_1; that is,

$$p(x) = b_0 + b_1(x - z) + b_2(x - z)(x - x_0) + b_3(x - z)(x - x_0)(x - x_1).$$

It follows that $b_0 = p(z)$, which is why this algorithm is the preferred method for evaluating a polynomial in Newton form at a given point z.

It should be noted that the algorithm can be derived by writing $p(x)$ in the *nested* form

$$p(x) = c_0 + (x - x_0)[c_1 + (x - x_1)[c_2 + (x - x_2)c_3]]$$

and computing $p(z)$ as follows:

$$\begin{aligned} p(z) &= c_0 + (z - x_0)[c_1 + (z - x_1)[c_2 + (z - x_2)c_3]] \\ &= c_0 + (z - x_0)[c_1 + (z - x_1)[c_2 + (z - x_2)b_3]] \\ &= c_0 + (z - x_0)[c_1 + (z - x_1)b_2] \\ &= c_0 + (z - x_0)b_1 \\ &= b_0. \end{aligned}$$

Initially, we have

$$p(x) = 3 - 7(x + 1) + 8(x + 1)x - 6(x + 1)x(x - 1),$$

so the coefficients of $p(x)$ in this Newton form are

$$c_0 = 3, \quad c_1 = -7, \quad c_2 = 8, \quad c_3 = -6,$$

with the centers

$$x_0 = -1, \quad x_1 = 0, \quad x_2 = 1.$$

Applying nested multiplication to these coefficients and centers, with $z = 0$, yields

$$\begin{aligned} b_3 &= -6 \\ b_2 &= 8 + (0 - 1)(-6) \\ &= 14 \\ b_1 &= -7 + (0 - 0)(14) \\ &= -7 \\ b_0 &= 3 + (0 - (-1))(-7) \\ &= -4. \end{aligned}$$

It follows that

$$\begin{aligned} p(x) &= -4 + (-7)(x-0) + 14(x-0)(x-(-1)) + (-6)(x-0)(x-(-1))(x-0) \\ &= -4 - 7x + 14x(x+1) - 6x^2(x+1), \end{aligned}$$

and the centers are now 0, -1 and 0.

For the second application of nested multiplication, we have

$$p(x) = -4 - 7x + 14x(x+1) - 6x^2(x+1),$$

so the coefficients of $p(x)$ in this Newton form are

$$c_0 = -4, \quad c_1 = -7, \quad c_2 = 14, \quad c_3 = -6,$$

with the centers

$$x_0 = 0, \quad x_1 = -1, \quad x_2 = 0.$$

Applying nested multiplication to these coefficients and centers, with $z = 0$, yields

$$\begin{aligned} b_3 &= -6 \\ b_2 &= 14 + (0-0)(-6) \\ &= 14 \\ b_1 &= -7 + (0-(-1))(14) \\ &= 7 \\ b_0 &= -4 + (0-0)(7) \\ &= -4. \end{aligned}$$

It follows that

$$\begin{aligned} p(x) &= -4 + 7(x-0) + 14(x-0)(x-0) + (-6)(x-0)(x-0)(x-(-1)) \\ &= -4 + 7x + 14x^2 - 6x^2(x+1), \end{aligned}$$

and the centers are now 0, 0 and -1.

For the third and final application of nested multiplication, we have

$$p(x) = -4 + 7x + 14x^2 - 6x^2(x+1),$$

so the coefficients of $p(x)$ in this Newton form are

$$c_0 = -4, \quad c_1 = 7, \quad c_2 = 14, \quad c_3 = -6,$$

with the centers

$$x_0 = 0, \quad x_1 = 0, \quad x_2 = -1.$$

Applying nested multiplication to these coefficients and centers, with $z = 0$, yields

$$\begin{aligned} b_3 &= -6 \\ b_2 &= 14 + (0 - (-1))(-6) \\ &= 8 \\ b_1 &= 7 + (0 - 0)(8) \\ &= 7 \\ b_0 &= -4 + (0 - 0)(7) \\ &= 1. \end{aligned}$$

It follows that

$$\begin{aligned} p(x) &= -4 + 7(x - 0) + 8(x - 0)(x - 0) + (-6)(x - 0)(x - 0)(x - 0) \\ &= -4 + 7x + 8x^2 - 6x^3, \end{aligned}$$

and the centers are now 0, 0 and 0. Since all of the centers are equal to zero, the polynomial is now in the power form.

It should be noted that this is not the most efficient way to convert the Newton form of $p(x)$ to the power form. To see this, we observe that after one application of nested multiplication, we have

$$\begin{aligned} p(x) &= b_0 + b_1(x - z) + b_2(x - z)(x - x_0) + b_3(x - z)(x - x_0)(x - x_1) \\ &= b_0 + (x - z)[b_1 + b_2(x - x_0) + b_3(x - x_0)(x - x_1)]. \end{aligned}$$

Therefore, we can apply nested multiplication to the second-degree polynomial

$$q(x) = b_1 + b_2(x - x_0) + b_3(x - x_0)(x - x_1),$$

which is the quotient obtained by dividing $p(x)$ by $(x - z)$. Because

$$p(x) = b_0 + (x - z)q(x),$$

it follows that once we have changed all of the centers of $q(x)$ to be equal to z, then all of the centers of $p(x)$ will be equal to z as well. In summary, we can convert a polynomial of degree n from Newton form to power form by applying nested multiplication n times, where the jth application is to a polynomial of degree $n - j + 1$, for $j = 1, 2, \ldots, n$.

Since the coefficients of the appropriate Newton form of each of these polynomials of successively lower degree are computed by the nested multiplication algorithm, it follows that we can implement this more efficient procedure simply by proceeding exactly as before, except that during the jth application of nested multiplication, we do not compute the coefficients $b_0, b_1, \ldots, b_{j-2}$, because they will not change anyway, as can be seen from the previous computations. For example, in the second application, we did not need to compute b_0, and in the third, we did not need to compute b_0 and b_1. □