Jim Lambers
MAT 772
Fall Semester 2010-11
Lecture 5 Notes

These notes correspond to Sections 6.2 and 6.3 in the text.

Lagrange Interpolation

Calculus provides many tools that can be used to understand the behavior of functions, but in most cases it is necessary for these functions to be continuous or differentiable. This presents a problem in most "real" applications, in which functions are used to model relationships between quantities, but our only knowledge of these functions consists of a set of discrete data points, where the data is obtained from measurements. Therefore, we need to be able to construct continuous functions based on discrete data.

The problem of constructing such a continuous function is called *data fitting*. In this lecture, we discuss a special case of data fitting known as *interpolation*, in which the goal is to find a linear combination of n known functions to fit a set of data that imposes n constraints, thus guaranteeing a unique solution that fits the data exactly, rather than approximately. The broader term "constraints" is used, rather than simply "data points", since the description of the data may include additional information such as rates of change or requirements that the fitting function have a certain number of continuous derivatives.

When it comes to the study of functions using calculus, polynomials are particularly simple to work with. Therefore, in this course we will focus on the problem of constructing a polynomial that, in some sense, fits given data. We first discuss some algorithms for computing the unique polynomial $p_n(x)$ of degree n that satisfies $p_n(x_i) = y_i$, $i = 0, \ldots, n$, where the points (x_i, y_i) are given. The points $x_0, x_1, \ldots, x_n$ are called *interpolation points*. The polynomial $p_n(x)$ is called the *interpolating polynomial* of the data (x_0, y_0), (x_1, y_1), ..., (x_n, y_n). At first, we will assume that the interpolation points are all distinct; this assumption will be relaxed in a later lecture.

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. However, different algorithms for computing the interpolating polynomial use a different A, since they each use a different basis for the space of polynomials of degree $\leq n$.

The most straightforward method of computing the interpolation polynomial is to form the system $A\mathbf{x} = \mathbf{b}$ where $b_i = y_i$, $i = 0, \ldots, n$, and the entries of A are defined by $a_{ij} = p_j(x_i)$, $i, j = 0, \ldots, n$, where $x_0, x_1, \ldots, x_n$ are the points at which the data $y_0, y_1, \ldots, y_n$ are obtained, and $p_j(x) = x^j$, $j = 0, 1, \ldots, n$. The basis $\{1, x, \ldots, x^n\}$ of the space of polynomials of degree $n + 1$ is called the *monomial basis*, and the corresponding matrix A is called the *Vandermonde matrix*

for the points $x_0, x_1, \ldots, x_n$. Unfortunately, this matrix can be ill-conditioned, especially when interpolation points are close together.

In *Lagrange interpolation*, the matrix A is simply the identity matrix, by virtue of the fact that the interpolating polynomial is written in the form

$$p_n(x) = \sum_{j=0}^{n} y_j \mathcal{L}_{n,j}(x),$$

where the polynomials $\{\mathcal{L}_{n,j}\}_{j=0}^{n}$ have the property that

$$\mathcal{L}_{n,j}(x_i) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}.$$

The polynomials $\{\mathcal{L}_{n,j}\}$, $j = 0, \ldots, n$, are called the *Lagrange polynomials* for the interpolation points x_0, x_1, ..., x_n. They are defined by

$$\mathcal{L}_{n,j}(x) = \prod_{k=0, k\neq j}^{n} \frac{x - x_k}{x_j - x_k}.$$

As the following result indicates, the problem of polynomial interpolation can be solved using Lagrange polynomials.

Theorem *Let $x_0, x_1, \ldots, x_n$ be $n + 1$ distinct numbers, and let $f(x)$ be a function defined on a domain containing these numbers. Then the polynomial defined by*

$$p_n(x) = \sum_{j=0}^{n} f(x_j)\mathcal{L}_{n,j}$$

is the unique polynomial of degree n that satisfies

$$p_n(x_j) = f(x_j), \quad j = 0, 1, \ldots, n.$$

The polynomial $p_n(x)$ is called the *interpolating polynomial* of $f(x)$. We say that $p_n(x)$ *interpolates* $f(x)$ at the points $x_0, x_1, \ldots, x_n$.

Example We will use Lagrange interpolation to find the unique polynomial $p_3(x)$, of degree 3 or less, that agrees with the following data:

i	x_i	y_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 Lagrange polynomials $\{\mathcal{L}_{3,j}(x)\}_{j=0}^{3}$, using the formula

$$\mathcal{L}_{n,j}(x) = \prod_{i=0, i\neq j}^{3} \frac{(x - x_i)}{(x_j - x_i)}.$$

This yields

$$\begin{aligned}
\mathcal{L}_{3,0}(x) &= \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} \\
&= \frac{(x-0)(x-1)(x-2)}{(-1-0)(-1-1)(-1-2)} \\
&= \frac{x(x^2-3x+2)}{(-1)(-2)(-3)} \\
&= -\frac{1}{6}(x^3-3x^2+2x) \\
\mathcal{L}_{3,1}(x) &= \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} \\
&= \frac{(x+1)(x-1)(x-2)}{(0+1)(0-1)(0-2)} \\
&= \frac{(x^2-1)(x-2)}{(1)(-1)(-2)} \\
&= \frac{1}{2}(x^3-2x^2-x+2) \\
\mathcal{L}_{3,2}(x) &= \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} \\
&= \frac{(x+1)(x-0)(x-2)}{(1+1)(1-0)(1-2)} \\
&= \frac{x(x^2-x-2)}{(2)(1)(-1)} \\
&= -\frac{1}{2}(x^3-x^2-2x) \\
\mathcal{L}_{3,3}(x) &= \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} \\
&= \frac{(x+1)(x-0)(x-1)}{(2+1)(2-0)(2-1)} \\
&= \frac{x(x^2-1)}{(3)(2)(1)}
\end{aligned}$$

$$= \frac{1}{6}(x^3 - x).$$

By substituting x_i for x in each Lagrange polynomial $\mathcal{L}_{3,j}(x)$, for $j = 0, 1, 2, 3$, it can be verified that

$$\mathcal{L}_{3,j}(x_i) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}.$$

It follows that the Lagrange interpolating polynomial $p_3(x)$ is given by

$$\begin{aligned}
p_3(x) &= \sum_{j=0}^{3} y_j \mathcal{L}_{3,j}(x) \\
&= y_0\mathcal{L}_{3,0}(x) + y_1\mathcal{L}_{3,1}(x) + y_2\mathcal{L}_{3,2}(x) + y_3\mathcal{L}_{3,3}(x) \\
&= (3)\left(-\frac{1}{6}\right)(x^3 - 3x^2 + 2x) + (-4)\frac{1}{2}(x^3 - 2x^2 - x + 2) + (5)\left(-\frac{1}{2}\right)(x^3 - x^2 - 2x) + \\
&\quad (-6)\frac{1}{6}(x^3 - x) \\
&= -\frac{1}{2}(x^3 - 3x^2 + 2x) + (-2)(x^3 - 2x^2 - x + 2) - \frac{5}{2}(x^3 - x^2 - 2x) - (x^3 - x) \\
&= \left(-\frac{1}{2} - 2 - \frac{5}{2} - 1\right)x^3 + \left(\frac{3}{2} + 4 + \frac{5}{2}\right)x^2 + (-1 + 2 + 5 + 1)\,x - 4 \\
&= -6x^3 + 8x^2 + 7x - 4.
\end{aligned}$$

Substituting each x_i, for $i = 0, 1, 2, 3$, into $p_3(x)$, we can verify that we obtain $p_3(x_i) = y_i$ in each case. □

While the Lagrange polynomials are easy to compute, they are difficult to work with. Furthermore, if new interpolation points are added, all of the Lagrange polynomials must be recomputed. Unfortunately, it is not uncommon, in practice, to add to an existing set of interpolation points. It may be determined after computing the kth-degree interpolating polynomial $p_k(x)$ of a function $f(x)$ that $p_k(x)$ is not a sufficiently accurate approximation of $f(x)$ on some domain. Therefore, an interpolating polynomial of higher degree must be computed, which requires additional interpolation points.

To address these issues, we consider the problem of computing the interpolating polynomial *recursively*. More precisely, let $k > 0$, and let $p_k(x)$ be the polynomial of degree k that interpolates the function $f(x)$ at the points $x_0, x_1, \ldots, x_k$. Ideally, we would like to be able to obtain $p_k(x)$ from polynomials of degree $k - 1$ that interpolate $f(x)$ at points chosen from among $x_0, x_1, \ldots, x_k$. The following result shows that this is possible.

Theorem *Let n be a positive integer, and let $f(x)$ be a function defined on a domain containing the $n+1$ distinct points $x_0, x_1, \ldots, x_n$, and let $p_n(x)$ be the polynomial of degree n that interpolates $f(x)$ at the points $x_0, x_1, \ldots, x_n$. For each $i = 0, 1, \ldots, n$, we define $p_{n-1,i}(x)$ to be the polynomial*

of degree $n-1$ that interpolates $f(x)$ at the points $x_0, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$. If i and j are distinct nonnegative integers not exceeding n, then

$$p_n(x) = \frac{(x - x_j)p_{n-1,j}(x) - (x - x_i)p_{n-1,i}(x)}{x_i - x_j}.$$

This result leads to an algorithm called *Neville's Method* that computes the value of $p_n(x)$ at a given point using the values of lower-degree interpolating polynomials at x. We now describe this algorithm in detail.

Algorithm Let $x_0, x_1, \ldots, x_n$ be distinct numbers, and let $f(x)$ be a function defined on a domain containing these numbers. Given a number x^*, the following algorithm computes $y^* = p_n(x^*)$, where $p_n(x)$ is the nth interpolating polynomial of $f(x)$ that interpolates $f(x)$ at the points $x_0, x_1, \ldots, x_n$.

for $j = 0$ **to** n **do**
    $Q_j = f(x_j)$
end
for $j = 1$ **to** n **do**
    for $k = n$ **to** j **do**
        $Q_k = [(x - x_k)Q_{k-1} - (x - x_{k-j})Q_k]/(x_{k-j} - x_k)$
    end
end
$y^* = Q_n$

At the jth iteration of the outer loop, the number Q_k, for $k = n, n-1, \ldots, j$, represents the value at x of the polynomial that interpolates $f(x)$ at the points $x_k, x_{k-1}, \ldots, x_{k-j}$.

The preceding theorem can be used to compute the polynomial $p_n(x)$ itself, rather than its value at a given point. This yields an alternative method of constructing the interpolating polynomial, called *Newton interpolation*, that is more suitable for tasks such as inclusion of additional interpolation points.

Convergence

In some applications, the interpolating polynomial $p_n(x)$ is used to fit a known function $f(x)$ at the points $x_0, \ldots, x_n$, usually because $f(x)$ is not feasible for tasks such as differentiation or integration that are easy for polynomials, or because it is not easy to evaluate $f(x)$ at points other than the interpolation points. In such an application, it is possible to determine how well $p_n(x)$ approximates $f(x)$.

To that end, we assume that x is *not* one of the interpolation points $x_0, x_1, \ldots, x_n$, and we define

$$\varphi(t) = f(t) - p_n(t) - \frac{f(x) - p_n(x)}{\pi_{n+1}(x)}\pi_{n+1}(t),$$

where

$$\pi_{n+1}(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$$

is a polynomial of degree $n + 1$. Because x is not one of the interpolation points, it follows that $\varphi(t)$ has at least $n + 2$ zeros: x, and the $n + 1$ interpolation points $x_0, x_1, \ldots, x_n$. Furthermore, $\pi_{n+1}(x) \neq 0$, so $\varphi(t)$ is well-defined.

If f is $n+1$ times continuously differentiable on an interval $[a, b]$ that contains the interpolation points and x, then, by the Generalized Rolle's Theorem, $\varphi^{(n+1)}$ must have at least one zero in $[a, b]$. Therefore, at some point $\xi(x)$ in $[a, b]$, that depends on x, we have

$$0 = \varphi^{(n+1)}(\xi(x)) = f^{(n+1)}(t) - \frac{f(x) - p_n(x)}{\pi_{n+1}(x)}(n + 1)!,$$

which yields the following result.

Theorem *(Interpolation error) If f is $n + 1$ times continuously differentiable on $[a, b]$, and $p_n(x)$ is the unique polynomial of degree n that interpolates $f(x)$ at the $n+1$ distinct points $x_0, x_1, \ldots, x_n$ in $[a, b]$, then for each $x \in [a, b]$,*

$$f(x) - p_n(x) = \prod_{j=0}^{n}(x - x_j)\frac{f^{(n+1)}(\xi(x))}{(n + 1)!},$$

where $\xi(x) \in [a, b]$.

It is interesting to note that the error closely resembles the Taylor remainder $R_n(x)$.

If the number of data points is large, then polynomial interpolation becomes problematic since high-degree interpolation yields oscillatory polynomials, when the data may fit a smooth function.

Example Suppose that we wish to approximate the function $f(x) = 1/(1 + x^2)$ on the interval $[-5, 5]$ with a tenth-degree interpolating polynomial that agrees with $f(x)$ at 11 equally-spaced points $x_0, x_1, \ldots, x_{10}$ in $[-5, 5]$, where $x_j = -5 + j$, for $j = 0, 1, \ldots, 10$. Figure 1 shows that the resulting polynomial is not a good approximation of $f(x)$ on this interval, even though it agrees with $f(x)$ at the interpolation points. The following MATLAB session shows how the plot in the figure can be created.

```
>> % create vector of 11 equally spaced points in [-5,5]
>> x=linspace(-5,5,11);
>> % compute corresponding y-values
>> y=1./(1+x.^2);
>> % compute 10th-degree interpolating polynomial
>> p=polyfit(x,y,10);
>> % for plotting, create vector of 100 equally spaced points
>> xx=linspace(-5,5);
```

```
>> % compute corresponding y-values to plot function
>> yy=1./(1+xx.^2);
>> % plot function
>> plot(xx,yy)
>> % tell MATLAB that next plot should be superimposed on
>> % current one
>> hold on
>> % plot polynomial, using polyval to compute values
>> % and a red dashed curve
>> plot(xx,polyval(p,xx),'r--')
>> % indicate interpolation points on plot using circles
>> plot(x,y,'o')
>> % label axes
>> xlabel('x')
>> ylabel('y')
>> % set caption
>> title('Runge''s example')
```

In general, it is not wise to use a high-degree interpolating polynomial and equally-spaced interpolation points to approximate a function on an interval $[a, b]$ unless this interval is sufficiently small. The example shown in Figure 1 is a well-known example of the difficulty of high-degree polynomial interpolation using equally-spaced points, and it is known as *Runge's example.* □

Is it possible to choose the interpolation points so that the error is minimized? To answer this question, we introduce the *Chebyshev polynomials*

$$T_k(x) = \cos(k \cos^{-1}(x)),$$

which satisfy the three-term recurrence relation

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x), \quad T_0(x) \equiv 1, \quad T_1(x) \equiv x.$$

These polynomials have the property that $|T_k(x)| \leq 1$ on the interval $[-1, 1]$, while they grow rapidly outside of this interval. Furthermore, the roots of these polynomials lie within the interval $[-1, 1]$. Therefore, if the interpolation points $x_0, x_1, \ldots, x_n$ are chosen to be the images of the roots of the $(n + 1)$st-degree Chebyshev polynomial under a linear transformation that maps $[-1, 1]$ to $[a, b]$, then it follows that

$$\left| \prod_{j=0}^{n} (x - x_j) \right| \leq \frac{1}{2^n}, \quad x \in [a, b].$$

Therefore, the error in interpolating $f(x)$ by an nth-degree polynomial is determined entirely by $f^{(n+1)}$.

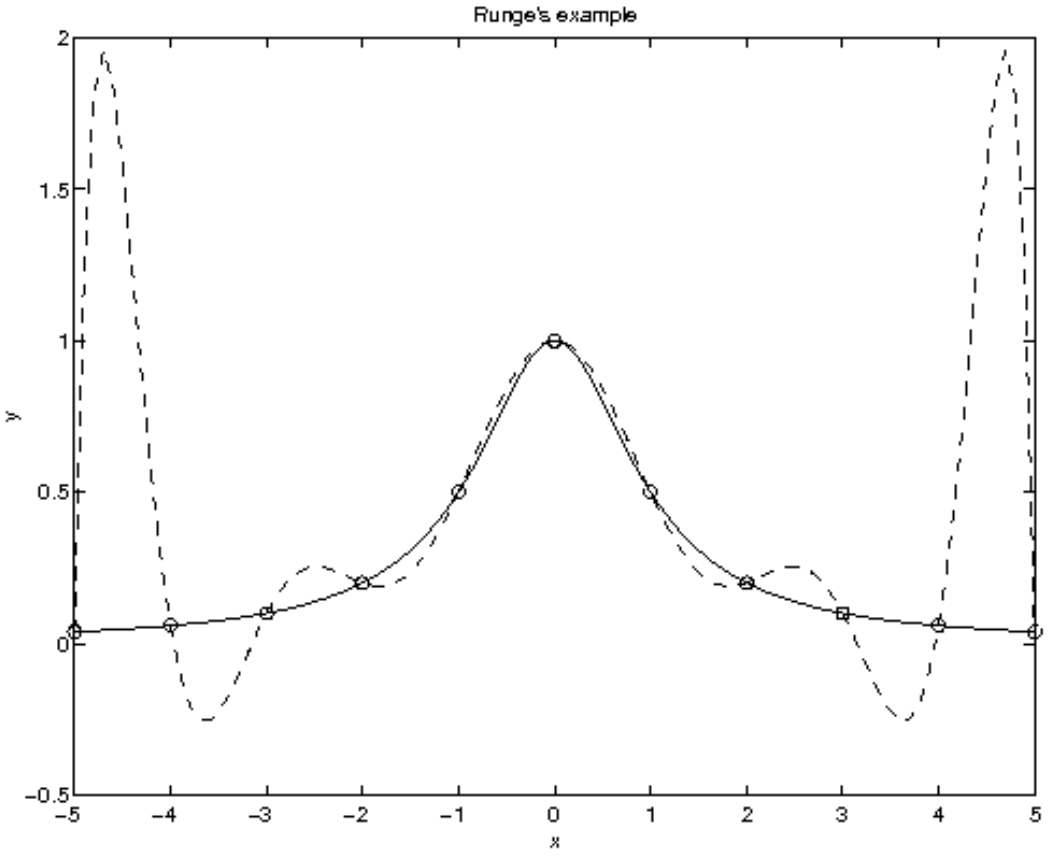

Figure 1: The function $f(x) = 1/(1+x^2)$ (solid curve) cannot be interpolated accurately on $[-5, 5]$ using a tenth-degree polynomial (dashed curve) with equally-spaced interpolation points. This example that illustrates the difficulty that one can generally expect with high-degree polynomial interpolation with equally-spaced points is known as *Runge's example*.