Jim Lambers
MAT 460/560
Fall Semester 2009-10
Lecture 20 Notes

These notes correspond to Section 3.4 in the text.

Piecewise Polynomial Interpolation

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')
```

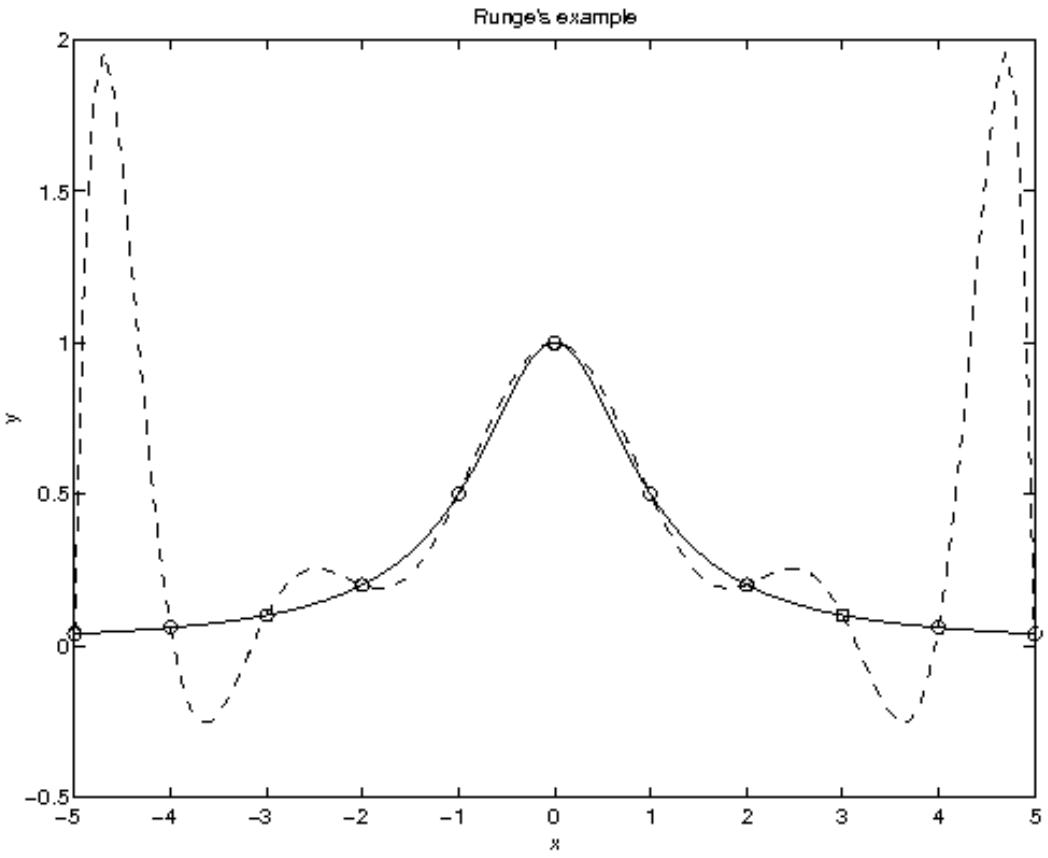

Figure 1: The function $f(x) = 1/(1+x^2)$ (solid curve) cannot be interpolated accurately on $[-5, 5]$ using a tenth-degree polynomial (dashed curve) with equally-spaced interpolation points. This example that illustrates the difficulty that one can generally expect with high-degree polynomial interpolation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

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. For example, one can interpolate this same function $f(x) = 1/(1 + x^2)$ with reasonable accuracy on the interval $[-1, 1]$ with a tenth-degree polynomial and equally-spaced interpolation points (try

it).

In general, when one is able to choose the interpolation points arbitrarily, the *Chebyshev points*, to be discussed later, yield much greater accuracy. 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.* □

If the fitting function is only required to have a few continuous derivatives, then one can construct a *piecewise polynomial* to fit the data.

We now precisely define what we mean by a piecewise polynomial.

Definition *(Piecewise polynomial) Let $[a, b]$ be an interval that is divided into subintervals $[x_i, x_{i+1}]$, where $i = 0, \ldots, n-1$, $x_0 = a$ and $x_n = b$. A* **piecewise polynomial** *is a function $p(x)$ defined on $[a, b]$ by*

$$p(x) = p_i(x), \quad x_i \leq x \leq x_{i+1}, \quad i = 0, 1, \ldots, n-1,$$

where, for $i = 0, 1, \ldots, n-1$, each function $p_i(x)$ is a polynomial defined on $[x_i, x_{i+1}]$. The **degree** *of $p(x)$ is the maximum degree of each polynomial $p_i(x)$, for $i = 0, 1, \ldots, n-1$.*

It is essential to note that by this definition, a piecewise polynomial defined on $[a, b]$ is equal to some polynomial on each subinterval $[x_i, x_{i+1}]$ of $[a, b]$, for $i = 0, 1, \ldots, n-1$, but a different polynomial may be used for each subinterval.

Typically, piecewise polynomials are used to fit smooth functions, and therefore are required to have a certain number of continuous derivatives. This requirement imposes additional constraints on the piecewise polynomial, and therefore the degree of the polynomials used on each subinterval must be chosen sufficiently high to ensure that these constraints can be satisfied.

Cubic Spline Interpolation

A *spline* is a piecewise polynomial of degree k that has $k-1$ continuous derivatives. The most commonly used spline is a *cubic spline*, which we now define.

Definition *(Cubic Spline) Let $f(x)$ be function defined on an interval $[a, b]$, and let $x_0, x_1, \ldots, x_n$ be $n+1$ distinct points in $[a, b]$, where $a = x_0 < x_1 < \cdots < x_n = b$. A* **cubic spline**, *or* **cubic spline interpolant**, *is a piecewise polynomial $s(x)$ that satisifes the following conditions:*

1. *On each interval $[x_{i-1}, x_i]$, for $i = 1, \ldots, n$, $s(x) = s_i(x)$, where $s_i(x)$ is a cubic polynomial.*
2. *$s(x_i) = f(x_i)$ for $i = 0, 1, \ldots, n$.*
3. *$s(x)$ is twice continuously differentiable on (a, b).*
4. *Either of the following boundary conditions are satisfied:*
   (a) *$s''(a) = s''(b) = 0$, which is called* **free** *or* **natural boundary conditions**, *and*

(b) $s'(a) = f'(a)$ *and* $s'(b) = f'(b)$*, which is called* **clamped boundary conditions**.

If $s(x)$ *satisfies free boundary conditions, we say that* $s(x)$ *is a* **natural spline**. *The points* $x_0, x_1, \ldots, x_n$ *are called the* **nodes** *of* $s(x)$.

Clamped boundary conditions are often preferable because they use more information about $f(x)$, which yields a spline that better approximates $f(x)$ on $[a, b]$. However, if information about $f'(x)$ is not available, then free boundary conditions must be used instead.

Constructing Cubic Splines

Suppose that we wish to construct a cubic spline interpolant $s(x)$ that fits the given data (x_0, y_0), $(x_1, y_1), \ldots, (x_n, y_n)$, where $a = x_0 < x_1 < \cdots < x_n = b$, and $y_i = f(x_i)$, for some known function $f(x)$ defined on $[a, b]$. From the preceding discussion, this spline is a piecewise polynomial of the form

$$s(x) = s_i(x) = d_i(x - x_i)^3 + c_i(x - x_i)^2 + b_i(x - x_i) + a_i, \quad i = 0, 1, \ldots, n-1, \quad x_i \leq x \leq x_{i+1}.$$

That is, the value of $s(x)$ is obtained by evaluating a different cubic polynomial for each subinterval $[x_i, x_{i+1}]$, for $i = 0, 1, \ldots, n-1$.

We now use the definition of a cubic spline to construct a system of equations that must be satisfied by the coefficients a_i, b_i, c_i and d_i for $i = 0, 1, \ldots, n-1$. We can then compute these coefficients by solving the system. Because $s(x)$ must fit the given data, we have

$$a_i = y_i, \quad i = 0, 1, \ldots, n-1.$$

If we define $h_i = x_{i+1} - x_i$, for $i = 0, 1, \ldots, n-1$, and define $a_n = y_n$, then the requirement that $s(x)$ is continuous at the interior nodes implies that we must have $s_i(x_{i+1}) = s_{i+1}(x_{i+1})$ for $i = 0, 1, \ldots, n-2$. Furthermore, because $s(x)$ must fit the given data, we must also have $s(x_n) = s_{n-1}(x_n) = y_n$. These conditions lead to the constraints

$$d_i h_i^3 + c_i h_i^2 + b_i h_i + a_i = a_{i+1}, \quad i = 0, 1, \ldots, n-1.$$

To ensure that $s(x)$ has a continuous first derivative at the interior nodes, we require that $s_i'(x_{i+1}) = s_{i+1}'(x_{i+1})$ for $i = 0, 1, \ldots, n-2$, which imposes the constraints

$$3d_i h_i^2 + 2c_i h_i + b_i = b_{i+1}, \quad i = 0, 1, \ldots, n-2.$$

Similarly, to enforce continuity of the second derivative at the interior nodes, we require that $s_i''(x_{i+1}) = s_{i+1}''(x_{i+1})$ for $i = 0, 1, \ldots, n-2$, which leads to the constraints

$$3d_i h_i + c_i = c_{i+1}, \quad i = 0, 1, \ldots, n-2.$$

There are $4n$ coefficients to determine, since there are n cubic polynomials, with 4 coefficients each. However, we have only prescribed $4n-2$ constraints, so we must specify 2 more in order to determine a unique solution. If we use free boundary conditions, then these constraints are

$$\begin{aligned} c_0 &= 0, \\ 3d_{n-1}h_{n-1} + c_{n-1} &= 0. \end{aligned}$$

On the other hand, if we use clamped boundary conditions, then our additional constraints are

$$\begin{aligned} b_0 &= z_0, \\ 3d_{n-1}h_{n-1}^2 + 2c_{n-1}h_{n-1} + b_{n-1} &= z_n, \end{aligned}$$

where $z_i = f'(x_i)$ for $i = 0, 1, \ldots, n$. In the next lecture, we will learn how to use these equations to construct cubic spline interpolants.