Lecture `interp2`: Interpolation: Newton's Method

October 12, 2022

Summary: Newton's method for polynomial interpolation. Discussion of operation count for interpolant determination, introduction of Newton's divided differences.

References: Some aspects taken from C. F. Van Loan's *Introduction to Scientific Computing*. See Section 8.1, pages 181-189.

Introduction

The previous lecture showed how to compute an interpolant with respect to a general basis of functions. We can interpolate with polynomials, trigonometric functions, a logarithmic class of functions, or whatever by constructing the appropriate Vandermonde matrix. Solving the corresponding N–by–N Vandermonde linear system $V\boldsymbol{c} = \boldsymbol{y}$ requires a direct inversion of V which generally costs $O(N^3)$ operations. (Moreover, V is potentially a poorly conditioned matrix, in which case the resulting interpolation may not be sufficiently accurate.) While this is not an issue for small N, for large problems such matrix inversion is costly. To mitigate this cost, we'll find a better method to compute the interpolant coefficients $\boldsymbol{c}$. To do so, we first simplify our interpolation problem: we'll restrict the problem to polynomial interpolation. It turns out that we can introduce new polynomial bases that allow for more efficient computation of interpolants. One such basis is due to Newton, and yields *Newton's Method* for interpolation.

Motivation

Consider an interpolation problem with data $D_4 = \{(x_i, y_i)\}_{i=1}^4$ (here $N = 4$), where for polynomial interpolation we previously used the basis $\mathcal{B}_4 = \left\{x^{i-1}\right\}_{i=1}^4 = \{1, x, x^2, x^3\}$ of monomials. The associated Vandermonde matrix is full,

$$V = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ 1 & x_4 & x_4^2 & x_4^3 \end{pmatrix} \tag{1}$$

To introduce the Newton Method, let us consider the same data set D_4, but a new basis defined by $\widehat{\mathcal{B}}_4 = \{1, (x - x_1), (x - x_1)(x - x_2), (x - x_1)(x - x_2)(x - x_3)\}$. Notice that the new basis $\widehat{\mathcal{B}}_4$ depends on the data nodes x_i. We then seek an interpolant $p(x)$ of the form

$$p(x) = c_0 + c_1(x - x_1) + c_2(x - x_1)(x - x_2) + c_3(x - x_1)(x - x_2)(x - x_3), \tag{2}$$

such that $p(x_i) = y_i$ for all $i = 1, 2, 3, 4$. Now the associated Vandermonde matrix is the following:

$$\widehat{V} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & (x_2 - x_1) & 0 & 0 \\ 1 & (x_3 - x_1) & (x_3 - x_1)(x_3 - x_2) & 0 \\ 1 & (x_4 - x_1) & (x_4 - x_1)(x_4 - x_2) & (x_4 - x_1)(x_4 - x_2)(x_4 - x_3). \end{pmatrix} \tag{3}$$

$\widehat{V}$ is a lower-triangular matrix! This means that $\widehat{V}\boldsymbol{c} = \boldsymbol{y}$ can be solved with forward substitution. Solution of a lower–triangular (or upper–triangular) system costs $O(N^2)$ operations, instead of $O(N^3)$ for a full system. Note that nothing has changed; the interpolant function will be the same whether we use the basis $\mathcal{B}_4$ or $\widehat{\mathcal{B}}_4$. However, the coefficients $\boldsymbol{c}$ will change: the solutions to $V\boldsymbol{c} = \boldsymbol{y}$ and $\widehat{V}\boldsymbol{c} = \boldsymbol{y}$ are different.

Let us start the process of solving this system, using the augmented matrix approach and Gaussian elimination rather than a direct forward substitution,

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & y_1 \\ 1 & (x_2 - x_1) & 0 & 0 & y_2 \\ 1 & (x_3 - x_1) & (x_3 - x_1)(x_3 - x_2) & 0 & y_3 \\ 1 & (x_4 - x_1) & (x_4 - x_1)(x_4 - x_2) & (x_4 - x_1)(x_4 - x_2)(x_4 - x_3) & y_4 \end{array}\right].$$

The first step is to eliminate all the ones in the first column below the first row, at it gives

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & y_1 \\ 0 & (x_2 - x_1) & 0 & 0 & y_2 - y_1 \\ 0 & (x_3 - x_1) & (x_3 - x_1)(x_3 - x_2) & 0 & y_3 - y_1 \\ 0 & (x_4 - x_1) & (x_4 - x_1)(x_4 - x_2) & (x_4 - x_1)(x_4 - x_2)(x_4 - x_3) & y_4 - y_1 \end{array}\right].$$

As a second step, we divide each row by the coefficient in the second column to get

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & y_1 \\ 0 & 1 & 0 & 0 & (y_2 - y_1)/(x_2 - x_1) \\ 0 & 1 & x_3 - x_2 & 0 & (y_3 - y_1)/(x_3 - x_1) \\ 0 & 1 & x_4 - x_2 & (x_4 - x_2)(x_4 - x_3) & (y_4 - y_1)/(x_4 - x_1) \end{array}\right].$$

We would need to go further to complete the solution; however, let us stop and note the following. Via 3 row–subtractions and 3 row–divisions, we've reduced the original 4–by–4 interpolation problem to a 3–by–3 interpolation problem. Indeed, the lower–right 3–by–3 submatrix is exactly the Vandermonde matrix for the basis $\{1, x - x_2, (x - x_2)(x - x_3)\}$ with the data nodes x_2, x_3, x_4. The ability to recursively obtain smaller interpolation problems is a key idea behind Newton's Method. To continue we need a new definition.

Newton divided differences

Definition 1 (Newton divided differences)
Let $f(x)$ be a function and $\{x_1, x_2, \cdots, x_N\}$ a collection of distinct data nodes. Define the zeroth–order divided differences as $f[x_i] = f(x_i)$ for $i = 1, 2, \cdots N$. Then, for all $i = 1, 2, \cdots, N - k$, the k–th order divided differences are given by

$$f[x_i, x_{i+1}, \cdots, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \cdots, x_{i+k}] - f[x_i, x_{i+1}, \cdots, x_{i+k-1}]}{x_{i+k} - x_i}. \tag{4}$$

An example will help us see through this definition.

Example 1 (Computing divided differences)
Let $D_5 = \{(1,1),(2,-2),(3,4),(4,-1),(5,0)\}$. We provide the following table as a guide for computing the associated divided differences.

$$\begin{array}{c|ccccc}
x_1 & f[x_1] & & & & \\
 & & f[x_1,x_2] & & & \\
x_2 & f[x_2] & & f[x_1,x_2,x_3] & & \\
 & & f[x_2,x_3] & & f[x_1,x_2,x_3,x_4] & \\
x_3 & f[x_3] & & f[x_2,x_3,x_4] & & f[x_1,x_2,x_3,x_4,x_5] \\
 & & f[x_3,x_4] & & f[x_2,x_3,x_4,x_5] & \\
x_4 & f[x_4] & & f[x_3,x_4,x_5] & & \\
 & & f[x_4,x_5] & & & \\
x_5 & f[x_5] & & & &
\end{array}$$

We start at the left of the table, and use

$$f[1] = f(1) = 1, \quad f[2] = f(2) = -2, \quad f[3] = f(3) = 4, \quad f[4] = f(4) - 1, \quad f[5] = f(5) = 0.$$

Then to compute each next column, we use the entries at the current column that straddle the desired next entry from the left. Therefore, the entries in the second row are

$$f[1,2] = \frac{f[2]-f[1]}{2-1} = -3, \quad f[2,3] = \frac{f[3]-f[2]}{3-2} = 6,$$
$$f[3,4] = \frac{f[4]-f[3]}{4-3} = -5, \quad f[4,5] = \frac{f[5]-f[4]}{5-4} = 1.$$

Those in third row are

$$f[1,2,3] = \frac{f[2,3]-f[1,2]}{3-1} = \frac{9}{2}, \qquad f[2,3,4] = \frac{f[3,4]-f[2,3]}{4-2} = -\frac{11}{2},$$
$$f[3,4,5] = \frac{f[4,5]-f[3,4]}{5-3} = 3.$$

The fourth row is

$$f[1,2,3,4] = \frac{f[2,3,4]-f[1,2,3]}{4-1} = -\frac{10}{3}, \quad f[2,3,4,5] = \frac{f[3,4,5]-f[2,3,4]}{5-2} = \frac{17}{6}.$$

Finally, the sole entry in the fifth row is

$$f[1,2,3,4,5] = \frac{f[2,3,4,5]-f[1,2,3,4]}{5-1} = \frac{37}{24}.$$

Newton's solution to the interpolation problem

We have introduced the divided differences with little motivation. However, let us return to our initial example: interpolation of the data $D_4 = \{(x_1,y_1),(x_2,y_2),(x_3,y_3),(x_4,y_4)\}$ with respect to the basis $\widehat{\mathcal{B}}_4$. To solve the problem, we know of two paths open to us:

1. Use MATLAB's built–in `inv` or backslash function to invert the Vandermonde matrix.
2. Given the simple form of the Vandermonde matrix, program a forward–substitution algorithm.

However, we now present a third alternative; it is no faster than the second option, but it utilizes the divided differences we have just computed. We state the following without proof.

Given data $D_N = \{(x_k, y_k)\}_{k=1}^N$, where we view the samples $y_k = f(x_k)$ as coming from a function $f(x)$ [the $f_{\text{true}}(x)$ of Lecture `interp1`], construct the Newton basis $\mathcal{B}_N = \{\phi_k(x)\}_{k=1}^N$,

$$\begin{aligned}
\phi_1(x) &= 1 \\
\phi_2(x) &= (x - x_1) \\
\phi_3(x) &= (x - x_1)(x - x_2) \\
&\vdots \\
\phi_N(x) &= (x - x_1)(x - x_2) \cdots (x - x_{N-1}).
\end{aligned}$$

The kth basis element is $\phi_k(x) = \prod_{j=1}^{k-1}(x - x_j)$, and the polynomial $p(x)$ which interpolates the data is given by

$$\begin{aligned}
p(x) &= \sum_{i=1}^{N} f[x_1, x_2, \cdots, x_i]\phi_i(x) \\
&= f[x_1]\phi_1(x) + f[x_1, x_2]\phi_2(x) + f[x_1, x_2, x_3]\phi_3(x) + \cdots + f[x_1, x_2, \cdots, x_N]\phi_N(x). \quad (5)
\end{aligned}$$

Here of course $p(x_k) = f(x_k) = y_k$. Put differently, for the Newton basis the interpolation coefficients are

$$c_i = f[x_1, x_2, \cdots, x_i] \quad (6)$$

Thus, instead of inverting the Vandermonde matrix, we may instead perform the exercise of **Example 1**, and compute divided differences to obtain $\boldsymbol{c}$. We note that computing the Newton divided differences $f[\cdot]$ up to order N also takes $O(N^2)$ operations; the same as using forward substitution to invert the Vandermonde system. In fact the approaches are equivalent, just the emphasis is different.

Example 2
Use Newton's divided differences to find the interpolating polynomial passing through $(0, 1)$, $(2, 2)$, $(3, 4)$.

The divided–difference table is as follows.

```
x1  | f[x1]
    |        f[x1,x2]
x2  | f[x2]             f[x1,x2,x3]
    |        f[x2,x3]
x3  | f[x3]
```

Explicitly, we have the table.

```
0   |   1
    |           1/2
2   |   2               1/2
    |            2
3   |   4
```

Therefore, the polynomial is $p(x) = 1 + \frac{1}{2}(x-0) + \frac{1}{2}(x-0)(x-2) = 1 + \frac{1}{2}x + \frac{1}{2}x(x-2)$.

Example 3
Add the fourth data point $(1, 0)$ to **Example 2**.

An advantage of Newton's method is that we can keep most of our work. The new divided difference table is

```
 0   |   1
     |           1/2
 2   |   2               1/2
     |            2               -1/2
 3   |   4                 0      -----
     |            2       ---
 1   |   0       ---
---      ---
```

Note only the underlined entries have been added to get the new table. Therefore, the new interpolating polynomial = old interpolating polynomial $-\frac{1}{2}(x-0)(x-2)(x-3)$, that is $p_{\text{new}}(x) = 1 + \frac{1}{2}x + \frac{1}{2}x(x-2) - \frac{1}{2}x(x-2)(x-3)$.