Evaluating a polynomial

Suppose we are given the coefficients α_i of $p_n(x)$

$$p_n(x) = \alpha_0 + \alpha_1 x + \ldots + \alpha_n x^n$$

The coefs are stored in array $a(0:n)$.
How do we compute $p_n(x)$ for a given x?

$p = a(0)$
for $i = 1, n$
 $p = p + a(i) * x ** i$
end for

The above algo uses n multiplications, n additions, and n exponentiations.

A faster alternative : **Horner's rule**

Note that $p_n(x)$ can be written in the form

$$p_n(x) = ((((\alpha_n x + \alpha_{n-1})x + \alpha_{n-2})x + \ldots) + \alpha_1)x + \alpha_0$$

This suggests the code

$p = a(n)$
for $i = (n-1), 0$
 $p = p \cdot x + a(i)$
end for

This algo uses only n mults, and n adds.

A modification of this algo can be used to compute $p_n'(x)$ and $p_n(x)$ given in NDD form.

Polynomial interpolant with derivative data

In some cases it is desirable to construct poly that match not only given function values on a set of points, but also derivative values (of some order) on the same or another set of points.

Given a set of distinct pts $x_0, \ldots, x_n$ and $m_i \geq 0$, $i = 0, \ldots, n$ the **osculating poly** of $f \in \mathcal{C}^m$, $m = \max\{m_i\}$ is the poly $p(x)$ of the least degree s.t.

$$\frac{\partial^k p(x_i)}{\partial x^k} = \frac{\partial^k f(x_i)}{\partial x^k} \; k = 0, \ldots, m_i \text{ and } i = 0, \ldots, n$$

When $m_i = 0, \forall i$, we get the usual interp poly that matches only function values.

A special case of interest is $m_i = 1, \forall i$. Then we have the **Hermite poly**. It matches the function values and 1st derivative values at all x_i and is of deg $\leq 2n+1$.

Hermite–monomials

Suppose we are given data $(x_i, y_i, y_i'), i = 0, \ldots, n$ where the y_i' represent derivative values. We want to construct a poly of degree at most $2n+1$ that interpolates the data including the derivative data. Let

$$p_{2n+1}(x) = \sum_{j=0}^{2n+1} \alpha_j x^j$$

The interpolating conditions are

$$p_{2n+1}(x_i) = y_i, p_{2n+1}'(x_i) = y_i', i = 0, \ldots, n$$

But
$p_{2n+1}'(x) = \alpha_1 + 2\alpha_2 x +, \ldots, +(2n+1)\alpha_n x^{2n}$

In the form of a linear system, $B\alpha = y$, the interp. conditions are

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{2n+1} \\ 0 & 1 & 2x_0 & \dots & (2n+1)x_0^{2n} \\ 1 & x_1 & x_1^2 & \dots & x_1^{2n+1} \\ 0 & 1 & 2x_1 & \dots & (2n+1)x_1^{2n} \\ & & & \dots & \\ 1 & x_n & x_n^2 & \dots & x_n^{2n+1} \\ 0 & 1 & 2x_n & \dots & (2n+1)x_n^{2n} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_{2n} \\ \alpha_{2n+1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_0' \\ y_1 \\ y_1' \\ \vdots \\ y_n \\ y_n' \end{bmatrix}$$

It can be shown that B is non-singular iff the pts x_i are distinct. In this case we can uniquely solve the equation and compute $\alpha_0, \dots, \alpha_n$.

Hermite–NDD

Let $p_n(x) = \sum_{j=0}^{2n+1} \alpha_j b_j(x)$, where

$$b_j(x) = \begin{cases} \Pi_{i=0}^{\frac{j}{2}-1}(x - x_i)^2 & j \text{ even}, \\ \Pi_{i=0}^{\frac{j-1}{2}-1}(x - x_i)^2(x - x_{\frac{j-1}{2}}) & j \text{ odd} \end{cases}$$

To construct the NDD table

- Write the x_i's in one col and the y_i's in another. For each pair (x_i, y_i) duplicate the data by writing two rows.
- Write the y_i''s in another col, at rows where the DDs cannot be calculated due to division by zero.
- Fill in the rest of the table.

Example in class.

Existence and Uniqueness Theorem: Given data $(x_i, y_i, y_i'), i = 0, \dots, n$, with x_i distinct, there exists a unique interp poly of degree $\leq 2n+1$ which interp the data.

Error Theorem: If g has $2n + 2$ cts derivatives and $p_{2n+1}(x)$ is the Hermite poly of degree$\leq 2n+1$ that interp g and its derivatives at $x_0, \dots, x_n$ then for any x

$$g(x) - p_{2n+1}(x) = \frac{g^{(2n+2)}(\xi)}{(2n+2)!} w^2(x)$$

where ξ is a pt in $spr\{x, x_0, \dots, x_n\}$ and $w(x) = (x - x_0)(x - x_1)\dots(x - x_n)$.

Problems with polynomial interpolants

In order to get a fairly *good* approximation to a function $f(x)$ by an interpolating poly of nth degree, it is often necessary to use a fairly large n.

There are error and computational problems arising from constructing high degree interp poly.

- Poly of high degree often have oscillatory behaviour towards the end pts
- NDD procedure takes $O(n^2)$ flops
- Evaluation of NDD poly on one pt takes $O(n)$ flops

Piecewise poly interpolation is a good alternative that resolves the above problems since it

- avoids oscillatory behaviour for large n
- requires $O(n)$ flops for construction, $O(1)$ for evaluation on one pt
- guarantees that error decreases as n increases

Piecewise polynomials and splines

Let $P = \{x_0, x_1, \ldots, x_n\}$ be a set of distinct pts partitioning the interval $[a = x_0, b = x_n]$ into n subintervals.
A **piecewise** polynomial (pp) of degree k defined wrt P is a poly of degree k in each subinterval but possibly not a poly in $[a, b]$ due to lack of continuity or continuity of derivatives on the pts of the partition.

A pp is defined by a function with branches

$$pp(x) = \begin{cases} a_1 x + b_1 & \text{if } x_0 \le x \le x_1 \\ a_2 x + b_2 & \text{if } x_1 \le x \le x_2 \\ \vdots & \\ a_n x + b_n & \text{if } x_{n-1} \le x \le x_n \end{cases}$$

We usually impose some continuity conditions on the interior x_i.
i.e. $a_i x_i + b_i = a_{i+1} x_i + b_{i+1}, i = 1, \ldots, n-1$.

A pp of degree k is called a **spline** if it is cts and has cts derivatives up to order $k-1$ on the interior nodes. A spline of degree k has
$(k+1)n - k(n-1) = n + k$ free parameters.
i.e. a linear spline is cts on x_i, but does not have a cts derivative on x_i.

A linear pp has $2n$ free parameters a_i, b_i, a linear spline has $2n - (n-1) = n + 1$ free parameters.

Linear spline interpolant

Given a partition $P = \{x_0, x_1, \ldots, x_n\}$ and a set of data $\{(x_i, y_i), i = 0, \ldots, n\}$, the unique linear spline defined wrt P and the data is written as, for $i = 1, \ldots, n$,

$$L(x) = \left\{ \; y_{i-1} \frac{x - x_i}{x_{i-1} - x_i} + y_i \frac{x - x_{i-1}}{x_i - x_{i-1}} \quad \text{if } x_{i-1} \le x \le x_i \right.$$

$L(x)$ can be graphed by connecting the pts (x_i, y_i), $i = 0, \ldots, n$. It can be shown that for some constants K and K' that

$$|g(x) - L(x)| \le K h^2$$
$$|g'(x) - L'(x)| \le K' h,$$

where $h = \max_i \{|x_{i+1} - x_i|\}$

Implications of the error formulae

- Bound for the error does not involve high derivatives of g, which often take large values, so it is kept small.
- If we double n, we expect the error to decrease by a factor of 4 and the error of the derivative by a factor of 2. Thus, it is guaranteed that the bound of the error decreases as n increases.
- No computation is required to construct an interp linear spline.
- To evaluate it for an arbitrary x in $[a, b]$ we only need to find out in which interval x belongs to, and evaluate it at that branch with a constant number of flops.