

- Instructions: 465/565 students must answer all the questions unless noted otherwise.

1. We write Lagrange's interpolation formula as

$$p_n(x) = \sum_{j=0}^n f(x_j)l_j(x)$$

where $l_j(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n)}{(x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)}$ and the points x_j , $j = 0, 1, \dots, n$, are assumed to be distinct. Show that

$$\sum_{j=0}^n l_j(x) \equiv 1 \tag{1}$$

for all values of x . You may not use (2) in problem 4 since we used (1) to prove this result.

Hint: This argument should only require one or two lines, and no algebra!

2. Suppose you wish to construct a table of the exponential integral function

$$E_1(x) = \int_x^\infty \frac{e^{-t}}{t} dt \quad x > 0,$$

over the interval $x \in [1, 10]$ with stepsize h . How large can h be if a user of your table expects to obtain values at arbitrary x -locations with an absolute error $\leq 10^{-8}$ when using second degree polynomial interpolation?

Hint: Use the error formula for second degree polynomial interpolation

$$|p_2(x) - f(x)| \leq \frac{|(x-x_0)(x-x_1)(x-x_2)|}{3!} \max_{a \leq x \leq b} |f^{(3)}(x)|.$$

Bound the term $|(x-x_0)(x-x_1)(x-x_2)|$ for $x_0 = -h$, $x_1 = 0$, $x_2 = h$, and compute $\max_{a \leq x \leq b} |E_1^{(3)}(x)|$ for $a = 1$ and $b = 10$.

3. As discussed in class, Lagrange's interpolation formula can be rearranged into the magnificent *barycentric formula* (or more specifically the second (true) form of the barycentric formula):

$$p_n(x) = \frac{\sum_{j=0}^n \frac{w_j}{x-x_j} f_j}{\sum_{j=0}^n \frac{w_j}{x-x_j}}, \tag{2}$$

where

$$w_j = \frac{1}{\prod_{\substack{i=0 \\ i \neq j}}^n (x_j - x_i)}, \quad j = 0, 1, \dots, n.$$

Here x_j are the given nodes and f_j the corresponding function values.

- (a) Write a MATLAB function for computing the barycentric weights w_j in the above expression. Your function should take as input a vector containing the nodes x_j and output the weights w_j . Call your function `baryfit`. The function declaration should be something like:

```
function w = baryfit(x)
```

Write another function for evaluating the barycentric interpolant $p_n(x)$. This function should take as input a vector containing the nodes x_j , a vector containing the corresponding barycentric weights w_j (generated from your `baryfit` function), a vector containing the corresponding function values f_j , and the location (or a vector of locations) of where the interpolant should be evaluated. The output of the function should be the value of the interpolating polynomial at all the evaluation points. Call this function `baryeval`. The function declaration should be something like:

```
function p = baryval(x,w,f,xi)
```

Both of these functions should avoid unnecessary loops and FLOPs. Turn in a print out of both functions.

MATLAB has a function `prod` that can be used to compute the product of all entries in a vector.

- (b) The following three node sets are known as equispaced, Chebyshev, and Legendre, respectively:

- (i) $x_j = -1 + \frac{2j}{8}, j = 0, 1, \dots, 8$
- (ii) $x_j = -\cos \frac{j\pi}{8}, j = 0, 1, \dots, 8$
- (iii) $\mp 0.96816023950763 \quad \mp 0.83603110732664 \quad \mp 0.61337143270059 \quad \mp 0.32425342340381 \quad 0$

Using your `baryfit` function from part a, generate the barycentric weights $w_j, j = 0, 1, \dots, 8$ for each of these node sets. Plot the values of the weights versus the corresponding values of the nodes (i.e plot $(x_j, w_j), j = 0, 1, \dots, 8$) for each of the three node sets. Comment on the results.

- (c) For the three node sets from part b, use your `baryval` function to evaluate the 8th degree polynomial interpolant of the function $f(x) = 1/(1 + 25x^2)$ at 101 equally spaced points between $[-1, 1]$ (i.e. at the points $\xi_j = -1 + \frac{2j}{100}, j = 0, 1, \dots, 100$). Plot the error $E(x) = p_8(x) - f(x)$ in the polynomial interpolant at these evaluation points for each of the three node sets. Which node set seems to produce the best result? What criteria did you use to determine what ‘best’ means?
- (d) Repeat part (c), but for the function $f(x) = |x|$. Which node set seems to produce the best result? What criteria did you use to determine what ‘best’ means?
- (e) For certain sets of nodes x_j , it is possible to give explicit formulas for the barycentric weights w_j . The easiest case is when the nodes are *equally spaced* between $[-1, 1]$, (i.e. $x_j = -1 + \frac{2j}{n}, j = 0, 1, \dots, n$). Show that for these nodes

$$w_j = \frac{\binom{n}{2}^n (-1)^{n-j}}{n!} \binom{n}{j}.$$

Note that since w_j appear both in the numerator and denominator of (2), any factors common to all w_j can be factored out. Thus, we can reduce the above expression for the barycentric weights to $w_j = (-1)^j \binom{n}{j}$.

4. One way to improve upon the convergence rate of the secant method for finding a zero of a function $f(x)$ is to fit a parabola through the three previous iterates x_n, x_{n-1} , and x_{n-2} . The updated iterate x_{n+1} is then given from one (which one?) of the locations where this parabola crosses the x -axis. The iteration for this rootfinding method is given by

$$x_{n+1} = x_n - \frac{2f(x_n)}{w + \text{sign}(w)\sqrt{w^2 - 4f(x_n)f[x_n, x_{n-1}, x_{n-2}]}}$$

where

$$w = f[x_n, x_{n-1}] + f[x_n, x_{n-2}] - f[x_{n-2}, x_{n-1}].$$

Your goal is to derive the above expression.

This method can be applied to any rootfinding problem, but it is particularly good for approximating roots of polynomials. The reason is that the square root in the denominator allows this method capture complex roots when a real initial guess is made. Both Newton and secant require a complex starting guess to find a complex root. It can be shown that this method converges superlinearly with a rate of ≈ 1.83929 .

5. [565 only] The Vandermonde matrix for polynomial interpolation at the node points x_0, x_1, \dots, x_n is defined as

$$V_n = \begin{bmatrix} 1 & x_0 & \cdots & x_0^{n-1} & x_0^n \\ 1 & x_1 & \cdots & x_1^{n-1} & x_1^n \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & x_{n-1} & \cdots & x_{n-1}^{n-1} & x_{n-1}^n \\ 1 & x_n & \cdots & x_n^{n-1} & x_n^n \end{bmatrix}.$$

Prove that $\det(V_n) = \prod_{0 \leq j < i \leq n} (x_i - x_j)$.

Hint: One way to proceed is as follows. Let

$$\gamma_n(x) = \det \begin{bmatrix} 1 & x_0 & \cdots & x_0^{n-1} & x_0^n \\ 1 & x_1 & \cdots & x_1^{n-1} & x_1^n \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & x_{n-1} & \cdots & x_{n-1}^{n-1} & x_{n-1}^n \\ 1 & x & \cdots & x^{n-1} & x^n \end{bmatrix},$$

and show $\gamma_n(x)$ is a polynomial of degree n , and that its roots are x_0, x_1, \dots, x_{n-1} (which can be done by expanding the determinant in the last row by minors). Use this result to deduce that

$$\gamma_n = (x - x_0)(x - x_1) \cdots (x - x_{n-1})\gamma_{n-1}(x_{n-1}).$$

Next use induction to prove $\gamma_n(x_n) = \prod_{0 \leq j < i \leq n} (x_i - x_j)$.

6. [565 only] The error term for the $2n+1$ degree Hermite interpolating polynomial to the function $f(x)$ is given without proof by Bradie on p. 410 as

$$f(x) - p_{2n+1}(x) = [\Phi_{n+1}(x)]^2 \frac{f^{(2n+2)}(\xi)}{(2n+2)!}, \quad \text{where } x_0 < x_1 < \cdots < x_n, \xi \in [x_0, x_n],$$

$$\text{and } \Phi_{n+1}(x) = \prod_{i=0}^n (x - x_i),$$

(here we have let $p_{2n+1}(x)$ replace $P(x)$ in Bradie's formula). The proof is a simple extension of the proof for the standard Lagrange interpolating polynomial error formula given in class and proceeds by first defining the function

$$G(t) = [\Phi_{n+1}(t)]^2 \{f(x) - p_{2n+1}(x)\} - [\Phi_{n+1}(x)]^2 \{f(t) - p_{2n+1}(t)\}.$$

Next, find the zeros of $G(t)$ and their multiplicity. Then use generalized Rolle's theorem, etc. Complete this argument to prove the above error formula.

Hint: Rolle's theorem and its generalization can also be applied to a function with roots of multiplicity > 1 .