Spring 2021: Numerical Analysis
Assignment 5: Polynomial interpolation/approximation
Due April 21st 2pm EST

1. **[Quadratic interpolation and root finding, 8 pts]** When solving an equation of the form $f(x) = 0$, Newton's method uses explicit derivative information of f to calculate the tangent line and its x-intercept (its root). The secant method uses two points $(x_{k-1}, f(x_{k-1}))$, and $(x_k, f(x_k))$, to fit a linear function approximating the tangent line and then finds its root.

   (a) [4 pts] On the other hand, Muller's method uses three points, $(x_{k-2}, f(x_{k-2}))$, $(x_{k-1}, f(x_{k-1}))$, $(x_k, f(x_k))$, to fit a parabola (i.e. interpolate a parabola). It then uses the quadratic formula to find the root of this parabola that is closest to x_k. Using Lagrange interpolation, derive Muller's method, i.e., write as compact of a formula for how to obtain x_{k+1} from the previous three points as you can.

   (b) [4 pts] A core part of Matlab's default algorithm in *fsolve* (called Brent's method) is to use inverse quadratic interpolation approach, which flips the role of x and y. The idea is to interpolate a quadratic polynomial through $(f(x_{k-2}), x_{k-2})$, $(f(x_{k-1}), x_{k-1})$, $(f(x_k), x_k)$, and then simply evaluate this polynomial at $y = 0$ to estimate the root, thus avoiding solving quadratic equations and possibly complex roots. Derive the iteration formula for this method.
   Note: I am aware that both methods are described in Wikipedia, and you are welcome to reference that. However, you must give complete steps of the derivation with explanations to get credit. Writing just the final formula will get no points.

2. **[Cost of barycentric interpolation, 6 pts]** The degree-n Second Barycentric Interpolation Formula that interpolates the data $(x_0, y_0), \ldots, (x_n, y_n)$ is given by:

$$p_n(x) = \frac{\sum_{k=0}^{n} y_k \, w_k/(x - x_k)}{\sum_{k=0}^{n} w_k/(x - x_k)},$$

   where

$$w_k = \prod_{j=0,\, j\neq k}^{n} \frac{1}{x_k - x_j}.$$

   [2pts] Compute the computational cost (i.e. the number of floating point operations) required for evaluatingp_n at a single point $\tilde{x} \neq x_j$.
   [2pts] Then give the optimal (as far as you can see) computational cost for evaluating p_n at N points.
   [2pts] Lastly, what is the computational cost of evaluating p_n at N fixed points for M different functions (i.e., different y data).

3. **[Space of polynomials, 8pts]** Let $\mathcal{P}_n$ be the space of functions defined on $[-1, 1]$ that can be described by polynomials of degree less of equal to n with coefficients in $\mathbb{R}$. The space $\mathcal{P}_n$ is a linear space in the sense of abstract linear algebra. The $n + 1$ monomials $\{1, x, x^2, \ldots, x^n\}$ are a basis for $\mathcal{P}_n$.

   (a) [2pts] Show that for pairwise distinct points $x_0, x_1, \ldots, x_n \in [-1, 1]$, the Lagrange polynomials $L_k(x)$ are in $\mathcal{P}_n$, and that they are linearly independent. Note that this implies that the $(n + 1)$ Lagrange polynomials also form a basis of $\mathcal{P}_n$.

   (b) [3pts] Since both the monomials and the Lagrange polynomials are a basis of $\mathcal{P}_n$, each $p \in \mathcal{P}_n$ can be written as linear combination of monomials as well as Lagrange polynomials, i.e.,

$$p(x) = \sum_{k=0}^{n} \beta_k L_k(x) = \sum_{k=0}^{n} \alpha_k x^k, \tag{1}$$

with appropriate coefficients $\alpha_k, \beta_k \in \mathbb{R}$. As you know from linear algebra, there exists a basis transformation matrix $\boldsymbol{V}$ that converts the coefficients $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_n)^T$ to the coefficients and vice versa $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_n)^T$. Write down this matrix, either for the transformation $\boldsymbol{\alpha} = V\boldsymbol{\beta}$ or $\boldsymbol{\beta} = V\boldsymbol{\alpha}$, whichever is easier for you.

(c) [3pts] Note that since $\boldsymbol{V}$ transforms one basis into another basis, it must be an invertible matrix. Write a computer code to compute the 2-norm condition number $\kappa_2(\boldsymbol{V})$ for $n = 5, 15, 25$ with uniformly spaced nodes $x_i = -1 + (2i)/n$, $i = 0, \ldots, n$. Based on the condition numbers, can this basis transformation be computed accurately numerically in double-precision floating-point arithmetic?

4. **[Polynomial interpolation versus least squares fitting, 4pts]** Previously, we have used a least squared approach to fit functions to data points. We are given the points:

i	0	1	2	3	4	5
X	0.0	0.5	1.0	1.5	2.0	2.5
Y	0.0	0.20	0.27	0.30	0.32	0.33

(a) [3pts] Write down the (probably overdetermined) linear system associated to finding the cubic best fit polynomial

$$Y = aX^3 + bX^2 + cX + d\,.$$

using (i) all six points, (ii) only the data for $i = 0, 1, 2, 3, 4$, and (iii) $i = 0, 1, 2, 3$. In each case solve the system and plot both the data points and the polynomial. Why is case (iii) different from the other two cases.

(b) [1pt] What is the degree of the polynomial you would have to use so that the solution interpolates (i.e., goes through) *all* six data points?

5. **[Errors in polynomial interpolation, 8pts]** Interpolate the function

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0, \end{cases}$$

on the domain $[-1, 1]$ using an interpolating polynomial with Chebyshev points as nodes.

(a) [3 pts] Write down an explicit formula for the polynomial of degree $n = 3$.

(b) [5 pts] Write code to evaluate the interpolating polynomial, using either the Lagrange or Barycentric formula (it is OK to reuse code from Worksheets or lectures but if you use code someone else wrote explain that and acknowledge the contribution). Note that you can use $n = 3$ from part (a) to test your code (or, use *polyfit/polyval* as illustrated in Worksheet 5).
Make plots of the function and its approximating polynomial for 8 interpolation points. [*Hint: The polynomial must pass through the interpolation points.*] Then plot the error $|f(x) - p(x)|$ with a log scale for the y axis, for $8, 16, 32, 64, 128, 256$ points. Comment on what you observe. Do you think $p(x) \to f(x)$ as $n \to \infty$ (points are given not based on whether the answer is right or wrong but for your reasoning/explanation)?