Fall 2018: Numerical Analysis
Assignment 5 (due November 20, 2018)

2 extra credit points will be given for cleanly plotted and labeled figures (see also rules on the first assignment). Use a legend and different line styles to label multiple graphs in one plot (no colors needed). Do not export figures using raster graphics (.jpg, .png) but use vector graphics (.eps, .pdf, .dxf) that do not mess up lines. Label axes and use titles. Use `help plot`, `help legend`, `help xlabel` to better understand MATLAB's (or Python's) plotting capabilities.

1. **[Space of polynomials P_n, 1+2+2pts]** Let 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 $\mathbb{R}$. P_n is a linear space in the sense of linear algebra, in particular, for $p, q \in P_n$ and $a \in \mathbb{R}$, also $p + q$ and ap are in P_n. Since the monomials $\{1, x, x^2, \dots, x^n\}$ are a basis for P_n, the dimension of that space is $n + 1$.

   (a) Show that for pairwise distinct points $x_0, x_1, \dots, x_n \in [-1, 1]$, the Lagrange polynomials $L_k(x)$ are in P_n, and that they are linearly independent, that is, for a linear combination of the zero polynomial with Lagrange polynomials with coefficients α_k, i.e.,

   $$\sum_{k=0}^{n} \alpha_k L_k(x) = 0 \text{ (the zero polynomial)}$$

   necessarily follows that $\alpha_0 = \alpha_1 = \ldots = \alpha_n = 0$. Note that this implies that the $(n + 1)$ Lagrange polynomials also form a basis of P_n.

   (b) Since both the monomials and the Lagrange polynomials are a basis of P_n, each $p \in P_n$ can be written as linear combination of monomials as well as Lagrange polynomials, i.e.,

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

   with appropriate coefficients $\alpha_k, \beta_k \in \mathbb{R}$. As you know from basic matrix theory, there exists a basis transformation matrix that converts the coefficients $\boldsymbol{\alpha} = (\alpha_0, \dots, \alpha_n)^T$ to the coefficients $\boldsymbol{\beta} = (\beta_0, \dots, \beta_n)^T$. Show that this basis transformation matrix is given by the so-called Vandermonde matrix $V \in \mathbb{R}^{n+1\times n+1}$ given by

   $$V = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} & x_n^n \end{pmatrix},$$

   i.e., the relation between $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in (1) is given by $\boldsymbol{\alpha} = V\boldsymbol{\beta}$. An easy way to see this is to choose appropriate x in (1).

(c) Note that since V transforms one basis into another basis, it must be an invertible matrix. Let us compute the condition number of V numerically.[1] Compute the 2-based condition number $\kappa_2(V)$ for $n = 5, 10, 20, 30$ with uniformly spaced nodes $x_i = -1 + (2i)/n$, $i = 0, \dots, n$. Based on the condition numbers, can this basis transformation be performed accurately?

2. **[Polynomial interpolation versus least squares fitting, 3+1pts]** Recall how Q7 in HW3 required you to fit a function to six given data points. This led to a least squares optimization problem. We are given the same points as in HW3:

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) Write down the least squares problem 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) not a least squares problem?

(b) 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?

3. **[Polynomial interpolation and error estimation, 2+2+2+2pt]** Let us interpolate the function $f : [0, 1] \to \mathbb{R}$ defined by $f(x) = \exp(3x)$ using the nodes $x_i = i/2$, $i = 0, 1, 2$ by a quadratic polynomial $p_2 \in \boldsymbol{P}_2$.

(a) Use the monomial basis $1, x, x^2$ and compute (numerically) the coefficients $c_j \in \mathbb{R}$ such that $p_2(x) = \sum_{j=0}^{2} c_j x^j$. Plot p_2 and f in the same graph.

(b) Give an alternative form for p_2 using Lagrange interpolation polynomials $L_0(x)$, $L_1(x)$ and $L_2(x)$. Plot the three Lagrange basis polynomials in the same graph.

(c) Compare the exact interpolation error $E_f(x) := f(x) - p_2(x)$ at $x = 3/4$ with the estimate

$$|E_f(x)| \leq \frac{M_{n+1}}{(n+1)!} |\pi_{n+1}(x)|,$$

where $M_{n+1} = \max_{z \in [0,1]} |f^{(n+1)}(z)|$, $f^{(n+1)}$ is the $(n + 1)$st derivative of f, and $\pi_{n+1}(x) = (x - x_0)(x - x_1)(x - x_2)$.

(d) Find a (Hermite) polynomial $p_3 \in \boldsymbol{P}_3$ that interpolates f and f' in x_0, x_1. Give the polynomial p_3 in the Hermite basis, plot f and p_3 in the same graph, and plot the four Hermite basis functions in another graph.

[1] MATLAB provides the function `vander`, which can be used to assemble V (actually, the transpose of V). Alternatively, one can use a simple loop to construct V.

4. **[Polynomial interpolation, 2+3pt]** Consider linear interpolation of $f(x) = x^3$ at $x_0 = 0$ and $x_1 = 1$.

   a) For given x, find the value of $\xi = \xi(x)$ for which

   $$f(x) - p_1(x) = \frac{f''(\xi)}{2}(x - x_0)(x - x_1).$$

   b) Repeat for $f(x) = (2x - 1)^4$.

5. **[Errors in polynomial interpolation, 3pt extra credit]** Interpolate the function

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

   on the domain $[-1, 1]$ using Lagrange polynomials with Chebyshev points.[2] You can use the following MATLAB function `lagrange_interpolant` to compute the values of the Lagrange interpolants p_n.

```
 function y0 = lagrange_interpolant(x, y, x0)
% x is the vector of abscissas.
% y is the matching vector of ordinates.
% x0 represents the target to be interpolated
% y0 represents the solution from the Lagrange interpolation
y0 = 0;
n = length(x);
 for j = 1 : n
     t = 1;
     for i = 1 : n
          if i~=j
               t = t * (x0-x(i))/(x(j)-x(i));
          end
     end
     y0 = y0 + t*y(j);
end
```

   Describe qualitatively what you see for $n = 2, 4, 8, 16, 32, 64, 128, 256$ interpolation points. Provide a table of the maximum errors[3]

   $$||p_n - f||_\infty = \max_{x \in [-1,1]} |p_n(x) - f(x)|,$$

[2] Recall that the Chebyshev points on the interval $[a, b]$ are

$$x_i = \frac{1}{2}(a + b) + \frac{1}{2}(b - a)\cos\left(\frac{i + \frac{1}{2}}{n + 1}\pi\right) \text{ for } i = 0, \ldots, n.$$

[3] You can approximate the maximum error by evaluating the error $p_n - f$ at a large number of uniformly distributed points, e.g., at $\sim 10n$ points, and determining the difference using the maximum absolute value, i.e.

$$||p_n - f||_\infty = \max_{x \in [-1,1]} |p_n(x) - f(x)| \approx \max_{j=0,\ldots,10n} |p_n(\xi_j) - f(\xi_j)|,$$

where $\xi_j = -1 + \frac{2}{10n}j$ for $j = 0, \ldots, 10n$.

and the L_2-errors[4]

$$||p_n - f||_2 = \sqrt{\int_{-1}^{1} (p_n(x) - f(x))^2 dx}$$

for each $n = 2, 4, 8, 16, 32, 64, 128, 256$. Do you expect convergence in the maximum norm? How about in the L_2 norm?

[4] You can approximate the L_2-error by evaluating the error $p_n - f$ at a large number of uniformly distributed points, e.g., at $\sim 10n$ points, and computing

$$||p_n - f||_2 = \sqrt{\int_{-1}^{1} (p_n(x) - f(x))^2 dx} \approx \sqrt{\frac{2}{10n} \sum_{j=0}^{10n} (p_n(\xi_j) - f(\xi_j))^2},$$

where $\xi_j = -1 + \frac{2}{10n} j$ for $j = 0, \ldots, 10n$.