Spring 2017: Numerical Analysis
Assignment 5 (due April 13, 2017)

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 plotting capabilities.

1. **[Eigenvalue/vector properties, 4pts]** Prove the following statements, using the basic definition of eigenvalues and eigenvectors, or give a counterexample showing the statement is not true. Assume $A \in \mathbb{R}^{n\times n}$, $n \geq 1$.

   (a) If λ is an eigenvalue of A and $\alpha \in \mathbb{R}$, then $\lambda + \alpha$ is an eigenvalue of $A + \alpha I$, where I is the identity matrix.

   (b) If λ is an eigenvalue of A and $\alpha \in \mathbb{R}$, then $\alpha\lambda$ is an eigenvalue of αA.

   (c) If λ is an eigenvalue of A, then for any positive integer k, λ^k is an eigenvalue of A^k.

   (d) If B is "similar" to A, which means that there is a nonsingular matrix S such that $B = SAS^{-1}$, then if λ is an eigenvalue of A, it is also an eigenvalue of B. How do the eigenvectors of B relate to the eigenvectors of A?

   (e) Every matrix with $n \geq 2$ has at least two distinct eigenvalues, say λ and μ, with $\lambda \neq \mu$.

   (f) Every real matrix has a real eigenvalue.

   (g) If A is singular, then it has an eigenvalue equal to zero.

   (h) If all the eigenvalues of a matrix A are zero, then $A = 0$.

2. **[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 $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_n)^T$ to the coefficients $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_n)^T$. 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 $x_i = -1 + (2i)/n$, $i = 0, \ldots, n$. Based on the condition numbers, can this basis transformation be performed accurately?

3. **[Polynomial interpolation versus least squares fitting, 3+1pts]** Recall how Q8 in HW3 required you to find the *cubic best fit* 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 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 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?

4. **[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 $p_2 \in \boldsymbol{P}_2$.

[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.

(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.

5. **[QR-algorithm, 1+1+2+2pts]** Let A be a symmetric, tridiagonal matrix. You learned that the matrices A_k defined by the QR-algorithm converge to a diagonal matrix that is similar to (and thus has the same eigenvalues as) A. The convergence speed depends on the absolute value of the ratio of consecutive eigenvalues. Let $r \in (0, 1)$ and

$$A = \begin{bmatrix} 1 & r \\ r & 1 \end{bmatrix}$$

(a) Calculate the eigenvalues of A as a function of r (by hand).

(b) Implement the QR-algorithm using MATLAB's (or Python's) implementation of the QR-factorization, `qr()`. Your code should run for a quadratic matrix of any size.

(c) Now define a tolerance, e.g., $\tau = 10^{-10}$. Introduce a stopping criterion in your code, causing it to stop when the maximal difference between the true eigenvalues of A and the diagonal entries of A_k is smaller than τ.[2]

(d) Use your code with the matrix given for at least five values of $r \in (0, 1)$ and make a plot with r versus the number of iterations needed to achieve the given tolerance. Explain your findings by examining the ratio between the eigenvalues of A using (a).

Please also hand in your code.

6. **[Inverse Iteration and Rayleigh Quotient, 2+2pts]** Given $\boldsymbol{x} \in \mathbb{R}^n$ and a symmetric matrix $A \in \mathbb{R}^{n\times n}$. Then the Rayleigh Quotient $R(\boldsymbol{x})$ is defined by

$$R(\boldsymbol{x}) = \frac{\boldsymbol{x}^T A \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}}$$

(a) Let $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ be an orthonormal basis of eigenvectors of A corresponding to the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, and $\boldsymbol{x} = \sum_{j=1}^{n} \alpha_j \boldsymbol{v}_j$. Show that

$$R(\boldsymbol{x}) = \frac{\sum_{j=1}^{n} \alpha_j^2 \lambda_j}{\sum_{j=1}^{n} \alpha_j^2}.$$

[2] You might want to sort the true and numerically computed eigenvalues before comparing them using `sort`.

(b) You already implemented the inverse iteration in HW4. Remember that it uses an initial *eigenvalue guess*, $\theta \in \mathbb{R}$. In class we discussed that given an approximation of an eigenvector $\boldsymbol{v}$, the Rayleigh Quotient is an estimate for the corresponding eigenvalue. Let

$$A = \begin{bmatrix} -2 & 1 & 4 \\ 1 & 1 & 1 \\ 4 & 1 & -2 \end{bmatrix}.$$

Start with your code from Q2(c), HW4. However, instead of fixing θ, calculate $\theta_k = R(\boldsymbol{x}_k)$ at each iteration and use it for the shift θ (this method is called the *Rayleigh Quotient Iteration*). Using the starting vector $\boldsymbol{x}_0 = (1, 2, -1)^T$ compare the first three iterates of your original implementation using a fixed $\theta = 2$ with the implementation using the Rayleigh Quotient. Which method converges faster? Please also hand in your code.