Fall 2018: Numerical Analysis
Assignment 2 (due Oct. 11, 2018)

1. **[2+2+1+1pt]** Newton's method computes the new iterate x_{k+1} as the x-intercept of the "line of best fit" through the point $(x_k, f(x_k))$, i.e., the line that passes through $(x_k, f(x_k))$ and whose first derivative is $f'(x_k)$. We will define a new method which finds the "quadratic of best fit" and uses it to compute the new iterate.
   (a) Find the quadratic of best fit through the point $(x_k, f(x_k))$, i.e., find the quadratic that goes through $(x_k, f(x_k))$ and whose first and second derivatives at x_k agree with $f'(x_k)$ and $f''(x_k)$, respectively.
   (b) Write down the new-Newton's method by finding the x-intercept for the quadratic of best fit.
   (c) What is the order of convergence for this method? (No justification required.)
   (d) How many steps are required for this method to find the solution of $f(x) = 0$, where f is a quadratic?

2. **[1+1+2+2pt]** Let $f : \mathbb{R}^2 \mapsto \mathbb{R}^2$ defined by $f(x, y) = (f_1(x, y), f_2(x, y))^T$, where

$$f_1(x, y) = x^2 + 4y^2 - 4, \quad f_2(x, y) = 2y - \sqrt{3}x^2.$$

   We want to find the roots of f, i.e., all pairs $(x, y) \in \mathbb{R}^2$ such that $f(x, y) = (0, 0)^T$.
   (a) Sketch or plot the sets $\mathcal{S}_i = \{(x, y) \in \mathbb{R}^2 : f_i(x, y) = 0\}$, $i = 1, 2$, i.e., the set of all zeros of f_1 and f_2. What geometrical shapes do these sets have?
   (b) Calculate analytically the roots of f, i.e., the intersection of the sets $\mathcal{S}_1$ and $\mathcal{S}_2$.
   (c) Calculate the Jacobian of f, defined by

$$J_f(x, y) = \begin{pmatrix} \partial_x f_1(x, y) & \partial_y f_1(x, y) \\ \partial_x f_2(x, y) & \partial_y f_2(x, y) \end{pmatrix} \in \mathbb{R}^{2\times 2}.$$

   Here, $\partial_x f_i(x, y)$ and $\partial_y f_i(x, y)$, $i = 1, 2$ denote the partial derivatives of f_i with respect to x and y, respectively.
   (d) The Newton method in 2D is as follows: Starting from an initial value $(x_0, y_0)^T \in \mathbb{R}^2$, compute the iterates

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} - [J_f(x_k, y_k)]^{-1} f(x_k, y_k), \text{ for } k = 0, 1, \dots,$$

   where $[J_f(x_k, y_k)]^{-1}$ is the inverse of the Jacobi matrix of f evaluated at (x_k, y_k). Implement the Newton method in 2D and use it to calculate the first 5 iterates for the starting values $(x_0, y_0) = (2, 3)$ and $(x_0, y_0) = (-1.5, 2)$. Plot these iterates in the xy-plane together with the curves $\mathcal{S}_1$ and $\mathcal{S}_2$. Please also hand in your code.[1]

[1] Some useful syntax: The MATLAB commands `b=[1;2]` and `A=[1, 2; 3, 4]` create the column vector $\boldsymbol{b} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ and the 2-by-2 matrix $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$. Moreover, `A*b` is a simple matrix multiplication and to obtain $A^{-1}\boldsymbol{b}$, you can use either `inv(A)*b`, which inverts the matrix A, or (much better!) the command `A\b`, which solves the linear system $A\boldsymbol{x} = \boldsymbol{b}$. You can use the command `surf` to make surface plots.

3. **[3pt]** Given is a tridiagonal matrix, i.e., a matrix with nonzero entries only in the diagonal, and the first upper and lower subdiagonals:

$$A = \begin{bmatrix} a_1 & c_1 & & & \\ b_1 & a_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-2} & a_{n-1} & c_{n-1} \\ & & & b_{n-1} & a_n \end{bmatrix}.$$

Assuming that A has an LU decomposition $A = LU$ with

$$L = \begin{bmatrix} 1 & & & \\ d_1 & 1 & & \\ & \ddots & \ddots & \\ & & d_{n-1} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} e_1 & f_1 & & \\ & \ddots & \ddots & \\ & & e_{n-1} & f_{n-1} \\ & & & e_n \end{bmatrix},$$

derive recursive expressions for d_i, e_i and f_i.

4. **[1+2pt]** We study basic properties of the LU-factorization.

   (a) Give an example of an invertible 3×3 matrix that does not have any zero entries, for which the LU decomposition without pivoting fails.

   (b) Show that the LU factorization of an invertible matrix $A \in \mathbb{R}^{n \times n}$ is unique. That is, if

   $$A = LU = L_1 U_1$$

   with upper triangular matrices U, U_1 and unit lower triangular matrices L, L_1, then necessarily $L = L_1$ and $U = U_1$. You can use the results we discussed in class about products of lower/upper triangular matrices, and their inverses.

5. **[4pt]** For a given dimension n, fix some k with $1 \leq k \leq n$. Now let $L \in \mathbb{R}^{n \times n}$ be a non-singular lower triangular matrix and let the vector $\boldsymbol{b} \in \mathbb{R}^n$ be such that $b_i = 0$ for $i = 1, 2, \ldots, k$.

   (a) Let the vector $\boldsymbol{y} \in \mathbb{R}^n$ be the solution of $L\boldsymbol{y} = \boldsymbol{b}$. Show, by partitioning L into blocks, that $y_j = 0$ for $j = 1, 2, \ldots, k$.

   (b) Use this to give an alternative proof of Theorem 2.1(iv), i.e., that the inverse of a non-singular lower triangular matrix is itself lower triangular.

6. **[4pt]** Let $n \geq 2$. Consider a matrix $A \in \mathbb{R}^{n \times n}$ for which every leading principal submatrix of order less than n is non-singular.

   (a) Show that A can be factored in the form $A = LDU$, where $L \in \mathbb{R}^{n \times n}$ is unit lower triangular, $D \in \mathbb{R}^{n \times n}$ is diagonal and $U \in \mathbb{R}^{n \times n}$ is unit upper triangular.

   (b) If the factorization $A = LU$ is known, where L is unit lower triangular and U is upper triangular, show how to find the LU-factors of the transpose A^T. Note that our requirement for an LU-factorization is that L is *unit* lower triangular, and U is upper triangular.

7. **[5pt]** Implement backward substitution to solve systems $U\boldsymbol{x} = \boldsymbol{b}$, i.e., write a function `x = backward(A,b)`, which expects as inputs an upper triangular matrix $U \in \mathbb{R}^{n\times n}$, and a right hand side vector $\boldsymbol{b} \in \mathbb{R}^n$, which returns the solution vector $\boldsymbol{x} \in \mathbb{R}^n$. The function should find the size n from the vector $\boldsymbol{b}$ and also check if the matrix and the vector sizes are compatible before it starts to solve the system. Please hand in your code. Apply your program for the computation of for $\boldsymbol{x} \in \mathbb{R}^4$, with

$$U = \begin{bmatrix} 1 & 2 & 6 & -1 \\ 0 & 3 & 1 & 0 \\ 0 & 0 & 4 & -1 \\ 0 & 0 & 0 & 2 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} -1 \\ -3 \\ -2 \\ 4 \end{bmatrix}.$$

8. **[3+2pt]** LU factorization without pivoting.

   (a) Implement the LU factorization using $(2.18), (2.19)$ from the textbook (hence assuming no permutations are required), and apply it to the matrix

   $$A = \begin{bmatrix} 6 & 2 & 1 & -1 \\ 2 & 4 & 1 & 0 \\ 1 & 1 & 4 & -1 \\ -1 & 0 & -1 & 3 \end{bmatrix}.$$

   (b) Generalize your code to handle input matrices A of any size $n \geq 2$. To avoid division by very small numbers or zero, check at each step that the absolute value of u_{jj} in (2.18) is not smaller than 10^{-8}. If it is, display an error message[2] and stop the code. Please also hand in your code.

9. **[2+2+2pt]** Let us use the LU-decomposition to compute the inverse of a matrix[3].

   (a) Describe an algorithm that uses the LU-decomposition of an $n \times n$ matrix A for computing A^{-1} by solving n systems of equations (one for each unit vector).

   (b) Calculate the floating point operation count of this algorithm.

   (c) Improve the algorithm by taking advantage of the structure (i.e., the zero entries—see question 5a) of the right-hand side. What is the new algorithm's floating point operation count?

[2]MATLAB has the command `error('message')` for doing that.

[3]This also illustrates that computing a matrix inverse is significantly more expensive than solving a linear system.