National University of Computer and Emerging Sciences

Islamabad Campus

Numerical Computing (CS-2008)

Date: May 21, 2024

Course Instructors

Mukhtar Ullah, Muhammad Ali, Imran Ashsraf, Almas Khan

Final Exam

Total Time (Hrs): 3

Total Marks: 84

Total Questions: 4

Roll No ____________ Section ____________ Student Signature ____________

Attempt all the questions
Use answer sheet to answer all questions
DO NOT WRITE BELOW THIS LINE

Question # 1 [Marks = 16]

(a) Perform LU factorization by hand to write matrices P, L, and U for the matrix: (8)

$$\begin{bmatrix} 1 & -2 & 0 \\ -3 & -2 & 1 \\ 0 & -2 & 8 \end{bmatrix}$$

Solution

Perform the row operations to get the upper triangular matrix

$$\begin{bmatrix} 1 & -2 & 0 \\ -3 & -2 & 1 \\ 0 & -2 & 8 \end{bmatrix} \xrightarrow{R_2 \leftarrow R_2 + 3R_1} \begin{bmatrix} 1 & -2 & 0 \\ 0 & -8 & 1 \\ 0 & -2 & 8 \end{bmatrix} \xrightarrow{R_3 \leftarrow R_3 - R_2/4} \begin{bmatrix} 1 & -2 & 0 \\ 0 & -8 & 1 \\ 0 & 0 & 31/4 \end{bmatrix} = U$$

Collect the multipliers from the row operations in the lower triangular matrix

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 1/4 & 1 \end{bmatrix}$$

Where p is identity matrix, because of no swaping.

(b) Write python code for LU decomposition with partial pivoting for a general matrix A to show P, L, and U. (8)

- **Python Code**:

```python
import numpy as np

def lu_decomposition_with_pivoting(A):
    n = A.shape[0]
    L = np.zeros((n, n))
    U = A.copy()
    P = np.eye(n)

    for i in range(n):
        # Partial pivoting
        max_row = np.argmax(np.abs(U[i:n, i])) + i
        if i != max_row:
```

```
            # Swap rows in U
            U[[i, max_row], :] = U[[max_row, i], :]
            # Swap rows in P
            P[[i, max_row], :] = P[[max_row, i], :]
            if i > 0:
                # Swap rows in L, but only the first i columns
                L[[i, max_row], :i] = L[[max_row, i], :i]

        # Compute L and U
        for j in range(i+1, n):
            L[j, i] = U[j, i] / U[i, i]
            U[j, i:] -= L[j, i] * U[i, i:]

    np.fill_diagonal(L, 1)
    return P, L, U
```

Question # 2 [Marks = 18]

Consider the following data:

$$A = \begin{bmatrix} 10 & -1 & 0 \\ -1 & 10 & -1 \\ 0 & -1 & 10 \end{bmatrix}, b = \begin{bmatrix} 4 \\ 8 \\ 20 \end{bmatrix}, x^{(0)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

(a) Using Gauss-Seidel iterative method, approximate the solution of the linear system $Ax = b$ up to 6

Solution

$$X_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} X_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} X_j^{k} \right)$$

first iteration. $X^0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

$k=0$

$i=1$

$$X_1^{(1)} = \frac{1}{a_{11}} \left(b_1 - a_{12} X_2^{k} - a_{13} X_3^{k} \right)$$

$i=2$

$$X_2^{(1)} = \frac{1}{a_{22}} \left(b_2 - \underline{\underline{a_{21} X_1^{k+1}}} - a_{23} X_3^{k} \right)$$

$i=3$

$$X_3^{(1)} = \frac{1}{a_{33}} \left(b_3 - \underline{\underline{a_{31} X_1^{k+1}}} - \underline{\underline{a_{32} X_2^{k+1}}} \right)$$

$K=0$

$x_1^{(1)} = \frac{1}{10}(4-(-1)(1)-0) = 0.5$

$x_2^{(1)} = \frac{1}{10}(8-(-1)(0.5)-(-1)(1)) = 0.95$

$x_3^{(1)} = \frac{1}{10}(20-0-(-1)(0.95)) = 2.095$

$\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow$

$k=1$

$x_1^{(2)} = \frac{1}{10}(4-(-1)(0.5)) = 0.45$

$x_2^{(2)} = \frac{1}{10}(8-(-1)(0.45)-(-1)(2.095)) = 1.0545$

$x_3^{(2)} = \frac{1}{10}(20+1.0545) = 2.10545$

$X^{(2)} = [0.45, 1.0545, 2.10545]$

$k=2$

$X^{(3)} = [0.5059, 1.06118, 2.106118]$

$k=3$ $X^{(4)} = [0.506118, 1.0612236, 2.10612236]$

$k=4$ $X^{(5)} = [0.50612236, 1.061224497, 2.10612245]$

$k=5$

$X^{(6)} = [0.50612245, 1.06122449, 2.10612245]$

(b) Calculate $\|x^{(2)} - x^{(1)}\|_2$ (4)

$\|X^{(2)} - X^{(1)}\|_2$.

$X^{(2)} - X^{(1)} = [-0.05, 0.1045, 0.01045]$

Now

$\|X^{(2)} - X^{(1)}\|_2 = \sqrt{(0.05)^2 + (0.1045)^2 + (0.01045)^2}$

$\simeq \underline{0.116}$

(c) Write the missing code in following function (on the answer sheet): (8)

Solution

```
def gauss_seidel(A, b, x, tol = 1.e-5, maxit = 100):
    n = len(b)
    err = 1.0
    iters = 0

    # Initialize the solution with the initial guess
    xnew = np.zeros_like(x)
    # Extract the lower triangular part of A
    M = np.tril(A)
    # Construct the upper triangular part of A
    U = A - M

    while (err > tol and iters < maxit):
        iters += 1
        # Compute the new approximation
        xnew = np.dot(npl.inv(M), b - np.dot(U, x))
        # Estimate convergence
        err = npl.norm(xnew-x)
        x = np.copy(xnew)
    return x
```

Question # 3 [Marks = 20]

(a) Consider the following three vectors in R^3:

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 8 \\ 1 \\ -6 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

(i) Apply Gram-Schmidt process to generate orthogonal vectors $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$. (9)

(ii) Using $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{u}_3$ compute the orthonormal vectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$. (3)

Q3 (a) Gram - Schmidt

Sol $u_1 = x_1 = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$

$$u_2 = x_2 - \text{Proj}_{u_1} x_2 = x_2 - \frac{x_2 \cdot u_1}{u_1 \cdot u_1} u_1$$

$$\Rightarrow u_2 = \begin{bmatrix} 6 \\ -3 \\ -6 \end{bmatrix}$$

$$u_3 = x_3 - \text{Proj}_{u_1} x_3 - \text{Proj}_{u_2} x_3$$

$$\Rightarrow u_3 = x_3 - \frac{x_3 \cdot u_1}{u_1 \cdot u_1} u_1 - \frac{x_3 \cdot u_2}{u_2 \cdot u_2} u_2$$

$$\Rightarrow u_3 = \begin{bmatrix} 4/9 \\ -2/9 \\ 5/9 \end{bmatrix}$$

(b) normalization

$$v_1 = \frac{u_1}{|u_1|} = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$$

$$v_2 = \frac{u_2}{|u_2|} = \frac{1}{3} \begin{bmatrix} 2 \\ -1 \\ -2 \end{bmatrix}$$

$$v_3 = \frac{u_3}{|u_3|} = \frac{1}{3\sqrt{5}} \begin{bmatrix} 4 \\ -2 \\ 5 \end{bmatrix}$$

Figure 1: Solution (a),(b)

(b) Comprehend the following piece of python code which contains comments which are numbered from 1 to 10. This numbering is done so that you only provide the comments in the answer sheet (without re-producing the source code in the answer sheet) to save time. Provide the missing comments with comment numbers. (8)

```
import imageio
import numpy as np
import numpy.linalg as npl

# Comment 1: Read the image into the array photo
photo = imageio.imread("Newton.jpg")

#Comment 2: extract Red, Green and Blue channels into separate matrices
Red[:,:,0] = photo[:,:,0]
Green[:,:,1] = photo[:,:,1]
Blue[:,:,2] = photo[:,:,2]

# Comment 3: perform SVD on each channel to get corresponding U, S and V
    componenets
U_r,S_r,V_r = npl.svd(Red)
U_g,S_g,V_g = npl.svd(Green)
U_b,S_b,V_b = npl.svd(Blue)

```

```
# Comment 4: set the number of singular values to be used
k=100

# Comment 5: perform compression by extracting only k dimensions from each
    component
U_r_c = U_r[:,0:k]; V_r_c = V_r[0:k,:]; S_r_c = np.diag(S_r[0:k])
U_g_c = U_g[:,0:k]; V_g_c = V_g[0:k,:]; S_g_c = np.diag(S_g[0:k])
U_b_c = U_b[:,0:k]; V_b_c = V_b[0:k,:]; S_b_c = np.diag(S_b[0:k])

# Comment 6: compute each channel back by using the compressed components
comp_img_r = np.dot(U_r_c, np.dot(S_r_c,V_r_c))
comp_img_g = np.dot(U_g_c, np.dot(S_g_c,V_g_c))
comp_img_b = np.dot(U_b_c, np.dot(S_b_c,V_b_c))

# Comment 7: zero initialize the result matrix which represents the
    computed image
comp_img = np.zeros((row, col, 3))

# Comment 8: add Red, Green and Blue channel back to the single matrix
    representing the computed image
comp_img[:,:,0] = comp_img_r
comp_img[:,:,1] = comp_img_g
comp_img[:,:,2] = comp_img_b

# Comment 9: clip values less than 0 and greater than 1
comp_img[comp_img < 0] = 0; comp_img[comp_img > 1] = 1

# Comment 10: show the comp_img
plt.imshow(comp_img)
plt.show()
```

Question # 4 [Marks = 30]

1. Gaussian elimination is
   (a) a direct method with finite precision in theory
   (b) a direct method with infinite precision in theory
   (c) an iterative method with finite precision in practice
   (d) an iterative method with infinite precision in theory
2. LU factorization is
   (a) a modification of Gaussian elimination
   (b) a decomposition into lower and upper triangular parts of a matrix
   (c) a method for forward substitution
   (d) a method for backward substitution
3. Pivoting strategies can resolve numerical issues arising in
   (a) forward substitution
   (b) backward substitution
   (c) LU factorization
   (d) all of the above
4. Naïve Gaussian elimination cannot be performed with

(a) $A_1 = \begin{bmatrix} 1 & 0 \\ 3 & 2 \end{bmatrix}$

(b) $A_2 = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$

(c) $A_3 = \begin{bmatrix} 3 & 1 \\ 2 & 0 \end{bmatrix}$

(d) $A_4 = \begin{bmatrix} 0 & 1 \\ 3 & 2 \end{bmatrix}$

5. How many cubic polynomials are typically used to construct a cubic spline with n data points?
   (a) n
   (b) $n - 1$
   (c) $2n - 1$
   (d) $n + 1$
6. What is a cubic spline used for?
   (a) Interpolation
   (b) Regression
   (c) Integration
   (d) Differentiation
7. In cubic spline interpolation, what condition must the spline satisfy at each data point?
   (a) The first derivative must be continuous
   (b) The second derivative must be continuous
   (c) Both first and second derivative must be continuous
   (d) Only the function value must be continuous
8. Which of the following is a permutation matrix?
   (a) $P_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
   (b) $P_2 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$
   (c) $P_3 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$
   (d) $P_4 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$
9. What is approximate solution of the inconsistent system $Ax = b$
   (a) $(A^TA)^{-1}A^Tb$
   (b) $(AA^T)^{-1}A^Tb$
   (c) $A^Tb(A^TA)^{-1}$
   (d) $A^Tb(AA^T)^{-1}$
10. What is the primary objective of least squares approximation?

(a) To maximize the sum of the residuals
(b) To minimize the sum of the squares of the residuals
(c) To maximize the correlation between variables
(d) To minimize the mean of the residuals

11. Failure of Cholesky factorization of a matrix A indicates that
    (a) A is upper triangular
    (b) A is lower triangular
    (c) $x^T Ax > 0$ for all nonzero vectors x
    (d) A is not positive definite
12. Which of the following is an iterative method?
    (a) Jacobi method
    (b) LU factorization
    (c) Cholesky factorization
    (d) Gaussian elimination
13. Which of the following is a numerical stable method?
    (a) Jacobi method
    (b) LU factorization
    (c) Cholesky factorization
    (d) Gauss-Seidel Method
14. Which of the following can be used to decide when pivoting is needed?
    (a) number of nonzero rows
    (b) number of nonzero columns
    (c) determinant
    (d) condition number
15. The choice between Jacobi and Gauss-Seidel methods is guided by the observation that
    (a) Jacobi method converges faster
    (b) Gauss-Seidel converges faster
    (c) Jacobi can be implemented in parallel computers
    (d) both b and c
16. Under certain conditions, pseudo-inverse of a matrix can be same as its classical inverse?
    (a) Always True
    (b) Always False
    (c) Conditionally True
    (d) Conditionally False
17. QR factorization decomposes a matrix A into which two matrices?
    (a) LU matrices
    (b) Diagonal and triangular matrices
    (c) Upper triangular and lower triangular matrices

(d) Orthogonal matrix and upper triangular matrix

18. Which of the following statements is true about the Gram-Schmidt process?
    (a) It is computationally expensive for large sets of vectors.
    (b) It can only be applied to square matrices.
    (c) It can be numerically unstable for ill-conditioned sets of vectors.
    (d) It always produces a unique set of orthonormal vectors.

19. Which of the following is NOT an advantage of using an orthonormal set of vectors in numerical computing?
    (a) Improved stability in calculations involving the vectors
    (b) Easier computation of vector norms
    (c) Simpler projection operations onto the subspace spanned by the vectors
    (d) Reduced storage requirements compared to the original set

20. Application domains of linear least squares fitting include?
    (a) Image processing
    (b) Financial modeling
    (c) Curve fitting
    (d) All of the above

21. Which of the following functions in NumPy is used to generate evenly spaced numbers over a specified range?
    (a) numpy.linspace
    (b) numpy.arange
    (c) numpy.random.rand
    (d) numpy.zeros

22. Which of the following iterative scheme is always convergent
    (a) Newton-Raphson Method
    (b) Fixed Point Iteration
    (c) Bisection Method
    (d) All of them

23. What function in NumPy can be used to implement the Gram-Schmidt process?
    (a) numpy.linalg.orthogonalize
    (b) numpy.orthogonalize
    (c) numpy.linalg.qr
    (d) numpy.linalg.gramschmidt

24. Given $x = \begin{bmatrix} 1 \\ 2 \\ -3 \end{bmatrix}$. $\|x\|_2$ is
    (a) Positive
    (b) Negative

(c) Non-negative
(d) Non-positive

25. Given $x = \begin{bmatrix} 1 \\ 2 \\ -3 \end{bmatrix}$. $\|x\|_1$ is
   (a) Positive
   (b) Negative
   (c) Non-negative
   (d) Non-positive

26. Which of the following statement about SVD is true?
   (a) It can only be applied to square matrices.
   (b) It can be used to determine the rank of any matrix.
   (c) The singular values in σ are always positive.
   (d) The columns of U and V are always linearly independent.

27. Provided $A^T A$ is non-singular, pseudo-inverse of a A exists if A is
   (a) square non-singular matrix only
   (b) square singular matrix only
   (c) any rectangular matrix
   (d) rectangular singular matrix only

28. What is the computational complexity of Gaussian elimination for solving a system of linear equations of order n?
   (a) $O(n)$
   (b) $O(n^2)$
   (c) $O(n^3)$
   (d) $O(2^n)$

29. SVD is particularly useful for image compression because:
   (a) It reduces the computational cost of storing pixel values.
   (b) It separates image information into components with varying importance.
   (c) It directly removes redundant information from the image.
   (d) None of the above

30. Which of the following statements is TRUE about square matrices?
   (a) A matrix must have all zero entries to be non-invertible.
   (b) A matrix is invertible only if it has an equal number of rows and columns.
   (c) A matrix is invertible if and only if its columns (or rows) are linearly independent.
   (d) A matrix with a determinant of 0 is always invertible.

Final_2024-05-21_key

Q No	Correct
NC 2008 - MCQs	
1	B
2	A
3	C
4	D
5	B
6	A
7	C
8	A
9	A
10	B
11	D
12	A
13	C
14	D
15	D
16	C
17	D
18	C
19	D
20	D
21	A
22	C
23	C
24	A
25	A
26	C, D
27	C
28	C
29	B
30	C

Useful Formulae and Algorithms

$$proj_{\mathbf{v}}\mathbf{a} = \frac{\mathbf{a} \cdot \mathbf{v}}{\|\mathbf{v}\|^2}\mathbf{v}$$

Figure 2: Projection of vector a on v

Algorithm 31 *LU* factorization with partial pivoting

Given the array $\boldsymbol{A}$
for $k = 1 : n - 1$ **do**
  Find p such that $|\boldsymbol{A}(p,k)| = \max_{k \le p \le n} |\boldsymbol{A}(k:n,k)|$
  Swap rows $\boldsymbol{A}(k,:) \leftrightarrow \boldsymbol{A}(p,:)$
  Swap rows $perm(k) \leftrightarrow perm(p)$
  for $i = k+1 : n$ **do**
    if $\boldsymbol{A}(i,k) \neq 0$ **then**
      $m_{ik} = \boldsymbol{A}(i,k)/\boldsymbol{A}(k,k)$
      $\boldsymbol{A}(i,k+1:n) = \boldsymbol{A}(i,k+1,n) - m_{ik} \cdot \boldsymbol{A}(k,k+1:n)$
      $\boldsymbol{A}(i,k) = m_{ik}$
    end if
  end for
end for

Figure 3: LU factorization algorithm

$$\boldsymbol{x}^{(k+1)} = (\boldsymbol{D} - \boldsymbol{L})^{-1}\boldsymbol{U}\boldsymbol{x}^{(k)} + (\boldsymbol{D} - \boldsymbol{L})^{-1}\boldsymbol{b}$$

Figure 4: Gauss-Seidel iterative method

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) \Big/ a_{ii}. \quad i = 1, \ldots, n$$

Figure 5: Gauss-Seidel iterative method

$$\ell_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{N} \frac{x - x_j}{x_i - x_j}$$

Figure 6: Lagrange polynomials

☞ Given $N+1$ nodes $x_0 < x_1 < \cdots < x_N$ and the values $f(x_i)$ and $f'(x_i)$ for $i = 0, 1, \ldots, N$, the Hermite interpolating polynomial is the polynomial

$$H_{2N+1}(x) = \sum_{i=0}^{N} [\alpha_i(x) f(x_i) + \beta_i(x) f'(x_i)] \ ,$$

where α_i and β_i are given in terms of the Lagrange polynomials as

$$\alpha_i(x) = [1 - 2\ell_i'(x_i)(x - x_i)]\ell_i^2(x) \quad \text{and} \quad \beta_i(x) = (x - x_i)\ell_i^2(x) \ .$$

Figure 7: Hermite Interpolation