SDSU - COMP521
Fall 2023

Homework 01 - Due Date: 09/08/2023

Problem 1

For each of the following series (a,b,c):

1. Determine if they are convergent. The terms in the series are positive. You can use the ratio test.
2. Write a program that computes the first 50 partial sums (up to n = 50). Plot the partial sums versus n.

The series are:

(a) $\sum_{n=1}^{\infty} \frac{(n!)^2}{(2n)!}$

(b) $\sum_{n=0}^{\infty} \frac{(3)^{2n}}{(10)^n}$

(c) $\sum_{n=1}^{\infty} e^{-n^2}$

Deliverable: For the analytical part you can scan your handwritten solution and upload it in Blackboard. For the computer work you must submit your code (please use MATLAB) and additional information in a .PDF file. "Additional information" is words describing what you did, and anything interesting that shows up.

Problem 2

Given the function:

$$f(x) = ln(3x + 1)$$

1. Write the Taylor series centered at zero for the function $f(x)$. Write a program to plot the exact function and the Taylor series approximation on $x \in [0, 0.5]$ using 2 and 5 terms. For the plots use $\Delta x = 0.001$. Explain your results.
2. Write a program that computes the L_2-norm of the error vector that results from the difference between the approximated and the exact values on the specified interval x. Compute the norm using 2 and 5 terms. Explain the differences.

Deliverable: For the analytical part you can scan your handwritten solution and upload it in Canvas as a .PDF file. For the computer work you must submit your code (please use MATLAB) and additional information in a .PDF file.

Problem 3 (5 bonus points)

Program the subroutines (algorithms 1 and 2) specified by the pseudocodes in this document. *These are algorithms from the book NUMERICAL RECIPES IN C: THE ART OF SCIENTIFIC COMPUTING.*

- **Algorithm 1:** Given a $N \times N$ matrix A denoted as $\{a\}_{i,j=1}^{N,N}$, the routine replaces it by the LU decomposition of a rowwise permutation of itself. "a" and "N" are input. "a" is also output, modified to apply the LU decomposition; $\{indx_i\}_{i=1}^{N}$ is an output vector that records the row permutation effected by the partial pivoting; "d" is output and adopts ± 1 depending on whether the number of row interchanges was even or odd. This routine is used in combination with algorithm 2 to solve linear equations or invert a matrix.
- **Algorithm 2:** Solves the set of N linear equations $\mathbf{A} \,.\, x = \mathbf{b}$. Matrix $\{a\}_{i,j=1}^{N,N}$ is actually the LU decomposition of the original matrix A, obtained from algorithm 1. Vector $\{indx_i\}_{i=1}^{N}$ is input as the permutation vector returned by algorithm 1. Vector $\{b_i\}_{i=1}^{N}$ is input as the right-hand side vector B but returns with the solution vector X. Inputs $\{a\}_{i,j=1}^{N,N}$, N, and $\{indx_i\}_{i=1}^{N}$ are not modified in this algorithm.
- **Note:** The term **Abs** in algorithm 1 is a function. It gives the absolute value of its input parameter. You can use a predefined function for the absolute value or write your own.

Deliverable

- You must submit the source file to your code. Use the programming language/environment of your preference but please specify your choice. Include the files required for compilation when necessary.
- You must test your code and submit your results in a document. The test will show the output for the following cases:
  - **Case 1**

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & -1 \\ 6 & -5 & 4 \\ -9 & 8 & -7 \end{bmatrix}; \quad \mathbf{b} = \begin{bmatrix} 2\pi \\ 5\pi \\ -8\pi \end{bmatrix}$$

  - **Case 2**

$$\mathbf{A} = \begin{bmatrix} \pi & 3\pi & 2\pi \\ 0 & 1 & -2/3 \\ -\pi & -3\pi & 2\pi \end{bmatrix}; \quad \mathbf{b} = \begin{bmatrix} 3 \\ 0 \\ -1 \end{bmatrix}$$

Input: Matrix $\{a_{i,j}\}_{i,j=1}^{N,N}$ [*double*] ; Scalar N [*integer*]

Output: Vector $\{indx_i\}_{i=1}^{N}$ [*integer*] ; Scalar d [*double*]; Matrix $\{a_{i,j}\}_{i,j=1}^{N,N}$ *modified* [*double*]

```
Declare : i, imax, j, k [integer]
Declare : big, dum, sum, temp [double]
Declare : {vv}_{j=1}^{N} [double]
Define : TINY = 1.0e − 20
d ← 1.0
for i = 1 to N do
    big ← 0.0
    for j = 1 to N do
        temp ← Abs(a_{i,j})
        if temp > big then
            big ← temp
        end
    end
    if big = 0.0 then
        Print message "Singular matrix"
        Exit Algorithm
    end
    vv_i ← 1.0/big
end
for j = 1 to N do
    for i = 1 to (j − 1) do
        sum ← a_{i,j}
        for k = 1 to (i − 1) do
            sum ← sum − a_{i,k} ∗ a_{k,j}
        end
        a_{i,j} ← sum
    end
    big ← 0.0
    for i = j to N do
        sum ← a_{i,j}
        for k = 1 to (j − 1) do
            sum ← sum − a_{i,k} ∗ a_{k,j}
        end
        a_{i,j} ← sum
        dum ← vv_i ∗ Abs(a_{i,j})
        if dum ≥ big then
            big ← dum
            imax ← i
        end
    end
    if j ≠ imax then
        for k = 1 to N do
            dum ← a_{imax,k}
            a_{imax,k} ← a_{j,k}
            a_{j,k} ← dum
        end
        d ← −d
        vv_{imax} ← vv_j
    end
    indx_j ← imax
    if a_{j,j} = 0.0 then
        a_{j,j} ← TINY
    end
    if j ≠ N then
        dum ← 1.0/a_{j,j}
        for i = (j + 1) to N do
            a_{i,j} ← a_{i,j} ∗ dum
        end
    end
end
return {indx_i}_{i=1}^{N} , d, {a_{i,j}}_{i,j=1}^{N,N}
```

Algorithm 1: LUDECMP: LU Decomposition of a matrix A

Input: Vector $\{b_i\}_{i=1}^{N}$ [*double*]; Matrix $\{a_{i,j}\}_{i,j=1}^{N,N}$ [*double*] ;
Scalar N [*integer*]; Vector $\{indx_i\}_{i=1}^{N}$ [*integer*]

Output: Vector $\{b_i\}_{i=1}^{N}$ with the solution

Declare : i, ii, ip, j [*integer*]
$ii \leftarrow 0$
for $i = 1$ ***to*** N **do**
    $ip \leftarrow indx_i$
    $sum \leftarrow b_{ip}$
    $b_{ip} \leftarrow b_i$
    if (ii) **then**
        for $j = ii$ ***to*** $(i-1)$ **do**
            $sum \leftarrow sum - a_{i,j} * b_j$
        end
    else if (sum) **then**
        $ii \leftarrow i$
    end
    $b_i \leftarrow sum$
end
for $i = N$ ***to*** 1 **do**
    $sum \leftarrow b_i$
    for $j = (i+1)$ ***to*** N **do**
        $sum \leftarrow sum - a_{i,j} * b_j$
    end
    $b_i \leftarrow sum/a_{i,i}$
end
return $\{b_i\}_{i=1}^{N}$

Algorithm 2: LUBKSB: Linear system $Ax = b$ solution using backsubstitution after using LU Decomposition of a matrix A