**Computational Sciences and Informatics Course 690**

**Assignment 2**

**Jericho McLeod**

**Introduction:**

This paper reflects assigned work in CSI-690 pertaining to the use of Taylor expansions to estimate functions, determine the error of Taylor series expansions, determine matrix norms, and solve a matrix system using Gaussian elimination algorithmically and by hand. The purpose of these assignments is to further reinforce the use of numerical methods for the evaluation and implementation of algorithms.

**Methods:**

**Problem 1:**

The method used for the Taylor series expansion it the given formula:

The method used for absolute and relative error can be given as:

The Taylor series will be used iteratively by order and compared to the actual value obtained until the error has reached the desired level.

**Problem 2:**

The method for determining the maximum order for a Taylor series with a given error rate will rely upon the remainder estimation theorem:

Where M is the maximum value of in the interval, as explained by Dr. Klipper, and solving the inequality. (Klipper, 2014)

**Problem 3:**

Matrix Norms will be calculated using the following formulae:

**Problem 4:**

Gaussian elimination will be accomplished using the following provided algorithm:

DOFOR k = 1, n -1

DOFOR i = k + 1, n

Factor = aij / akk

DOFOR j = k + 1 to n

aij = aij – ( factor \* akj )

END DO

bi = bi – ( factor \* bk )

END DO

END DO

Xn = bn / an,n

DOFOR i = n – 1, 1, -1

Sum = bi

DOFOR j = i + 1, n

sum = sum – ( aij \* xj )

END DO

Xi = sum / aii

END DO

LU Decomposition will be achieved using the following provided algorithm:

SUB Decompose (a, n)

DOFOR k = 1, n-1

DOFOR i = k +1, n

Factor = aik/akk

aik = Factor

DOFOR j = k + 1, n

aij = aij – ( Factor \* akj )

END DO

END DO

END DO

These algorithms were both obtained in course textbook. (Chapra & Canale, 2016)

The error of these algorithms will be evaluated based on the relationship of Ax to B, based on the input formula Ax=B, using the solved values in place of x and evaluating B-B̂̂ for absolute error, then dividing absolute error by the actual values to the expressions to obtain relative error.

**Results:**

The assigned problems’ solutions are as follows; for additional information detailing the arrival at these solutions please see Appendix A.

The first assigned problem was to use zero through third order Taylor series expansions to predict f(3) for the function , given a base point at x=1, then to compute the true percent error (εt) at each approximation.

The results for this problem were:

* 0th order value and relative error: -62, 111.19%
* 1st order value and relative error: 78, 85.92%
* 2nd order value and relative error: 354, 36.10%
* 3rd order value and relative error: 554, 0.0%

The second assigned problem asked required the determination of which order of a Taylor Series expansion would be the highest attainable while retaining error less than or equal to 0.015. The formula given was , where the base point for the interval .

Solving using gives the solution as the highest order of a Taylor Expansion that will retain error of 0.015 or less.

The third assigned problem was to determine ||A||1, ||A||2, and ||A||∞ for the matrix [A] given below and to scale the matrix by making the maximum element in each row equal to one.

* The matrix scaled such that the highest value is 1 is:
* The maximum magnitude norm, or ||A||∞ , for the matrix is .
* The 2-norm, also called the Euclidean Norm is:
* The 1-norm, or column sum norm, is:

The fourth and final problem was to solve the set of linear equations listed below with Gaussian Elimination both analytically and by writing code, complete LU factorization, and to determine the error in the computation. The code written for this problem is available in Appendix B.

2x1 + 3x2 - 2x3 = 2

-x1 + x2 + x3 = 4

x1 + 3x2 + 4x3 = 8

Gaussian Elimination completed analytically, verified with python, then verified with an original Gaussian Elimination algorithm implementation:

The LU Factorization of this matrix yields:

The absolute error of this matrix is 0, thus so is the relative error, indicating there is no introduced error in this solution.

# **Bibliography**

Chapra, S. C., & Canale, R. P. (2016). *Numerical Methods for Engineers, 7th Edition.* McGraw-Hill Education.

Klipper, M. (2014, 12 2). *Math 2260: Calculus II For Science And Engineering.* Retrieved from UGA.edu: http://alpha.math.uga.edu/~mklipper/2260/F14/taylorerror.pdf

**Appendix A:**

**Problem 1:**

*The true value of f(x) = 25x^3 - 6x^2 + 7x – 88 when f(3) =25(3)^3 – 6(3)^2 + 7(3) – 88 = 554*

*The derivatives are:*

*f’(x) = 75x^2 – 12x + 7*

*f’’(x) = 150x – 12*

*f’’’(x) = 150*

*0th order: f(3) =25(1)^3 – 6(1)^2 7(1) – 88 = -62*

*True Percent Relative Error: (554—62)/554 = 111.19%*

*1st order: f(3) = (-62) + (75(1)^2 -12+7) \* (3-1) = 78*

*True Percent Relative Error: (554-78)/554 = 85.92%*

*2nd order: f(3) = 78 + ((150(1) – 12) / 2!) \* (3-1)^2 = 354*

*True Percent Relative Error: (554-354)/554 = 36.10%*

*3rd order: f(3) = 354 +150/3! \*(3-1)^3 = 554*

*True Percent Relative Error: (554-554)/554 = 0.0%*

**Problem 2:**

*Finding the highest derivative order necessary to keep error to 0.015 can be completed by using the Remainder Estimation Theorem:*

*Where M is the maximum value of f n+1in the interval.*

*The derivatives of the function are:*

*f’(x) = 1 - 0.5cos(x) = 1*

*f’’(x) = -0.5 -sin(x) = 0.5*

*f’’’(x) = -0.5 -cos(x) = 0*

*f’’’’(x) = -0.5sin(x) = -0.5*

*f’’’’’(x) = -0.5cos(x) = 0 ….*

*In this case, f(π) is the highest value in the interval, thus M = π – 1 - .5sin(π) = 2.14159.*

*Resolving this calculation with the desired level of error, 0.015, yields:*

*0.015 <= 2.14159 / (n+1)!*

*Rearranged, (n+1)! <= 2.14159 / .015*

*(n+1)! <= 142.77*

*(n+1)! <= 6!*

*n+1 <= 6*

*n<= 5*

**Problem 3:**

*Scale the matrix:*

*determination:*

*Choose the maximum of [2, 1.444444, 1.466666] = 2.*

*determination:*

*determination:*

*Choose the maximum of [2.8, 0.377777, 1.733333] = 2.8*

**Problem 4:**

*The equations:*

*2x1 + 3x2 - 2x3 = 2*

*-x1 + x2 + x3 = 4*

*x1 + 3x2 + 4x3 = 8*

*Organized as a Matrix and use Gaussian Elimination:*

*Midpoint of manual solution:*

*Gaussian Elimination completed manually:*

*The python script used to initially verify the results:*

*A = np.matrix([[2,3,-2],[-1,1,1],[1,3,4]])*

*b = np.matrix([[2],[4],[8]])*

*x=np.linalg.solve(A,b)*

*print(x)*

*np.allclose(np.dot(A,x),b) # Prints ‘True’ if the system is accurately solved*

*And its results:*

*[[-1.2]*

*[ 2. ]*

*[ 0.8]]*

*Out[44]: True*

*The error calculations:*

**Appendix B:**

‘’’Solving system for X using NumPy’’’

import numpy as np

# Input Matrices

A = np.matrix([[2,3,-2],[-1,1,1],[1,3,4]])

b = np.matrix([[2],[4],[8]])

# Use Numpy to solve for X vector

x=np.linalg.solve(A,b)

print(x)

# Use Numpy to verify X vector is a valid solution

function = np.allclose(np.dot(A,x),b)

print(function)

‘’’ Solve for X algorithmically ‘’’

# Input Matrices

A = np.matrix([[2.0,3.0,-2.0],[-1.0,1.0,1.0],[1.0,3.0,4.0]])

b = np.matrix([[2.0],[4.0],[8.0]])

# Solve for X algorithmically

n=len(A)

for k in range(0,n-1):

for i in range(k+1,n):

factor = A[i,k]/A[k,k]

for j in range(k+1,n):

A[i,j]=A[i,j]-(factor\*A[k,j])

b[i]=b[i]-(factor\*b[k])

for i in range(n-1,1,-1):

sum\_=b[i]

for j in range(i+1,n):

sum\_=sum\_-A[i,j]\*(b[j]/A[j,j])

x[i]=sum\_/A[i,i]

print(x)

#source: Numerical Methods for Engineers, 7th Ed, by Chapra and Canale, pg254

‘’’LU Factorization using Scipy’’’

import scipy

A = scipy.array([[2.0,3.0,-2.0],[-1.0,1.0,1.0],[1.0,3.0,4.0]])

P, L, U = scipy.linalg.lu(A)

print(A)

print()

print(P)

print()

print(L)

print()

print(U)

‘’’Calculate Residual using machine:

Note: this will only be effective to the level of machine precision’’’

A = np.array([[2.0,3.0,-2.0],[-1.0,1.0,1.0],[1.0,3.0,4.0]])

B = np.array([2.0,4.0,8.0])

X = np.array([-1.2,2,0.8])

Residual = np.dot(A,X) - B

print(Residual)