Assignment 3

Arifur Rahman
400300356
Feb 11, 2023

```
from tabulate import tabulate
import numpy as np

def generateHb(n):
    H = []
    for i in range(n):
        row = []
        for j in range(n):
            row.append(1/(i + j + 1))
        H.append(row)

    x = [1 for i in range(n)]
    b = [0 for i in range(n)]
    for i in range(n):
        for j in range(n):
            b[i] += H[i][j] * x[j]

    return (H, b)

def forward_substitution(L, b):
    n = len(b)
    x = [[0.0] * n for i in range(n)]
    for i in range(n):
        if L[i][i] == 0.0:
            return None
        sum = 0
        for j in range(i):
            sum += L[i][j] * x[j]
        x[i] = (b[i] - sum) / L[i][i]
    return x

def backward_substitution(U, b):
    n = len(b)
    x = [[0.0] * n for i in range(n)]
    for i in range(n - 1, -1, -1):
        if U[i][i] == 0.0:
            return None
        sum = 0.0
        for j in range(i + 1, n):
            sum += U[i][j] * x[j]
        x[i] = (b[i] - sum) / U[i][i]
```

```
    return x



def upper_triangular_matrix_U(k, n, L, U, A):
    # Get the upper triangular matrix U
    for j in range(k, n):
        sum = 0.0
        for p in range(k):
            sum += L[k][p] * U[p][j]
        U[k][j] = A[k][j] - sum
    # print(f"upper triangular matrix step {k+1}: {U}")

def lower_triangular_matrix_L(k, n, L, U, A):
    # Get the lower triangular matrix L
    for i in range(k + 1, n):
        sum = 0.0
        for p in range(k):
            sum += L[i][p] * U[p][k]
        L[i][k] = (A[i][k] - sum) / U[k][k]
    # print(f"Lower triangular matrix step {k+1}: {L}\n")

def LU_decomposition(A):
    n = len(A)
    L = [[0.0] * n for i in range(n)]
    U = [[0.0] * n for i in range(n)]

    for k in range(n):
        if A[k][k] == 0:
            return None

        L[k][k] = 1
        upper_triangular_matrix_U(k, n, L, U, A)
        lower_triangular_matrix_L(k, n, L, U, A)

    return L, U

def gauss_elimination(A, b):

    L, U = LU_decomposition(A)
    y = forward_substitution(L, b)
    x = backward_substitution(U, y)
    return x

def residual_norm(H, x, b):
    return np.linalg.norm(b - np.dot(H, x), np.inf)

def error_norm(x, x_true):
```

```
    return np.linalg.norm(x - x_true, np.inf)
def condition_number(H):
    return np.linalg.cond(H, np.inf)

def toPrecisionString(val, pre):
    return np.format_float_positional(np.float32(val), unique=False,
precision=pre)


if __name__ == '__main__':
    # Generate Hilbert Matrix
    print("")
    print("======================================== Generate Hilbert Matrix
====================================|")
    print(f"|n             | error                 |
residual                                  | Cond(H)         |")
    print("================================================================================
==================================|")
    for n in range(2, 100):
        H, b = generateHb(n)
        x = np.ones(n)
        x_approx = gauss_elimination(H, b)
        res = residual_norm(H, x_approx, b)
        err = error_norm(x, x_approx)
        error_percent = 100 * (err / np.linalg.norm(x_approx, np.inf))
        cond = condition_number(H)
        if error_percent <= 100:
            print(f"n = {n}         | error% = {toPrecisionString(error_percent,
3)}         | residual = {toPrecisionString(res, 20)}           | Cond(H) =
{toPrecisionString(cond, 2)}")
        else:
            break
    print("")
```

Output:

N	Error%	Residual	Cond(H)
n = 2	error% = 0.000	0.00000000000000000000	27.00
n = 3	error% = 0.000	0.00000000000000011102	748.00
n = 4	error% = 0.000	0.00000000000000011102	28375.00
n = 5	error% = 0.000	0.00000000000000044409	943656.00
n = 6	error% = 0.000	0.00000000000000044409	29070280.00
n = 7	error% = 0.000	0.00000000000000022204	985194880.00
n = 8	error% = 0.000	0.00000000000000000000	33872791552.00
n = 9	error% = 0.003	0.00000000000000022204	1099652005888.00
n = 10	error% = 0.060	0.00000000000000044409	35356846063616.00
n = 11	error% = 1.829	0.00000000000000044409	1234532783095808.00
n = 12	error% = 1.145	0.00000000000000044409	42553994802888704.00
n = 13	error% = 94.463	0.00000000000000088818	778165816821547008.00
n = 14	error% = 95.048	0.00000000000000133227	1148964018661097472.00
n = 15	error% = 98.851	0.00000000000000044409	1041727000336662528.00
n = 16	error% = 90.425	0.00000000000000066613	10083374847729074176.00
n = 17	error% = 92.067	0.00000000000000044409	2644625356597755904.00
n = 18	error% = 99.446	0.00000000000000066613	2202886939412004864.00
n = 19	error% = 96.537	0.00000000000000044409	2299806552938250240.00

Actual Console Output:

```
======================================= Generate Hilbert Matrix ====================================|
|n             | error                  | residual                                    | Cond(H)         |
====================================================================================================|
n = 2         | error% = 0.000          | residual = 0.00000000000000000000          | Cond(H) = 27.00
n = 3         | error% = 0.000          | residual = 0.00000000000000011102          | Cond(H) = 748.00
n = 4         | error% = 0.000          | residual = 0.00000000000000011102          | Cond(H) = 28375.00
n = 5         | error% = 0.000          | residual = 0.00000000000000044409          | Cond(H) = 943656.00
n = 6         | error% = 0.000          | residual = 0.00000000000000044409          | Cond(H) = 29070280.00
n = 7         | error% = 0.000          | residual = 0.00000000000000022204          | Cond(H) = 985194880.00
n = 8         | error% = 0.000          | residual = 0.00000000000000000000          | Cond(H) = 33872791552.00
n = 9         | error% = 0.003          | residual = 0.00000000000000022204          | Cond(H) = 1099652005888.00
n = 10         | error% = 0.060          | residual = 0.00000000000000044409          | Cond(H) = 35356846063616.00
n = 11         | error% = 1.829          | residual = 0.00000000000000044409          | Cond(H) = 1234532783095808.00
n = 12         | error% = 1.145          | residual = 0.00000000000000044409          | Cond(H) = 42553994802888704.00
n = 13         | error% = 94.463          | residual = 0.00000000000000088818          | Cond(H) = 778165816821547008.00
n = 14         | error% = 95.048          | residual = 0.00000000000000133227          | Cond(H) = 1148964018661097472.00
n = 15         | error% = 98.851          | residual = 0.00000000000000044409          | Cond(H) = 1041727000336662528.00
n = 16         | error% = 90.425          | residual = 0.00000000000000066613          | Cond(H) = 10083374847729074176.00
n = 17         | error% = 92.067          | residual = 0.00000000000000044409          | Cond(H) = 2644625356597755904.00
n = 18         | error% = 99.446          | residual = 0.00000000000000066613          | Cond(H) = 2202886939412004864.00
n = 19         | error% = 96.537          | residual = 0.00000000000000044409          | Cond(H) = 2299806552938250240.00
```