Código disponível em: [Código - 187793 - Pastebin.com]

Estou usando as seguintes bibliotecas no meu código

```
import numpy as np
from numpy import linalg as LA
```

1.
d)

```
L = np.tril(A)
U = np.triu(A, 1)

C = -LA.inv(L) @ U
g = LA.inv(L) @ b

print("C_gs =\n", tabulate.tabulate(C, tablefmt="fancy_grid", floatfmt=".4f"), sep = '')
print(LA.norm(C, np.inf))
```

A nesse caso é a matriz da função gs (que foi passado M), definido da seguinte maneira `def gs(A, b, eps):` e invocada assim `gs(M, d, 1e-2)`

O resultado desses prints foram

```
C_gs =
```

0.0000	0.0714	0.0000	0.0000	0.1429	0.1429	0.0000	0.1429
0.0000	0.0102	0.0000	0.0000	0.0918	0.0204	0.0000	0.0204
0.0000	0.0034	0.0000	0.1667	0.0306	0.0068	0.0000	0.0068
0.0000	0.0006	0.0000	0.0278	0.0051	0.0011	0.0000	0.1678
0.0000	0.0009	0.0000	0.0432	0.0079	0.0018	0.0000	0.0388
0.0000	0.0009	0.0000	0.0417	0.0077	0.0017	0.5000	0.0017
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0294	0.0000	0.0420	0.0649	0.0589	0.0000	0.0663

```
0.5535714285714285
```

e nossa Norma - inf (C_gs) é 0.5535714285714285

e) Definição da função:

```python
def jacobi(A, b, eps):
    n = len(A)

    D = np.zeros(n * n).reshape(n, n)
    C = np.zeros(n * n).reshape(n, n)
    for i in range(n):
        D[i][i] = A[i][i].copy()
        for j in range(n):
            if i != j:
                C[i][j] = A[i][j].copy()
    C = -LA.inv(D) @ C
    g = LA.inv(D) @ b

    print("C_j =\n", tabulate.tabulate(C, tablefmt="fancy_grid", floatfmt=".4f"), sep = '')
    print("Norm-inf (C_j) =", LA.norm(C, np.inf))

    k = 0
    x = [np.array([5, 2, 2, 4, 3, 2, 0, 11]).reshape(n, 1)]
    table = []
    table.append([0, x[0], "-", LA.norm((A @ x[0]) - b, np.inf)])
    while k == 0 or (LA.norm(x[k] - x[k - 1], np.inf) >= eps and LA.norm((A @ x[k]) - b, np.inf) >= eps):
        x.append(C @ x[k] + g)
        k += 1
        table.append([k, x[k], LA.norm(x[k] - x[k - 1], np.inf), LA.norm((A @ x[k]) - b, np.inf)])

    print(tabulate.tabulate(table, tablefmt="fancy_grid"))
    return x
```

Invocação:

```python
jacobi(M, d, 1e-2)
```

Output:

```
C_j =
```

0.0000	0.0714	0.0000	0.0000	0.1429	0.1429	0.0000	0.1429
0.1429	0.0000	0.0000	0.0000	0.0714	0.0000	0.0000	0.0000
0.0000	0.3333	0.0000	0.1667	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.1667	0.0000	0.0000	0.0000	0.0000	0.1667
0.0000	0.0000	0.2222	0.2222	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.2500	0.0000	0.0000	0.0000	0.5000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.4000	0.0000	0.2000	0.0000	0.2000	0.0000	0.0000	0.0000

```
Norm-inf (C_j) = 0.8
```

0	[[5] [2] [2] [4] [3] [2] [0] [11]]	-	0.4
1	[[5.57142857] [1.64285714] [2.5] [3.83333333] [2.44444444] [1.5] [0.2] [11.]]	0.5714285714285721	0.123413
2	[[5.39512472] [1.68480726] [2.3531746] [3.91666667] [2.51851852] [1.725] [0.2] [11.21746032]]	0.2250000000000001	0.0537509
3	[[5.47191178] [1.664912] [2.38104686] [3.92843915] [2.50440917] [1.68829365] [0.2] [11.13238851]]	0.0850718065003786	0.0167337
4	[[5.45107819] [1.67487377] [2.37637719] [3.9189059] [2.51321911] [1.69526172] [0.2] [11.16585592]]	0.033467408127272336	0.00542263

g) Definição da função

```python
def gs(A, b, eps, printtable = False):
    n = len(A)

    L = np.tril(A)
    U = np.triu(A, 1)

    C = -LA.inv(L) @ U
    g = LA.inv(L) @ b

    print("C_gs =\n", tabulate.tabulate(C, tablefmt="fancy_grid", floatfmt=".4f"), sep = '')
    print(LA.norm(C, np.inf))
    # print("g-gs\n", g)

    k = 0
    x = [np.array([5, 2, 2, 4, 3, 2, 0, 11]).reshape(n, 1)]
    table = []
    table.append([0, x[0], "-", LA.norm((A @ x[0]) - b, np.inf)])
    while k == 0 or (LA.norm(x[k] - x[k - 1], np.inf) >= eps and LA.norm((A @ x[k]) - b, np.inf) >= eps):
        x.append(C @ x[k] + g)
        k += 1
        table.append([k, x[k], LA.norm(x[k] - x[k - 1], np.inf), LA.norm((A @ x[k]) - b, np.inf)])

    if printtable:
        print(tabulate.tabulate(table, tablefmt="fancy_grid"))
    return x
```

Invocação:

```python
gs(M, d, 1e-2, printtable = True)
```

Output:

```
C_gs =
```

0.0000	0.0714	0.0000	0.0000	0.1429	0.1429	0.0000	0.1429
0.0000	0.0102	0.0000	0.0000	0.0918	0.0204	0.0000	0.0204
0.0000	0.0034	0.0000	0.1667	0.0306	0.0068	0.0000	0.0068
0.0000	0.0006	0.0000	0.0278	0.0051	0.0011	0.0000	0.1678
0.0000	0.0009	0.0000	0.0432	0.0079	0.0018	0.0000	0.0388
0.0000	0.0009	0.0000	0.0417	0.0077	0.0017	0.5000	0.0017
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0294	0.0000	0.0420	0.0649	0.0589	0.0000	0.0663

```
0.5535714285714285
```

0	[[5] [2] [2] [4] [3] [2] [0] [11]]	-	0.4
1	[[5.57142857] [1.7244898] [2.40816327] [3.90136054] [2.51322751] [1.60204082] [0.2] [11.21284958]]	0.5714285714285721	0.0809637
2	[[5.45576612] [1.67319712] [2.3746258] [3.9312459] [2.51241593] [1.69365645] [0.2] [11.15971479]]	0.11566245545837361	0.00531348