COMPUTING ASSIGNMENT 5

NAVJOT KAUR (301404765)

This assignment concerns with cantilever beam of length L=1.5m, which is free to move vertically with constant force F=0.4 N/m distributed over its length L. The goal here is to solve the system of linear equations Az=b that the beam yields, where A is in a banded form(given). To do so, the beam is divided into n parts of length h=L/n and the load vector b is defined by constants($b_i = F * (h^4)$), where i = 1, 2, ……, n.

Editor - C:\Users\user\Documents\MACM316\CA5\bandedMatrices.m

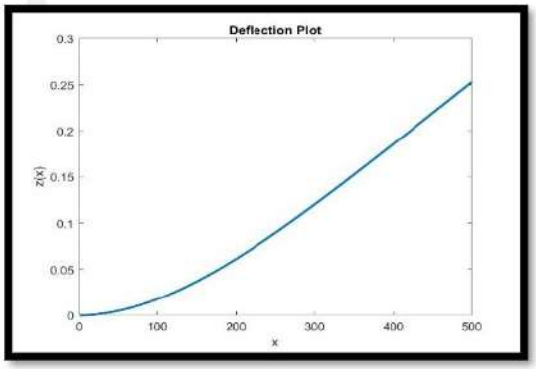
Command Window

```
n = 20
Asparse cost : 2.557000e-04,    Asparse norm : 6.067726e-01
Adense cost : 2.192000e-04,     Adense norm : 6.067726e-01
GS cost : 1.656680e-01,     GS norm : 7.584536e-01
PartB cost : 7.270000e-05,  PartB norm : 6.067726e-01
Condition number of Asparse : 325360
Condition number of U : 800
rho(T) : 9.999701e-01
Bending Stiffness : 1.311360e+00

n = 50
Asparse cost : 2.955000e-04,    Asparse norm : 9.276228e-01
Adense cost : 1.211000e-04,     Adense norm : 9.276228e-01
GS cost : 2.573946e-01,     GS norm : 5.294365e+00
PartB cost : 7.590000e-05,  PartB norm : 9.276228e-01
Condition number of Asparse : 12583400
Condition number of U : 5000
rho(T) : 9.999993e-01
Bending Stiffness : 1.324496e+00

n = 100
Asparse cost : 9.850000e-05,    Asparse norm : 1.297201e+00
Adense cost : 2.010000e-04,     Adense norm : 1.297201e+00
GS cost : 5.439840e-01,     GS norm : 8.642953e+00
PartB cost : 8.060000e-05,  PartB norm : 1.297201e+00
Condition number of Asparse : 200666800
Condition number of U : 20000
rho(T) : 1.000000e+00
Bending Stiffness : 1.328903e+00

n = 500
Asparse cost : 2.494000e-04,    Asparse norm : 2.874659e+00
Adense cost : 3.986500e-03,     Adense norm : 2.874658e+00
GS cost : 1.138841e+01,     GS norm : 2.178610e+01
PartB cost : 9.980000e-05,  PartB norm : 2.874658e+00
Condition number of Asparse : 125083334000
Condition number of U : 500000
rho(T) : 1.000000e+00
Bending Stiffness : 1.332445e+00
```



In part a, Az=b is solved using three methods: 1. GE+PP for sparse matrix using MATLAB's backslash operator. 2. GE+PP for dense version of A using backslash. 3. Gauss-Siedel iterative algorithm using initial guess $z_0 = (1, 1, \dots, 1)^T, tol = 10^{-8}, \ maxiter = \ 10^5$. The solutions are computed for n=20, 50, 100, 500 points as shown on the left. The first two methods are similar in terms of cost and norms, however Gauss-Siedel method is slow as expected because of the number of calculations it must perform to calculate the iterative matrix. The three answers agree with each other as can be seen from the results that the norms of first two methods are similar whereas the GS method have larger norm. The most accurate result is the first method which uses sparse matrix with the use of backslash operator, which have a nature of first identifying the matrix and then performing operations on it, since the matrix is sparse and very few calculations are used, the first method is most accurate among the three.

For Gauss Siedel, as n increases, the amount of time taken to compute iterations also increases because of many calculations performed. As far as norms are concerned, they increase as well because of the iteration algorithms that gs2.m performs for finding T. And these algorithms yield approximate solutions where subtractive cancellation errors are involved.

In part b, the upper triangular matrix and its transpose are calculated which gives A matrix back, i.e, $A = U * U^T$, hence counting towards accuracy. In method 1, $U \ and \ U^T$ are calculated first and then the substitution is done whereas in this part, the $U \ and \ U^T$ are already given so MATLAB just does the forward and backward substitution. Hence it can be concluded that this algorithm is quite fast however the norms remain similar.

In part c, the condition numbers are calculated for A and U, which are calculated in part b. As n increases, the condition numbers become very high and the matrix is very close to singular(ill-conditioned nature of matrix) because the condition numbers can only be estimated for these types of matrices. The spectral radius (rho) is also calculated for the iterative matrix T, used in gs2.m. As n increases, rho is very close to 1 so it takes longer to converge hence slowing the algorithm which in turn means it has to perform more operations.

In part d, the most accurate solution is the solution calculated for part b as it is similar to method 1 in part a, the only difference is that in part b, the decomposed matrices are passed so only forward and backward substitutions are done, which reduces the error and cost while doing the substitutions, hence making it more accurate. The plot is given on the right where the plots for n=20,50,100 and 500 all lie on the same line as shown.

The bending stiffness is also calculated by using the formula $EI = \frac{F*L^4}{(6*zMax)}, where \ zMax = \max(solution \ of \ part \ b)$.

This bending stiffness is a material property and is fairly same(1.33) as n increases because it is independent of n.

MATLAB CODE

```
n_vector = [20, 50, 100, 500];
for n = n_vector
    fprintf("n = %i", n);
    e = ones(n,1);
    Asparse = spdiags([e, -4*e, 6*e, -4*e, e], [-2, -1, 0, 1, 2], n, n);
    Asparse(1, 1) = 9;
    Asparse(n, n) = 1;
    Asparse(n-1, n-1) = 5;
    Asparse(n, n-1) = -2;
    Asparse(n-1, n) = -2;
    F = 0.4; L = 1.5;
    h = L/n;
    Bi = F*(h^4);
    b = Bi * e;
    %part a I.
    tic
    zsparse = Asparse \ b;
    AStime = toc;
    ASnorm = norm(zsparse, 2);
    %part a II.
    Adense = full(Asparse);
    tic
    zdense = Adense \ b;
    ADtime = toc;
    ADnorm = norm(zdense, 2);
    %part a III.
    tic
    [x, niter] = gs2( Asparse, b, e, 1e-8, 1e5);
    AGStime = toc;
    AGSnorm = norm(x, 2);
    fprintf("\nAsparse cost : %d, \tAsparse norm : %d", AStime, ASnorm);
    fprintf("\nAdense cost : %d, \tAdense norm : %d", ADtime, ADnorm);
    fprintf("\nGS cost : %d, \tGS norm : %d", AGStime, AGSnorm);
    %part b
    U = spdiags([e, -2*e, e], [0, 1, 2], n, n);
    U(1, 1) = 2;
    Utranspose = U.';
    A = U * Utranspose;
    tic
    x = U \ b;
    xBack = Utranspose \ x;
    xTime = toc;
    xNorm = norm(xBack, 2);
    fprintf("\nPartB cost : %d, \tPartB norm : %d", xTime, xNorm);
    %part c
    Asparse_U = triu(Asparse, 1);
    DpL= Asparse - Asparse_U;
    T  = -DpL \ Asparse_U;
    condAsparse = condest(Asparse, 2);
    condU = condest(U, 2);
    fprintf("\nCondition number of Asparse : %i", condAsparse);
    fprintf("\nCondition number of U : %i", condU);
    rhoT = max(abs(eig(full(T))));
    fprintf("\nrho(T) : %i", rhoT);
    %part d
    x = 1:n;
    plot(x,xBack, "LineWidth", 2);
    xlabel("x");ylabel("z(x)"); title("Deflection Plot");
    zMax = max(xBack);
    bStiffness = (F * ((L)^4))/(6*zMax);
    fprintf("\nBending Stiffness : %d\n\n",bStiffness);
end
```