CS-MATH 143M PROJECT- I (20 points) NAME: \_\_\_\_Jinhan Han\_\_\_\_\_\_\_\_\_\_\_\_

(Dr. Saleem) FALL 2020 DUE DATE Sunday 10-11-2020

The purpose of this program is to test Gaussian Elimination (without pivoting) on Hilbert's Matrix which is known to be very ill-conditioned. We will also do an operation count and compute the errors in our solution.

DEFINITION : Hilbert Matrix, H, has each of its elements given by: a = 1/(i + j -1) where i,j go from 1 to n.

MATLAB command >> hilb(4) will create a Hilbert Matrix of order 4x4. For example, in FORMAT RAT, if H denotes the 4x4 Hilbert Matrix, then its first row is 1 1/2 1/3 1/4 and second row is 1/2 1/3 1/4 1/5 RAT is for Rational. We will do calculations in “format short”, so our answers will have 4 decimal digits only.

PROBLEM

Consider three systems of equations defined by: H **x** = **b** , n = size of H. We will take n = 11,12 and 13, where b is a vector chosen in such a way that the exact solution of our system is [1 1 1 1 .... 1].

(a) Write a program or use the one from our book’s website(<https://sites.google.com/site/numericalanalysis1burden/home> ), that performs Gaussian Elimination (without pivoting) to compute the solution for each n (3 solution vectors in all). Your program should also keep track of the number of multiplications (divisions). The OUTPUT should consist of the solution vector **x**, and the norm of the error vector, as shown in the example below:

* for n = 5, the exact solution is = Transpose of [1.0 1.0 1.0 ……… …… ]
* computed solution = Transpose of [0.9937 0.999 1.0001 .....]
* error = exact solution minus computed solution = Transpose of [0.0063 0.001 0.0001 ..... ]
* infinity norm of the error vector is = 0.0063
* Euclidean norm of the error vector is = 0.0235
* Number of multiplications in my computer program = yyyy
* Number of multiplications for n=5, using the formula in our book, my answer should have been: \_\_\_\_\_\_\_\_\_\_

As shown above, write the seven bullet items for each case, n=11, case n=12 and case n=13. Put the answers here and proceed to part (b)

Source code in matlab

h\_1 = zeros(11, 11); %hilebert 11 matrix

h\_1 = hilb(11);

disp(h\_1);

b\_1=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1] %actual x soln

disp(b\_1);

disp(h\_1);

result\_1 = h\_1 \* b\_1 % calculated matrix b

disp(result\_1);

h\_2 = zeros(12, 12); %hilebert 12 matrix

h\_2 = hilb(12);

b\_2=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1] %actual x soln

disp(h\_2);

result\_2 = h\_2 \* b\_2 % calculated matrix b

disp(result\_2);

h\_3 = zeros(13, 13); %hilebert 13 matrix

h\_3 = hilb(13);

b\_3=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1] %actual x soln

disp(h\_3);

result\_3 = h\_3 \* b\_3 % calculated matrix b

disp(result\_3);

A=h\_3;

b=result\_3;

count = 0;

[n, n] = size(A); % Find size of matrix A

[n, k] = size(b); % Find size of matrix b

x = zeros(n,k); % Initialize x

for i = 1:n-1

m = -A(i+1:n,i)/A(i,i); % make multipliers

for j = i+1:n

fprintf('Multiply the multiplier into the %i row, subtract %i row from the %i row\n', j, j, j+1);

count=count+(n-i-1);

end

fprintf('Multipliers in order : %6.4f\n',m);

A(i+1:n,:) = A(i+1:n,:) + m\*A(i,:); %multiplication

b(i+1:n,:) = b(i+1:n,:) + m\*b(i,:); %multiplication

disp(A);

disp(b);

end;

% Use back substitution to find unknowns

x(n,:) = b(n,:)/A(n,n); %divide vector b by left element in matrix H

for j = n:-1:i

count=count+i;

end

fprintf('x1....%i is :\n ', n ); % mark

error\_list=zeros(11,1); % preallocate

for i = n:-1:1

x(i,:) = (b(i,:) - A(i,i+1:n)\*x(i+1:n,:))/A(i,i); %calculated equation divide vector b by left element in matrix H

error\_list(i) = x(i,:); %add error element between exact solution and computed solution

disp(x(i,:));

for j = i:n

count = count +1;

end

end

g = b\_3 - error\_list;

disp('error lists between exact solution and computed solution : ')

for i = n:-1:1 %The loop is the printer for error list(exact soln - computed soln)

fprintf(' %6.4f ', g(i));

end

fprintf('\nnumber of multiple count : ')

fprintf([num2str(count)]); % number of multiplication and division

error\_norm=norm(g, 'Inf') %infinity norm of the error vector

Euclid\_error\_norm = norm(g, 2)

hilb(11) ans:

* For n=11 the exact solution is = Transpose of [1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1]
* computed solution = Transpose of [0.9993 1.0037 0.9919 1.0097 0.9929 1.0033 0.9991 1.0002 1.0000 1.0000 1.0000]
* error = exact solution minus computed solution = Transpose of [0.0007 -0.0037 0.0081 -0.0097 0.0071 -0.0033 0.0009 -0.0002 0.0000 -0.0000 0.0000 ]
* infinity norm of the error vector is = 0.0097
* Euclidean norm of the error vector is = 0.0154
* Number of multiplications in my computer program = 0416
* Number of multiplications for n=11, using the formula in our book, my answer should have been: \_0533\_\_\_\_\_\_\_\_\_

hilb(12) ans:

* For n=12 the exact solution is = Transpose of [1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1]
* computed solution = Transpose of [1.0110 0.9385 1.1487 0.7958 1.1754 0.9023 1.0354 0.9919 1.0011 0.9999 1.0000 1.0000]
* error = exact solution minus computed solution = Transpose of [-0.0110 0.0615 -0.1487 0.2042 -0.1754 0.0977 -0.0354 0.0081 -0.0011 0.0001 -0.0000 0.0000]
* infinity norm of the error vector is = 0.2042
* Euclidean norm of the error vector is = 0.3307
* Number of multiplications in my computer program = 0540
* Number of multiplications for n=12, using the formula in our book, my answer should have been: \_\_0688\_\_\_\_\_\_\_\_

hilb(13) ans:

* For n=11 the exact solution is = Transpose of [1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1]
* computed solution = Transpose of [1.2688 -0.6182 5.2709 -5.4935 7.2839 -3.0361 2.7409 0.5010 1.0920 0.9898 1.0006 1.0000 1.0000]
* error = exact solution minus computed solution = Transpose of [0.0007 -0.0037 0.0081 -0.0097 0.0071 -0.0033 0.0009 -0.0002 0.0000 -0.0000 0.0000 ]
* infinity norm of the error vector is = 6.4935
* Euclidean norm of the error vector is = 11.0527
* Number of multiplications in my computer program = 0687
* Number of multiplications for n=13, using the formula in our book, my answer should have been: \_\_\_\_\_0817\_\_\_\_\_

**(b) Comment on the sources of error for parts (a). Type your answer here:**

I calculated error with Matlab. The answer is sort of different from exact soln because of error gap. As the number of size and number of calculation is increasing, the error values was getting bigger.

I commented extra information on the source codes.

Hilb(11)

Graphical user interface, application, Word

Description automatically generated

Hilb(12)

Graphical user interface, application, Word

Description automatically generated

Hilb(13)Graphical user interface, application, Word

Description automatically generated

**(c) Over here, copy the Gaussian Elimination computer program that you used in part (a).**

\*I paste Gaussian part only\*

A=h\_3;

b=result\_3;

count = 0;

[n, n] = size(A); % Find size of matrix A

[n, k] = size(b); % Find size of matrix b

x = zeros(n,k); % Initialize x

for i = 1:n-1

m = -A(i+1:n,i)/A(i,i); % make multipliers

for j = i+1:n

fprintf('Multiply the multiplier into the %i row, subtract %i row from the %i row\n', j, j, j+1);

count=count+(n-i-1);

end

fprintf('Multipliers in order : %6.4f\n',m);

A(i+1:n,:) = A(i+1:n,:) + m\*A(i,:); %multiplication

b(i+1:n,:) = b(i+1:n,:) + m\*b(i,:); %multiplication

disp(A);

disp(b);

end;

% Use back substitution to find unknowns

x(n,:) = b(n,:)/A(n,n); %divide vector b by left element in matrix H

for j = n:-1:i

count=count+i;

end

fprintf('x1....%i is :\n ', n ); % mark

error\_list=zeros(11,1); % preallocate

for i = n:-1:1

x(i,:) = (b(i,:) - A(i,i+1:n)\*x(i+1:n,:))/A(i,i); %calculated equation divide vector b by left element in matrix H

error\_list(i) = x(i,:); %add error element between exact solution and computed solution

disp(x(i,:));

for j = i:n

count = count +1;

end

end

(d) Upload this project (maybe two or three pages) to CANVAS. From the navigation menu, click on **Assignments** and locate the blue **SUBMIT** **ASSIGNMENT** button.