2021

HW3

AERO 5830 - S. HOSDER

MATT PAHAYO

Question 1

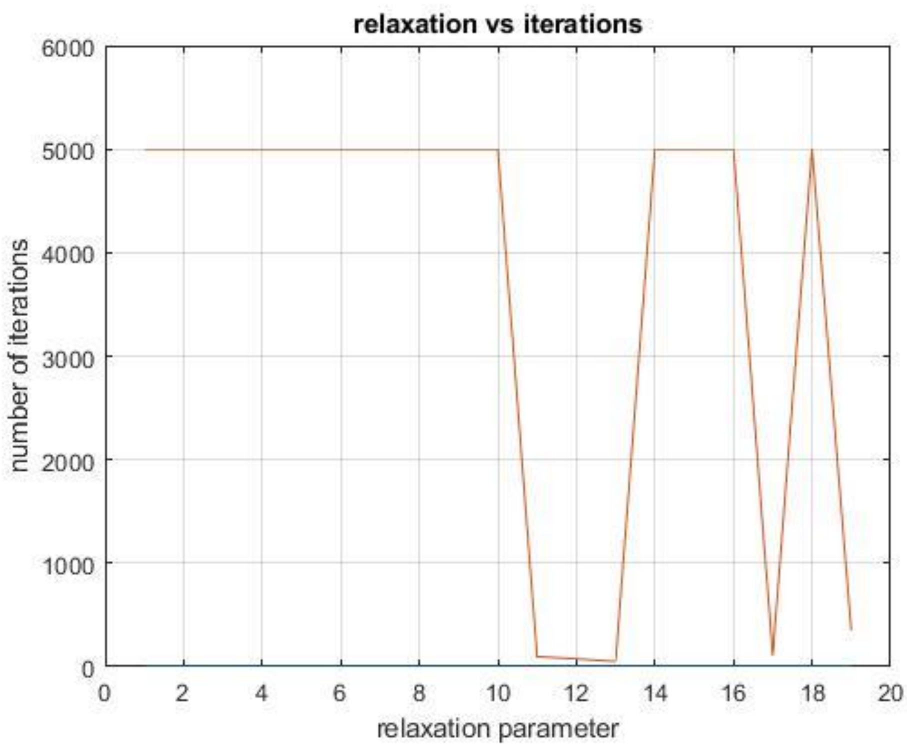
Develop a computer routine to solve a set of linear equations ($Ax = b$) using the Gauss-Seidel scheme with over/under relaxation. Your routine should take A (the coefficient matrix), b (the right hand side vector), ω (the relaxation factor), $NMAX$ (maximum number of iterations allowed), and p (tolerance to stop the iteration process) as inputs. Your output should include the solution vector x. In your program, define the residual vector $r^{(k)} = b - Ax^{(k)}$ where k indicates the iteration number. Stop your iteration process if

$$\frac{||r^k||_2}{||r^0||_2} \leq 10^{-p} \qquad or \qquad k \geq NMAX$$

Use your program to solve $Ax = b$ where A is a $n \times n$ Hilbert matrix and the vector b is given by $b_i = 1.0$ ($i = 1, 2, ..., n$). Obtain the solution vector for $n = 2$ and $n = 3$ with a precision goal of $p = 8$. Set $NMAX$ to 5000 iterations at each case. For $n = 2$, start your iteration with $x^0 = \{1, 1\}^T$ and use $x^0 = \{1, 1, 1\}^T$ for $n = 3$. For each case, make a plot of the number of iterations to achieve the precision goal (or the maximum number of iterations) versus the relaxation factor by changing ω between 0.1 and 1.9 with at least 0.1 increments.

Results

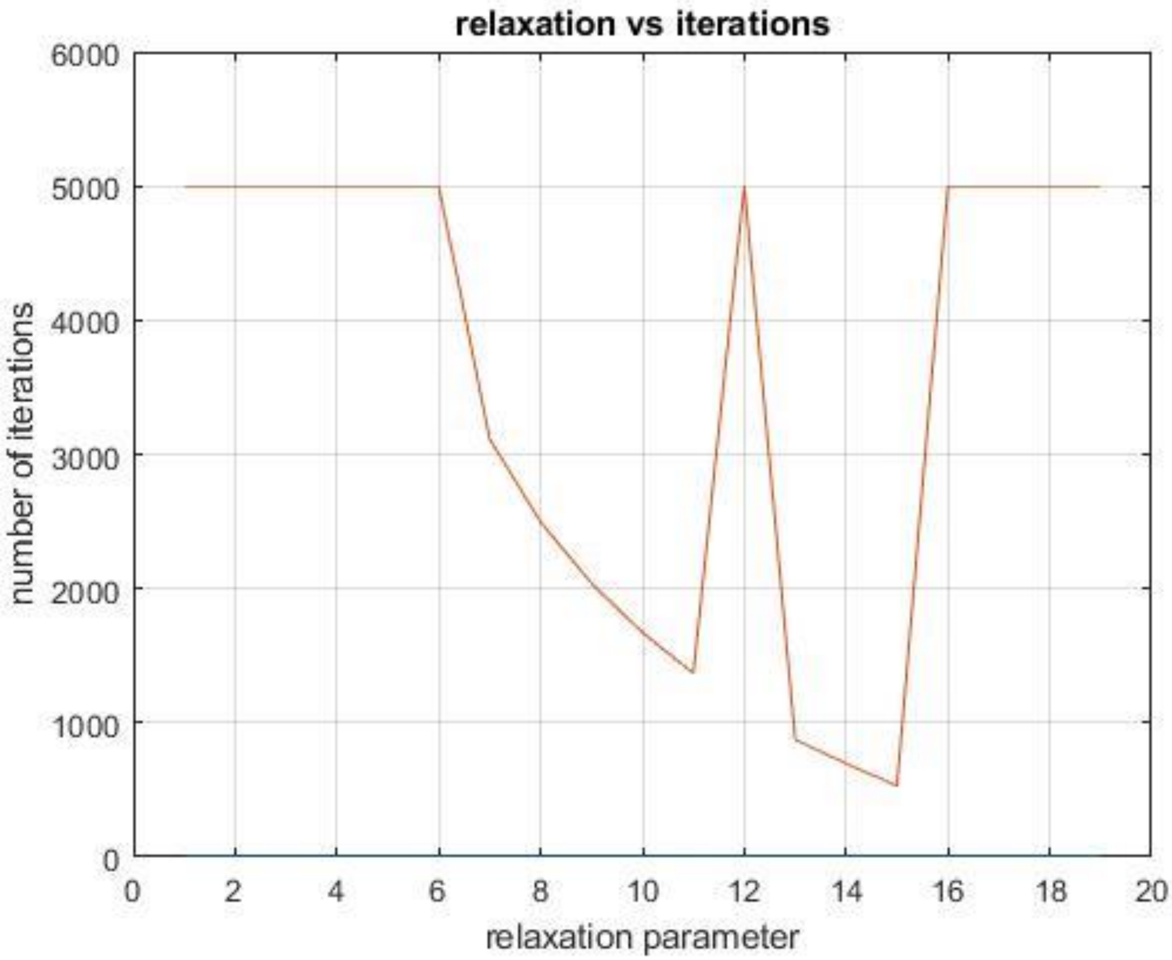
n = 2: $\vec{x} = \begin{bmatrix} -2 \\ 6 \end{bmatrix}$



note the relaxation parameter is multiplied by 10 on both n=2 and n=3

in increments of 0.1

n = 3: $\vec{x} = \begin{bmatrix} 3 \\ -24 \\ 30 \end{bmatrix}$



<u>Method</u>

1. Solve Ax=b with initial guess
2. Using the most updated x vector continue till stopping criteria is met.
3. Use

<u>Where lambda is the relaxation parameter</u>

$$x_i^{new} = \lambda x_i^{new} + (1 - \lambda) x_i^{old}$$

Question 2

Develop a computer program to solve n non-linear equations with n unknowns using the Newton's Method. Use your program to find the solution vector of the following system of equations:

$$\begin{aligned}
4x_1 - x_2 + x_3 &= x_1x_4 \\
-x_1 + 3x_2 - 2x_3 &= x_2x_4 \\
x_1 - 2x_2 + 3x_3 &= x_3x_4 \\
x_1^2 + x_2^2 + x_3^2 &= 1
\end{aligned}$$

Obtain the solution using three different starting vectors:
$x^0 = \{1,1,1,1\}^T$, $x^0 = \{3,3,3,3\}^T$, and $x^0 = \{6,6,6,6\}^T$. For each case, use the following stopping criteria

$$\frac{||f^k||_2}{||f^0||_2} \leq 10^{-16} \qquad or \qquad k \geq NMAX$$

where $NMAX = 100$ and f^k is the function vector evaluated at k^{th} iteration. Note that for this question you can use the methods (direct or indirect) you have developed or build-in functions (or commands) in Matlab for solving the linear set of equations at each iteration.

Results

$x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$: does not converge

$x_0 = \begin{bmatrix} 3 \\ 3 \\ 3 \\ 3 \end{bmatrix}$: converges to $x_8 = \begin{bmatrix} 0.816496580927726 \\ 0.408248290463863 \\ -0.408248290463863 \\ 3.00000000000000 \end{bmatrix}$

$x_0 = \begin{bmatrix} 6 \\ 6 \\ 6 \\ 6 \end{bmatrix}$: does not converge

$$x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}:$$

iteration	$\|f^k\|_2$	$\|f^k\|_2 / \|f^0\|_2$
0	1.96798967126543	1
1	1.96798967126543	1

$$x_0 = \begin{bmatrix} 3 \\ 3 \\ 3 \\ 3 \end{bmatrix}:$$

iteration	$\|f^k\|_2$	$\|f^k\|_2 / \|f^0\|_2$
0	5.27625073457944	1
1	5.27625073457944	1
2	5.62731433871138	1.06653656579115
3	2.77025625671907	0.525042572098288
4	1.30284328905105	0.246925962125432
5	0.516751370470491	0.0979391231511821
6	0.118614966457973	0.0224809192028338
7	0.00698578350756078	0.00132400521866359
8	2.43999882041128e-05	4.62449368529824e-06

$$x_0 = \begin{bmatrix} 6 \\ 6 \\ 6 \\ 6 \end{bmatrix}:$$

iteration	$\|f^k\|_2$	$\|f^k\|_2 / \|f^0\|_2$
0	10.7721300873577	1
1	10.7721300873577	1
2	15.7009819225848	1.45755591468502
3	7.83461666894592	0.727304312648498
4	3.88578334494246	0.360725623755959
5	1.88158355239809	0.174671447256873
6	0.830753224357385	0.0771206082381389

Method

1. Solve $[f'] \overrightarrow{\Delta x} = -\vec{f}$ with gauss elimination.
2. $x^{k+1} = \overrightarrow{\Delta x} + x^k$
3. Repeat step 1 and 2 until stopping criteria is met.

f(x) = 0

```matlab
% matthew Pahayo
% main.m
clc
clear all
close all

format longg
%==========================================================================%
                      % q1
%==========================================================================%
n = 3
A = Hilbert(n);
b = ones(n,1);
x = b;
imax = 5000;
es = 10^-8;
lambda = .1;
i = 1;
while lambda <= 2
    [x,w] = IM.gauSei(A,b,n,b,imax,es,lambda)
    y(i,1) = w(1,1);
    y(i,2) = w(1,2);
    lambda = lambda + .1
    i = i + 1
end

plot(y)
xlabel("relaxation parameter")
ylabel("number of iterations")
title("relaxation vs iterations")
ylim([0 6000])
grid on
%==========================================================================%
                      % q2
%==========================================================================%

syms x1 x2 x3 x4
p = -16;
kmax = 100;
f = symfun([4*x1-x2+x3-x1*x4 -x1+3*x2-2*x3-x2*x4 x1-2*x2+3*x3-x3*x4...
x1^2+x2^2+x3^2-1],[x1,x2,x3,x4]);
q = transpose([6,6,6,6]);
[q,k] = IM.newRap(f,q,p,kmax)
```

```
% Iterative Methods class
%IM.m
classdef IM
    methods (Static)
%=========================================================================%
                      % Gauss-Seidel Method
%=========================================================================%
function [x,w] = gauSei(A,b,n,x,imax,es,lambda)
    for i = 1:n
        dum = A(i,i);
        for j = 1:n
            A(i,j) = A(i,j)/dum;
        end
        b(i) = b(i)/dum;
    end
    for i = 1:n
        sum = b(i);
        for j = 1:n
            if i~= j
                sum = sum - A(i,j)*x(j);
            end
            x(i) = sum;
        end
    end
    iter = 1;
    sen = 0;
    L2norm_0 = norm(b-A*x)^(1/2);
    while sen == 0
        sen = 1;
        for i = 1:n
            old = x(i);
            sum = b(i);
            for j = 1:n
                if i~= j
                    sum = sum - A(i,j)*x(j);
                end
            end
            x(i) = lambda*sum + (1-lambda)*old;
            L2norm = norm(b-A*x)^(1/2);
            if sen == 1 && x(i) ~= 0
                ea = abs(L2norm/L2norm_0);
                if ea > es
                    sen = 0;
                end
            end

        end
        iter = iter + 1;
        if iter >= imax
            break
        end
    end
    w = [lambda iter];
end
%=========================================================================%
                      % Newton-Raphson Method
%=========================================================================%
```

```
function [q,t] = newRap(f,q,p,kmax)
    syms x1 x2 x3 x4
    fp = jacobian(f,[x1 x2 x3 x4]);
    b = transpose(double(f(q(1),q(2),q(3),q(4))));
    b_0 = b ;
    k = 0;
    while (norm(b)/norm(b_0))^(1/2) > 10^p && k<kmax
        L2norm = (norm(b)/norm(b_0))^(1/2);
        l = norm(b)^(1/2);
        A = double(fp(q(1),q(2),q(3),q(4)));
        b = transpose(double(f(q(1),q(2),q(3),q(4))));
        del = gauss(A,-b);
        q = q+del;
        t(k+1,1) = k;
        t(k+1,2) = l;
        t(k+1,3) = L2norm;
        k = k + 1;
    end
end

    end
end
```

```
%gauss.m
function [x] = gauss(a,b)
% gauss elimination

n = length(a);

k = 1 ;
p = k ;
big = abs(a(k,k));

%****************************************************
% pivoting portion
%****************************************************
for ii=k+1:n
    dummy = abs(a(ii,k));
    if dummy > big
       big = dummy;
       p = ii ;
    end
end
if p ~= k
    for jj = k:n
        dummy = a(p,jj);
        a(p,jj) = a(k,jj);
        a(k,jj) = dummy;
    end
    dummy = b(p);
    b(p)=b(k);
    b(k) = dummy;
end

%****************************************************
% elimination step
%****************************************************
for k=1:(n-1)
    for i=k+1:n
        factor = a(i,k)/a(k,k);
       for j=k+1:n
           a(i,j) = a(i,j) - factor*a(k,j);
       end
       b(i) = b(i) - factor*b(k);
    end
end

%****************************************************
% back substitution
%****************************************************
x(n,1) = b(n)/a(n,n);
for i = n-1:-1:1
    sum = b(i);
   for j = i + 1:n
       sum = sum - a(i,j)*x(j,1);
   end
   x(i,1) = sum/a(i,i);
end
end
```

```
 % Hilbert.m
function [A] = Hilbert(n)
%HILBERT Summary of this function goes here
%   Detailed explanation goes here
for i = 1:n
    for j = 1:n
        A(i,j) = 1/(i+j-1);
    end
end
end
```