ME 594 – Numerical Methods – HW04

Viral Panchal | Due Date: 10/19

'I pledge my honor that I have abided by the stevens honor system'

VIRAL PANCHAL
HW 04

Q.1) Gauss - Seidel iterative method.

Given:

$$\begin{bmatrix} 4 & 0 & 1 & 0 & 1 \\ 2 & 5 & -1 & 1 & 0 \\ 1 & 0 & 3 & -1 & 0 \\ 0 & 1 & 0 & 4 & -2 \\ 1 & 0 & -1 & 0 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 32 \\ 19 \\ 14 \\ -2 \\ 41 \end{bmatrix}$$

$4x_1 + x_3 + x_5 = 32$; $x_1 = \dfrac{32 - x_3 - x_5}{4}$

$2x_1 + 5x_2 - x_3 + x_4 = 19$; $x_2 = \dfrac{19 - 2x_1 + x_3 - x_4}{5}$

$x_1 + 3x_3 - x_4 = 14$; $x_3 = \dfrac{14 - x_1 + x_4}{3}$

$x_2 + 4x_4 - 2x_5 = -2$; $x_4 = \dfrac{-2 - x_2 + 2x_5}{4}$

$x_1 - x_3 + 5x_5 = 41$; $x_5 = \dfrac{41 - x_1 + x_3}{5}$

(I) $x_1 = \dfrac{32 - 0 - 0}{4} = 8$; $x_2 = \dfrac{19 - 2(8) + 0 - 0}{5} = 0.6$

$x_3 = \dfrac{14 - 8 + 0}{3} = 2$; $x_4 = \dfrac{-2 - (0.6) + 2(0)}{4} = -0.65$

$x_5 = \dfrac{41 - 8 + 2}{5} = 7$

(ii) $x_1 = \dfrac{32 - 2 - 7}{4} = 5.75$, $x_2 = \dfrac{19 - 2(5.75) + 2 + 065}{5}$

$x_2 = 2.03$

$x_3 = \dfrac{14 - 8 + (-0.65)}{3} = 2.533$

$x_4 = \dfrac{-2 - 2.03 + 14}{4} = 2.4925$

$x_5 = \dfrac{41 - 5.75 + 2.533}{5} = 7.5567$

(iii) $x_1 = \dfrac{32 - 2.533 - 7.5567}{4} = 5.4775$ ✓

$x_2 = \dfrac{19 - 2(5.4775) + 2.5333 - 2.4925}{5} = 1.6172$ ✓

$x_3 = \dfrac{14 - (5.4775) + (2.4425)}{3} = 3.6717$ ✓

$x_4 = \dfrac{-2 - (1.6172) + 2(7.5567)}{4} = 2.8740$ ✓

$x_5 = \dfrac{41 - (5.4775) + 3.6717}{5} = 7.8388$ ✓

$$\therefore X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 5.4775 \\ 1.6172 \\ 3.6717 \\ 2.8740 \\ 7.8388 \end{bmatrix}$$ Ans..

Q 2) Condition number:

Given:

$$A = \begin{bmatrix} 10 & 12 & 0 \\ 0 & 2 & 8 \\ 2 & 4 & 8 \end{bmatrix}$$

$$\text{cond}[A] = \|A\|_1 \, \|A^{-1}\|_1$$

$$\|A\|_1 = \max_{1 \le i \le N} \sum_{i=1}^{N} |a_{ij}|$$

$$= \max \begin{bmatrix} |10| + |0| + |2| \\ |12| + |2| + |4| \\ |0| + |8| + |8| \end{bmatrix}^T = \max \begin{bmatrix} 12 \\ 18 \\ 16 \end{bmatrix}^T$$

$$\therefore \|A\|_1 = 18$$

$$A^{-1} = \begin{bmatrix} -0.5 & -3 & 3 \\ 0.5 & 2.5 & -2.5 \\ -0.125 & -0.5 & 0.625 \end{bmatrix}$$

$$\|A^{-1}\| = \max \begin{bmatrix} |-0.5| + |0.5| + |-0.125| \\ |-3| + |2.5| + |-0.5| \\ |3| + |-2.5| + |0.625| \end{bmatrix}^T$$

$$= \max \begin{bmatrix} 1.125 \\ 6 \\ 6.125 \end{bmatrix}^T = 6.125$$

$$\therefore \text{cond}[A] = 18 \times 6.125 = 110.25 \; // \; \text{Ans}$$

Q.3) Fixed point iteration.

$$\left.\begin{array}{l} f_1 = y - x^3 + 3x^2 - 4x \\ \\ f_2 = y^2 - x - 2 = 0 \end{array}\right\} \text{Given}$$

$f_1 = y - x^3 + 3x^2 - 4x$ $\qquad$ $f_2 = y^2 - x - 2$

$\therefore x = \dfrac{y - x^3 + 3x^2}{4}$ $\qquad$ $\therefore y^2 = x + 2$

$y = \pm\sqrt{x+2}$

,

Q3. Program

- ***Script for fixed point algorithm***

```
% Function for fixed point algorithm
function [P,rel_error,n_iters] = fixed_point(G,P,tol,max_iter)

% rel_error: relative error in the solution
% P: Initial guess during input and Fixed point approximation during
% output.
% G = non linear system saved in it's own function file.

n = length(P);

for i = 1:max_iter
    x = feval(G,P);
    error = norm(x-P);
    rel_error = error/(norm(x) + eps);
    P = x;
    n_iters = i;
    if (rel_error<tol)
        break
    end
end
```

- ***Script for non-linear system G1***

```
function z = G1(x)
z = zeros(1,2);
z(1) = (x(2)-(x(1)^3)+(3*x(1)^2))/4;
z(2) = -sqrt(x(1)+2);
end
```

- ***Script for non-linear system G2***

```
function z = G2(x)
z = zeros(1,2);
z(1) = (x(2)-(x(1)^3)+(3*x(1)^2))/4;
z(2) = sqrt(x(1)+2);
end
```

- ***Driver to run Q3***

```
% Q3 driver

close all
clear all
clc

tol = 10^(-6);
max_iter = 100;
P1 = [-0.3 -1.3];
```

```
[P1,rel_error_1,n_iters_1] = fixed_point('G1',P1,tol,max_iter);
fprintf("First point:\n");
disp(P1)
fprintf("Relative error: \n");
disp(rel_error_1)
fprintf("Number of iterations: \n");
disp(n_iters_1)

P2 = [0.3 1];
[P2,rel_error_2,n_iters_2] = fixed_point('G2',P2,tol,max_iter);
fprintf("Second point:\n");
disp(P2)
fprintf("Relative error: \n");
disp(rel_error_2)
fprintf("Number of iterations: \n");
disp(n_iters_2)
```

- ***MATLAB output:***

```
First point:
   -0.2695   -1.3155

Relative error:
   6.4627e-07

Number of iterations:
    11

Second point:
    0.6699    1.6340

Relative error:
   8.4306e-07

Number of iterations:
    42
```

Published with MATLAB® R2021a

Q4. Newton’s method to solve given equations:

- ***MATLAB Program:***
- ***Script defining given equations:***

```
% Given equations for the system
function z = f(x)

a = x(1);
b = x(2);
c = x(3);
z = zeros(3,1);
z(1) = a^2 - a + b^2 + c^2 -5;
z(2) = a^2 + b^2 - b + c^2 -4;
z(3) = a^2 + b^2 + c^2 + c - 6;
end
```

- ***Script for getting Jacobian:***

```
% Function to jacobian
function w = Jacobian(x)

a = x(1);
b = x(2);
c = x(3);
w = [2*a-1 2*b 2*c; 2*a 2*b-1 2*c; 2*a 2*b 2*c+1];
end
```

- ***Script for Newton’s method:***

```
% Function for newton's method
function [P,iter,rel_error,est_error] = Newton(f,Jacobian,P,eps,max_iter)

% P: initial guess during input and approximation to solution during output
% f: calling given equations
% Jacobian: calling output of jacobian.m

n=length(P);
y=feval(f,P);

for i = 1:max_iter
    jacobian = feval(Jacobian,P);
    aug_mat = [jacobian -y];
    delta_P = zeros(n,1);

    for j = 1:n-1
        [max_element,k] = max(abs(aug_mat(j:n,j)));
        if (k>1)
            temp_row = aug_mat(j,:);
            aug_mat(j,:) = aug_mat(j+k-1,:);
            aug_mat(j+k-1,:) = temp_row;
        end
        for l = j+1:n
            pivot = aug_mat(l,j)/aug_mat(j,j);
```

```
            aug_mat(l,:) = aug_mat(l,:)-pivot.*aug_mat(j,:);
        end
    end

    % Not to do back substitution
    delta_P(n) = aug_mat(n,n+1)/aug_mat(n,n);
    for m = n-1:-1:1
        delta_P(m) = (aug_mat(m,n+1)-
aug_mat(m,m+1:n)*delta_P(m+1:n))/aug_mat(m,m);
    end

    temp_point = P + delta_P;
    y=feval(f,temp_point);

    error = norm(temp_point-P);
    rel_error = error/(norm(temp_point)+eps);
    est_error = norm(y);

    P = temp_point;
    iter = i;
    if(rel_error<eps || (est_error<eps))
        break
    end
end
```

- ***Driver for above function:***

```
%Q4 driver

close all
clear all
clc

eps = 10^-8;
max_iter = 100;
fprintf('Initial estimated point:\n');
P = [0 0 0]';
disp(P)

fprintf('Numerical solution: \n');
[P,iter,rel_error,est_error] = Newton('f','Jacobian', P, eps, max_iter)
```

- ***MATLAB output:***

```
Initial estimated point:
     0
     0
     0
```

```
Numerical solution:

P =

   -0.8471
    0.1529
    1.8471


iter =

     7


rel_error =

   5.1387e-06


est_error =

   1.8994e-10
```

- *Published with MATLAB® R2021a*