ME 594 – Numerical Methods – HW05

Viral Panchal | Due Date: 10/26

“I pledge my honor that I have abided by the stevens honor system”

## Q1) Inverse Power Method

**MATLAB Program:**

* ***Script for Inverse Power Method:***

% Function to determine smallest eigen value and its eigen vector using

% inverse power method.

function [lambda\_n,v\_n,n\_iter] = inverse\_power(A,x0,tol,max\_iter)

[n,n] = size(A);

P = eye(n);

L = zeros(n,n);

L(1,1) = 1;

u = A;

for i = 1:n-1

[max\_element,k] = max(abs(u(i:n,i)));

if (k > 1)

temp\_row = u(i,:);

u(i,:) = u(i+k-1,:);

u(i+k-1,:) = temp\_row;

temp\_row = P(i,:);

P(i,:) = P(i+k-1,:);

P(i+k-1,:) = temp\_row;

end

L(i+1,i+1) = 1;

for j = i+1:n

pivot = u(j,i)/u(i,i);

L(j,i) = pivot;

u(j,i:n) = u(j,i:n) - pivot.\*u(i,i:n);

end

end

x\_k = x0/norm(x0);

y=zeros(n,1);

for k = 1:max\_iter

b = P\*x\_k;

y(1) = b(1)/L(1,1);

for i=2:n

y(i) = (b(i)-L(i,1:i-1)\*y(1:i-1))/L(i,i);

end

x\_k(n) = y(n)/u(n,n);

for i = n-1:-1:1

x\_k(i) = (y(i)-u(i,i+1:n)\*x\_k(i+1:n))/u(i,i);

end

x\_k = x\_k/norm(x\_k);

lambda\_n = x\_k'\*A\*x\_k;

error = norm(A\*x\_k-lambda\_n\*x\_k);

if (error < tol)

v\_n = x\_k;

n\_iter = k;

break

end

v\_n = x\_k;

n\_iter = -1;

end

* ***Script to run the above function:***

% Q1 driver  
close all  
clear all  
clc  
  
n = 25;  
A = zeros(n);  
A(1,1) = -2;  
A(1,2) = 1;  
A(n,n-1) = 1;  
A(n,n) = -2;  
for i=2:n-1  
 A(i,i-1) = 1;  
 A(i,i) = -2;  
 A(i,i+1) = 1;  
end  
  
x0 = ones(n,1);  
tol = 10^(-6);  
max\_iter = 1000;  
  
[lambda\_min,v\_min,n\_iter] = inverse\_power(A,x0,tol,max\_iter);  
lambda\_min  
lamda\_theoretical = -pi^2/(n+1)^2  
v\_min = [0 v\_min' 0]';  
x = linspace(0,1,n+2);  
plot(x,v\_min)  
xlabel("x")  
ylabel("y")  
grid on  
title("Buckling deflection mode")

* ***MATLB Output:***

lambda\_min =  
  
 -0.0146  
  
  
lamda\_theoretical =  
  
 -0.0146

Chart, line chart

Description automatically generated

[*Published with MATLAB® R2021a*](https://www.mathworks.com/products/matlab)

## Q2) Givens Rotation – QR Iteration

**MATLAB Program:**

* ***Script for matrix decomposition using givens method***

% Function to decompose A using Givens method

function [Q,R] = givens\_qr(A)

[n,n] = size(A);

Q = eye(n);

for i = 1:n-1

for j=i+1:n

[c,s] = givens\_params(A(i,i),A(j,i));

A = givens\_mul(A,i,j,c,s);

Q = givens\_mul(Q,i,j,c,s);

end

end

R = A;

Q = Q';

* ***Script to determine givens parameters***

% Functions to determine the givens parameters (c,s)

function [c,s] = givens\_params(x\_i,x\_j)

if (x\_j==0)

c = 1;

s = 0;

elseif (abs(x\_j)>abs(x\_i))

t = x\_i/x\_j;

s = 1/sqrt(1+t^2);

c = s\*t;

else

t = (x\_j)/(x\_i);

c = 1/sqrt(1+t^2);

s = c\*t;

end

* ***Script to generate the givens multiple matrix***

% Function to make a givens multiple matrix

function A = givens\_mul(A,i,j,c,s)

a = A(i,:);

b = A(j,:);

A(i,:) = (c\*a)+(s\*b);

A(j,:) = (-s\*a)+(c\*b);

* ***Script to perform QR iteration***

% Function to perform QR iteration and determine the eigen values

function [eig\_vals] = qr\_iter(A)

for i=1:40

[Q,R] = givens\_qr(A);

A = R \* Q;

end

eig\_vals = diag(A);

* ***Driver script to run Q2***

% Q2 driver  
close all  
clear all  
clc  
  
A = [2.5 -2 2.5 0.5  
 0.5 5 -2.5 -0.5  
 -1.5 1 3.5 -2.5  
 2 3 -5 3]  
  
[eig\_vals] = qr\_iter(A)

***MATLAB Output:***

A =  
  
 2.5000 -2.0000 2.5000 0.5000  
 0.5000 5.0000 -2.5000 -0.5000  
 -1.5000 1.0000 3.5000 -2.5000  
 2.0000 3.0000 -5.0000 3.0000  
  
  
eig\_vals =  
  
 6.0000  
 4.0000  
 3.0000  
 1.0000

[*Published with MATLAB® R2021a*](https://www.mathworks.com/products/matlab)

## Q3) Householder Reflections – QR iteration

***MATLAB Program***

* ***Script to decompose given matrix using HHR method:***

% Function to decompose given matrix using HHR method

function [Q,R] = hhr\_qr(A)

% Input - A matrix (3\*3)

% Output - Q --> dialgonal matrix | R-->Upper triangular matrix

R = A;

[n,n] = size(R);

I = eye(n);

Q=I;

for i = 1:n-1

col=R(:,i);

if(i>1)

col(1:i-1)=0;

end

max\_col = max(abs(col));

col = col/max\_col;

e(1:n,1)=0;

e(i)=1;

mag\_col = sqrt(col'\*col);

if(col(i) >= 0)

u = col + mag\_col\*e;

else

u = col - mag\_col\*e;

end

beta = 2/(u'\*u);

H = I - (beta\*u\*u');

Q = Q\*H;

R = H\*R;

end

* ***Script to perform QR iteration:***

% Function to perform QR iteration

function [eig\_vals] = qr\_iter(A)

for i=1:20

[Q,R] = hhr\_qr(A);

A = R\*Q;

end

eig\_vals = diag(A);

end

* ***Driver to run Q3***

% Q3 driver  
close all  
clear all  
clc  
  
A = [7 6 -3  
 -12 -20 24  
 -6 -12 16]  
  
[eig\_vals] = qr\_iter(A)

***MATLAB Output:***

A =  
  
 7 6 -3  
 -12 -20 24  
 -6 -12 16  
  
  
eig\_vals =  
  
 -2.0000  
 4.0000  
 1.0000

[*Published with MATLAB® R2021a*](https://www.mathworks.com/products/matlab)

## Q4) Jacobi Method to determine eigen pairs:

**MATLAB Program:**

* ***Function for Jacobi method***

% Function to determine eigen pairs using Jacobi method.

function [V,D,num\_sweeps] = jacobi\_cyclic(A,eps,max\_sweeps)

%Input - Matrix A (4\*4)

%Output - V-->Eigen vectors | D-->Diagonal matrix with eigen values

D=A;

[n,n] = size(A);

V = eye(n);

num\_sweeps = 0;

tol\_reached = false;

while((num\_sweeps <= max\_sweeps) && (~tol\_reached))

for i = 1:n-1

for j = i+1:n

tau = (D(j,j) - D(i,i))/2/D(i,j);

if (tau >= 0)

t = 1/(tau + sqrt(tau^2 + 1));

else

t = -1/(-tau + sqrt(tau^2 + 1));

end

c = 1/sqrt(1+t^2);

s = c\*t;

R = [c s; -s c];

D([i j],:) = R'\*D([i j],:);

D(:,[i j]) = D(:,[i j])\*R;

V(:,[i j]) = V(:,[i j])\*R;

end

end

off\_A = 0;

for i = 1:n-1

for j = i+1:n

off\_A = off\_A + D(i,j)^2;

end

end

off\_A = sqrt(2\*off\_A);

if (off\_A < eps)

tol\_reached = true;

end

num\_sweeps = num\_sweeps + 1;

end

D = diag(diag(D));

* ***Driver to run Q4***

% Q4 driver  
  
close all  
clear all  
clc  
  
A = [-8 16 23 -13  
 16 9 2 3  
 23 2 1 -23  
 -13 3 -23 -7]  
  
eps = 10^(-6);  
max\_sweeps = 100;  
  
[V,D,num\_sweeps] = jacobi\_cyclic(A,eps,max\_sweeps)

**MATLAB Output:**  
A =  
  
 -8 16 23 -13  
 16 9 2 3  
 23 2 1 -23  
 -13 3 -23 -7  
  
  
V =  
  
 0.6928 0.5461 -0.1499 0.4464  
 -0.2301 0.2943 -0.8784 -0.2980  
 -0.6473 0.6355 0.2742 0.3193  
 -0.2191 -0.4596 -0.3615 0.7811  
  
  
D =  
  
 -30.6917 0 0 0  
 0 38.3264 0 0  
 0 0 12.3412 0  
 0 0 0 -24.9760  
  
  
num\_sweeps =  
  
 4

[*Published with MATLAB® R2021a*](https://www.mathworks.com/products/matlab)