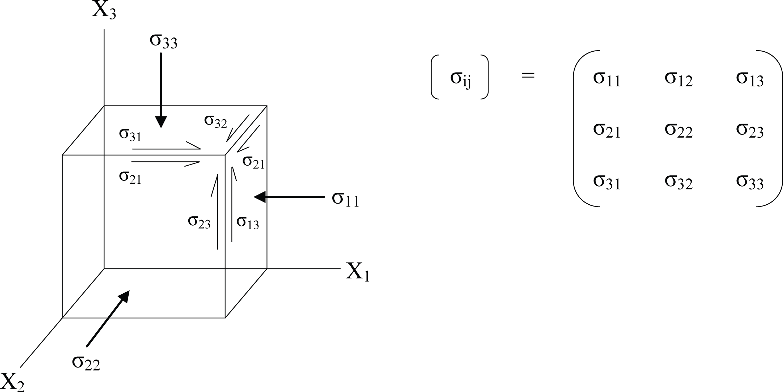
# Assignment: Finding Eigenvalue and Eigenvectors

## What you need to submit

* Submit the report+source files as a zip file: **Assignment\_eigenValue\_Name\_ID.zip**
* **Report:** with pseudocode, output result and source code as instructed
* **Src Code Files:**
  + Assignment\_eigenValue\_ID.cpp
  + myNP.h, myNP.cpp, myMatrix.h, myMatrix.cpp

**Problem:** The 3-D state of stress at a point is given by the stress tensor. Find the principal stresses and the principal directions at the point which are the eigenvalues and the eigenvectors, respectively.



30 15 20

**  *A*  15 22 26 MPa

 

20 26 40

# Procedure

## Find real eigenvalues

* 1. Write a pseudocode for finding eigenvalue **[5pt]**

[lamda]= eig(A) A1=A

For i=1 to N *// or until Ai becomes U*

*// Part1. QRdecomp using household matrix*

[Q R] = QRHousehold (Ai);

*// Part2. Make into similar matrix*

Ai+1 = RQ;

End U= Ai

lamda= diag(U) return lamda

End of function

[Q R] = QRHousehold (Ai)

function [Q R] = QRdecomp(A)

R=A;

n = size(R,1);

I = eye(n);

Q = I;

for j = 1:n-1

for k = 1:n-1

c = R(:,k);

c(1:k-1) = 0;

e(1:n,1)=0;

if (c(k) >= 0)

e(k,1) = 1.

else

e(k,1) =-1;

end

v = c + norm(c,2)\*e;

H=I-2\*(v\*v')/(v'\*v);

R=H\*R;

Q=Q\*H;

end

end

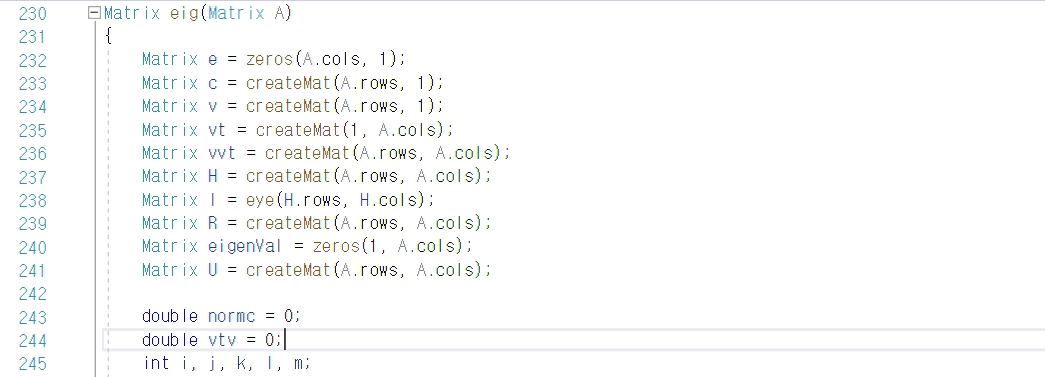
return Q, R End of function

* 1. Create C functions of eig(A) that returns a vector of eigenvalues **[10pt]**

Matrix eig(Matrix A);

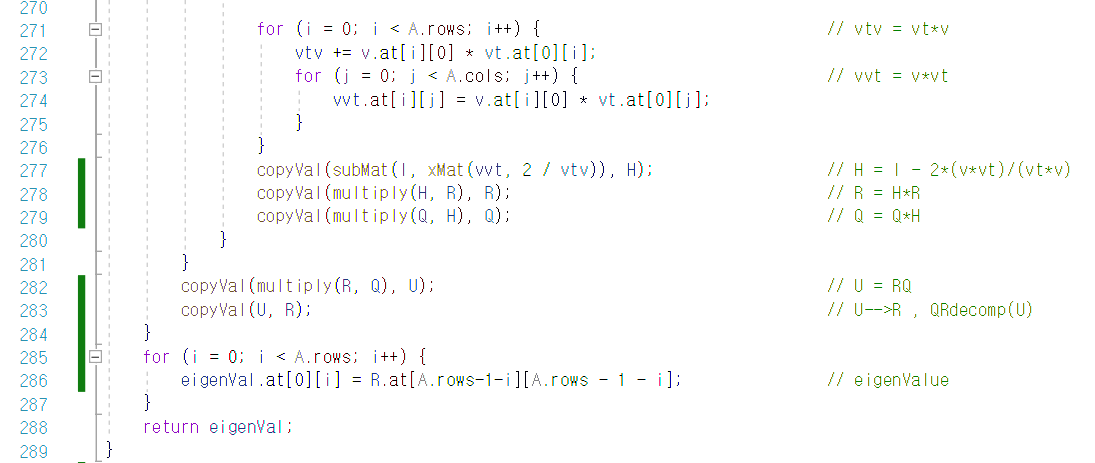
* + - Input: matrix **A**(n x n)
    - Output: vector nx1 Lamda = [lamda1, lamda2,…]T
* Must check if the input matrix is a square matrix.
* Must check if there are any division by zero

// your code



텍스트이(가) 표시된 사진

자동 생성된 설명



## Find Eigenvectors (Simple method, 2x2 or 3x3 matrix)

* Use the simple method learnt in class to find eigenvectors
* Assume, this function is for 2x2 ~ 3x3 matrix only.

1. Write a pseudocode for finding eigenvector **[5pt]**

**For k = 0 to n-1 {**

**B = A-lamda \* I**

**If (k == 0) {**

**For i = 0 to n - 2**

**For j = 0 to n – 2**

**subB(i)(j) = B(i+1)(j+1) // get subB**

**For i = 0 to n – 2**

**vecB(i)(0) = -B(i+1)(k) // get vecB**

**V1 = inv(subB) \* vecB**

**V1 = V1 / |V1| // normalization**

**}**

**Else If (k > 0 && k < n-1) {**

**Repeat for V2**

**V2 = inv(subB) \* vecB**

**V2 = V2 / |V2| // normalization**

**}**

**Else If (k >= n-1) {**

**Repeat for V3**

**V3 = inv(subB) \* vecB**

**V3 = V3 / |V3| // normalization**

**}**

**Return V  
}**

1. Create C function of eigvec(A) **[10pt]**

Matrix eigvec(Matrix A);

* + Input: matrix A(n x n)
  + Output: Matrix **V** (nxn)= [v1; v2; v3]
* Must check if the input matrix is a square matrix, 2x2 or 3x3.
* Must check if there are any division by zero
* Use your own function for matrix inverse

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

Result: The principal directions are

텍스트이(가) 표시된 사진

자동 생성된 설명

**Stresses = 3.461223, 15.507761, 73.031016**

## Show the output results for eigenvalues and vectors [10pt]

Result: The principal stresses and directions are

텍스트이(가) 표시된 사진

자동 생성된 설명

**Stresses = 3.461223, 15.507761, 73.031016**

**Directions =**

**V1 = [0.040641 V2 = [-0.863751 V3 = [0.502278**

**-0.824479 0.254981 0.505193**

**0.564432] 0.434649] 0.701782]**

## Compare your answer with MATLAB [5pt]

Write Matlab code

// your MATLAB code

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

// MATLAB output

텍스트이(가) 표시된 사진

자동 생성된 설명

**Appendix**

**-** A simple method of calculating the eigenvector can be as follows.

After finding the eigenvectors, we are interested in finding the eigenvector v: (*A-λ*I*)v=0*

Let B=(A-λI), and A,B are 3 by 3 matrix Let the eigenvectors are initially set as

1  12  2

*v*13 

For the first eigenvector *V1*

*V*  *v*  , *V*   1  , *V*  *v*

 1  *v*22 

*v*31 

 

*v*23 

3  32 

 1 



*b*11

*b*

 21

*b*31

*b*

*b*12

22

*b*32

*b*  *v*   0 ,

*b*13   1  0

*b*

 *v*  *b* 

23   12 

*b*33  *v*13 

 

0

22



*b*

*b*32

*b*33  *v*13  *b*31 

23 12 21

     

If the inverse matrix exists, then

*v*  *b*

   

12

22

*b*

23

 *b* 

1

21

*v*13  *b*32

*b*33   31 

 *b* 

Normalize the eigenvector *V1*

*V*  *V*1

1

*V*

1

For the other eigenvector of V2,

*b*11

*b*

*b*12

*b*13  *v*22  0

 21

*b*31

*b*

*b*   1   0 ,

*b*

11



*b*

13 12 12

     

 *v*

 *b* 

*b*32 *b*33  *v*23 

22 23  

  

0

*b*31 *b*33  *v*13  *b*32 

Repeat for V3