```
close all
clear
clc

%---test codes---------------
disp("Test codes for question (B): ");
Q=[sqrt(0.2),sqrt(0.3),sqrt(0.4),sqrt(0.1)];
Rotation=Quat_to_Rot(Q);
disp("Rotation Matrix R: ");
disp(Rotation);
Quat=Rot_to_Quat(Rotation);
disp("Input Quaternion for Comparison: ");
disp(Q);
disp("Quaternion: ");
disp(Quat);
%---test ends----------------

%--------question (c)---------
disp("*****************************************************");
disp("Question (C) section: ")
% first of all, generate 5 sets of random rpy angles and convert them into
% SO(3) Rotation Group
numOfRandom = 5;
% declare the equality.
equality=false;
for i=1:numOfRandom
    % generate random set of rpy vector
    rpy=2*pi*rand(1,3);
    % convert it into Rotation Matrix
    R=RPY_to_Rot(rpy);
    % Convert random Rotation Matrix into Quaternions using functions below
    Quaternion=Rot_to_Quat(R);
    % Convert the Quaternions back to Rotation Matrix using functions below
    R_c=Quat_to_Rot(Quaternion);
    % Compute the difference between the input and output Rotation Matrix
    diff=R_c-R;
    % display the results
    disp("-----------------------------------------------");
    disp("The random set " + i + ": ");
    disp("    The input Rotation Matrix: ");
    disp(R);
    disp("    The output Rotation Matrix: ");
    disp(R_c);
    disp("    The difference is: ");
    disp(diff);
    if max(diff)>1E-10
        disp("The codes fail and the random generation of matrix stops!");
        equality=false;
        break;
    else
        disp("Equality for set " + i + " holds!" + newline);
        equality=true;
    end
end

% check the overall equality and make conclusion
disp("-------------------------------------------------");
disp(newline+"Conclusion:");
switch equality
    case false
        disp("    The codes fail!");
    otherwise
        disp("    The codes are verified to be correct!");
```

```
end




%-----question (a)------------
function R = Quat_to_Rot(Q)
% Assign the elements in Quaternions for convenience
q0 = Q(1);
q1 = Q(2);
q2 = Q(3);
q3 = Q(4);
% Compute the Rotation Matrix Elements using Elements of Quaternions above
r11 = q0^2+q1^2-q2^2-q3^2;
r12 = 2*(q1*q2-q0*q3);
r13 = 2*(q1*q3+q0*q2);
r21 = 2*(q1*q2+q0*q3);
r22 = q0^2+q2^2-q1^2-q3^2;
r23 = 2*(q2*q3-q0*q1);
r31 = 2*(q1*q3-q0*q2);
r32 = 2*(q2*q3+q0*q1);
r33 = q0^2+q3^2-q1^2-q2^2;
% Assign the elements into the Rotation Matrix
R = [r11,r12,r13;r21,r22,r23;r31,r32,r33];
end
%-----question (a) ends-------

%-----question (b)------------
function Q = Rot_to_Quat(R)
% Assign the elements in Rotation matrix for convenience
r11 = R(1,1);
r12 = R(1,2);
r13 = R(1,3);
r21 = R(2,1);
r22 = R(2,2);
r23 = R(2,3);
r31 = R(3,1);
r32 = R(3,2);
r33 = R(3,3);
% declare column vector b expressed in Equation (2.23) in lecture notes
b = [r11;r22;r33;1];
% declare the inverse of a matrix that represents the coefficients
% expressed in equation (2.21) in lecture notes. The inverse of that matrix
% is just 0.25 times the transpose of that matrix itself!
invA = 0.25*[1 1 1 1;1 -1 -1 1;-1 1 -1 1;-1 -1 1 1];
% compute the Qsquare term
Qsquare = invA*b;
% find the index of the maximum value in the vector, if there are multiple
% maximum values (less likely), then return the first one.
maxIndex=find(Qsquare==max(Qsquare),1,'first');
qknown=sqrt(max(Qsquare));

% determine which solution will be used.
switch maxIndex
    % solution 1 will be used
    case 1
        q0=qknown;
        q1=(r32-r23)/(4*q0);
        q2=(r13-r31)/(4*q0);
        q3=(r21-r12)/(4*q0);
    % solution 2 will be used
    case 2
        q1=qknown;
        q0=(r32-r23)/(4*q1);
        q2=(r12+r21)/(4*q1);
```

```
        q3=(r13+r31)/(4*q1);
    % solution 3 will be used
    case 3
        q2=qknown;
        q0=(r13-r31)/(4*q2);
        q1=(r12+r21)/(4*q2);
        q3=(r23+r32)/(4*q2);
    otherwise
        q3=qknown;
        q0=(r21-r12)/(4*q3);
        q1=(r13+r31)/(4*q3);
        q2=(r23+r32)/(4*q3);
end

Q=[q0,q1,q2,q3];

end
%-----question (b) ends-------




%--------question (c)---------
function R = RPY_to_Rot(RPY_angles)
% assign the RPY angles for convenience
% Roll angle
gamma = RPY_angles(1);
% Pitch angle
beta = RPY_angles(2);
% Yaw angle
alpha = RPY_angles(3);
% Compute each elementary rotation matrices for RPY.
% Roll Matrix
R1 = [1,0,0;0,cos(gamma),-sin(gamma);0,sin(gamma),cos(gamma)];
% Pitch Matrix
R2 = [cos(beta),0,sin(beta);0,1,0;-sin(beta),0,cos(beta)];
% Yaw Matrix
R3 = [cos(alpha),-sin(alpha),0;sin(alpha),cos(alpha),0;0,0,1];
% Total Rotation Matrix
R = R3*R2*R1;
end
%--------question (c) ends-----
```

```
Test codes for question (B):
Rotation Matrix R:
   -0.0000    0.4100    0.9121
    0.9757    0.2000   -0.0899
   -0.2193    0.8899   -0.4000

Input Quaternion for Comparison:
    0.4472    0.5477    0.6325    0.3162

Quaternion:
    0.4472    0.5477    0.6325    0.3162

***************************************************
Question (C) section:
-----------------------------------------------
The random set 1:
    The input Rotation Matrix:
    0.1146    0.0145    0.9933
    0.2143    0.9760   -0.0389
   -0.9700    0.2173    0.1088
```

```
    The output Rotation Matrix:
    0.1146    0.0145    0.9933
    0.2143    0.9760   -0.0389
   -0.9700    0.2173    0.1088

    The difference is:
   1.0e-15 *

   -0.1388    0.0069    0.1110
         0    0.1110    0.0069
         0         0   -0.2220

Equality for set 1 holds!

----------------------------------------------
The random set 2:
    The input Rotation Matrix:
   -0.0274    0.3495   -0.9365
    0.9097   -0.3796   -0.1683
   -0.4143   -0.8566   -0.3075

    The output Rotation Matrix:
   -0.0274    0.3495   -0.9365
    0.9097   -0.3796   -0.1683
   -0.4143   -0.8566   -0.3075

    The difference is:
   1.0e-15 *

   -0.0902   -0.0555         0
         0   -0.0555   -0.1110
         0   -0.1110    0.1110

Equality for set 2 holds!

----------------------------------------------
The random set 3:
    The input Rotation Matrix:
   -0.2759   -0.3450   -0.8971
    0.4093    0.8023   -0.4344
    0.8697   -0.4871   -0.0802

    The output Rotation Matrix:
   -0.2759   -0.3450   -0.8971
    0.4093    0.8023   -0.4344
    0.8697   -0.4871   -0.0802

    The difference is:
   1.0e-15 *

    0.1665   -0.0555         0
    0.1110         0   -0.0555
         0         0    0.1804

Equality for set 3 holds!

----------------------------------------------
The random set 4:
    The input Rotation Matrix:
   -0.0862   -0.1309   -0.9876
   -0.1700    0.9787   -0.1149
    0.9817    0.1580   -0.1066

    The output Rotation Matrix:
   -0.0862   -0.1309   -0.9876
```

```
    -0.1700    0.9787   -0.1149
     0.9817    0.1580   -0.1066

     The difference is:
    1.0e-15 *

    -0.2082   -0.0555    0.1110
     0.0555   -0.1110    0.0555
          0   -0.0555   -0.1388

Equality for set 4 holds!

-----------------------------------------------
The random set 5:
    The input Rotation Matrix:
     0.6066   -0.6332    0.4807
    -0.5229   -0.7732   -0.3586
     0.5988   -0.0338   -0.8002

     The output Rotation Matrix:
     0.6066   -0.6332    0.4807
    -0.5229   -0.7732   -0.3586
     0.5988   -0.0338   -0.8002

     The difference is:
    1.0e-15 *

    -0.1110         0   -0.0555
          0    0.3331   -0.0555
    -0.1110   -0.0416         0

Equality for set 5 holds!

------------------------------------------------

Conclusion:
    The codes are verified to be correct!
```

Published with MATLAB® R2018b