ELEC-448: Assignment #1

Jonathon Malcolm | 10135148

1.a) Matlab Code

function Rot2ZYZ( RotMatrix )  
%Rot2ZYZ converts a rotation matrix into the ZYZ Euler Angles  
% This function takes in a 3x3 rotation matrix and calculates the ZYZ  
% Euler Angles while also checking for special cases where Phi and Psi  
% can not be found.  
  
if (RotMatrix(1,3) == 0 && RotMatrix(2,3) == 0 && RotMatrix(3,1) == 0 && RotMatrix(3,2) == 0)  
 if (RotMatrix(3,3) == 1)  
 fprintf('Nu = 0 therefore Psi and Phi cannot be found separately.\n');  
 x = (180/pi) \* atan2(RotMatrix(2,1), RotMatrix(2,2));  
 fprintf('Phi + Psi = %0.2f\n\n', x);  
 else  
 fprintf('Nu = 180 therefore Psi and Phi cannot be found separately.\n');  
 x = (180/pi) \* atan2(RotMatrix(2,1), RotMatrix(2,2));  
 fprintf('Phi - Psi = %0.2f\n\n', x);  
 end  
else  
 Phi1 = atan2(RotMatrix(2,3),RotMatrix(1,3));  
 Nu1 = atan2(sqrt(RotMatrix(1,3)^2+RotMatrix(2,3)^2),RotMatrix(3,3));  
 Psi1 = atan2(RotMatrix(3,2),-RotMatrix(3,1));  
 fprintf('Assuming 0 < Nu < 180 \nPhi = %0.2f degrees \nNu = %0.2f degrees \nPsi = %0.2f degrees\n\n',Phi1, Nu1, Psi1);  
  
 Phi2 = atan2(-RotMatrix(2,3),-RotMatrix(1,3));  
 Nu2 = atan2(-sqrt(RotMatrix(1,3)^2+RotMatrix(2,3)^2),RotMatrix(3,3));  
 Psi2 = atan2(-RotMatrix(3,2),RotMatrix(3,1));  
 fprintf('Assuming -180 < Nu < 0 \nPhi = %0.2f degrees \nNu = %0.2f degrees \nPsi = %0.2f degrees\n\n',Phi2, Nu2, Psi2);  
end  
  
end

1.b) Flowchart of Program

1.c) Output with the following inputs:

Rot1:

Assuming 0 < Nu < 180   
Phi = 0.00 degrees   
Nu = 0.52 degrees   
Psi = 0.52 degrees  
  
Assuming -180 < Nu < 0   
Phi = -3.14 degrees   
Nu = -0.52 degrees   
Psi = -2.62 degrees

Rot2:  
Nu = 0 therefore Psi and Phi cannot be found separately.  
Phi + Psi = 60.00

Rot3:  
Nu = 180 therefore Psi and Phi cannot be found separately.  
Phi - Psi = 30.00

2.a) Matlab Code

function Rot2EqAngle( RotMatrix )  
%Rot2EqAngle converts a rotation matrix into the Equivalent Angle and axis  
% This function takes in a 3x3 rotation matrix and calculates the  
% Equivalent Angle (v) and the axis (r) while also checking for special  
% cases where Phi and Psi can not be found.  
  
Nu = acosd((RotMatrix(1,1) + RotMatrix(2,2) + RotMatrix(3,3) - 1)/2);  
if (Nu == 0)  
 fprintf('Nu = 0 is a special case so the unit vector R is arbitrary.\n\n');  
elseif (Nu == 180)  
 fprintf('Nu = 180 is a special case so the unit vector R is arbitrary.\n\n');  
else  
 r = [RotMatrix(3,2)-RotMatrix(2,3);RotMatrix(1,3)-RotMatrix(3,1);RotMatrix(2,1)-RotMatrix(1,2)];  
 R = 1/(2\*sind(Nu)).\*r;  
 fprintf('Nu = %0.2f \nR = (%0.2f, %0.2f, %0.2f)\n\n',Nu,R(1),R(2),R(3));  
end  
  
end

2.b) Flowchart of Program

2.c) Output with the following inputs:

Rot3:  
Nu = 180 is a special case so the unit vector R is arbitrary.

Rot4:  
Nu = 30.00   
R = (0.00, 0.71, 0.71)

Rot5:  
Nu = 0 is a special case so the unit vector R is arbitrary.

3.a) Matlab Code

function Rot2UQuater( RotMatrix )  
%Rot2UQuater converts a rotation matrix into the Unit Quaternions  
% This function takes in a 3x3 rotation matrix and calculates the  
% Euler parameters (Eta and Esp).  
  
Eta = sqrt(RotMatrix(1,1) + RotMatrix(2,2) + RotMatrix(3,3) + 1)/2;  
e = [sign(RotMatrix(3,2) - RotMatrix(2,3))\*sqrt(RotMatrix(1,1) + RotMatrix(2,2) + RotMatrix(3,3) + 1);  
 sign(RotMatrix(1,3) - RotMatrix(3,1))\*sqrt(RotMatrix(2,2) + RotMatrix(3,3) + RotMatrix(1,1) + 1);  
 sign(RotMatrix(3,2) - RotMatrix(2,3))\*sqrt(RotMatrix(3,3) + RotMatrix(1,1) + RotMatrix(2,2) + 1)];  
Esp = e ./ 2;  
  
fprintf('Eta = %0.2f \nEsp = (%0.2f, %0.2f, %0.2f)\n\n', Eta, Esp(1), Esp(2), Esp(3));  
  
end

3.b) Flowchart of Program

3.c) Output with the following inputs:

Rot4:  
Eta = 0.00   
Esp = (0.00, 0.00, 0.00)

Rot5:  
Eta = 0.97   
Esp = (0.00, 0.97, 0.00)

Rot6:  
Eta = 1.00   
Esp = (0.00, 0.00, 0.00)