Queen's University

Electrical and Computer Engineering

MREN 348 Intro to Robotics

Assignment 1

Kay Burnham

Student ID: 20220414

January 30th, 2025

1.

a. Not considering special cases, there are two possible sets of ZYZ Euler angles -- one with angles of rotation of angles (a, b, c) and a second with (a+180, -b, c+180).

b. See below for code producing ZYZ Euler angles:

```
%% QUESTION 1B
function Rot2ZYZ(RotMatrix)
% calculate Euler angles from rotation matrix
phi = atan2(RotMatrix(2,3), RotMatrix(1,3)) * 180/pi;
nu  = atan2(sqrt(RotMatrix(1,3)^2+RotMatrix(2,3)^2), RotMatrix(3,3)) * 180/pi;
psi = atan2(RotMatrix(3,2), -RotMatrix(3,1)) * 180/pi;

%detect special cases
if (nu == 0)
    % display message describing case
    fprintf("No Y' rotation \n");
    %recalculate Z rotation angle
    alpha = atan2(RotMatrix(2,1), RotMatrix(1,1)) * 180/pi;
    % display results in degrees
    fprintf("Total rotation about Z: %0.2f \n", alpha);
elseif (nu == 180)
    % display message describing case
    fprintf("Y' rotation of 180 degrees: Z and Z'' aligned \n");
    %recalculate Z rotation angle
    alpha = atan2(RotMatrix(2,1), RotMatrix(1,1)) * 180/pi;
    % display results in degrees
    fprintf("Total rotation about Z: %0.2f \n", alpha);
else
    % display results in degrees
    fprintf("Z: %0.2f \n Y': %0.2f \n Z'': %0.2f \n", phi, nu, psi);
end
```

c. See below for output results:

```
rotation matrix 1:
Z: -68.99
 Y': 4.00
 Z'': 20.05
rotation matrix 2:
Y' rotation of 180 degrees: Z and Z'' aligned
Total rotation about Z: -169.00
rotation matrix 3:
No Y' rotation
Total rotation about Z: 97.00
```

2.

a. See below for code producing an equivalent angle and axis:

```
%% QUESTION 2A
function Rot2EqAngle(RotMatrix)
nu = acos((RotMatrix(1,1)+RotMatrix(2,2)+RotMatrix(3,3)-1)/2) * 180/pi;
a = 1/(2*sin(nu));
r = [a * (RotMatrix(3,2)-RotMatrix(2,3));
     a * (RotMatrix(1,3)-RotMatrix(3,1));
     a * (RotMatrix(2,1)-RotMatrix(1,2))];

if (nu == 0)
    % display message describing case
    fprintf("No rotation, r is arbitrary \n");
    % recalculate

    % display results [Nu (in degrees) and R (vector)]
    fprintf("Nu: %0.2f \n r: [", nu);
    fprintf(" %0.2g " , r);
    fprintf("] \n ");
elseif (nu == 180)
    % display message describing case
    fprintf("Y' rotation of 180 degrees: r flipped \n");
    %recalculate

    % display results [Nu (in degrees) and R (vector)]
    fprintf("Nu: %0.2f \n r: [", nu);
    fprintf(" %0.2g " , r);
    fprintf("] \n ");
else
    % display results [Nu (in degrees) and R (vector)]
    fprintf("Nu: %0.2f \n r: [", nu);
    fprintf(" %0.2g " , r);
    fprintf("] \n ");
end
```

b. See below for output results:

```
rotation matrix 4:
Nu: 25.00
 r: [ -0  -3.2  -0 ]
 rotation matrix 5:
Y' rotation of 180 degrees: r flipped
Nu: 180.00
 r: [ -0  -0  -0 ]
 rotation matrix 6:
No rotation, r is arbitrary
Nu: 0.00
 r: [ NaN  NaN  NaN ]
```

3.

a. See below for code producing Unit Quaternions:

```
%%QUESTION 3A
function Rot2UQuater(RotMatrix)
eta = 0.5 * sqrt(RotMatrix(1,1)+RotMatrix(2,2)+RotMatrix(3,3)+1);
esp = [0.5 * signum((RotMatrix(3,2)-RotMatrix(2,3))) * sqrt(RotMatrix(1,1)-
RotMatrix(2,2)-RotMatrix(3,3)+1);
       0.5 * signum((RotMatrix(1,3)-RotMatrix(3,1))) * sqrt(RotMatrix(2,2)-
RotMatrix(3,3)-RotMatrix(1,1)+1);
       0.5 * signum((RotMatrix(2,1)-RotMatrix(1,2))) * sqrt(RotMatrix(3,3)-
RotMatrix(1,1)-RotMatrix(2,2)+1)];


    % display results [Eta (scalar) and Esp (vector)]
    fprintf(" Eta: %0.2f \n Esp: [", eta);
    fprintf(" %0.2g " , esp);
    fprintf("] \n");

end

%modified signum function
function result = signum (value)
    result = sign(value);
    if (value == 0)
        result = 1;
    end
end
```

b. See below for output results:

```
rotation matrix 4:
 Eta: 0.98
 Esp: [ 0  0.22  0 ]
rotation matrix 5:
 Eta: 0.00
 Esp: [ 0.33  0.67  0.67 ]
rotation matrix 6:
 Eta: 1.00
 Esp: [ 0  0  0 ]
```