**Assignment #4 Yufeng Sun**

4.1

Estimated Attitude by **TRIAD** method using **v1** and **v2**:

0.4156 -0.8551 0.3100

-0.8339 -0.4943 -0.2455

0.3631 -0.1566 -0.9185

Error Principal Angle: 0.0840 rad

Estimated Attitude by **TRIAD** method using **v1** and **v3**:

0.4167 -0.8563 0.3051

-0.8370 -0.4924 -0.2387

0.3546 -0.1559 -0.9219

Error Principal Angle: 0.0786 rad

Estimated Attitude by **TRIAD** method using **v1** and **v4**:

0.4279 -0.8736 0.2317

-0.8750 -0.4646 -0.1359

0.2263 -0.1446 -0.9633

Error Principal Angle: 0.1148 rad

The least accurate measurement is k=4 as the error principle angle is the largest

4.2

Estimated Attitude by **Davenport’s** **q-method**:

0.4158 -0.8551 0.3096

-0.8360 -0.4934 -0.2403

0.3582 -0.1589 -0.9200

Error Principal Angle: 0.0057 rad

Estimated Attitude by **QUEST method**:

0.4158 -0.8551 0.3096

-0.8360 -0.4934 -0.2403

0.3582 -0.1589 -0.9200

Error Principal Angle: 0.0057 rad

Estimated Attitude by **OLAE method**:

0.4158 -0.8548 0.3105

-0.8352 -0.4941 -0.2416

0.3599 -0.1588 -0.9194

Error Principal Angle: 0.0040 rad

%--------------------------------------------------------------------------

% AEEM6036 Assignment 4

% Description:

% Find the attitude control matrix between body frame and inertial frame

% with multiple sensing vector

% Author: Yufeng Sun

% Date: Oct 29, 2020

%--------------------------------------------------------------------------

**4 unit vectors in body frame and inertial frame**

bv1 = [0.8273 0.5541 -0.0920];

bv2 = [-0.8285 0.5522 -0.0955];

bv3 = [0.2155 0.5522 0.8022];

bv4 = [0.5570 -0.7442 -0.2884];

nv1 = [-0.1517 -0.9669 0.2050];

nv2 = [-0.8393 0.4494 -0.3044];

nv3 = [-0.0886 -0.5856 -0.8000];

nv4 = [0.8814 -0.0303 0.5202];

**4-1 TRIAD Method**

C\_BN12 = TRIAD(bv1,bv2,nv1,nv2);

C\_BN13 = TRIAD(bv1,bv3,nv1,nv3);

C\_BN14 = TRIAD(bv1,bv4,nv1,nv4);

C\_Ref = (C\_BN12+C\_BN13+C\_BN14)/3;

err\_1 = Accuracy(C\_BN12, C\_Ref);

err\_2 = Accuracy(C\_BN13, C\_Ref);

err\_3 = Accuracy(C\_BN14, C\_Ref);

disp('Estimated Attitude by TRIAD method using v1 and v2:');

disp(C\_BN12);

disp('Error Principal Angle:');

disp(err\_1)

disp('Estimated Attitude by TRIAD method using v1 and v3:');

disp(C\_BN13);

disp('Error Principal Angle:');

disp(err\_2)

disp('Estimated Attitude by TRIAD method using v1 and v4:');

disp(C\_BN14);

disp('Error Principal Angle:');

disp(err\_3)

disp('The least accurate measurement is k=4 as the error principle angle is the largest');

**4-2 Davenport's q-method**

B = 2\*bv1'\*nv1+bv2'\*nv2+bv3'\*nv3;

S = B + B';

Z = [B(2,3)-B(3,2) B(3,1)-B(1,3), B(1,2)-B(2,1)]';

sigma = trace(B);

K = [sigma Z';

Z S-sigma\*eye(3)];

[V,D] = eigs(K,1); % the lagest eigen value of K and corresponding vector

C\_BN\_q = EP2DCM(V);

err\_q = Accuracy(C\_BN\_q, C\_BN12);

disp('Estimated Attitude by q-method:');

disp(C\_BN\_q);

disp('Error Principal Angle:');

disp(err\_q)

**4-2 QUEST method**

s0 = 2+1+1;

s = fsolve(@CharacterEqn,s0);

rou\_quest = ((s+sigma)\*eye(3)-S)^-1\*Z;

C\_BN\_quest = RP2DCM(rou\_quest);

err\_quest = Accuracy(C\_BN\_quest, C\_BN12);

disp('Estimated Attitude by QUEST-method:');

disp(C\_BN\_quest);

disp('Error Principal Angle:');

disp(err\_quest)

**4-2 OLAE method**

d1 = bv1-nv1;

d2 = bv2-nv2;

d3 = bv3-nv3;

D = [d1';d2';d3'];

s1 = bv1+nv1;

s2 = bv2+nv2;

s3 = bv3+nv3;

SK = [Skew(s1);Skew(s2);Skew(s3)];

W = eye(9);

for i=1:3

W(i,i)=2;

end

rou\_olae = (SK'\*W\*SK)^-1\*SK'\*W\*D;

C\_BN\_olae = RP2DCM(rou\_olae);

err\_olae = Accuracy(C\_BN\_olae, C\_BN12);

disp('Estimated Attitude by OLAE-method:');

disp(C\_BN\_olae);

disp('Error Principal Angle:');

disp(err\_olae)

**support functions**

% skew matrix of a vector

function mat = Skew(vec)

x = vec(1);

y = vec(2);

z = vec(3);

mat = [0 -z y;

z 0 -x;

-y x 0];

end

%

% TRIAD method for computing the matrix between body frame and inertial

% frame with given 2 pairs of unit vector measured by sensors

%

function C\_BN = TRIAD( bv1, bv2, nv1, nv2)

% intermediate t frame respective to body frame

bt1 = bv1 / norm( bv1 );

bt2 = cross( bv1, bv2 ) / norm( cross( bv1, bv2 ) );

bt3 = cross( bt1, bt2 ) / norm( cross( bt1, bt2 ) );

C\_BT = [ bt1' bt2' bt3' ];

% intermediate t frame respective to inertial frame

nt1 = nv1 / norm( nv1 );

nt2 = cross( nv1, nv2 ) / norm( cross( nv1, nv2 ) );

nt3 = cross( nt1, nt2 ) / norm( cross( nt1, nt2 ) );

C\_NT = [ nt1' nt2' nt3' ];

C\_BN = C\_BT\*C\_NT';

end

% characteristic equation used in QUEST method

function f = CharacterEqn(s)

bv1 = [0.8273 0.5541 -0.0920];

bv2 = [-0.8285 0.5522 -0.0955];

bv3 = [0.2155 0.5522 0.8022];

nv1 = [-0.1517 -0.9669 0.2050];

nv2 = [-0.8393 0.4494 -0.3044];

nv3 = [-0.0886 -0.5856 -0.8000];

B = 2\*bv1'\*nv1+bv2'\*nv2+bv3'\*nv3;

S = B + B';

Z = [B(2,3)-B(3,2) B(3,1)-B(1,3), B(1,2)-B(2,1)]';

sigma = trace(B);

K = [sigma Z';

Z S-sigma\*eye(3)];

f = det(K-s\*eye(4));

end

% Measure the attitude matrix accuracy

function err = Accuracy(C\_BN, C\_BN\_Ref)

C\_BB = C\_BN\*C\_BN\_Ref';

[phi,axis] = DCM2PRE(C\_BB);

err = phi;

end

% priciple rotation elements from direct cosine angles

function [phi,axis] = DCM2PRE(dcm)

phi = acos(0.5\*(trace(dcm)-1));

e1 = dcm(2,3)-dcm(3,2);

e2 = dcm(3,1)-dcm(1,3);

e3 = dcm(1,2)-dcm(2,1);

axis = (1/(2\*sin(phi)))\*[e1 e2 e3]';

end

function mat = EP2DCM(epv)

b0 = epv(1);

b1 = epv(2);

b2 = epv(3);

b3 = epv(4);

mat = [b0^2+b1^2-b2^2-b3^2 2\*(b1\*b2+b0\*b3) 2\*(b1\*b3-b0\*b2);

2\*(b1\*b2-b0\*b3) b0^2-b1^2+b2^2-b3^2 2\*(b2\*b3+b0\*b1);

2\*(b1\*b3+b0\*b2) 2\*(b2\*b3-b0\*b1) b0^2-b1^2-b2^2+b3^2];

end

function mat = RP2DCM(rpv)

r1 = rpv(1);

r2 = rpv(2);

r3 = rpv(3);

mat = (1/(1+rpv'\*rpv))\*[1+r1^2-r2^2-r3^2 2\*(r1\*r2+r3) 2\*(r1\*r3-r2);

2\*(r1\*r2-r3) 1-r1^2+r2^2-r3^2 2\*(r2\*r3+r1);

2\*(r1\*r3+r2) 2\*(r2\*r3-r1) 1-r1^2-r2^2+r3^2];

end

[*Published with MATLAB® R2018b*](https://www.mathworks.com/products/matlab/)