# Table of Contents



```
function HW5_LiamHood()

clear ; close all ; clc ;
neg = [ -1 0 0 ]' ;
nsg = [ 0 1 0 ]' ;
```

# a

```
Cbg = C_321Euler( 0 , 0 , pi/4 ) ;
disp( 'a' )
disp( 'Cbg' )
disp( Cbg )
disp( ' ' )
```

```
a
Cbg
    0.7071    0.7071         0
   -0.7071    0.7071         0
         0         0    1.0000
```

# b

```
neb = Cbg*neg ;
nsb = Cbg*nsg ;
disp( 'b' )
disp( 'neb' )
disp( neb )
disp( 'nsb' )
disp( nsb )
disp( ' ' )
```

```
b
neb
   -0.7071
    0.7071
         0
```

```
nsb
    0.7071
    0.7071
         0
```

# c

```
s1a = nsg ;
s1b = nsb ;
s2a = neg ;
s2b = neb ;
    xta = s1a ;
    yta = cross( s1a , s2a )/( norm( cross( s1a,s2a ))) ;
    zta = cross( xta , yta ) ;
    xtb = s1b ;
    ytb = cross( s1b , s2b )/( norm( cross( s1b,s2b ))) ;
    ztb = cross( xtb , ytb ) ;
```

# d

```
    Cat = [ xta , yta , zta ] ;
    Cbt = [ xtb , ytb , ztb ] ;
```

# e

```
Ctriad = Triad( nsg , nsb , neg , neb ) ;
disp( 'C from Triad' )
disp( Ctriad )
disp( ' ' )
```

```
C from Triad
    0.7071    0.7071         0
   -0.7071    0.7071         0
         0         0    1.0000
```

# f

```
w = [ 1 1 ] ;
sb = [ nsb , neb ] ;
sg = [ nsg , neg ] ;

[ Cd ] = DavenportQ( w , sb , sg ) ;
disp( 'Cbg from Davenport-q' )
disp( Cd )
[ Cq ] = Quest( w , sb , sg ) ;
disp( 'Cbg from Quest' )
disp( Cq )
```

```
Why is DavenQ transposed?
Cbg from Davenport-q
    0.7071    0.7071         0
   -0.7071    0.7071         0
         0         0    1.0000

Cbg from Quest
    0.7071    0.7071         0
   -0.7071    0.7071         0
         0         0    1.0000
```

# Functions

```
function C = Triad( s1a , s1b , s2a , s2b )
% s1b , s2b are vectors in body frame; s1a and s2a are vectors in
 inertial
% space
    xta = s1a ;
    yta = cross( s1a , s2a )/( norm( cross( s1a,s2a ))) ;
    zta = cross( xta , yta ) ;
    xtb = s1b ;
    ytb = cross( s1b , s2b )/( norm( cross( s1b,s2b ))) ;
    ztb = cross( xtb , ytb ) ;
    Cat = [ xta , yta , zta ] ;
    Cbt = [ xtb , ytb , ztb ] ;
    C = Cbt*Cat' ;
end

function [ C ] = DavenportQ( w , sb , sg )

    Bt = zeros( 3,3 ) ;
    for ii = 1:length(w)
        Bt = Bt + w(ii)*sb(:,ii)*sg(:,ii)' ;
    end
    B = Bt' ;
    k22 = trace( B ) ;
    k12 = [ B(2,3) - B(3,2) ; B(3,1) - B(1,3) ; B(1,2) - B(2,1) ] ;
    k11 = B + B' - k22*eye(3) ;
    K = [ k11 , k12 ; k12' , k22 ] ;
    [ evec , lam ] = eig( K ) ;
    [ ~ , column ] = max( max( lam ) ) ;
    qbook = evec( : , column )' ;
    q = [ qbook(4) , qbook(1:3) ] ;
    C = quat2rotm( q ) ;
    disp( 'Why is DavenQ transposed? ' )

end

function [ C ] = Quest( w , sb , sg )
    Bt = zeros( 3,3 ) ;
    for ii = 1:length(w)
        Bt = Bt + w(ii)*sb(:,ii)*sg(:,ii)' ;
```

```
    end
    B = Bt' ;
    s = B + B' ;
    k22 = trace( B ) ;
    k12 = [ B(2,3) - B(3,2) ; B(3,1) - B(1,3) ; B(1,2) - B(2,1) ] ;
    k11 = s - k22*eye(3) ;
    K = [ k11 , k12 ; k12' , k22 ] ;

        lambda = sum( w ) ;
        ii = 1 ;
        ratio = 1 ;
        lim = 1e4 ;
        tol = 1e-8 ;
        a = k22^2 - trace( adjoint(s) ) ;
        b = k22^2 + k12'*k12 ;
        c = det(s) + k12'*s*k12 ;
        d = k12'*s^2*k12 ;
        f = @(lambda) lambda^4 - (a+b)*lambda^2 - c*lambda + ( a*b +
 c*k22 - d ) ;
        fprime = @(lambda) 4*lambda^3 - (a+b)*lambda*2 - c ;
            while abs(ratio(ii)) >= tol
                ratio(ii+1) = f(lambda(ii))/fprime(lambda(ii)) ;
                lambda(ii+1) = lambda(ii) - ratio(ii+1) ;
                ii = ii + 1 ;
                    if ii > lim
                        error([ 'Ran ' , num2str( lim ) , ' times
 without a solution' ])
                    end
            end
    lambda = lambda( ii ) ;

    alpha = lambda^2 - k22^2 + trace( adjoint( s ) ) ;
    beta = lambda - k22 ;
    gamma = ( lambda + k22 )*alpha + det(s) ;
    xbar = ( alpha*eye(3) + beta*s + s^2 )*k12 ;

    q = [ gamma , xbar' ] ;
    C = quat2rotm( q ) ;
end

end
```

*Published with MATLAB® R2019a*