PROSJEKTOPPGAVE

I

TEK4040:

MATEMATISK MODELLERING AV DYNAMISKE SYSTEMER

AV

FURKAN KAYA

1. *Finn treghetsmatrisen for en murstein med sidekanter 5,10 og 20 cm (langs hhv x, y og z-aksene (når tettheten er 2 kg/dm^3. Plasser b-systemet i massesenteret med akser parallelle med sidekantene.*

Vi baserer vårt svar på denne oppgaven på definisjonen av treghetsmatrisen fra side 168 i Craigs bok fra pensum. Treghetsmatrisen relativ til ramme {A} er uttrykt i matriseform som:

$$J^A = \begin{pmatrix} J_{xx} & -J_{xy} & -J_{xz} \\ -J_{xy} & J_{yy} & -J_{yz} \\ -J_{xz} & -J_{yz} & J_{zz} \end{pmatrix}$$

Så bruker vi 6.17 til å finne de forskjellige komponentene med oppgitte parametere fra oppgaveteksten. Gjør det også i den rekkefølgen oppgitt i 6.17.

$$J_{xx} = \iiint_V (y^2 + z^2) * 2 \, dV$$

Ved å bruke Wolfram Alpha får vi svaret: 3.33e-5. Denne prosedyren gjør vi for alle de andre komponenter i matrisen. Det gir oss da følgende resultater i tabellform nedenfor:

J_yy	2.83e-5
J_zz	8.33e-6
J_xy	2.5e-6
J_xz	5e-6
J_yz	1e-5

Dette setter vi da inn i treghetsmatrisen ovenfor:

$$J^A = \begin{pmatrix} 3.33e-5 & -2.5e-6 & -5e-6 \\ -2.5e-6 & 2.83e-5 & -1e-5 \\ -5e-6 & -1e-5 & 8.33e-6 \end{pmatrix}$$

Så skal vi plassere b-systemet i massesenteret med akser parallelle med sidekantene. Vi har origo i A og kan derfor skrive spinnsatsen på følgende måte. Det bør sies at spinnsatsen for stive legemer skal gi sammenhengen mellom ytre moment, treghetsmatrise og vinkelakselerasjon. Den representeres av n. Det gir da:

$$n_x = J_{xx}^b w_x' + w_y w_z (J_{zz}^b - J_{yy}^b)$$

$$n_y = J_{yy}^b w_y' + w_z w_x (J_{xx}^b - J_{zz}^b)$$

$$n_z = J_{zz}^b w_z' + w_x w_y (J_{yy}^b - J_{xx}^b)$$

2. *Definer den kinetiske energiellipsoida og spinellipsoida. Tegn opp disse to ellipsoidene med felles sentrum (3D, opake og med to forskjellige farger) for tre tilfeller som gir skjæringer nær de tre hovedaksene (den kinetiske enrgiellipsoida holdes konstant). Se vedlegg for valg av de tre tilfellene.*

Vi definerer først den kinetiske energiellipsen. Følger da forelesning 13 i dette.

Multipliserer Euler-ligningene fra forrige side med hhv. w_x, w_y, w_z for de tre ligningene og summerer. Sette da:

$$0 = \sum_{i_1}^{3} J_{ii} w_i' \, w_i, \;\; med \; x = 1, y = 2 \; og \; z = 3$$

Etter en integrasjon med tiden som variabel får vi:

$$\frac{w_1^2(t)}{\frac{2K_0}{J_{11}}} + \frac{w_2^2(t)}{\frac{2K_0}{J_{22}}} + \frac{w_3^2(t)}{\frac{2K_0}{J_{33}}} = 1$$

Dette er da definisjonen av kinetisk energiellipsoide.

Så ser vi på spinnellipsoiden. Spinnellipsoiden blir på samme måte utledet til:

$$\frac{w_1^2}{\frac{h_0^2}{{J_{11}^b}^2}} + \frac{w_2^2}{\frac{h_0^2}{{J_{22}^b}^2}} + \frac{w_3^2}{\frac{h_0^2}{{J_{33}^b}^2}} = 1$$

Jeg legger herved de to ellipsoidene og forklarer hva som er hva i figurteksten. Vi begynner med den kinetiske energiellipsoiden.

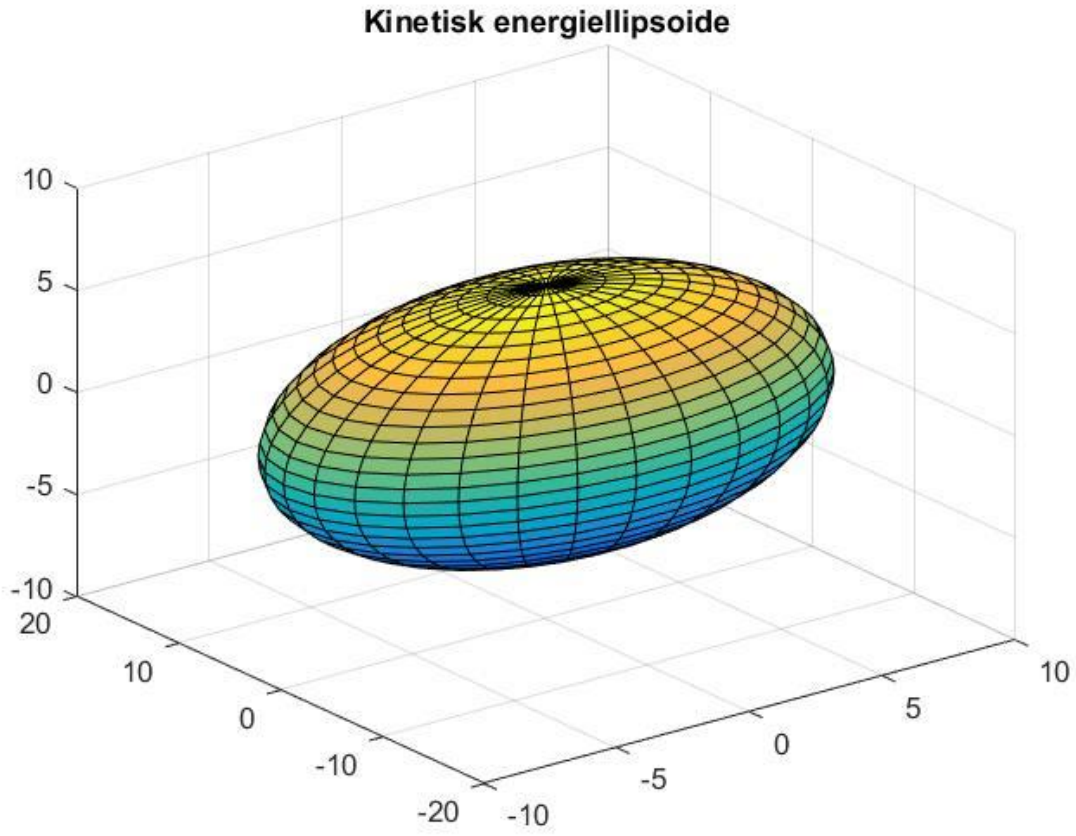


Figure 1: viser den kinetiske energiellipsoiden

Så følger vi med spinnellipsoiden.

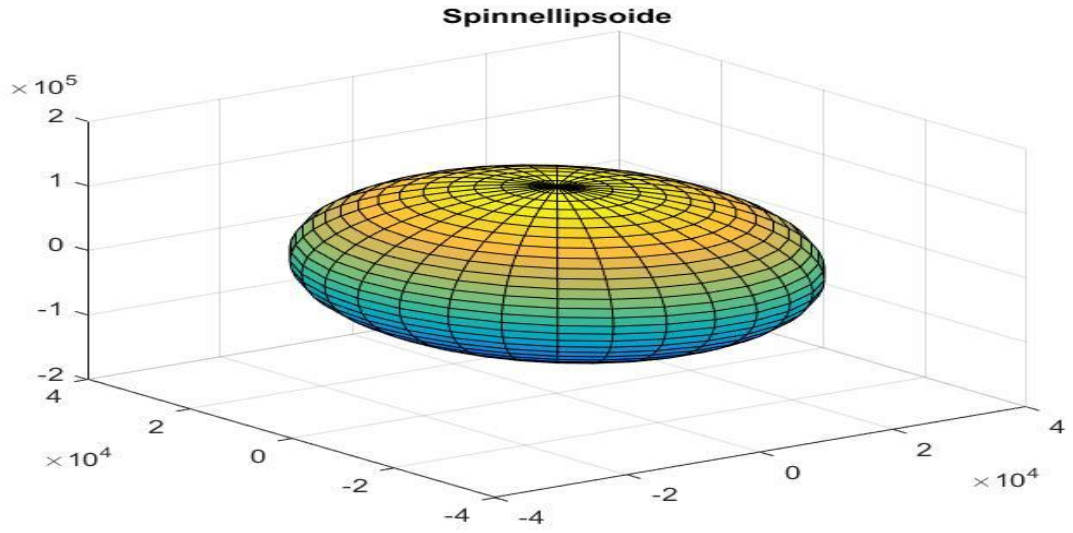


Figure 2: viser Spinnellipsoiden

Scriptene brukt til å få disse to figurer følger nedenfor. I samme rekkefølge som figurene.

For kinetisk energiellipsoide:

```
J33 = 8.33e-6;
J22 = 2.83e-5;
J11 = 3.33e-5;

K0 = (2*pi.^2)*I33;

barw1 = (2*K0./J11);
barw2 = (2*K0./J22);
barw3 = 2*pi;

w3 = barw3./10;

w1 = barw1*(sqrt((1-((w3.^2)./(barw3.^2)))));
w2 = barw2*(sqrt((1-((w3.^2)./(barw3.^2)))));

[w1, w2, w3] = ellipsoid(0,0,0,barw1,barw2,barw3,30);
figure
surf(w1,w2,w3)
%axis equal

title('Kinetisk energiellipsoide');
```

For spinnellipsoide:

```
J33 = 8.33e-6;
J22 = 2.83e-5;
J11 = 3.33e-5;
h0 = 1;

K0 = (2*pi.^2)*I33;

barw1 = (2*K0./J11);
barw2 = (2*K0./J22);
barw3 = 2*pi;

w3 = barw3./10;

w1 = barw1*(sqrt((1-((w3.^2)./(barw3.^2)))));
w2 = barw2*(sqrt((1-((w3.^2)./(barw3.^2)))));

[w1, w2, w3] =
ellipsoid(0,0,0,(h0.^2)./(J11),(h0.^2)./(J22),(h0.^3)./(J3
3),30);
figure
```

```
surf(w1,w2,w3)
%axis equal
title('Spinnellipsoide');
```

3. *Sett opp Euler-likningene og integrer disse i MATLAB for tre tilfeller med initialbetingelsene som gir samme løsning som de skjæringene vi fikk i 2. Tegn opp trajektoriene i en 3D figur.*

Etter litt omdanning får vi at Euler-likningene blir:

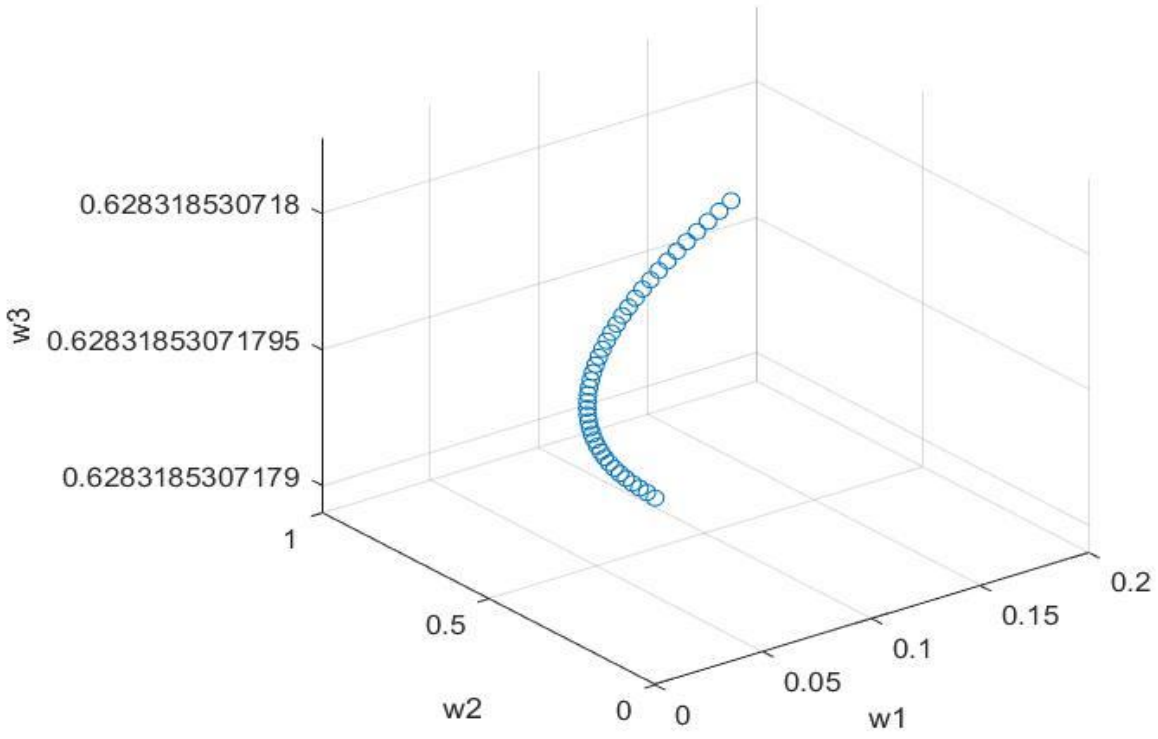
$$w_1' = \frac{J_{22} - J_{33}}{J_{11}} w_2 w_3$$

$$w_2' = \frac{J_{33} - J_{11}}{J_{22}} w_3 w_1$$

$$w_3' = \frac{J_{11} - J_{22}}{J_{33}} w_1 w_2$$

Torque er utelatt fra likningene.

Jeg legger ved figurene som viser trajeksjonen i 3D før jeg legger ved koden. Satte w3 som i vedlegget.

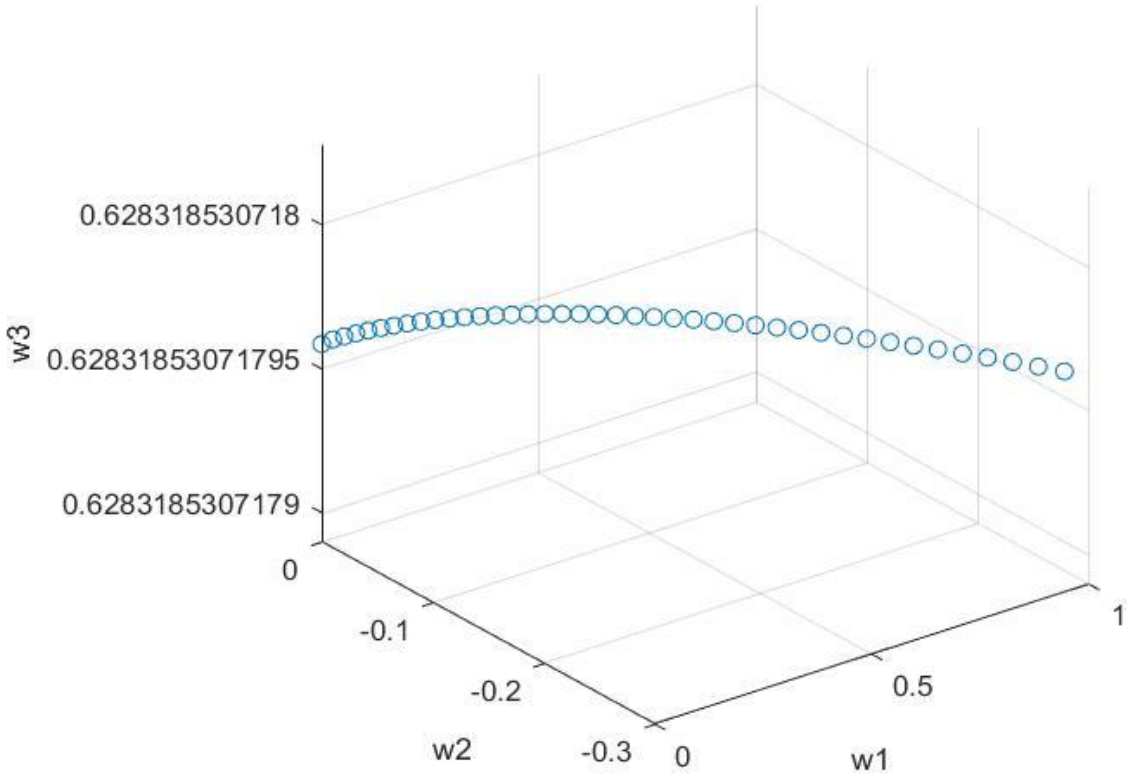


Figur 1: viser trajeksjonen i 3D for w1

```
function [wprikk] = deriv(w2,w1)

w3 = 2*pi./10;

wprikk = ((2.83e-5) - 8.33e-6)./(3.33e-5)*w2*w3;

%wprikk2 = ((8.33e-6) - 3.33e-5)./(2.83e-5)*w3*w1;

%wprikk3 = ((3.33e-5) - (2.83e-5)./(8.33e-6))*w1*w2;

end
```

```
[w2,w1] = ode45('TEK4040_c_2',[0,1],0)

scatter3(w1,w2,w3);

xlabel('w1');
ylabel('w2');
zlabel('w3');
```

Så ser vi på de to andre trajeksjonene. Legger ikke ved koden her ettersom det bare er marginale forskjeller fra første kode i denne oppgaven.



Figur 2: viser trajeksjonen til den andre Euler-likningen

4. *Sett opp de kinematiske ligningene for 3-2-1 Eulervinkler og løs Euler- og spinn-likningene for de samme tre tilfellene som ovenfor. Lag tre animasjonsfilmer som viser hvordan mursteinen roterer sett fra treghetsrommet i de tre tilfellene.*

De kinematiske ligningene for 3-2-1 Eulervinkler er som følgende:

$$\begin{pmatrix} c_1 & s_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} c_2 & 0 & -s_2 \\ 0 & 1 & 0 \\ s_2 & 0 & c_2 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_3 & s_3 \\ 0 & -s_3 & c_3 \end{pmatrix}$$

Legger ved stlread-filen som ble lastet ned fra Mathworks:

```
function varargout = stlread(file)
% STLREAD imports geometry from an STL file into MATLAB.
%    FV = STLREAD(FILENAME) imports triangular faces from
the ASCII or binary
%    STL file idicated by FILENAME, and returns the patch
struct FV, with fields
%    'faces' and 'vertices'.
%
%    [F,V] = STLREAD(FILENAME) returns the faces F and
vertices V separately.
%
%    [F,V,N] = STLREAD(FILENAME) also returns the face
normal vectors.
%
%    The faces and vertices are arranged in the format
used by the PATCH plot
%    object.
% Copyright 2011 The MathWorks, Inc.
    if ~exist(file,'file')
        error(['File ''%s'' not found. If the file is not
on MATLAB''s path' ...
               ', be sure to specify the full path to the
file.'], file);
    end

    fid = fopen(file,'r');
    if ~isempty(ferror(fid))
        error(lasterror); %#ok
    end

    M = fread(fid,inf,'uint8=>uint8');
    fclose(fid);

    [f,v,n] = stlbinary(M);
```

```
    %if( isbinary(M) ) % This may not be a reliable test
     %   [f,v,n] = stlbinary(M);
    %else
     %  [f,v,n] = stlascii(M);
    %end

    varargout = cell(1,nargout);
    switch nargout
        case 2
            varargout{1} = f;
            varargout{2} = v;
        case 3
            varargout{1} = f;
            varargout{2} = v;
            varargout{3} = n;
        otherwise
            varargout{1} = struct('faces',f,'vertices',v);
    end
end
function [F,V,N] = stlbinary(M)
    F = [];
    V = [];
    N = [];

    if length(M) < 84
        error('MATLAB:stlread:incorrectFormat', ...
              'Incomplete header information in binary STL
file.');
    end

    % Bytes 81-84 are an unsigned 32-bit integer
specifying the number of faces
    % that follow.
    numFaces = typecast(M(81:84),'uint32');
    %numFaces = double(numFaces);
    if numFaces == 0
        warning('MATLAB:stlread:nodata','No data in STL
file.');
        return
    end

    T = M(85:end);
    F = NaN(numFaces,3);
    V = NaN(3*numFaces,3);
    N = NaN(numFaces,3);

    numRead = 0;
```

```
    while numRead < numFaces
        % Each facet is 50 bytes
        %  - Three single precision values specifying the
face normal vector
        %  - Three single precision values specifying the
first vertex (XYZ)
        %  - Three single precision values specifying the
second vertex (XYZ)
        %  - Three single precision values specifying the
third vertex (XYZ)
        %  - Two unused bytes
        i1    = 50 * numRead + 1;
        i2    = i1 + 50 - 1;
        facet = T(i1:i2)';

        n  = typecast(facet(1:12),'single');
        v1 = typecast(facet(13:24),'single');
        v2 = typecast(facet(25:36),'single');
        v3 = typecast(facet(37:48),'single');

        n = double(n);
        v = double([v1; v2; v3]);

        % Figure out where to fit these new vertices, and
the face, in the
        % larger F and V collections.
        fInd  = numRead + 1;
        vInd1 = 3 * (fInd - 1) + 1;
        vInd2 = vInd1 + 3 - 1;

        V(vInd1:vInd2,:) = v;
        F(fInd,:)        = vInd1:vInd2;
        N(fInd,:)        = n;

        numRead = numRead + 1;
    end

end
function [F,V,N] = stlascii(M)
    warning('MATLAB:stlread:ascii','ASCII STL files
currently not supported.');
    F = [];
    V = [];
    N = [];
end
% TODO: Change the testing criteria! Some binary STL files
still begin with
% 'solid'.
```

```
function tf = isbinary(A)
% ISBINARY uses the first line of an STL file to identify
its format.
    if isempty(A) || length(A) < 5
        error('MATLAB:stlread:incorrectFormat', ...
      'File does not appear to be an ASCII or binary STL file.');
    end
    if strcmpi('solid',char(A(1:5)'))
        tf = false; % ASCII
    else
        tf = true;  % Binary
    end
end
```

Så legger jeg ved animasjonsfilen jeg har laget.

```
clear
close all

brick = stlread('20mm_cube.stl'); %Downloaded stlreaded
from mathworks and a brick-like stl-file

brick_0 = brick;
for dx = 0:0.01:10;
    for i=1:length(brick.vertices(:,1))
        brick.vertices(i,:) = brick_0.vertices(i,:) +
[dx,0,0];
end
figure(1);
patch(brick);
view(3);
pause(0.05);
end
```