# PROSJEKTOPPGAVE I TEK4040 MATEMATISK MODELLERING AV DYNAMISKE SYSTEMER

Daniel Fremming

November 3, 2024

## 1 Introduksjon

Prosjektoppgaven utgjør det obligatoriske arbeidet i faget TEK4040 Matematisk modellering av dynamiske systemer. Formålet er å modellere og simulere et stivt legeme som roterer. Bevegelsen er basert på løsninger av kinematikk- og spinnlikningene gitt av kinematikklikngene for 3-2-1 Eulervinkler. Denne oppgaven ble gjennomført i MATLAB.

## 2 Fremgangsmåte

Oppgaven har 4 deloppgaver, der følgende besvares:

1. Finne treghetsmatrisa for en murstein med sidekanter 5, 10 og 20 cm (langs hhv. x-, y- og z-aksene) og tettheten $2kg/dm^3$

2. Definér den kinetiske energiellipsoiden og spinnellipsoiden for 3 tilfeller. Her holdes den kinetiske energiellipsoiden konstant. Ellipsoidene plottes i hvert sitt koordinatsystem for hvert tilfellet.

3. Eulerlikningene settes opp og løses i MATLAB for de 3 tilfellene sammen med initialbetingelsene som gir samme løsning som skjæringskurvene i punktet over.

4. Tre animasjonsfilmer som viser rotasjonen til mursteinen sett fra treghetsrommet i hvert av de 3 tilfellene. Disse baseres på kinematikklikningene for 3-2-1 Euler vinkler og løsningen med kinematikk- og spinnlikningene for repsektive tilfeller.

# 3 Matematiske likninger

Sentrale likninger brukt for oppgaven er oppgitt i denne delen. Først regnes ut treghetsmatrisen til mursteinen, den kinetiske energien, de karakteristiske vinkelhastighetene for hvert tilfelle og til slutt kinematikken gitt av 3-2-1 Eulervinklene.

## 3.1 Treghetsmatrise

Treghetsmomentene beregnes komponentvis:

$$J_{11} = \frac{1}{12}m(b^2 + c^2) \tag{1}$$

$$J_{22} = \frac{1}{12}m(a^2 + c^2) \tag{2}$$

$$J_{33} = \frac{1}{12}m(a^2 + b^2) \tag{3}$$

der $a$, $b$ og $c$ er dimensjonene langs henholdsvis x-, y- og z-aksene, og $m$ er massen. Disse verdiene var gitt av oppgave beskrivelsen. Dette ga følgende numeriske treghetsmatrise:

$$J = \begin{bmatrix} 0.0083 & 0 & 0 \\ 0 & 0.0071 & 0 \\ 0 & 0 & 0.0021 \end{bmatrix}$$

## 3.2 Kinetisk energiellipsoide og spinnellipsoide - utregning av vinkelhastighet

### 3.2.1 Initialbetingelser

Starter med rotasjon om hovedakse 3 ($\vec{b}_3$) med periode $T = 1$ s:

$$\tilde{\omega}_3 = \frac{2\pi}{T} = 2\pi \approx 6.28 \text{ rad/s} \tag{4}$$

$$\vec{\omega}_b^{ib} = [0; 0; \tilde{\omega}_3] \tag{5}$$

### 3.2.2 Kinetisk Energi

Den kinetiske energien blir da:

$$K_0 = \frac{1}{2}J_{33}\tilde{\omega}_3^2 \tag{6}$$

$$= \frac{1}{2} \cdot \frac{m}{12}(x^2 + y^2) \cdot (2\pi)^2 \tag{7}$$

Gitt dimensjonene:

$$\begin{aligned} x &= 0.05 \text{ m} \\ y &= 0.10 \text{ m} \\ z &= 0.20 \text{ m} \\ \rho &= 2000 \text{ kg/m}^3 \end{aligned}$$

Beregner massen:

$$\begin{aligned} m &= \rho \cdot x \cdot y \cdot z \\ &= 2000 \cdot 0.05 \cdot 0.10 \cdot 0.20 \\ &= 2 \text{ kg} \end{aligned}$$

Treghetsmomentene blir:

$$\begin{aligned} J_{11} &= \frac{m}{12}(y^2 + z^2) = \frac{2}{12}(0.1^2 + 0.2^2) = 0.00717 \text{ kg} \cdot \text{m}^2 \\ J_{22} &= \frac{m}{12}(x^2 + z^2) = \frac{2}{12}(0.05^2 + 0.2^2) = 0.00675 \text{ kg} \cdot \text{m}^2 \\ J_{33} &= \frac{m}{12}(x^2 + y^2) = \frac{2}{12}(0.05^2 + 0.1^2) = 0.00208 \text{ kg} \cdot \text{m}^2 \end{aligned}$$

Den kinetiske energien blir da:

$$\begin{aligned} K_0 &= \frac{1}{2} \cdot 0.00208 \cdot (2\pi)^2 \\ &= 0.0410 \text{ J} \end{aligned}$$

### 3.2.3 Vinkelhastigheter

Energien beregnet i forrige del kan vi brukes til å regne ut de karakteristiske vinkelhastighetene for hver av aksene:

$$\begin{aligned} \tilde{\omega}_1 &= \sqrt{\frac{2K_0}{J_{11}}} = \sqrt{\frac{2 \cdot 0.0410}{0.00717}} = 3.38 \text{ rad/s} \\ \tilde{\omega}_2 &= \sqrt{\frac{2K_0}{J_{22}}} = \sqrt{\frac{2 \cdot 0.0410}{0.00675}} = 3.49 \text{ rad/s} \\ \tilde{\omega}_3 &= 2\pi = 6.28 \text{ rad/s} \end{aligned}$$

### 3.2.4 Initial vinkelhastigheter for tilfelle 1-3

Tilfelle 1: Rotasjon nær $\vec{b}_1$ (x-akse) Med $\omega_3 = \tilde{\omega}_3/10$:

$$\begin{aligned}\omega_3 &= 0.628 \text{ rad/s} \\ \omega_1 &= \tilde{\omega}_1\sqrt{1 - \frac{\omega_3^2}{\tilde{\omega}_3^2}} \\ &= 3.38\sqrt{1 - \frac{0.628^2}{6.28^2}} \\ &= 3.35 \text{ rad/s}\end{aligned}$$

Gir vinkelhastighetesvektor:

$$\vec{\omega}_b^{ib} = [3.35; 0; 0.628] \text{ rad/s}$$

Tilfelle 2: Rotasjon nær $\vec{b}_2$ (y-akse):

$$\begin{aligned}\omega_3 &= 0.628 \text{ rad/s} \\ \omega_2 &= \tilde{\omega}_2\sqrt{1 - \frac{\omega_3^2}{\tilde{\omega}_3^2}} \\ &= 3.49\sqrt{1 - \frac{0.628^2}{6.28^2}} \\ &= 3.46 \text{ rad/s}\end{aligned}$$

Gir vinkelhastighetesvektor:

$$\vec{\omega}_b^{ib} = [0; 3.46; 0.628] \text{ rad/s}$$

Tilfelle 3: Rotasjon nær $\vec{b}_3$ (z-akse):

$$\vec{\omega}_b^{ib} = [0.338; 0; 6.22] \text{ rad/s}$$

hvor $\omega_1 = \tilde{\omega}_1/10$ og $\omega_3 = 0.99\tilde{\omega}_3$.

## 3.3 Initialbetingelser

De beregnede initialbetingelsene for hvert tilfelle sikrer at den kinetiske energien er bevart for alle 3 tilfellene, dvs. at den kinetiske energien er den samme for hvert tilfelle. For plot av ellipsoidene se 3.5.1, 3.5.2, 3.5.3:

$$\text{Tilfelle 1: } \vec{\omega}_b^{ib} = [3.35; 0; 0.628] \text{ rad/s}$$
$$\text{Tilfelle 2: } \vec{\omega}_b^{ib} = [0; 3.46; 0.628] \text{ rad/s}$$
$$\text{Tilfelle 3: } \vec{\omega}_b^{ib} = [0.338; 0; 6.22] \text{ rad/s}$$

## 3.4 Kinematisk løsning for 3-2-1 Eulervinkler

Vi har at løsningen for Eulersvinkelhastigheter er gitt av:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

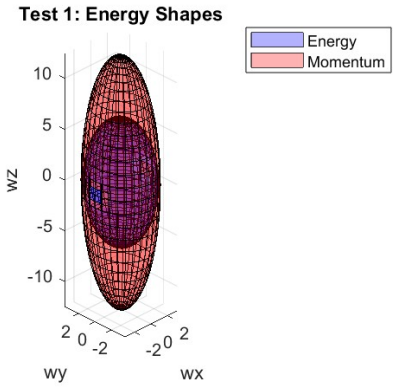
Som komponentvis kan skrives:

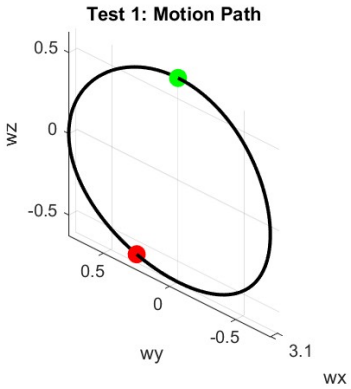
$$\dot{\phi} = \omega_1 + \omega_2 \sin(\phi)\tan(\theta) + \omega_3 \cos(\phi)\tan(\theta)$$
$$\dot{\theta} = \omega_2 \cos(\phi) - \omega_3 \sin(\phi)$$
$$\dot{\psi} = \frac{\omega_2 \sin(\phi) + \omega_3 \cos(\phi)}{\cos(\theta)}$$

## 3.5 Plots

### 3.5.1 Tilfelle 1: Skjæring nær $b_1$-aksen

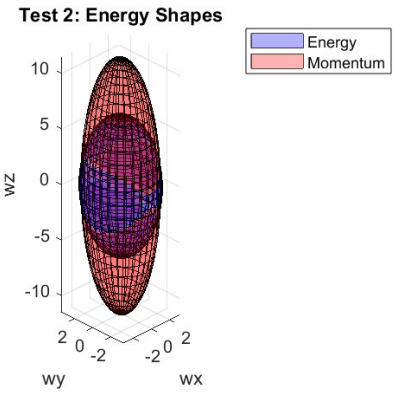
Vinkelhastigheten $\omega_1$ beregnes til å være $\omega_1 = \tilde{\omega_1}\sqrt{1 - (\frac{\omega_3}{\tilde{\omega_3}})^2}$.

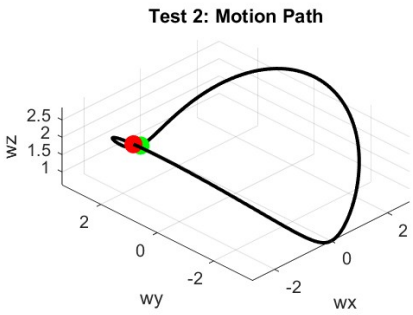



### 3.5.2 Tilfelle 2: Skjæring nær $b_2$-aksen

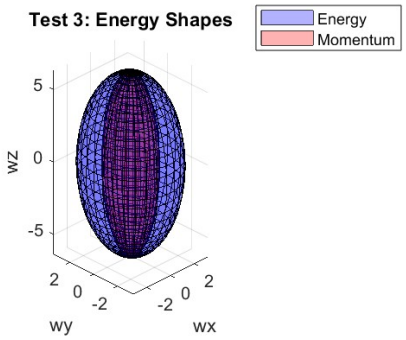
Vinkelhastigheten $\omega_2$ beregnes til å være $\omega_2 = \tilde{\omega_2}\sqrt{1 - (\frac{\omega_3}{\tilde{\omega_3}})^2}$.

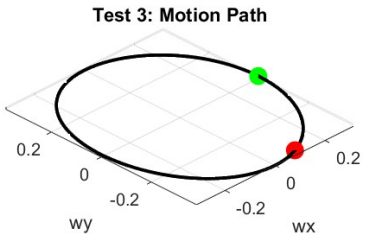


### 3.5.3 Tilfelle 3: skjæring nær $b_3$-aksen

I dette tilfellet roterer legemet kun om $b_3$-aksen, slik at $\omega_1 = 0$ og $\omega_2 = 0$, og $\omega_3 = \tilde{\omega_3}$




# 4 MATLAB-kode

MATLAB-koden for implementeringen er strukturert i flere funksjoner. Disse er inkludert under og navngitt ut i fra deres funksjoner. Disse er satt opp i kronologisk rekekfølge etter hvilke oppgave de ble brukt til å løse.

## 4.1 Utregning av treghetsmatrise

```
% Box size
length = 0.05;  % x
width = 0.10;   % y
height = 0.20;  % z

% Basic properties
density = 2000;  % kg/m^3
mass = density * length * width * height;

% Calculate moment of inertia
% Using basic formulas from textbook
Ixx = mass/12 * (width^2 + height^2);
Iyy = mass/12 * (length^2 + height^2);
Izz = mass/12 * (length^2 + width^2);
I = [Ixx 0 0; 0 Iyy 0; 0 0 Izz];
disp(I)
```

Listing 1: Treghetsmatrise

## 4.2 Plotting og utregning av kinetiske energi- og spinnellipsoider

```
% Initial rotation speed
base_speed = 2*pi;
energy = 0.5 * Izz * base_speed^2;

% Calculate other speeds to keep same energy
wx = sqrt(2*energy/Ixx);
wy = sqrt(2*energy/Iyy);

% Three different starting conditions
initial_conditions = {
    [wx * 0.99, 0, base_speed * 0.1],      % Mostly around x
    [0, wy * 0.99, base_speed * 0.1],      % Mostly around y
    [wx * 0.1, 0, base_speed * 0.99]       % Mostly around z
```

Listing 2: Initial betingelser

```
    % Calculate initial energy
    init_cond = initial_conditions{test};
    energy = 0.5 * init_cond * I * init_cond';
    momentum = norm(I * init_cond');

    % Draw energy surface
    surf(X*sqrt(2*energy/Ixx), Y*sqrt(2*energy/Iyy), Z*sqrt(2*energy/Izz), ...
        'FaceColor', 'b', 'FaceAlpha', 0.3);
    hold on

    % Draw momentum surface
    surf(X*momentum/Ixx, Y*momentum/Iyy, Z*momentum/Izz, ...
        'FaceColor', 'r', 'FaceAlpha', 0.3);

    grid on
```

```
    axis equal
    view([-45, 30]);
    xlabel('wx'); ylabel('wy'); zlabel('wz');
    title(['Test ' num2str(test) ': Energy Shapes']);
    legend('Energy', 'Momentum');
```

Listing 3: Utregning og plotting av ellipsoider

## 4.3 Utregning av løsning for Eulerlikingene

```
% Function to calculate motion
function dw = calculate_motion(~, w, I)
    % Basic Euler equations
    dw = zeros(3,1);
    dw(1) = ((I(2,2) - I(3,3)) * w(2) * w(3)) / I(1,1);
    dw(2) = ((I(3,3) - I(1,1)) * w(3) * w(1)) / I(2,2);
    dw(3) = ((I(1,1) - I(2,2)) * w(1) * w(2)) / I(3,3);
end
```

Listing 4: Eulerlikning

```
% Solve differential equations

    for test = 1:3
    % Solve differential equations
    [t, w] = ode45(@(t,w) calculate_motion(t, w, I), time
        , initial_conditions{test}, options);

    % Make a new figure
    figure('Position', [50, 500-200*test, 800, 300]);

    % Plot the motion path
    subplot(1, 2, 2);
    plot3(w(:,1), w(:,2), w(:,3), 'k', 'LineWidth', 2);
    hold on
    % Mark start and end
    plot3(w(1,1), w(1,2), w(1,3), 'go', 'MarkerFaceColor'
        , 'g', 'MarkerSize', 10);
    plot3(w(end,1), w(end,2), w(end,3), 'ro', '
       MarkerFaceColor', 'r', 'MarkerSize', 10);
    grid on
    axis equal
    view([-45, 30]);
```

```
    xlabel('wx'); ylabel('wy'); zlabel('wz');
    title(['Test ' num2str(test) ': Motion Path']);
```

Listing 5: Plotting av skjæringskurvene

## 4.4 Rotasjon av murstein

```
% Update rotation angles
function angles = update_box_angles(angles, omega, dt)
    phi = angles(1);
    theta = angles(2);

    % Keep theta in safe range
    if theta > pi/2 - 0.01
        theta = pi/2 - 0.01;
    elseif theta < -pi/2 + 0.01
        theta = -pi/2 + 0.01;
    end

    % Calculate angle changes
    dphi = omega(1) + tan(theta) * (sin(phi)*omega(2) +
       cos(phi)*omega(3));
    dtheta = cos(phi)*omega(2) - sin(phi)*omega(3);
    dpsi = (sin(phi)*omega(2) + cos(phi)*omega(3)) / cos(
       theta);

    % Update angles
    angles = angles + dt * [dphi dtheta dpsi];

    % Keep angles between 0 and 2pi
    angles = mod(angles, 2*pi);
end

% Make rotation matrix
function R = make_rotation_matrix(angles)
    phi = angles(1);
    theta = angles(2);
    psi = angles(3);

    % Basic rotation matrices
    Rx = [1 0 0; 0 cos(phi) -sin(phi); 0 sin(phi) cos(phi
       )];
```

```
33      Ry = [cos(theta) 0 sin(theta); 0 1 0; -sin(theta) 0
            cos(theta)];
34      Rz = [cos(psi) -sin(psi) 0; sin(psi) cos(psi) 0; 0 0
            1];
35
36      % Combine rotations
37      R = Rx * Ry * Rz;
38  end
```

Listing 6: Rotasjon