IMU-Daten einer Trajektorie

Christoph Kurz

March 22, 2022

Inhaltsverzeichnis

Anwendung

Im Folgenden soll ein Matlab-Tool vorgestellt werden, das es ermöglicht, anhand einer vorgegebenen Trajektorie und einer vorgegebenen Orientierung die Ausgangsdaten einer IMU (Inertialen Messeinheit) zu simulieren. Die Vorgabe der Trajektorie $\vec{s}(t)$ und der Orientierung $\vec{\varphi}(t)$ soll mithilfe einer Funktion (zur symbolischen Berechnung) und mithilfe eines Vektors (zur nummerischen Berechnung) möglich sein:

Vorgabe mithilfe von Funktionen:

```
% Trajektorie als Funktion
sx = @(t) (0.8 .* cos(2*pi*f*t)); % m
sy = @(t) (0.8 .* sin(2*pi*f*t)); % m
sz = @(t) (0.3 .* cos(2*pi*3*f*t)); % m

% Orientierung als Funktion
phix = @(t) (0 .* t); % rad
phiy = @(t) (0 .* t); % rad
phiz = @(t) (0 .* t); % rad
```

Vorgabe mithilfe von Vektoren:

```
% Zeit Vektor
t = (0 : 0.01 : 5); % s

% Trajektorie als Vektor
sx = (0.8 .* cos(2*pi*f*t)); % m
sy = (0.8 .* sin(2*pi*f*t)); % m
sz = (0.3 .* cos(2*pi*3*f*t)); % m

% Orientierung als Vektor
phix = (0 .* t); % rad
phiy = (0 .* t); % rad
phiz = (0 .* t); % rad
```

Entlang dieser Vorgegebenen Trajektorie bewegt sich der Schwerpunkt S eines starren Körpers. Die Inertiale Messeinheit ist dabei im Abstand $\vec{r}$ vom Schwerpunkt entfernt.

```
r = [-75e-3, -75e-3, 25e-3]; % m
```

Um dies zu Verdeutlichen wird in der Abbildung 1 die oben definierte Trajektorie abgebildet.

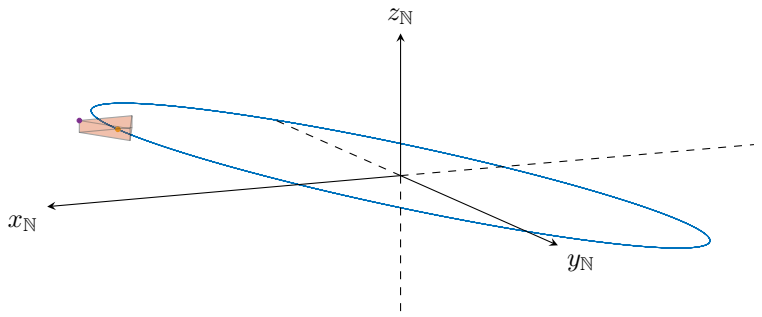


Figure 1: Beispiel einer Trajektorie mit einer IMU

Hierbei wird der starre Körper als rotes Dreieck dargestellt das sich entlang der Trajektorie fortbewegt. Die Orientierung des starren Körpers wurde so gewählt, dass er sich entlang des Tangentialvektors der Trajektorie ausrichtet. Der Schwerpunkt und die Position der IMU werden mit einem gelben und violetten Punkt markiert.

Grundlagen

Bevor näher auf die Algorithmen eingegangen wird, werdend die Grundlagen beschrieben, die im Weiteren von Bedeutung sind. Hierbei sind der Weg $\vec{s}$ und die Orientierung $\vec{\varphi}$ von Bedeutung. Daraus ergeben sich 6 Freiheitsgrade, damit Position und Lage eines Körpers im Raum beschrieben werden kann. Weiters sind die folgenden Ableitungen dieser Größen von Bedeutung:

- die Geschwindigkeit $\vec{v} = \dot{\vec{s}}$
- die Winkelgeschwindigkeit $\vec{\omega} = \dot{\vec{\varphi}}$
- die lineare Beschleunigung $\vec{a} = \dot{\vec{v}}$
- die Winkelbeschleunigung $\vec{\alpha} = \dot{\vec{\omega}}$

Wobei $\dot{(\cdot)}$ die zeitliche Ableitung kennzeichnet. Die beschriebenen Größen lassen sich in lineare Größen (Weg $\vec{s}$, Geschwindigkeit $\vec{v}$, lineare Beschleunigung $\vec{a}$) und in Drehgrößen (Orientierung $\vec{\varphi}$, Winkelgeschwindigkeit $\vec{\omega}$, Winkelbeschleunigung $\vec{\alpha}$) unterteilen.

Orientierung im dreidimensionalen Raum mithilfe von Euler-Winkeln und Drehmatrizen

Um die Orientierung von Körpern zu beschreiben werden Euler-Winkel verwendet. Aus diesen können Drehmatrizen gewonnen werden, die eine Rotation des Koordinatensystem (eine Änderung der Orientierung) ermöglichen. Dazu müssen verschiedenen Koordinatensysteme eingeführt werden:

- Inertialsystem $\mathbb{N}$
- Körperkoordinatensystem $\mathbb{B}$

Das Inertialsystem wird nach ENU-Konvention definiert. ENU bedeutet East, North, Up (also Osten, Norden, oben) und beschreibt die Richtung der Koordinaten-Achsen (Osten = x, Norden = y und oben = z). Die Körperkoordinatensysteme beschreibt das Koordinatensystem des starren Körpers.

Euler-Winkel

Die Euler-Winkel beschreiben die Drehlage/Orientierung eines Körpers durch 3 Winkel φ_x, φ_y und φ_z der jeweiligen Achse. Zusammen beschreiben sie den Vektor $\vec{\varphi}$:

$$\vec{\varphi} = \begin{bmatrix} \varphi_x \\ \varphi_y \\ \varphi_z \end{bmatrix} \tag{1}$$

Es wird unterschieden zwischen den klassischen Euler-Winkeln und den Tait-Bryan-Winkeln, wobei diese wieder in unterschiedliche Konventionen eingeteilt werden können.

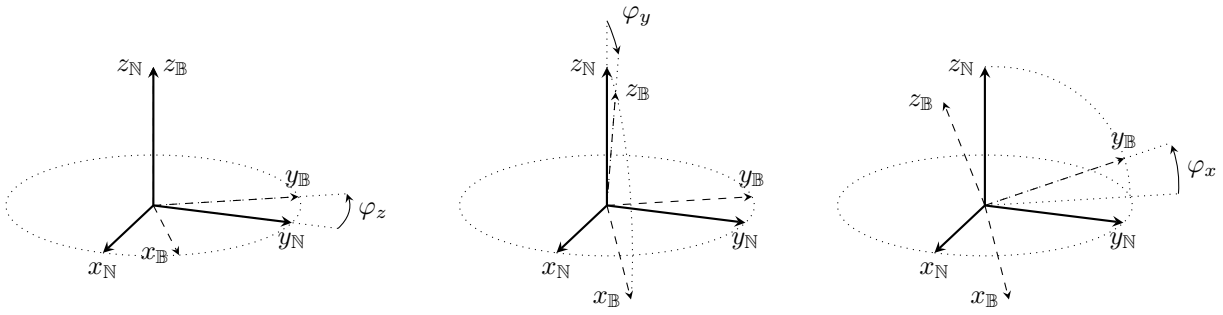


Figure 2: Darstellung der Tait-Bryan-Winkel nach z-y-x (intrinsisch) Konvention

Die am häufigsten verwendete Konvention ist die z-y-x-Konvention (intrinsische Konvention), wobei z-y-x die Reihenfolge der zu drehenden Achsen definiert. Intrinsisch bedeutet, dass jeweils um die neu entstehende Achse gedreht wird. Abbildung 2 zeigt schrittweise wie mit dieser Konvention Koordinatensysteme gedreht werden. Im

Gegensatz zu intrinsischen Drehungen werden extrinsische Drehungen um die Achse des alten Koordinatensystems gedreht. Jede intrinsische Drehung kann in eine extrinsische Drehung umgewandelt werden und vice versa. Die Drehung z-y-x ist beispielsweise intrinsisch äquivalent mit der extrinsischen x-y-z-Drehung.

Drehmatrix

Eine andere Möglichkeit, die Euler-Winkel zu manipulieren, ist die Drehmatrix. Sie ergibt sich aus der Multiplikation dreier Matrizen, die eine Rotation um die jeweilige Koordinaten-Achse beschreiben:

$$\mathbf{R}(\vec{\varphi}) = \underbrace{\begin{bmatrix} \cos(\varphi_z) & -\sin(\varphi_z) & 0 \\ \sin(\varphi_z) & \cos(\varphi_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{R_z}(\varphi_z)} \cdot \underbrace{\begin{bmatrix} \cos(\varphi_y) & 0 & \sin(\varphi_y) \\ 0 & 1 & 0 \\ -\sin(\varphi_y) & 0 & \cos(\varphi_y) \end{bmatrix}}_{\mathbf{R_y}(\varphi_y)} \cdot \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi_x) & -\sin(\varphi_x) \\ 0 & \sin(\varphi_x) & \cos(\varphi_x) \end{bmatrix}}_{\mathbf{R_x}(\varphi_x)} \tag{2}$$

Um einen Vektor vom Inertialsystem $\mathbb{N}$ in das Körpersystem $\mathbb{B}$ (Rotation des Punktes) zu transformieren, wird der zu rotierende Vektor $\vec{v}_\mathbb{N} = (v_x, v_y, v_z)^T$ mit der Drehmatrix $\mathbf{R}(\vec{\varphi})$ multipliziert:

$$\vec{v}_\mathbb{B} = \mathbf{R}(\vec{\varphi}) \cdot \vec{v}_\mathbb{N} \tag{3}$$

Die Indizes geben dabei das Koordinatensystem des jeweiligen Vektors an. Soll ein Vektor vom Körpersystem in das Inertialsystem (Rotation des Koordinatensystems) transformiert werden, muss der zu rotierende Vektor mit der inversen Drehmatrix multipliziert werden:

$$\vec{v}_\mathbb{B} = (\mathbf{R}(\vec{\varphi}))^{-1} \cdot \vec{v}_\mathbb{N} \tag{4}$$

Bestimmung der abgeleiteten Größen

Um die Geschwindigkeit oder die Beschleunigung zu bestimmen muss der Weg durch Differenzieren nach der Zeit bestimmt werden:

$$\vec{v}(t) = \frac{\mathrm{d}\vec{s}(t)}{\mathrm{d}t} \tag{5}$$

$$\vec{a}(t) = \frac{\mathrm{d}\vec{v}(t)}{\mathrm{d}t} \tag{6}$$

Die Winkelgeschwindigkeit wird nicht über die Euler-Winkel, sondern über die Rotationsmatrix bestimmt:

$$\begin{bmatrix} 0 & -\omega_z(t) & \omega_y(t) \\ \omega_z(t) & 0 & -\omega_x(t) \\ -\omega_y(t) & \omega_x(t) & 0 \end{bmatrix} = \mathbf{\Omega}(t) = \frac{\mathrm{d}\mathbf{R}(t)}{\mathrm{d}t}\mathbf{R}(t)^{-1} \tag{7}$$

Anschließend kann die Winkelbeschleunigung mit anschließendem differenzieren gewonnen werden:

$$\vec{\alpha}(t) = \frac{\mathrm{d}\vec{\omega}(t)}{\mathrm{d}t} \tag{8}$$

Mit Matlab kann die Differentiation sowohl symbolisch (mit der Symbolic Math Toolbox) als auch nummerisch (mit der Funktion `x_ = gradient(x,t)`) durchgeführt werden

Mechanische Grundlagen

Um die Daten der IMU (am Punkt I) anhand der Größen am Schwerpunkt S zu bestimmen, kann folgende kinematische Beziehung angewandt werden:

$$\vec{a_I}(t) = \vec{a_S}(t) + \vec{\omega}(t) \times (\vec{\omega}(t) \times \vec{r}) + \vec{\alpha}(t) \times \vec{r} \tag{9}$$

Die einzelnen Größen können folgendermaßen interpretiert werden:

- $\vec{a_S}$ ist die Beschleunigung im Schwerpunkt S und setzt sich aus Gravitationsbeschleunigung $\vec{g}$ und linearer Beschleunigung $\vec{a_l}$ zusammen.
- $\vec{\omega}(t) \times (\vec{\omega}(t) \times \vec{r})$ entspricht der Zentrifugalbeschleunigung $\vec{a_c}$, die durch die Drehung des starren Körpers entsteht.
- $\vec{\alpha}(t) \times \vec{r}$ entspricht der Euler-Kraft und entsteht durch eine beschleunigte Drehung des starren Körpers mit der Winkelbeschleunigung $\vec{\alpha}(t)$.
- $\vec{a_I}$ ist die resultierende Beschleunigung an der Punkt der IMU.

Tangentialvektor / Tangentialwinkel

In den meisten Fällen ist es üblich, dass bewegte Fahrzeuge (Autos, Flugzeuge) tangential zur Trajektorie ausgerichtet sind (bei Drohen ist das nicht der Fall). Der Tangentialvektor $\vec{t}(t) = (t_x(t), t_y(t), t_z(t))^T$ kann über die lineare Geschwindigkeit $\vec{v}(t)$ bestimmt werden:

$$\vec{t}(t) = \frac{\vec{v}(t)}{|\vec{v}(t)|} \tag{10}$$

Anhand dieses Vektors können die Euler-Winkel bestimmt werden die eine Rotation zwischen dem Inertialsystem und dem Tangentialvektor $\vec{t}(t)$ beschreibt. Da es sich aber nur um einen Vektor handel können nur zwei der drei Euler-Winkel (φ_z und φ_y) gefunden werden, der dritte Euler-Winkel (φ_x) wird 0 gesetzt:

$$\varphi_z(t) = \text{atan2}(t_y(t), t_x(t)) \tag{11}$$

$$\varphi_y(t) = -\text{asin}\left(\frac{t_z(t)}{|\vec{t}(t)|}\right) \tag{12}$$

$$\varphi_x(t) = 0 \tag{13}$$

Funktionen

Im Nachfolgendem werden die einzelnen Funktionen beschrieben, um Daten einer IMU zu bestimmen und darzustellen.

Lineare Größen `my_lin`

Mit der Funktion `my_lin` werden die lineare Geschwindigkeit $\vec{v}(t)$ und lineare Beschleunigung $\vec{a}(t)$ anhand der Trajektorie $\vec{s}(t)$ bestimmt. Dazu werden (5) und (6) angewandt.

Syntax

```
[s_calc, v_calc, a_calc] = my_lin(s, option, t_)
```

Parameter

- `s` 3D-Vektor oder Cell mit 3 Vektorfunktionen der Trajektorie
- `option` String: `'num'` oder `'sym'` zur Auswahl für nummerische oder symbolische Berechnung
- `t_` 1D-Zeitvektor
- `s_calc` berechneter 3D-Vektor für die Trajektorie (im nummerischen fall ident mit `s`)
- `v_calc` berechneter 3D-Vektor für die lineare Geschwindigkeit
- `a_calc` berechneter 3D-Vektor für die lineare Beschleunigung

Beispiel

Nummerisch

```
%% Zeit
t_start = 0; % s
t_stop = 5; % s
t_step = 0.005; % s

t = (t_start:t_step:t_stop); % s

%% Frequenz
f = 1; % Hz

%% Trajektorie
sx = 0.8 * 5 .* cos(2*pi*f*t); % m
sy = 0.8 * 5 .* sin(2*pi*f*t); % m
sz = 0.3 * 5 .* cos(2*pi*f*t); % m

s = [sx', sy', sz'];

%% Berechnung lineare Größen
[s, v, a] = my_lin(s,"num",t);
```

Symbolisch

```
%% Zeit
t_start = 0; % s
t_stop = 5; % s
t_step = 0.005; % s

t = (t_start:t_step:t_stop); % s

%% Frequenz
f = 1; % Hz
```

```
%% Trajektorie
sx = @(t) (0.8 .* cos(2*pi*f*t)); % m
sy = @(t) (0.8 .* sin(2*pi*f*t)); % m
sz = @(t) (0.3 .* cos(2*pi*f*t)); % m

s = {sx, sy, sz};

%% Berechnung lineare Größen
[s, v, a] = my_lin(s,"sym",t);
```

Tangentialwinkel `my_tang`

Mit der Funktion `my_tang` können die Euler-Winkel bestimmt werden, die benötigt werden um einen Vektor vom Inertialsystem in ein Körperkoordinatensystem (ausgerichtet zum Tangentialvektor) zu drehen. Dazu werden (10) - (13) angewandt.

Syntax

`[phi_tz, phi_ty, phi_tx, ta] = my_tang(v, t)`

Parameter

- `v` 3D-Vektor der linearen Geschwindigkeit
- `t` Zeit
- `phi_tz` 1D-Vektor für z-Komponente der Euler-Winkel
- `phi_ty` 1D-Vektor für y-Komponente der Euler-Winkel
- `phi_tx` 1D-Vektor für x-Komponente der Euler-Winkel
- `ta` 3D-Vektor für Tangentialvektor

Beispiel

```
%% Zeit
t_start = 0; % s
t_stop = 5; % s
t_step = 0.005; % s

t = (t_start:t_step:t_stop); % s

%% Frequenz
f = 1; % Hz

%% Trajektorie
sx = 0.8 * 5 .* cos(2*pi*f*t); % m
sy = 0.8 * 5 .* sin(2*pi*f*t); % m
sz = 0.3 * 5 .* cos(2*pi*f*t); % m

s = [sx', sy', sz'];

%% Berechnung lineare Größen
[s, v, a] = my_lin(s,"num",t);

%% Berechnung Tangentialwinkel
[phi_tz, phi_ty, phi_tx, ta] = my_tang(v, t);
```

Winkel Größen `my_ang`

Mit der Funktion `my_ang` werden die Winkelgeschwindigkeit $\vec{\omega}(t)$ und Winkelbeschleunigung $\vec{\alpha}(t)$ anhand der Euler-Winkel $\vec{\varphi}(t)$ bestimmt. Dazu werden (7) und (8) angewandt.

Syntax

```
[phi_calc, omega_calc, alpha_calc] = my_ang(phi, option, t_)
```

Parameter

- `phi` 3D-Vektor oder Cell mit 3 Vektorfunktionen der Eulerwinkel
- `option` String: `'num'` oder `'sym'` zur Auswahl für nummerische oder symbolische Berechnung
- `t_` 1D-Zeitvektor
- `phi_calc` berechneter 3D-Vektor für die Euler-Winkel (im nummerischen fall ident mit `s`)
- `omega_calc` berechneter 3D-Vektor für die Winkelgeschwindigkeit
- `alpha_calc` berechneter 3D-Vektor für die Winkelbeschleunigung

Beispiel

Nummerisch

```
%% Zeit
t_start = 0; % s
t_stop = 5; % s
t_step = 0.005; % s

t = (t_start:t_step:t_stop); % s

%% Frequenz
f = 1; % Hz

%% Trajektorie
phix = 0 .* t; % rad
phiy = 0 .* t; % rad
phiz = 0.3 .* cos(2*pi*f*t); % rad

phi = [phix', phiy', phiz'];

%% Berechnung Winkelgrößen
[phi, omega, alpha] = my_ang(phi,"num",t);
```

Symbolisch

```
%% Zeit
t_start = 0; % s
t_stop = 5; % s
t_step = 0.005; % s

t = (t_start:t_step:t_stop); % s

%% Frequenz
f = 1; % Hz

%% Trajektorie
phix = @(t) (0.8 .* cos(2*pi*f*t)); % m
phiy = @(t) (0.8 .* sin(2*pi*f*t)); % m
```

```
phiz = @(t) (0.3 .* cos(2*pi*f*t)); % m

phi = {phix, phiy, phiz};

%% Berechnung Winkelgrößen
[phi, omega, alpha] = my_ang(phi,"sym",t);
```

Rotation `my_rotate`

Mit der Funktion `my_rotate` können Punkte oder Vektoren von einem Koordinatensystem (z.B. das Inertialsystem) in ein anderes Koordinatensystem (z.B. das Körperkoordinatensystem gedreht werden). Dazu wird (2) und (3) verwendet.

Syntax

`[aB] = my_rotate(phi, aN, t)`

Parameter

- `phi` 3D-Vektor der Euler-Winkel (Orientierung zwischen Inertialsystem und Körperkoordinatensystem)
- `aN` 3D-Vektor im Inertialsystem
- `t` Zeit
- `aB` 3D-Vektor im Körperkoordinatensystem
- `phi_tx` 1D-Vektor für x-Komponente der Euler-Winkel

Beispiel

```
%% Zeit
t_start = 0; % s
t_stop = 5; % s
t_step = 0.005; % s

t = (t_start:t_step:t_stop); % s

%% Frequenz
f = 1; % Hz

%% Trajektorie
sx = 0.8 * 5 .* cos(2*pi*f*t); % m
sy = 0.8 * 5 .* sin(2*pi*f*t); % m
sz = 0.3 * 5 .* cos(2*pi*f*t); % m

s = [sx', sy', sz'];

%% Berechnung lineare Größen
[s_N, v_N, a_N] = my_lin(s,"num",t);
% lineare Größen im Schwerpunkt S und referenzerit auf das Inertialsystem N

%% Berechnung Tangentialwinkel
[phi_tz, phi_ty, phi_tx, ta] = my_tang(v_N, t);

phi = [phi_tx, phi_ty, phi_tz];

%% Rotation der lineare Beschleunigung
```

```
[a_B] = my_rotate(phi, a_N, t);
% lineare Beschleunigung im Schwerpunkt S und referenzerit auf das Körperkoorinatensystem B
```

IMU Größen `my_imu`

Mit der Funktion `my_imu` können anhand der Größen im Schwerpunkt S die Winkelgeschwindigkeit und die lineare Beschleunigung im Punkt I bestimmt werden. Dazu kann (9) verwendet werden. Damit die Daten aus der Sicht der IMU bestimmt werden müssen Größen im Schwerpunkt S ins Körperkoordinatensystem $\mathbb{B}$ gedreht werden. Dazu kann die Funktion `my_rot` verwendet werden.

Syntax

```
[aIMU] = my_imu(aB, omega, alpha, r, t)
```

Parameter

- `aS` 3D-Vektor der linearen Beschleunigung im Körperkoordinatensystem $\mathbb{B}$ am Schwerpunkt S
- `omega` 3D-Vektor der Winkelgeschwindigkeit
- `alpha` 3D-Vektor der Winkelbeschleunigung
- `r` Abstandsvektor zwischen Schwerpunkt und IMU
- `t` 1D-Vektor der Zeit
- `aIMU` 3D-Vektor der linearen Beschleunigung im Körperkoordinatensystem $\mathbb{B}$ am Punkt I

Beispiel

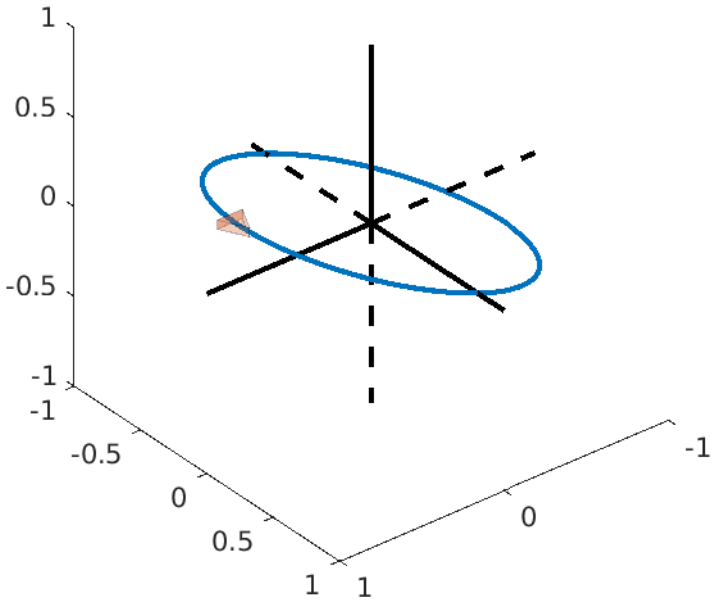
```
%% Zeit
t_start = 0; % s
t_stop = 5; % s
t_step = 0.005; % s

t = (t_start:t_step:t_stop); % s

%% Frequenz
f = 1; % Hz

%% Trajektorie
sx = 0.8 * 5 .* cos(2*pi*f*t); % m
sy = 0.8 * 5 .* sin(2*pi*f*t); % m
sz = 0.3 * 5 .* cos(2*pi*f*t); % m

s = [sx', sy', sz'];

%% Position IMU
r = [-150e-3,-150e-3,50e-3]./2; % m

%% Berechnung lineare Größen
[s_S_N, v_S_N, a_S_N] = my_lin(s,"num",t);
% lineare Größen im Schwerpunkt S und referenzerit auf das Inertialsystem N

%% Berechnung Tangentialwinkel
[phi_tz, phi_ty, phi_tx, ta] = my_tang(v_S_N, t);

phi = [phi_tx, phi_ty, phi_tz];

%% Berechnung Wineklgrößen
```

```
[phi, omega, alpha] = my_ang(phi,"num",t);
% Winkelgrößen des starren Körpers (sind am gesammten Körper ident)

%% Rotation der lineare Beschleunigung
[a_S_B] = my_rotate(phi, a_S_N, t);
% lineare Beschleunigung im Schwerpunkt S und referenzerit auf das Körperkoorinatensystem B

%% Berechnung IMU Daten
[a_I_B] = my_imu(a_S_B, omega, alpha, r, t);

% a_I_B entspricht gemessenen Beschleunigung der IMU
a_IMU = a_I_B;
% omega entspricht der gemessenen Winkelgeschwindigkeit der IMU
omega_IMU = omega;
```

Beispiele

Fehler der numerischen Differentiation im Vergleich zur symbolischen Differentiation

Im Folgenden Soll der Fehler zwischen der nummerischen Berechnung und der symbolischen Berechnung dargestellt



werden. Als Trajektorie wird ein im Raum gekippter Kreis verwendet (siehe Abbildung **??**).

```
%%
clear all
close all
clc

%% Zeit
t_start = 0; % s
t_stop = 1; % s
t_step = 0.005; % s

t = (t_start:t_step:t_stop); % s

%% Frequenz
f = 1; % Hz

%% Trajektorie Nummerisch
sx_n = 0.8 .* cos(2*pi*f*t); % m
sy_n = 0.8 .* sin(2*pi*f*t); % m
sz_n = 0.3 .* cos(2*pi*f*t); % m

phix_n = 0 .* t; % rad
phiy_n = asin((3*sqrt(2)*sin(2*pi*f*t))./sqrt(- 9*cos(4*t*f*pi) + 137)); % rad
phiz_n = 2*pi*f*t + pi/2; % rad
```

```
s_n = [sx_n', sy_n', sz_n'];
phi_n = [phix_n', phiy_n', phiz_n'];

%% Trajektorie Symbolisch
sx_s = @(t) (0.8 .* cos(2*pi*f*t)); % m
sy_s = @(t) (0.8 .* sin(2*pi*f*t)); % m
sz_s = @(t) (0.3 .* cos(2*pi*f*t)); % m

phix_s = @(t) (0 .* t); % rad
phiy_s = @(t) (asin((3*sqrt(2)*sin(2*pi*f*t))./sqrt(- 9*cos(4*t*f*pi) + 137))); % rad
phiz_s = @(t) (2*pi*f*t + pi/2); % rad

s_s = {sx_s, sy_s, sz_s};
phi_s = {phix_s, phiy_s, phiz_s};

%% Berechnung lineare Größen nummerisch
[s_n, v_n, a_n] = my_lin(s_n,"num",t);

%% Berechnung lineare Größen symbolisch
[s_s, v_s, a_s] = my_lin(s_s,"sym",t);

%% Berechnung Winkelgrößen nummerisch
[phi_n, omega_n, alpha_n] = my_ang(phi_n,"num",t);

%% Berechnung Winkelrößen symbolisch
[phi_s, omega_s, alpha_s] = my_ang(phi_s,"sym",t);
m([0,2])

subplot(2,2,2)
plot(t, abs(a_s - a_n)./max(v_s).*100)
xlabel('t')
ylabel('Fehler a(t) in %')
legend('x','y','z');
ylim([0,2])

subplot(2,2,3)
plot(t, abs(omega_n-omega_s)./max(v_s).*100)
xlabel('t')
ylabel('Fehler \omega(t) in %')
legend('x','y','z');
ylim([0,2])

subplot(2,2,4)
plot(t, abs(alpha_s - alpha_n)./max(v_s).*100)
xlabel('t')
ylabel('Fehler \alpha(t) in %')
legend('x','y','z');
ylim([0,2])
%% Differenz zwischen symbolische und nummerische Berechnung
figure
subplot(2,2,1)
plot(t, abs(v_n-v_s)./max(v_s) .* 100)
xlabel('t')
ylabel('Fehler v(t) in %')
legend('x','y','z');
ylim([0,2])
```

```
subplot(2,2,2)
plot(t, abs(a_s - a_n)./max(v_s).*100)
xlabel('t')
ylabel('Fehler a(t) in %')
legend('x','y','z');
ylim([0,2])

subplot(2,2,3)
plot(t, abs(omega_n-omega_s)./max(v_s).*100)
xlabel('t')
ylabel('Fehler \omega(t) in %')
legend('x','y','z');
ylim([0,2])

subplot(2,2,4)
plot(t, abs(alpha_s - alpha_n)./max(v_s).*100)
xlabel('t')
ylabel('Fehler \alpha(t) in %')
legend('x','y','z');
ylim([0,2])
```

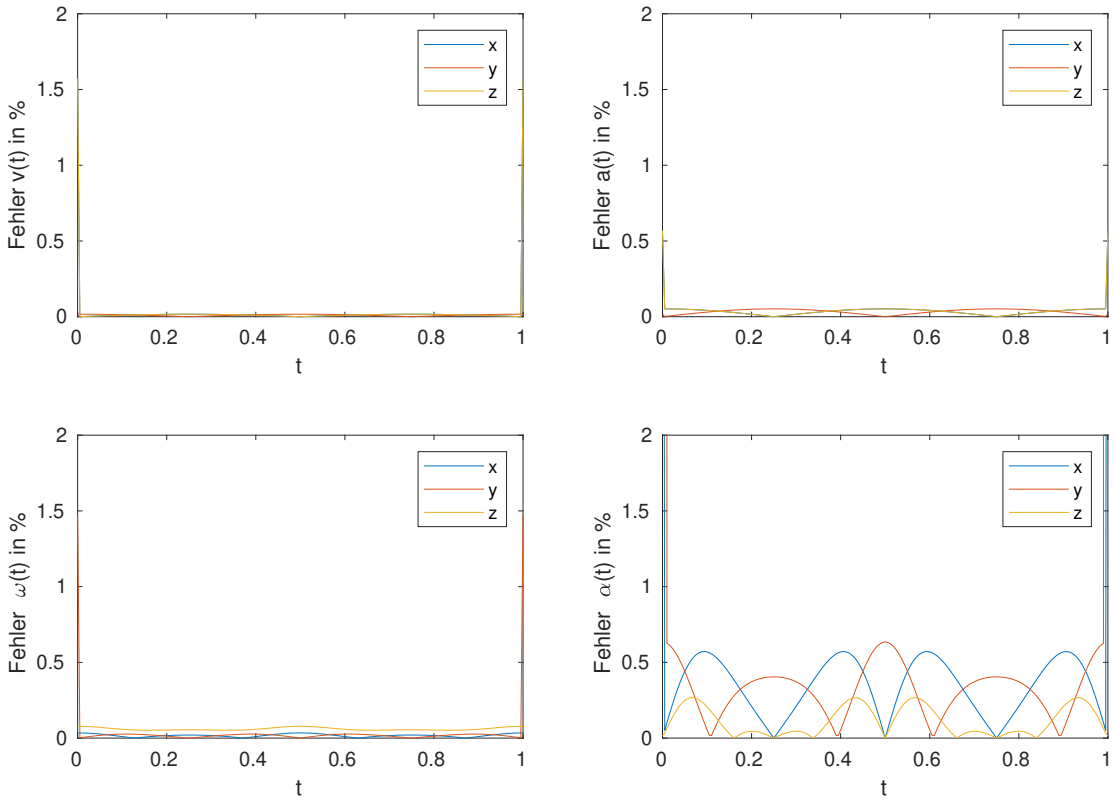


Figure 3: Fehler der numerischen Differentiation

Abegrundetes Viereck

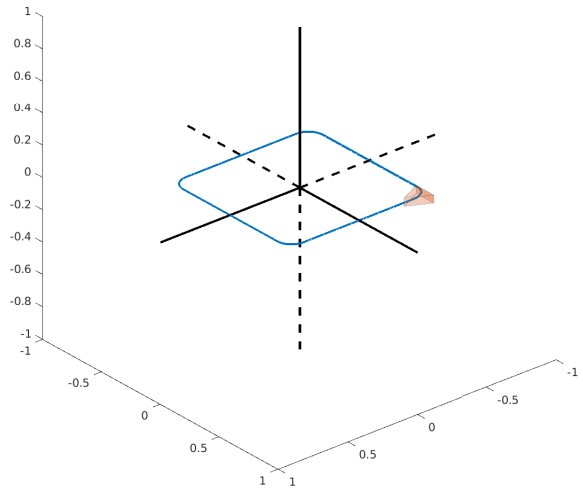


Figure 4: Trajektorie des Abgerundeten Vierecks

```
%%
clear all
close all
clc

%% Konstanten
g = 9.81; %m/s^2 Gravitationskonstante

%% Zeit
t_start = 0; % s
t_stop = 4.7854; % s
t_step = 0.0005; % s

t = (t_start:t_step:t_stop); % s

%% IMU Position
r = [0,0,0]./2; % IMU im Schwerpunkt

%% Trajektorie Nummerisch
v = 0.8; % Geschwindigkeit
rad = 0.1; % Radius der Rundung
omega_ = v / rad; % Winkelgeschwindigkeit in den Rundungen

w2 = 0.5; % Halbe Breite des Rechtecks
h2 = 0.5; % Halbe Höhe des Rechtecks

% 1. line 1
t0 = 0;
```

```
st0 = -(w2-rad);
s01 = st0 - v * t0;

st1 = w2-rad;
t1 = (st1 - s01) / v;

% 2. circle 1
t2 = t1 + 2*pi/(4*omega_);
phi2 = - omega_ * t1;

% 3. line 2
st2 = (h2-rad);
s02 = st2 + v * t2;

st3 = -(h2-rad);
t3 = (st3 - s02) / (-v);

% 4. circle 2
t4 = t3 + 2*pi/(4*omega_);
phi4 = - omega_ * t3;

% 5. line 3
st4 = (w2-rad);
s04 = st4 + v * t4;

st5 = -(w2-rad);
t5 = (st5 - s04) / (-v);

% 6. circle 3
t6 = t5 + 2*pi/(4*omega_);
phi6 = - omega_ * t5;

% 7. line 4
st6 = -(h2-rad);
s06 = st6 - v * t6;

st7 = (h2-rad);
t7 = (st7 - s06) / (v);

% 8. circle 4
t8 = t7 + 2*pi/(4*omega_);
phi8 = - omega_ * t8;



sx =    ( s01 + v .* t) .* (t >= t0 & t < t1) +...
        ( rad*sin(omega_*t + phi2) + (+w2-rad)) .* (t >= t1 & t <= t2) +...
        ( w2 + 0.0 .* t) .* (t > t2 & t < t3) + ...
        ( rad*cos(omega_*t + phi4) + (+w2-rad) ) .* (t >= t3 & t <= t4) +...
        ( s04 - v .* t) .* (t > t4 & t < t5) + ...
        (-rad*sin(omega_*t + phi6) + (-w2+rad) ) .* (t >= t5 & t <= t6) + ...
        (-w2 - 0.0 .* t) .* (t > t6 & t < t7) + ...
        ( rad*sin(omega_*t + phi8) + (-w2+rad) ) .* (t >= t7 & t <= t8);


sy =    ( h2 + 0.0 .* t) .* (t >= 0 & t < t1) +...
```

```
        ( rad*cos(omega_*t + phi2) + (+h2-rad) ) .* (t >= t1 & t <= t2) + ...
        ( s02 - v .* t) .* (t > t2 & t < t3) +...
        (-rad*sin(omega_*t + phi4) + (-h2+rad) ) .* (t >= t3 & t <= t4) +...
        (-h2 + 0.0 .* t) .* (t > t4 & t < t5) + ...
        (-rad*cos(omega_*t + phi6) + (-h2+rad) ) .* (t >= t5 & t <= t6) + ...
        ( s06 + v .* t) .* (t > t6 & t < t7) + ...
        ( rad*cos(omega_*t + phi8) + (+h2-rad) ) .* (t >= t7 & t <= t8);



sz =    0.0 .* t; % m

s_S_N = [sx', sy', sz'];

%% Berechnung lineare Größen nummerisch
[s_S_N, v_S_N, a_S_N] = my_lin(s_S_N, "num",t);

%% Gravitation
a_S_N = a_S_N + [0,0,-g];

%% Berechnung Tangentialwinkel
[phi_z, phi_y, phi_x, ta] = my_tang(v_S_N, t);

phi = [phi_x, phi_y, phi_z];

%% Berechnung Winkelgrößen nummerisch
[phi, omega, alpha] = my_ang(phi,"num",t);

%% Rotation der lineare Beschleunigung
[a_S_B] = my_rotate(phi, a_S_N, t);

%% Berechnung IMU Daten
[a_I_B] = my_imu(a_S_B, omega, alpha, r, t);

a_IMU = a_I_B;
omega_IMU = omega;

%% Plot Trajektorie
figure
plot(sx,sy,'-')
xlabel("s_x in m")
ylabel("s_y in m")
xlim([-0.6,0.6]);
ylim([-0.6,0.6]);

%% Plot lineare Größen
figure
subplot(3,1,1);
plot(t,s_S_N,'-')
xlabel("t in s")
ylabel("s in m")
legend('x','y','z');

subplot(3,1,2);
plot(t,v_S_N,'-')
xlabel("t in s")
```

```
ylabel("v in m/s")
legend('x','y','z');

subplot(3,1,3);
plot(t,a_S_N,'-')
xlabel("t in s")
ylabel("a in m/s^2")
legend('x','y','z');

%% Plot Winkelgrößen
figure
subplot(3,1,1);
plot(t,phi,'-')
xlabel("t in s")
ylabel("\phi in rad")
legend('x','y','z');

subplot(3,1,2);xlabel("t in s")
ylabel("s in m")
plot(t,omega,'-')
xlabel("t in s")
ylabel("\omega in rad/s")
legend('x','y','z');

subplot(3,1,3);
plot(t,alpha,'-')
xlabel("t in s")
ylabel("\alpha in rad/s^2")
legend('x','y','z');

%% Plot IMU-Daten
figure
subplot(2,1,1);
plot(t,omega_IMU,'-')
xlabel("t in s")
ylabel("\omega in rad/s")
legend('x','y','z');

subplot(2,1,2);
plot(t,a_IMU,'-')
xlabel("t in s")
ylabel("a in m/s^2")
legend('x','y','z');

%% Animation
body.width  = 150e-3;
body.length = 150e-3;
body.heigth = 50e-3;
body.r = r;

figure
tmr = my_traAnim(t, s_S_N, phi, body);
%start(tmr);
%stop(tmr);
```

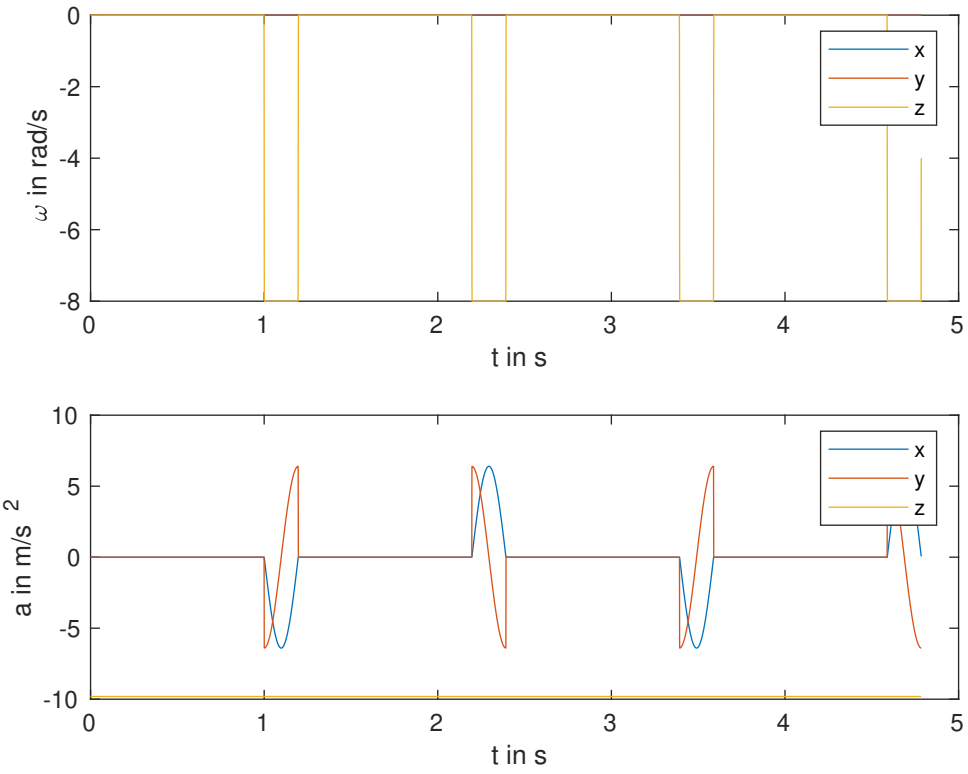


Figure 5: Berechnete IMU-Daten der Trajektorie

8-Trajektorie

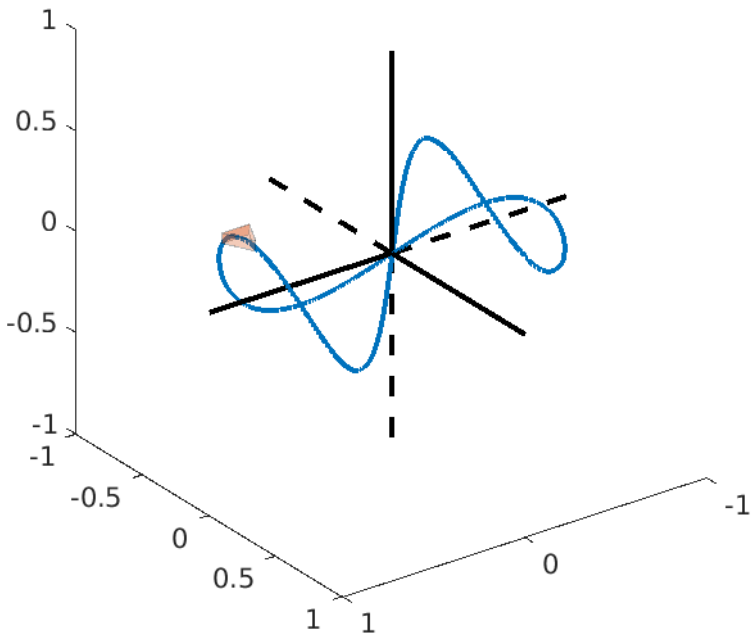


Figure 6: Berechnete IMU-Daten der Trajektorie

```
%%
clear all
close all
clc

%% Konstanten
g = 9.81; %m/s^2 Gravitationskonstante

%% Zeit
t_start = 0; % s
t_stop = 1; % s
t_step = 0.0005; % s

t = (t_start:t_step:t_stop); % s

%% IMU Position
r = [-150e-3,-150e-3,50e-3]./2;

%% Trajektorie Nummerisch
f = 1; %Hz

sx = (0.8 .* cos(2*pi*f*t)); % m
sy = (0.8 .* sin(2*pi*f*t) .* cos(2*pi*f*t)); % m
sz = (0.3 .* cos(2*pi*3*f*t)); % m
```

```
s_S_N = [sx', sy', sz'];

%% Berechnung lineare Größen nummerisch
[s_S_N, v_S_N, a_S_N] = my_lin(s_S_N, "num",t);

%% Gravitation
a_S_N = a_S_N + [0,0,-g];

%% Berechnung Tangentialwinkel
[phi_z, phi_y, phi_x, ta] = my_tang(v_S_N, t);

phi = [phi_x, phi_y, phi_z];

%% Berechnung Winkelgrößen nummerisch
[phi, omega, alpha] = my_ang(phi,"num",t);

%% Rotation der lineare Beschleunigung
[a_S_B] = my_rotate(phi, a_S_N, t);

%% Berechnung IMU Daten
[a_I_B] = my_imu(a_S_B, omega, alpha, r, t);

a_IMU = a_I_B;
omega_IMU = omega;

%% Plot Trajektorie
figure
plot3(sx,sy,sz,'-')
xlabel("s_x in m")
ylabel("s_y in m")
zlabel("s_z in m")

%% Plot lineare Größen
figure
subplot(3,1,1);
plot(t,s_S_N,'-')
xlabel("t in s")
ylabel("s in m")
legend('x','y','z');

subplot(3,1,2);
plot(t,v_S_N,'-')
xlabel("t in s")
ylabel("v in m/s")
legend('x','y','z');

subplot(3,1,3);
plot(t,a_S_N,'-')
xlabel("t in s")
ylabel("a in m/s^2")
legend('x','y','z');

%% Plot Winkelgrößen
figure
subplot(3,1,1);
```

```
plot(t,phi,'-')
xlabel("t in s")
ylabel("\phi in rad")
legend('x','y','z');

subplot(3,1,2);xlabel("t in s")
ylabel("s in m")
plot(t,omega,'-')
xlabel("t in s")
ylabel("\omega in rad/s")
legend('x','y','z');

subplot(3,1,3);
plot(t,alpha,'-')
xlabel("t in s")
ylabel("\alpha in rad/s^2")
legend('x','y','z');

%% Plot IMU-Daten
figure
subplot(2,1,1);
plot(t,omega_IMU,'-')
xlabel("t in s")
ylabel("\omega in rad/s")
legend('x','y','z');

subplot(2,1,2);
plot(t,a_IMU,'-')
xlabel("t in s")
ylabel("a in m/s^2")
legend('x','y','z');

%% Animation
body.width  = 150e-3;
body.length = 150e-3;
body.heigth = 50e-3;
body.r = r;

figure
tmr = my_traAnim(t, s_S_N, phi, body);
%start(tmr);
%stop(tmr);
```

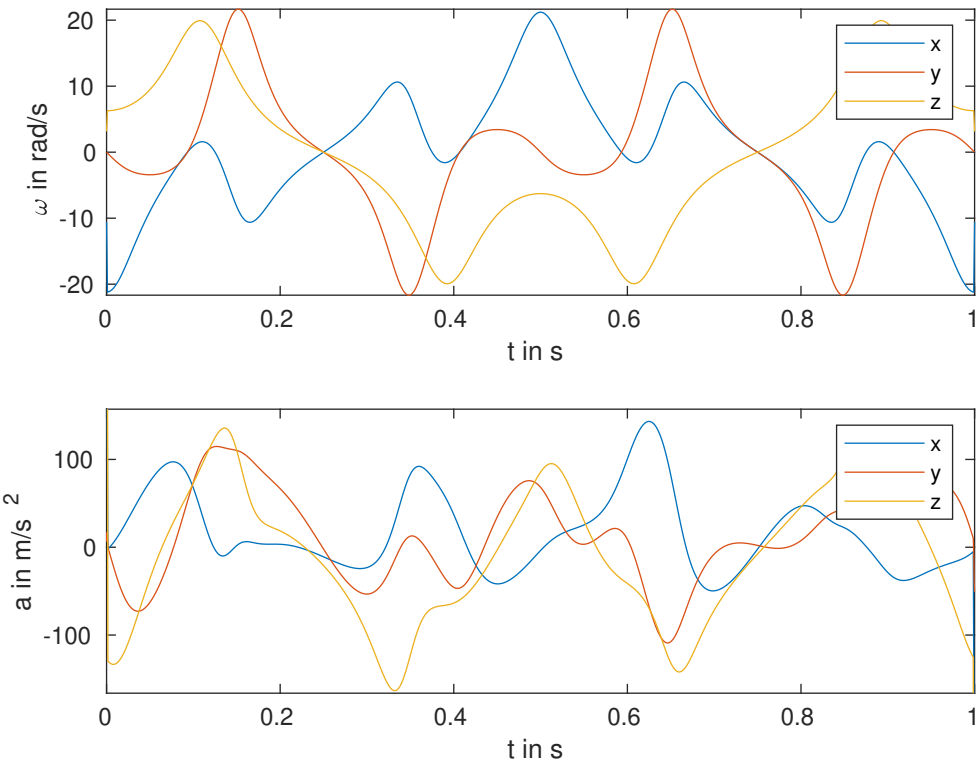


Figure 7: Berechnete IMU-Daten der Trajektorie