Nieliniowy Model Matematyczny Łodzi (3-DOF)

Koło Naukowe PUT Powertrain - Szymon Markowski

February 28, 2026

Abstract

Niniejszy dokument przedstawia wyprowadzenie nieliniowego modelu matematycznego łodzi o 3 stopniach swobody (3-DOF). Model oparty jest na klasycznych równaniach dynamiki obiektów pływających (według T.I. Fossena).

1 Wstęp i Zmienne Stanu

Do opisu ruchu płaskiego łodzi (bez uwzględniania przechyłów i nurkowania) wykorzystujemy model o 3 stopniach swobody (3-DOF): *surge* (ruch wzdłużny), *sway* (ruch poprzeczny) oraz *yaw* (odchylenie/obrót).

Ruch analizowany jest w dwóch układach współrzędnych:

1. **Układ globalny (NED - North-East-Down)**: Nieruchomy względem Ziemi. Służy do opisu pozycji i orientacji na mapie.
2. **Układ lokalny (Body-fixed)**: Związany ze środkiem masy łodzi. Służy do opisu prędkości liniowych, kątowych oraz sił działających na kadłub.

Wektor stanu układu dzieli się na wektor pozycji globalnej $\boldsymbol{\eta}$ oraz wektor prędkości lokalnej $\boldsymbol{\nu}$:

$$\boldsymbol{\eta} = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}, \quad \boldsymbol{\nu} = \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{1}$$

2 Model Kinematyki

Zależność między prędkościami w układzie związanym z obiektem (lokalnym), a tempem zmiany pozycji w układzie inercjalnym (globalnym) wyraża równanie:

$$\dot{\boldsymbol{\eta}} = \mathbf{J}(\psi)\boldsymbol{\nu} \tag{2}$$

gdzie $\mathbf{J}(\psi)$ to macierz rotacji względem osi Z, zależna wyłącznie od kąta kursu ψ.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{3}$$

3 Model Dynamiki (Równania Stanu)

3.1 Ogólne Równanie Dynamiki

Ogólne, wektorowo-macierzowe równanie ruchu jednostki pływającej, opracowane przez T.I. Fossena, wyraża się wzorem:

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} + \boldsymbol{\tau}_{env} \tag{4}$$

gdzie:

- $\mathbf{M}$ – macierz stałej bezwładności i mas dodanych,
- $\mathbf{C}(\boldsymbol{\nu})$ – macierz sił Coriolisa i odśrodkowych,
- $\mathbf{D}(\boldsymbol{\nu})$ – macierz tłumienia hydrodynamicznego,
- $\boldsymbol{\tau}$ – wektor sił i momentów napędowych,
- $\boldsymbol{\tau}_{env}$ – wektor zakłóceń środowiskowych (wiatr, prądy, fale).

3.2 Model Wektorowania Ciągu (Thrust Allocation)

W klasycznych, uproszczonych modelach akademickich często przyjmuje się brak siły poprzecznej z silnika ($\tau_v = 0$). Jednak w przypadku łodzi wyposażonej w obracany pędnik, wychylenie go o kąt α powoduje powstanie znacznej siły poprzecznej, prowadzącej w pierwszej fazie skrętu do zjawiska nieminimalnofazowego (tzw. "kopnięcie rufy" – *stern kick*).

Dla silnika generującego łączny ciąg T [N], umieszczonego w odległości L_m [m] za środkiem ciężkości łodzi, wektor sił i momentów $\boldsymbol{\tau}$ rozkłada się następująco:

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_u \\ \tau_v \\ \tau_r \end{bmatrix} = \begin{bmatrix} T\cos(\alpha) \\ T\sin(\alpha) \\ -L_m \cdot T\sin(\alpha) \end{bmatrix} \tag{5}$$

Znak ujemny przy momencie obrotowym τ_r wynika z faktu, że pchnięcie rufy w lewo (dodatnie τ_v) powoduje obrót dziobu w prawo (ujemne r).

3.3 Struktura Macierzy Modelu (3-DOF)

Dla symetrycznej łodzi ze środkiem układu w środku ciężkości, poszczególne macierze przyjmują postać:

Macierz bezwładności i mas dodanych M:

$$\mathbf{M} = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & 0 \\ 0 & 0 & I_z - N_{\dot{r}} \end{bmatrix} \tag{6}$$

Skośno-symetryczna macierz sił Coriolisa $\mathbf{C}(\boldsymbol{\nu})$:

$$\mathbf{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -(m - Y_{\dot{v}})v \\ 0 & 0 & (m - X_{\dot{u}})u \\ (m - Y_{\dot{v}})v & -(m - X_{\dot{u}})u & 0 \end{bmatrix} \tag{7}$$

Macierz liniowego tłumienia hydrodynamicznego D:

$$\mathbf{D} = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & 0 \\ 0 & 0 & -N_r \end{bmatrix} \tag{8}$$

3.4 Równania Skalarne Przyspieszeń

Odwracając ogólne równanie dynamiki ($\dot{\boldsymbol{\nu}} = \mathbf{M}^{-1}[...]$), otrzymujemy nieliniowe równania stanu dla przyspieszeń w układzie lokalnym. Uwzględniają one wyliczone składowe ciągu oraz zakłócenia środowiskowe:

$$\dot{u} = \frac{1}{m - X_{\dot{u}}}\Big(\tau_u + \tau_{env,u} + (m - Y_{\dot{v}})vr + X_u u\Big) \tag{9}$$

$$\dot{v} = \frac{1}{m - Y_{\dot{v}}}\Big(\tau_v + \tau_{env,v} - (m - X_{\dot{u}})ur + Y_v v\Big) \tag{10}$$

$$\dot{r} = \frac{1}{I_z - N_{\dot{r}}}\Big(\tau_r + \tau_{env,r} + (X_{\dot{u}} - Y_{\dot{v}})uv + N_r r\Big) \tag{11}$$

4 Symulator Dynamiki Statku (python

4.1 Opis modelowanego obiektu

Na potrzeby testowania algorytmów nawigacyjnych zaimplementowano środowisko symulacyjne w języku Python. Symulator stanowi tzw. pusty model fizyczny obiektu, całkowicie odseparowany od algorytmów sterowania.

Fizyka obiektu opiera się na 3-stopniowym (3-DOF) nieliniowym modelu kinematyki i dynamiki statku, wykorzystującym równania T.I. Fossena. Uwzględniono w nim zjawisko wektorowania ciągu oraz tzw. *stern kick* (zarzucanie rufą przy wychyleniu pędnika). Równania różniczkowe rozwiązywane są numerycznie za pomocą solvera nieliniowego `solve_ivp` (metoda RK45) z biblioteki SciPy.

4.2 Struktura skryptu i obsługa pętli głównej

Skrypt podzielony jest na trzy główne bloki: definicję parametrów i równań różniczkowych, główną pętlę symulacji oraz moduł telemetrii i wizualizacji z wykorzystaniem pakietu `matplotlib.gridspec`.

Pętla główna symulatora iteruje po zdefiniowanym wektorze czasu `time_steps`. Z punktu widzenia projektowania algorytmów sterowania, najważniejszym miejscem w kodzie jest sekcja zadawania wejść. Użytkownik (lub w przyszłości zaimplementowany regulator) pełni tu rolę sternika, modyfikując dwie zmienne:

- `T_cmd` – Wartość zadana uciągu silnika wyrażona w Niutonach [N].
- `alpha_cmd` – Wartość zadana kąta wychylenia silnika wyrażona w radianach [rad].

Poniższy listing prezentuje sposób obsługi pętli. Zmienne zadane (`_cmd`) są przed przekazaniem do solvera odpowiednio nasycane (funkcja `np.clip`), co symuluje fizyczne i mechaniczne ograniczenia pędnika (maksymalna moc 200 N oraz mechaniczna blokada obrotu serwomechanizmu przy 45 stopniach).

Listing 1: Struktura pętli głównej i zadawanie wejść symulatora

```
for i in range(len(time_steps)):
    t = time_steps[i]  # Aktualny czas symulacji
    states_sim[i, :] = current_state

    # --- MIEJSCE NA ALGORYTM STEROWANIA ---
    if t < 5.0:
        T_cmd = 0.0           # Silnik wylaczony
        alpha_cmd = 0.0       # Ster prosto
    elif 5.0 <= t < 10.0:
        T_cmd = 150.0         # Skok jednostkowy ciagu
        alpha_cmd = np.radians(-15.0) # Skret w lewo
    else:
        T_cmd = 150.0
        alpha_cmd = 0.0       # Wyjscie z zakretu

    # Nasycenia sprzetowe (fizyczne limity aktuatorow)
    T = np.clip(T_cmd, 0.0, 200.0)
    alpha = np.clip(alpha_cmd, np.radians(-45), np.radians(45))

    # Rozwiazanie rownan fizyki w biezacym kroku (RK45)
    sol = solve_ivp(...)
    current_state = sol.y[:, -1]
```

Dzięki tak zaprojektowanej architekturze, zintegrowanie algorytmu nawigacyjnego polega wyłącznie na zastąpieniu instrukcji warunkowych `if/elif` odpowiednimi równaniami regulatora. Reszta logiki – w tym dynamika nieliniowa, nasycenia sprzętowe oraz odrysowywanie telemetrii na żywo – pozostaje nienaruszona.