Sprawozdanie z projektu z MMM

Projekt nr. 10

Osoby w zespole:
Kewin Kisiel 197866
Mateusz Kuczerowski 197900

Temat projektu:

Projekt 10. Dany jest układ opisany za pomocą transmitancji:

$$G(s) = \frac{a_3 s^3 + a_2 s^2 + a_1 \ s + a_0}{b_4 s^4 + b_3 s^3 + b_2 \ s^2 + b_1 s + b_0}$$

gdzie a_i, b_i to parametry modelu. Należy zaimplementować symulator tego układu umożliwiając uzyskanie odpowiedzi czasowych układu na pobudzenie przynajmniej trzema rodzajami synagłów wejściowych (prostokątny o skończonym czasie trwania, trójkątny, harmoniczny). Symulator powinien umożliwiać zmianę wszystkich parametrów transmitancji oraz sygnałów wejściowych. Należy określać stabilność układu oraz wykreślić charakterystyki częstotliwościowe Bodego (amplitudową i fazową) oraz odpowiedź układu.

Cel Projektu:

Celem projektu było zaimplementowanie numerycznego symulatora odpowiedzi układu opisanego transmitancją o ogólnej postaci:

$$G(s) = \frac{a_3 s^3 + a_2 s^2 + a_1 s^1 + a_0}{b_4 s^4 + b_3 s^3 + b_2 s^2 + b_1 s^1 + b_0}$$

Symulator miał umożliwiać:

- modyfikację parametrów transmitancji oraz sygnałów z poziomu interfejsu,
- analizę odpowiedzi czasowych dla różnych typów sygnałów wejściowych (prostokątny, trójkątny, sinusoidalny, oraz dwa dodatkowe: impuls jednostkowy i skok jednostkowy),
- ocenę stabilności układu,
- wykreślenie charakterystyk częstotliwościowych Bodego (amplitudowej i fazowej).

Działanie programu:

Użytkownik wpisuje współczynniki licznika i mianownika transmitancji, a następnie wybiera typ sygnału wejściowego:
- Prostokątny
- Trójkątny
- Harmoniczny
- Impuls jednostkowy
- Skok jednostkowy

W przypadku wybrania sygnału okresowego, pojawia się możliwość podania częstotliwości i amplitudy, a przy wyborze impulsu i skoku tylko wyboru amplitudy.
Po kliknięciu przycisku „Symuluj”, program rysuje wykresy:
- sygnału wejściowego i wyjściowego w funkcji czasu,
- charakterystyki Bodego (amplitudowa i fazowa),
- biegunów układu w płaszczyźnie zespolonej.

Program w dwóch miejscach pokazuje wzór z transmitancją:
- Na górze z symbolicznymi nazwami współczynników (a_3, a_2 itp.)
- Na dole z rzeczywistymi wartościami wpisanymi przez użytkownika

Obie wersje są generowane jako równania LaTeX.
Równanie LaTeX to sposób graficznego przedstawienia wzoru matematycznego transmitancji, zapisanego w składni języka używanego do profesjonalnego formatowania matematyki.

Główny program main.py oparty jest o trzy zewnętrzne funkcje:
euler.py – symulacja odpowiedzi czasowej
stability.py – znajdowanie biegunów i ocena stabilności
body.py – ręczne obliczanie charakterystyki Bodego

Biblioteki użyte w programie:

import tkinter as tk - to standardowa biblioteka Pythona służąca do tworzenia graficznego interfejsu (GUI). Umożliwia tworzenie okien, przycisków, pól tekstowych oraz innych elementów, które pozwalają użytkownikowi na interakcję z aplikacją.

from tkinter import messagebox - moduł messagebox jest częścią tkintera i pozwala na wyświetlanie okien dialogowych, takich jak komunikaty o błędach, ostrzeżenia czy informacje. W programie przekazuje użytkownikowi informacje zwrotne w trakcie działania programu.

import numpy as np - biblioteka numpy jest podstawowym narzędziem do obliczeń numerycznych w Pythonie. Zapewnia wydajne struktury danych (tablice) oraz funkcje do operacji matematycznych.

import matplotlib.pyplot as plt - to biblioteka służąca do tworzenia wykresów i wizualizacji danych. Umożliwia rysowanie odpowiedzi czasowych układów dynamicznych, charakterystyk częstotliwościowych oraz dowolnych innych wykresów.

from scipy import signal - moduł signal z biblioteki scipy zawiera narzędzia do analizy i przetwarzania sygnałów oraz układów dynamicznych. Pozwala między innymi na definiowanie transmitancji, analizę charakterystyk Bodego, odpowiedzi impulsowej i skokowej systemów LTI (liniowych, niezmiennych w czasie).

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg - FigureCanvasTkAgg to element pośredniczący, który umożliwia osadzanie wykresów tworzonych w Matplotlib wewnątrz interfejsu graficznego opartego na Tkinterze. Dzięki temu użytkownik może oglądać wyniki symulacji bezpośrednio w oknie programu.

from euler import simulate_response_euler – funkcja implementuje numeryczną symulację dynamicznej odpowiedzi liniowego układu opisanego równaniem różniczkowym o maksymalnym rzędzie 4, przy użyciu prostego i popularnego schematu całkowania numerycznego - metody Eulera. Dzięki normalizacji współczynników oraz uwzględnieniu różnych rzędów układu, funkcja jest elastyczna.

Działanie funkcji:

1. Obliczenie kroku czasowego Ts jako różnicy między dwoma kolejnymi punktami czasowymi w wektorze t.
2. Uzupełnienie i odwrócenie współczynników “a” i “b”, by miały stałą długość (odpowiednio 4 i 5 elementów), oraz by pasowały do postaci równania różniczkowego.
3. Normalizacja współczynników przez pierwszy współczynnik b[0] mianownika, co upraszcza równanie.
4. Inicjalizacja wektorów odpowiedzi układu i ich pochodnych (do czwartego rzędu włącznie), a także pochodnych sygnału wejściowego.
5. Określenie rzędu układu na podstawie najwyższego współczynnika różnego od zera zarówno w liczniku, jak i mianowniku.
6. Wyznaczenie numerycznych pochodnych sygnału wejściowego (pierwsza, druga i trzecia pochodna).
7. Symulacja odpowiedzi układu za pomocą metody Eulera. W zależności od rzędu układu (od 0 do 4) stosowane jest odpowiednie równanie rekurencyjne, które wyznacza kolejne wartości odpowiedzi y oraz jej pochodnych.

Przykład dla transmitancji 4 rzędu:

Rozpisanie transmitancji z definicji;

$$G(s) = \frac{Y(s)}{U(s)} = \frac{a_3 s^3 + a_2 s^2 + a_1 s^1 + a_0}{b_4 s^4 + b_3 s^3 + b_2 s^2 + b_1 s^1 + b_0}$$

Przejście do równania różniczkowego:

$$b_4 \frac{d^4 y}{dt^4} + b_3 \frac{d^3 y}{dt^3} + b_2 \frac{d^2 y}{dt^2} + b_1 \frac{dy}{dt} + b_0 y \;=\; a_3 \frac{d^3 u}{dt^3} + a_2 \frac{d^2 u}{dt^2} + a_1 \frac{du}{dt} + a_0 u$$

Normalizacja równania (dzielimy oba boki równania przez b4 jeśli b4≠0, aby uzyskać postać):

$$\frac{d^4 y}{dt^4} \;=\; \frac{1}{b_4}\left(a_3 \frac{d^3 u}{dt^3} + a_2 \frac{d^2 u}{dt^2} + a_1 \frac{du}{dt} + a_0 u \;-\; b_3 \frac{d^3 y}{dt^3} - b_2 \frac{d^2 y}{dt^2} - b_1 \frac{dy}{dt} - b_0 y\right)$$

Aproksymacja pochodnych sygnałów wejściowego i wyjściowego:

$$\frac{du}{dt} \approx \frac{u[k]-u[k-1]}{T_s},$$

$$\frac{d^2 u}{dt^2} \approx \frac{du[k]-du[k-1]}{T_s},$$

itd. (analogicznie dla y(t)).

Całkujemy kolejne pochodne, aż do otrzymania y[k]:

$$d^4 y[k] = \frac{1}{b_4}(a_3 d^3 u[k] + a_2 d^2 u[k] + a_1 du[k] + a_0 u[k]) +$$

$$\frac{1}{b_4}(-b_3 d^3 y[k-1] - b_2 d^2 y[k-1] - b_1 dy[k-1] - b_0 y[k-1])$$

$$d^3 y[k] = d^3 y[k-1] + T_s \cdot d^4 y[k]$$

$$d^2 y[k] = d^2 y[k-1] + T_s \cdot d^3 y[k]$$

$$dy[k] = dy[k-1] + T_s \cdot d^2 y[k]$$

$$y[k] = y[k-1] + T_s \cdot dy[k]$$

from stability import find_roots, check_stability – pierwsza z tych funkcji oblicza bieguny rzeczywiste i zespolone mianownika transmitancji G(s), natomiast druga służy do sprawdzenia, czy dany układ jest stabilny, niestabilny, czy znajduje się na granicy stabilności.

find_roots - funkcja oblicza pierwiastki (bieguny) rzeczywiste i zespolone mianownika transmitancji G(s). Została zaimplementowana tak, aby uwzględniać wszystkie możliwe przypadki w zależności od stopnia mianownika.

Poniżej omówienie każdego przypadku:

a) Stopień 0 (stały mianownik):

Jeśli mianownik transmitancji jest stały (czyli składa się tylko ze współczynnika $a_0 \neq 0$, to nie istnieją żadne pierwiastki – równanie nie ma rozwiązania w dziedzinie zmiennej s. Funkcja w takim przypadku zwraca komunikat o braku pierwiastków.

b) Stopień 1:

Równanie ma postać:

$$a_1 s + a_0 \ = \ 0$$

Jeśli $a_0 \neq 0$, to pierwiastek wyznaczany jest ze wzoru:

$$s = -\frac{a_0}{a_1}$$

c) Stopień 2:

Dla równania kwadratowego:

$$a_2 s^2 + a_1 s + a_0 = 0$$

Najpierw obliczana jest delta:

$$\Delta = a_1^2 - 4a_2 a0 = 0$$

Następnie pierwiastki wyznaczane są ze wzoru kwadratowego:

$$s_{1,2} = \frac{-a_1 \pm \sqrt{\Delta}}{2a_2}$$

Funkcja uwzględnia przypadki:

$\Delta > 0$: dwa różne pierwiastki rzeczywiste,

$\Delta = 0$: jeden podwójny pierwiastek,

$\Delta < 0$: dwa pierwiastki zespolone sprzężone.

d) Stopień 3:

Dla równania trzeciego stopnia stosowana jest metoda Cardana, która umożliwia analityczne wyznaczenie wszystkich pierwiastków (zarówno rzeczywistych, jak i zespolonych). Metoda ta daje bardzo dobre wyniki i jest porównywalna z wynikami uzyskanymi przy pomocy narzędzi numerycznych, takich jak MATLAB.

e) Stopień 4:

Dla równań czwartego stopnia istnieje metoda Ferrariego, jednak nie daje ona stabilnych i dokładnych wyników przy dużych wartościach współczynników. Z tego powodu dla tego przypadku zdecydowano się na użycie wbudowanej funkcji języka Python (np. z biblioteki NumPy lub SymPy), która wykorzystuje sprawdzone metody numeryczne do obliczenia pierwiastków.

check_stability - do sprawdzenia stabilności wykorzystujemy kryterium Routha-Hurwitza, które przedstawia się następująco:

1. Na początku sprawdzamy, czy wszystkie współczynniki mianownika są dodatnie – jest to warunek konieczny, ale niewystarczający do określenia stabilności.

2. Następnie tworzymy tablicę Routha:

Liczba wierszy równa jest liczbie współczynników.

Liczba kolumn wynosi $\frac{n+1}{2}$, gdzie n to stopień mianownika (czyli liczba współczynników minus 1).

Wypełniamy dwa pierwsze wiersze wartościami współczynników, zaczynając od lewej strony – na przemian pierwszy wiersz, potem drugi – od najwyższego stopnia do najniższego.

3. Obliczamy kolejne wiersze tablicy zgodnie ze wzorem:

$$R[i][j] = \frac{R[i-2][0]R[i-1][j+1] - R[i-1][0]R[i-2][j+1]}{-R[i-1][0]}$$

Gdzie:

i – indeks wiersza,

j – indeks kolumny.

4. Na końcu sprawdzamy, ile razy znak zmienia się w pierwszej kolumnie. Jeśli występuje co najmniej jedna zmiana znaku, oznacza to, że transmitancja ma biegun(y) w prawej półpłaszczyźnie zespolonej, co świadczy o niestabilności układu.

from body import compute_bode_manual – służy do numerycznego wyznaczania charakterystyki Bodego dla zadanej transmitancji.

Działanie funkcji:

a) Strona matematyczna - Funkcja analizuje zachowanie transmitancji w dziedzinie częstotliwości. Proces ten można podzielić na następujące etapy:

1. Zamiana transmitancji G(s) na funkcję częstotliwościową:

- Dla transmitancji G(s) podstawiamy s = jω.
- Transmitancja staje się funkcją zespoloną zależną od częstotliwości.

2. Obliczanie wartości wielomianów zespolonych:

- Każdy składnik ak ·(jω)k jest liczony numerycznie.
- Można to porównać do podstawienia zmiennej do wielomianu, tylko że teraz zmienna jest zespolona.

3. Obliczanie modułu i fazy funkcji zespolonej:

- Amplituda : $|G(j\omega)| = \sqrt{Re^2 + Im^2}$
- Faza: $\arg(G(j\omega)) = \arctan\left(\frac{Im}{Re}\right)$

4. Logarytmiczna skala amplitudy:

- Przejście z liniowej do logarytmicznej skali: $Mag_{20dB} = 20\ \log_{10}(|G(j\omega)|)$

b) Kod w programie:

1. Tworzenie pustych list:

```
G_mag = []
G_phase = []
```

2. Pętla po częstotliwościach:

```
for w in w_range:
    jw = 1j * w
```

3. Obliczanie transmitancji dla kolejnych jw:

```
num_eval = sum(c * jw ** (len(num) - i - 1) for i, c in enumerate(num))
den_eval = sum(c * jw ** (len(den) - i - 1) for i, c in enumerate(den))
```

4. Sprawdzamy czy dzielimy przez zero:

```
if abs(den_eval) < 1e-12:
    G_mag.append(np.inf)
    G_phase.append(np.nan)
```

Jeśli mianownik bliski zera to odpowiednio wpisujemy moduł jako nieskończony a fazę niezdefiniowaną.

5. Obliczamy moduł i fazę:

```
else:
    G = num_eval / den_eval
    G_mag.append(20 * np.log10(abs(G)))
    G_phase.append(np.angle(G, deg=True))
```

6. Odwijanie fazy:

```
G_phase = -np.unwrap(np.radians(G_phase)) * (180 / np.pi)
```

- np.unwrap – usuwa skoki fazy ±360° (dzięki czemu wykres jest ciągły),
- Zamiana fazy z powrotem na stopnie po odwinięciu (unwrap działa na radianach),
- “-” na początku: zmiana znaku — bo w klasycznym wykresie Bodego faza opadająca jest przedstawiana z minusem.

Wnioski i podsumowanie:

W ramach projektu nr 10 zrealizowano symulator dynamicznego układu liniowego opisanego transmitancją dowolnego rzędu do 4 stopnia. Program umożliwia analizę odpowiedzi czasowej i częstotliwościowej dla różnych typów sygnałów wejściowych oraz ocenę stabilności układu. Wszystkie kluczowe procedury numeryczne w tym różniczkowanie, całkowanie, obliczanie charakterystyki Bodego oraz analiza stabilności zostały zaimplementowane samodzielnie, bez korzystania z gotowych funkcji analitycznych.

Wnioski:

1. Zastosowanie metody Eulera do symulacji odpowiedzi czasowej pozwala na elastyczne modelowanie układów różnego rzędu i dobrze sprawdza się przy odpowiednio dobranym kroku czasowym.
2. Ręczna implementacja kryterium Routha-Hurwitza umożliwia skuteczną ocenę stabilności układu, niezależnie od jego rzędu.
3. Dla wielomianów trzeciego stopnia zastosowanie metody Cardano dało dokładne i zadowalające wyniki.
4. W przypadku wielomianów czwartego stopnia metoda Ferrariego okazała się niewystarczająco dokładna dla większych współczynników, dlatego użyto funkcji bibliotecznej (co jest jedynym odstępstwem od pełnej niezależności obliczeniowej).
5. Charakterystyki Bodego (amplitudowa i fazowa) zostały obliczone numerycznie na podstawie zespolonej postaci transmitancji $G(j\omega)$, co pozwoliło w pełni zrozumieć proces analizy częstotliwościowej.
6. Program posiada przyjazny interfejs graficzny umożliwiający modyfikację parametrów transmitancji i sygnałów wejściowych oraz wizualizację wyników w postaci wykresów.

Podsumowując, projekt spełnił wszystkie założone cele i wymagania. Zrealizowany symulator jest funkcjonalnym i edukacyjnym narzędziem umożliwiającym analizę układów dynamicznych, zarówno pod kątem odpowiedzi czasowej, jak i charakterystyk częstotliwościowych.