Projekt z Metod Modelowania Matematycznego 2025

Zadanie 10: Implementacja symulatora układu danego za pomocą transmitancji

Natalia Sampławska 197573
Martyna Penkowska 197926

8 czerwca 2025

Okres trwania projektu: Semestr letni roku akademickiego 2025
Prowadzący projekt: dr inż. Marek Tatara

1 Cel

Implementacja symulatora w Pythonie układu danego 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 oraz b_i to zmienne parametry modelu, umożliwiającego uzyskanie odpowiedzi czasowych układu na pobudzenia sygnałami: sinusoidalnym, prostokątnym, piłokształtnym, trójkątnym, impulsem prostokątnym, impulsem jednostkowym i skokiem jednosktkowym. Parametry sygnałów można odpowiednio ustawiać w oknie symulacji. Symulator umożliwia określenie stabilności układu oraz wykreślanie charakterystyk częstotliwościowych Bodego - amplitudowej i fazowej.

2 Zaimplementowane funkcje symulatora

1. Wybór parametrów obiektu
2. Wybór sygnału wejściowego wraz z wszystkimi jego parametrami
3. Generowanie charakterystyk Bodego
4. Implementacja różniczkowania
5. Przedstawienie graficzne sygnału wejściowego i wyjściowego
6. Określenie stabilności układu

3 Opis funkcji

a. **Wybór parametrów obiektu i sygnału wejściowego.**

Z poziomu interfejsu można zmienić wszystkie parametry układu – współczynniki transmitancji oraz wybrany sygnał pobudzający wraz z wartościami, które go charakteryzują (amplitudą, częstotliwością, fazą, szerokością impulsu).

`check_input_values` — funkcja sprawdzająca poprawność wprowadzonych danych. Sprawdzenie czy wszystkie współczynniki licznika i mianownika są liczbami, czy licznik lub mianownik nie są zerowe, czy rząd licznika nie jest wyższy niż rząd mianownika oraz czy parametry sygnału wejściowego są poprawne: amplituda, częstotliwość i szerokość impulsu większe od zera, faza z zakresu od $-\pi$ do π. W przypadku błędnych danych funkcja generuje flagę błędu wyświetlającą się w oknie main view. Gdy dane są poprawne wyświetla się komunikat "Correct parameters". Wywołanie funkcji występuje po naciśnięciu przycisku set parameters oraz przed wykonaniem symulacji układu.

`update_selected_signal`, `update_signal_param_visibility` — funkcje aktualizujące dane sygnału wejściowego. Wybór pobudzenia jest zrealizowany za pomocą radiobuttonów. Po naciśnięciu odpowiedniego przycisku wyświetlają się pod nim parametry danego sygnału, których wartości można wprowadzać z klawiatury

b. **Okno programu.**

Interfejs graficzny jest zrealizowany za pomocą biblioteki PyQt5. Po uruchomieniu kodu otwiera się okno startowe `start_menu`, w którym znajduje się opis założeń projektu. Z niego po naciśnięciu przycisku menu przechodzimy

do głównej części programu `menu_display`. Z tego poziomu zmieniamy parametry transmitancji i pobudzenia. Naciśnięcie przycisku simulate umożliwia przejście do okna symulacji `simulation`, w którym wyświetlają się charakterystyki Bodego, ustawiona postać transmitancji oraz wykresy sygnałów wejściowych i wyjściowych.

c. **Pobudzenia wejściowe układu.**

Układ może być pobudzany następującymi sygnałami:

- **Sinusoidalnym:** $y(t) = A\sin(2\pi ft + \varphi)$
- **Prostokątnym:** $y(t) = A \cdot \operatorname{sgn}\left(\sin(2\pi ft + \varphi)\right)$
- **Piłokształtnym:** $y(t) = \left(\frac{2A}{T}\right)(t \bmod T) - A$
- **Trójkątnym:** $y(t) = A\left(1 - 4\left|\frac{t \bmod T}{T} - \frac{1}{2}\right|\right)$
- **Impulsem prostokątnym:** $y(t) = \begin{cases} A, & \text{dla } 0 < t < pulsewidth \\ 0, & \text{w przeciwnym razie} \end{cases}$
- **Skokiem jednostkowym:** $y(t) = A \cdot u(t)$
- **Impulsem jednostkowym:** $y(t) = A \cdot \delta(t)$

d. **Wyznaczenie wyjścia układu.**

Do wyznaczenia wyjścia wykorzystano różniczkowanie metodą Eulera.

`get_manual_input_derivatives` — aproksymacja pochodnych sygnału wejściowego

$$\dot{u}[i] \approx \frac{u[i] - u[i-1]}{\Delta t} \qquad \text{(pierwsza pochodna)}$$
$$\ddot{u}[i] \approx \frac{\dot{u}[i] - \dot{u}[i-1]}{\Delta t} \qquad \text{(druga pochodna)}$$
$$\dddot{u}[i] \approx \frac{\ddot{u}[i] - \ddot{u}[i-1]}{\Delta t} \qquad \text{(trzecia pochodna)}$$

gdzie Δt to krok czasowy.

`euler_output` — wyznaczenie kolejnych pochodnych sygnału wyjściowego za pomocą metody Eulera

$$y^{(4)}[k] = \frac{a_3 \cdot u^{(3)}[k] + a_2 \cdot u^{(2)}[k] + a_1 \cdot u^{(1)}[k] + a_0 \cdot u[k] - b_3 \cdot y^{(3)}[k-1] - b_2 \cdot y^{(2)}[k-1] - b_1 \cdot y^{(1)}[k-1] - b_0 \cdot y[k-1]}{b_4}$$
$$y^{(3)}[k] = y^{(3)}[k-1] + \Delta t \cdot y^{(4)}[k]$$
$$y^{(2)}[k] = y^{(2)}[k-1] + \Delta t \cdot y^{(3)}[k]$$
$$y^{(1)}[k] = y^{(1)}[k-1] + \Delta t \cdot y^{(2)}[k]$$
$$y[k] = y[k-1] + \Delta t \cdot y^{(1)}[k]$$

Analogicznie wyznaczamy najwyższą pochodną oraz pozostałe pochodne wyjścia dla układów niższych rzędów.

e. **Rysowanie wykresów sygnału wejściowego i wyjściowego.**

Do rysowania wykresów wykorzystano bibliotekę matplotlib. Rysowanie wykresów realizują funkcje `input_plot` i `output_plot` w klasie `input_function` i `output_compute`.

f. **Wyznaczenie charakterystyk Bodego.**

Za narysowanie charakterystyki amplitudowej i fazowej oraz za określenie stabilności odpowiedzialna jest klasa `bode_plot`. Inicjalizacja klasy pobiera potrzebne dane oraz przygotowuje zakres kreślonego wykresu. W funkcji `plotting_bode` obliczne są charakterystyki fazowe i amplitudowe układu. Dodatkowo, określana jest stabilność na podstawie zapasów fazy i wzmocnienia.

g. **Określenie stabilności układu.**

Stabilność układu określana jest w funkcji `plotting_bode`. Układy stabilne charakteryzują się brakiem biegunów dodatnich oraz dodatnimi zapasami fazy i wzmocnienia. Zapas fazy oblicza się, kiedy wykres wzmocnienia przechodzi do ujemnych wartości. Natomiast, zapas wzmocnienia określany jest, gdy wykres przecina wartość $-180°$. Jeżeli taki punkt nie występuje, zapas wzmocnienia przyrównywany jest do ∞. Dodatkowo dodano funkcję `plot_zeros_poles`, która odpowada za rysowanie zer i biegunów układu na płaszyźnie zespolonej.

4 Podsumowanie i wnioski

Założeniem projektu było zaimplementowanie symulatora układu danego transmitancją. Funkcje realizujące różniczkowanie zostały zaimplementowane przez autorów projektu (różniczkowanie metodą Eulera). Generowanie sygnałów wejściowych zaimplementowano bez wykorzystania gotowych funkcji analitycznych z wyjątkiem np.sin wykorzystanej dla sinudoidy i sygnału prostokątnego. Program umożliwia zadawanie parametrów symulacyjnych i zmianę parametrów w stosunku do symulacji pokazanej przed chwilą z poziomu interfejsu użytkownika.

1. Metoda Eulera jest metodą dość prostą w implementacji i bardzo elastyczną w modyfikowaniu dla dowolnego rzędu układu. Działanie jest prawidłowe pod warunkiem dobrania odpowiedniego kroku czasowego — w naszej symulacji dobrany został krok 0.01s, który pozwala na dość szybkie uzyskanie odpowiedzi programu, a zarazem dość optymalną dokładość wykreślania sygnału wyjściowego. Im wyższy rząd układu, tym mniejsza dokładność wyznaczania odpowiedzi w porównaniu z metodą różniczkowania Runnego-Kutty, która nie jest częścią głównego projektu, ale została zaimplementowana w pliku testowym dla porównania odpowiedzi.

2. Łatwość modyfikacji parametrów transmitancji i sygnałów pobudzających umożliwia szybkie testowanie wielu wariantów układu. W tym układów o różnych rzędach — od zerowego do czwartego.

3. Charakterystyki Bodego są wygodnym narzędziem, za pomocą którego można wyznaczać stabilność układu poprzez wyznaczenie zapasu amplitudy i fazy bezpośrednio z punktów charakterystycznych na wykresach — punktu przejścia wykresu amplitudy przez wartość 0dB oraz punktu przecięcia wykresu fazy z $-180°$.

4. Znajomość transmitancji układu pozwala na określenie jego stabilności i przewidywanie jego reakcji na różne pobudzenia. Postać transmitancyjna jest tak powszechnie stosowana w automatyce i modelowaniu układów ze względu na swoją funkcjonalność i uniwersalność.

Projekt spełnił wszystkie założone wymagania. Symulator jest dobrym narzędziem do testowania odpowiedzi różnych układów do 4 rzędu i można dość łatwo rozbudowywać go dodając inne pobudzenia, czy zwiększając rząd układu. Projekt stanowi praktyczne podsumowanie wiedzy z zakresu modelowania układów dynamicznych, wyznaczania ich odpowiedzi, określania stabilności za pomocą charakterystyk Bodego oraz stosowania metod numerycznych w rozwiązywaniu problemów w automatyce.