Sprawozdanie

Projekt – Metody Modelowania Matematycznego

Projekt nr. 8

Natalia Zapolnik 197627
Bartłomiej Tomczyk 198335
ACiR3

Cel projektu

Celem projektu było stworzenie aplikacji symulującej zachowanie układu $G(s) = \frac{a_1 s + a_0}{b_2 s^2 + b_1 s + b_0}$ sterowanego regulatorem PI w konfiguracji z ujemnym sprzężeniem zwrotnym, umożliwiającej:

- Wybór sygnału wejściowego (sinusoidalny, trójkątny, prostokątny, impulsowy) i jego parametrów,
- Zmianę parametrów układu i regulatora PI,
- Obliczenie odpowiedzi układu z regulacją oraz uchybu regulacji,
- Wizualizację wyników w czasie rzeczywistym przez graficzny interfejs użytkownika,
- Eksport sygnału do pliku CSV.

Organizacja projektu

Projekt został zaplanowany i zorganizowany w sposób czytelny, modułowy i zgodny z dobrymi praktykami inżynierii oprogramowania. Struktura katalogów oraz podział funkcjonalności umożliwiają łatwe zarządzanie kodem, testowanie, a także ewentualne rozwijanie w przyszłości.

Projekt został podzielony na dwie części:

- GUI – napisane w języku Python (moduł frontendowy)
- Biblioteka symulacyjna – napisana w języku C++ z wykorzystaniem pybind11 wykorzystywana przez GUI (moduł backendowy)

Struktura projektu:

```
MMMSimulator/
├── assets/
│   └── fonts/
├── data/
├── src/
│   ├── bindings/
│   │   ├── pybind11/
│   │   ├── CMakeLists.txt
│   │   └── cpp_backend.cpp
│   ├── config.py
│   └── gui.py
└── venv-MMMSimulator/
```

Ogólne cechy organizacji:

- Moduły są oddzielone – Python zajmuje się warstwą interakcji z użytkownikiem, a C++ odpowiada za obliczenia,
- Struktura katalogów dba o czytelność i pozwala na łatwe rozszerzenie projektu o kolejne moduły,
- Użycie wirtualnego środowiska niweluje potrzebę instalowania potrzebnych bibliotek globalnie,
- Całościowa struktura ułatwia pracę zespołową.

Biblioteka cpp_backend

Postawiliśmy na pisanie warstwy obliczeniowej w C++ ze względu na wydajność obliczeń oraz czytelność kodu. Biblioteka pybind11 pozwala nam na zaimportowanie jej jako biblioteki Python w celu łatwego użycia jej w kodzie odpowiedzialnym za GUI.
Biblioteka zawiera klasy Parameters (ułatwiającą wgrywanie parametrów do symulacji), Simulation (odpowiadającą za symulację układu) oraz funkcji export_to_csv (mówi samo za siebie).

Metody numeryczne

W projekcie do obliczeń wyników symulacji zastosowano model ARX (Auto-Regressive with eXogenous input) jako metodę numeryczną odwzorowującą zachowanie transmitancji układu w dziedzinie czasu. Metoda ta pozwala w prosty sposób przeprowadzać symulacje zachowania układu dynamicznego w dziedzinie czasu, bez konieczności stosowania transformacji Laplace'a. Model ARX opisuje układ dynamiczny za pomocą równania różnicowego, w którym aktualna wartość

sygnału wyjściowego y[k] zależy od przeszłych wartości tego sygnału oraz sygnału wejściowego. Model ten stanowi dyskretny odpowiednik klasycznej transmitancji w dziedzinie Laplace'a.

Zastosowane równania:

$$y[k] = -a_1 * y[k-1] - a_0 * y[k-2] + b_2 * u[k] + b_1 * u[k-1] + b_0 * u[k-2]$$

$$e[k] = r[k] - y[k-1]$$

$$u[k] = u[k-1] + K_p * (e[k] - e[k-1]) + K_i * dt * e[k]$$

Wszystkie metody do obliczeń zostały zaimplementowane ręcznie.

Budowanie i instalacja

Do zbudowania biblioteki używamy systemu CMake.

Budowanie:

a) cd src/bindings
b) mkdir build
c) cd build
d) cmake ..
e) cmake -build .

Zalecane jest użycie wirtualnego środowiska:

a) python -m venv venv-MMMSimulator
b) .\venv-MMMSimulator\Scripts\activate
c) pip install dearpygui

Wnioski dotyczące wpływu parametrów na zachowanie układu

Testowanie układu w różnych konfiguracjach parametrów transmitancji zaobserwowano wyraźny wpływ poszczególnych współczynników na odpowiedź układu. Parametry transmitancji (a_1, a_0, b_2, b_1, b_0) wpływają na szybkość odpowiedzi, zwiększenie wartości ujemnych powoduje szybszą odpowiedź, ale może też podnieść ryzyko niestabilności. Wpływają one na dynamikę układu, kształtując zależność między sygnałem sterującym a wyjściem. Wzmocnienia regulatora (K_p, K_i) wpływają na przyspieszenie reakcji układu oraz na wielkość uchybu. Wzrost wartości K_p przyspiesza reakcję na uchyb, ale może powodować oscylacje. Wyższe K_i zwiększa składową całkującą i redukuje uchyb ustalony, ale przy zbyt dużych wartościach może wywoływać przeregulowania. Zwiększenie amplitudy sygnału wejściowego silniej pobudza układ, co może wydobywać nieliniowości lub opóźnienia. Wzrost częstotliwości skraca okres sygnału, co wymaga szybszej reakcji regulatora. Mniejsze dt zwiększa dokładność symulacji,

ale wydłuża czas obliczeń. Zwiększenie czasu impulsu, dla sygnału impulsowego, przedłuża okres, w którym układ jest pobudzany maksymalnym sygnałem, co może uwidocznić przesterowania lub przeregulowania.