

# Projekt z przedmiotu Programowanie Obiektowe i Grafika Komputerowa

System amortyzacji samochodowej wraz ze stacją diagnostyczną wymuszającą drgania

Natalia SAMPLAWSKA 197573

Martyna PENKOWSKA 197926

16 czerwca 2025

Okres trwania projektu: Semestr letni roku akademickiego 2025

Prowadzący projekt: dr inż. MARCIN PAZIO

## 1 Cel

Stworzenie symulacji systemu amortyzacji samochodowej w języku Python pozwalającej na zmiany parametrów takich jak: masa, współczynniki sprężyny i tłumika oraz ruchu platformy wprawiającej układ w drgania. Dodatkowa ilustracja stabilności układu.

Symulowany układ jest układem 2 rzędu opisanym transmitancją:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{ms^2 + bs + k}$$

## 2 Zaimplementowane funkcje programu

1. Wybór parametrów obiektu
2. Wybór sygnału wejściowego
3. Generowanie charakterystyk Bodego
4. Przedstawienie graficzne sygnału wejściowego i wyjściowego
5. Wizualizacja 3D działania układu amortyzacji

## 3 Opis funkcji

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

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

`ParameterControl` — klasa z funkcjami odpowiedzialnymi za wizualizację okna wyboru parametrów układu i sygnału wejściowego oraz ich aktualizacje.

`update_parameters`, `update_simulation_data`, `update_visibility` — funkcje aktualizujące parametry na podstawie wyboru użytkownika. Dodatkowe zabezpieczenia przed niepoprawnymi wartościami, które mogłyby uniemożliwić realizację zadania (program wyświetla w oknie etykietę z informacją o prawidłowości wprowadzonych danych).

### b. Wyświetlanie okien programu.

Interfejs graficzny jest zrealizowany za pomocą biblioteki Pygame. Program składa się z 2 okien, które są aktywne jednocześnie poprzez `threading`. Po uruchomieniu kodu otwierają się okna programu. Naciśnięcie przycisku `simulate` aktywuje symulację w oknie symulacji. Możliwe jest również wyświetlenie charakterystyk Bodego, wykresów sygnałów wejściowych i wyjściowych po naciśnięciu odpowiadających im przycisków.

`ParameterControl` — klasa z wyglądem interfejsu ustawiania parametrów oraz logiką przycisków

### c. Pobudzenia wejściowe układu.

Układ może być pobudzany następującymi sygnałami:

- **Sinusoidalnym:**  $y(t) = A \sin(2\pi ft + \varphi)$
- **Prostokątnym:**  $y(t) = A \cdot \text{sgn}(\sin(2\pi ft + \varphi))$
- **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:**  $y(t) = \begin{cases} A, & \text{dla } 0 < t < \text{pulsewidth} \\ 0, & \text{w przeciwnym razie} \end{cases}$
- **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.

`InputOutputFunction` — klasa odpowiedzialna za obliczanie sygnału wyjściowego

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

$$y^{(2)}[k] = \frac{-b \cdot y^{(1)}[k-1] - k \cdot y[k-1] + u[k]}{m}$$

$$y^{(1)}[k] = y^{(1)}[k-1] + \Delta t \cdot y^{(2)}[k]$$

$$y[k] = y[k-1] + \Delta t \cdot y^{(1)}[k]$$

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

Do rysowania wykresów wykorzystano bibliotekę matplotlib. Rysowanie wykresów realizuje funkcja `input_output_plot` w klasie `InputOutputFunction`.

### f. Wyznaczenie charakterystyk Bodego.

Za narysowanie charakterystyki amplitudowej i fazowej oraz za określenie stabilności odpowiedzialna jest klasa `BodePlot`. Inicjalizacja klasy pobiera potrzebne dane oraz przygotowuje zakres kreślonego wykresu. W funkcji `plotting_bode` obliczne są charakterystyki fazowe i amplitudowe układu. Jest również na podstawie wykresów określany zapas amplitudy i fazy, który decyduje o stabilności.

## 4 Podsumowanie i wnioski

Cel projektu został zrealizowany. Program umożliwia obliczenie oraz wizualizację zachowania układu. Użytkownik ma możliwość sterowania parametrami symulacji, wyboru pobudzenia oraz obserwacji wyników zarówno numerycznych, jak i graficznych na symulacji.

### Elementy programowania obiektowego

W projekcie programowanie obiektowe zostało wykorzystane do podziału programu na uporządkowane klasy. Każdy istotny element układu (sprężyna, tłumik itd.) został odwzorowany za pomocą osobnych klas. Przykładowo:

- `Spring` — klasa reprezentująca sprężynę, przechowuje jej model graficzny 3D,
- `Attenuator` — reprezentuje tłumik i jego model graficzny 3D,
- `Wheel` - klasa odpowiedzialna za graficzne przedstawienie koła będącego częścią układu,
- `InputOutputFunction` — realizuje obliczenia związane z dynamiką układu,
- `ParameterControl` — odpowiada za interfejs użytkownika i aktualizację parametrów,
- `BodePlot` — oblicza i rysuje charakterystyki Bodego,

Dzięki takiemu podziałowi kod stał się przejrzysty. Poszczególne klasy odpowiadają poszczególnym zadaniom, co pozwala na łatwą zmianę programu bez konieczności ingerencji w cały system, ułatwia odnalezienie porządanego fragmentu kodu. Korzystano również z dziedziczenia i enkapsulacji — dane przechowywane są wewnątrz obiektów, a użytkownik korzysta z interfejsu.

## **Grafika komputerowa z OpenGL**

Biblioteka OpenGL umożliwia trójwymiarową wizualizację układu dynamicznego w czasie rzeczywistym. Dzięki połączeniu Pygame (zarządzającego oknem oraz zdarzeniami) z PyOpenGL możliwe było narysowanie obiektów takich jak sprężyna, masa oraz koło. Dynamiczne zmiany w układzie odwzorowują zmiany wartości obliczanych numerycznie, co tworzy spójną reprezentację fizyki i grafiki. Wizualizacja ta jest zsynchronizowana z częścią obliczeniową, co pozwala na intuicyjne zrozumienie wpływu parametrów na zachowanie układu.