# Sterowanie adaptacyjne i estymacja

Temat projektu:

Oprogramowanie, które z bieżących pomiarów $u(t)$ i $y(t)$ zdejmowanych co próbkę na przedziale $[0, T]$ będzie odtwarzało impulsową funkcję przejścia $g(t)$ i odpowiedź skokową $h(t)$ na bieżąco i na całym przedziale.

**Maciej Kazana**

**Anna Ryszka**

AGH, Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej, Kraków 2016

# Spis treści

# 1. Wstęp

Celem projektu wykonanego w ramach przedmiotu Sterowanie adaptacyjne i estymacja jest stworzenie oprogramowania w języku C lub C++, które będzie z podanych danych sterujących i wyjściowych dowolnego obiektu sterowania wyznaczał charakterystykę impulsową i skokową tego obiektu. Danymi wejściowymi tworzonego programu będą wektory $u$ i $y$ przedstawione na rysunku 1.1. Są one pobierane co pewien stały ustalony okres próbkowania. Wyjściem programu będą dwa wektory $g$ i $h$, które będą odpowiadały kolejno charakterystyce impulsowej i skokowej badanego obiektu. Wektory $u$ i $y$ są jedynymi dostępnymi danymi wejściowymi, transmitancja obiektu nie jest znana.


**Rysunek 1.1** Uzyskanie danych wejściowych do tworzonego oprogramowania

Oprogramowanie podzielono na 3 części, z czego druga odpowiada za wykonanie głównego celu projektu - jest programem obliczającym żądane charakterystyki z podanych wektorów sterowania i wyjścia. Pozostałe części są pomocnicze i służą do generowania danych wejściowych oraz wyświetlania i porównania danych wyjściowych. Komponenty oprogramowania:

1. Interfejs graficzny napisany w środowisku MATLAB służący do generowania danych wejściowych na podstawie podanej transmitancji (opisany w rozdziale 3.1)
2. Program napisany w języku C++, który oblicza zadane charakterystyki na podstawie wektorów wejściowych (opisany w rozdziale 3.2)
3. Interfejs graficzny napisany w środowisku MATLAB służący do wyświetlania uzyskanych wyników oraz porównania ich do odpowiedzi referencyjnych obliczonych przez środowisko MATLAB na podstawie transmitancji (opisany w rozdziale 3.3)

Interfejsy graficzne w środowisku MATLAB zostały napisane w wersji 2015b. Oprogramowanie zostało również z powodzeniem przetestowane w wersji 2013b. Próbowano także uruchomić interfejs na wersji 2011a, jednak w tej wersji brakuje już funkcji zaimplementowanych w nowszych wersjach środowiska, a są używane w interfejsie.

# 2. Wprowadzenie teoretyczne

Podstawowym stwierdzeniem, które umożliwia napisanie oprogramowania o założeniach opisanych w temacie i we wstępie jest to, że w układach dyskretnych (oraz w układach ciągłych, jednak projekt dotyczy układów dyskretnych) splot dowolnego ciągu sterującego i impulsowej funkcji przejścia służy do obliczenia odpowiedzi systemu. Można pokazać, że dysponując pomiarami dyskretnymi wejścia i wyjścia można odtworzyć impulsową charakterystykę przejścia, jednak jest to tylko możliwe przy zerowych warunkach początkowych ($x_0 = 0$). Takie wyznaczenie charakterystyki impulsowej jest nazywane identyfikacją nieparametryczną obiektu lub procedurą dyskretnej dekonwolucji. Pełny opis metody znajduje się w publikacji [1], natomiast w tym rozdziale opisano jedynie najważniejsze wzory, które zostały użyte do stworzenia oprogramowania.

Wzór na odpowiedź $y_k$ dla $k \geq 0$ ma postać (2.1).

$$y_k = \sum_{i=0}^{k} g_{k-i} u_i = \sum_{i=0}^{k} g_i u_{k-i} \tag{2.1}$$

Odpowiedź systemu na sterowanie impulsem $u_0 = \delta(0) = 1$ jest równa impulsowej funkcji przejścia.

Przy zerowych warunkach początkowych przebieg dowolnie długiego fragmentu charakterystyki impulsowej można wyznaczyć ze wzorów (2.2).

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} u_0 & 0 & \cdots & 0 \\ u_1 & u_0 & 0 & 0 \\ \vdots & \vdots & \ddots & 0 \\ u_{k-1} & u_{k-2} & \cdots & u_0 \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_k \end{bmatrix}$$

$$\begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_k \end{bmatrix} = \begin{bmatrix} u_0 & 0 & \cdots & 0 \\ u_1 & u_0 & 0 & 0 \\ \vdots & \vdots & \ddots & 0 \\ u_{k-1} & u_{k-2} & \cdots & u_0 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} \tag{2.2}$$

Równanie (2.2) może być dość problematyczne w obliczeniu dla dużej liczby pomiarów ze względu na konieczność odwrócenia macierzy, w której zapisane są sterowania. Będzie to generowało duży nakład obliczeń, który znacząco wydłuży działanie programu, bądź dla bardzo dużych macierzy całkowicie go uniemożliwi. W takim przypadku można zastosować wzór rekurencyjny (2.3).

$$g_1 = \frac{y_1}{u_0}$$
$$g_k = \frac{1}{u_0}\left[y_k - \sum_{i=1}^{k-1} g_{k-i}u_i\right] \quad (2.3)$$
$$dla\ k = 2,3,4\ldots$$

Szereg $g_0, g_1, g_2$ … jest nazywany szeregiem parametrów Markowa.

Odpowiedź na skok jednostkowy będzie wyrażała się wzorem (2.4).

$$h_k = \sum_{i=0}^{k} g_i \quad (2.4)$$

# 3. Opis oprogramowania

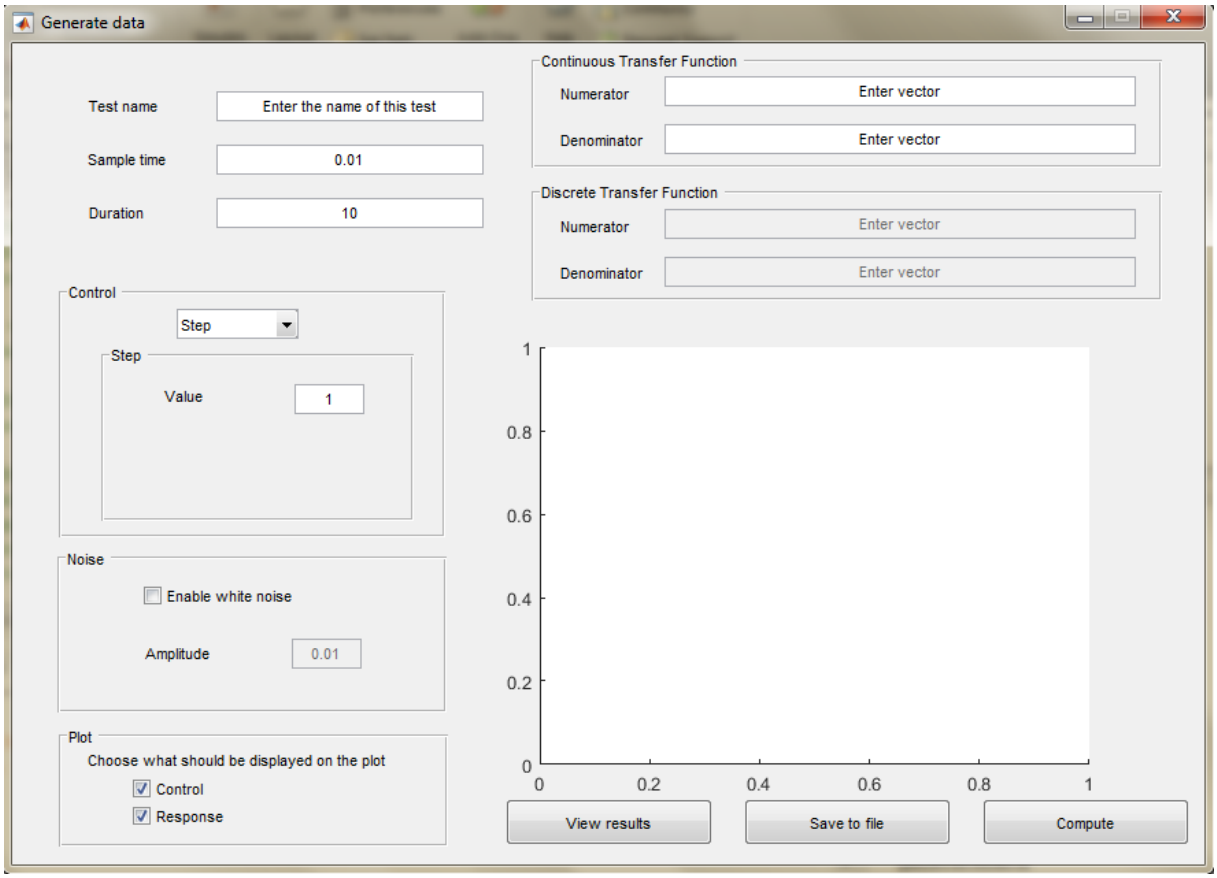
Stworzone oprogramowanie składa się z trzech części, które zostaną opisane w tym rozdziale.

## *3.1 Interfejs graficzny do generowania danych wejściowych*

Pierwsza część oprogramowania jest napisana w środowisku MATLAB. Jest to interfejs graficzny, który służy do generowania danych wejściowych do programu napisanego w języku C++. Zgodnie z założeniem tematu (rozdział 1), danymi wejściowymi mają być wektory sterowania i wyjścia z obiektu bez znajomości dynamiki obiektu. Dane wygenerowane w opisywanym GUI są zapisywane w pliku tekstowym, w którym oprócz sterowania i wyjścia znajduje się także odpowiadający wektor czasu. Wektory w pliku są zapisane w trzech kolumnach, w których kolejno znajdują się wektory: czasu, sterowania i wyjścia. Poniżej pokazano fragment przykładowego pliku, kolejne wartości są oddzielone spacją.

```
0.000000 1.000000 0.000000
0.010000 0.999950 0.000000
0.020000 0.999800 0.000001
0.030000 0.999550 0.000004
0.040000 0.999200 0.000010
…
```

Interfejs graficzny należy uruchomić wpisując komendę *generateData* w linii komend w środowisku MATLAB. Folder z oprogramowaniem musi być folderem bieżącym lub musi być dodany do ścieżki. Po wywołaniu skryptu ukaże się okno pokazane na rysunku 3.1.

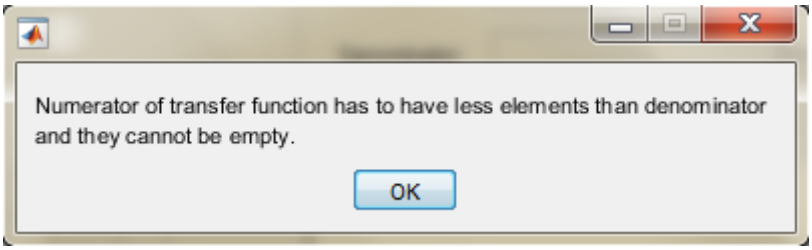



**Rysunek 3.1** Okno startowe interfejsu graficznego do generowania danych wejściowych

W lewym górnym rogu znajdują się trzy parametry, które muszą zostać ustawione. Są to:

- *Test name* – nazwa własna przeprowadzanego eksperymentu. Nazwa ta będzie podstawą nazw wszystkich plików tworzonych dla danego testu, rozszerzona o odpowiednie ciągi znaków w zależności od zawartości pliku. Plik zostanie zapisany w folderze, w którym znajdują się pliki źródłowe programów środowiska MATLAB; w przypadku gdy w folderze istnieje już eksperyment o podanej nazwie zostanie on nadpisany. Jeżeli nazwa nie zostanie wpisana, to zostanie ona ustawiona na pusty ciąg znaków, co nie wygeneruje żadnych błędów w działaniu programu.
- *Sample time* – czas próbkowania.
- *Duration*– czas trwania symulacji eksperymentu.

Kolejnym istotnym parametrem, który należy ustawić przed uruchomieniem symulacji jest transmitancja obiektu, którą należy podać w części *Continuous Transfer Function* w postaci jej licznika (*Numerator*) i mianownika (*Denominator*). Wektory można podać w formacie języka MATLAB, czyli jako ciąg współczynników stojących przy kolejnych składnikach wielomianu oddzielonych spacją. Wektory można wprowadzać w nawiasach kwadratowych lub bez. Należy pamiętać o tym, że transmitancja musi być podana w poprawnej formie (stopień licznika musi być mniejszy od stopnia mianownika). W przypadku gdy zasada ta nie będzie zachowana, albo licznik lub mianownik nie został wprowadzony, przy próbie wygenerowania danych (przyciski *View results* i *Save to file* opisane w dalszej części podrozdziału) zostanie wygenerowany błąd. Okno z komunikatem zostało pokazane na rysunku 3.2.

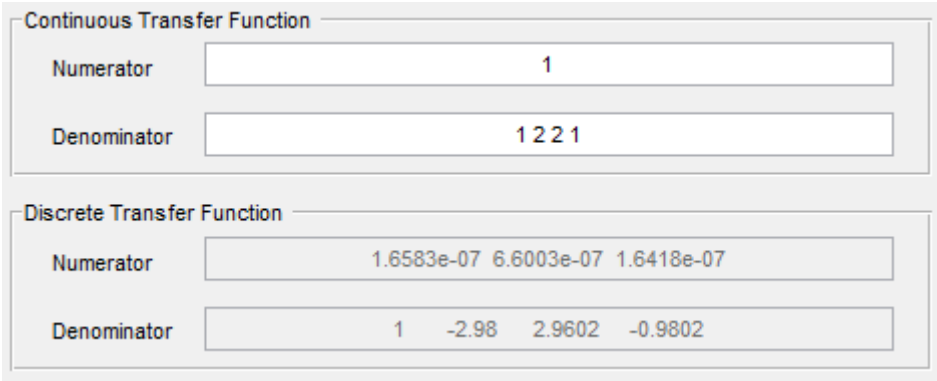



**Rysunek 3.2** Błąd wygenerowany w przypadku podania złej formy transmitancji

Poniżej bloku z transmitancją ciągłą znajduje się blok z transmitancją dyskretną. Model symulacyjny jest modelem dyskretnym, dlatego transmitancja w nim podawana jest transmitancją dyskretną. Jednak w celu ułatwienia korzystania z interfejsu wprowadzana jest transmitancja ciągła. Transmitancja dyskretna nie może być edytowana; jest ona wyświetlana po wprowadzeniu bądź zmodyfikowaniu transmitancji ciągłej. Przykładowo przy wprowadzeniu transmitancji:

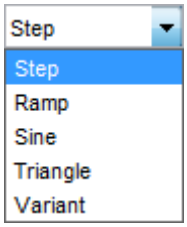
$$G(s) = \frac{1}{s^3 + 2s^2 + 2s + 1}$$

otrzyma się przedstawiony na rysunku 3.3 fragment GUI.




**Rysunek 3.3** Przykładowa transmitancja wprowadzona do interfejsu graficznego

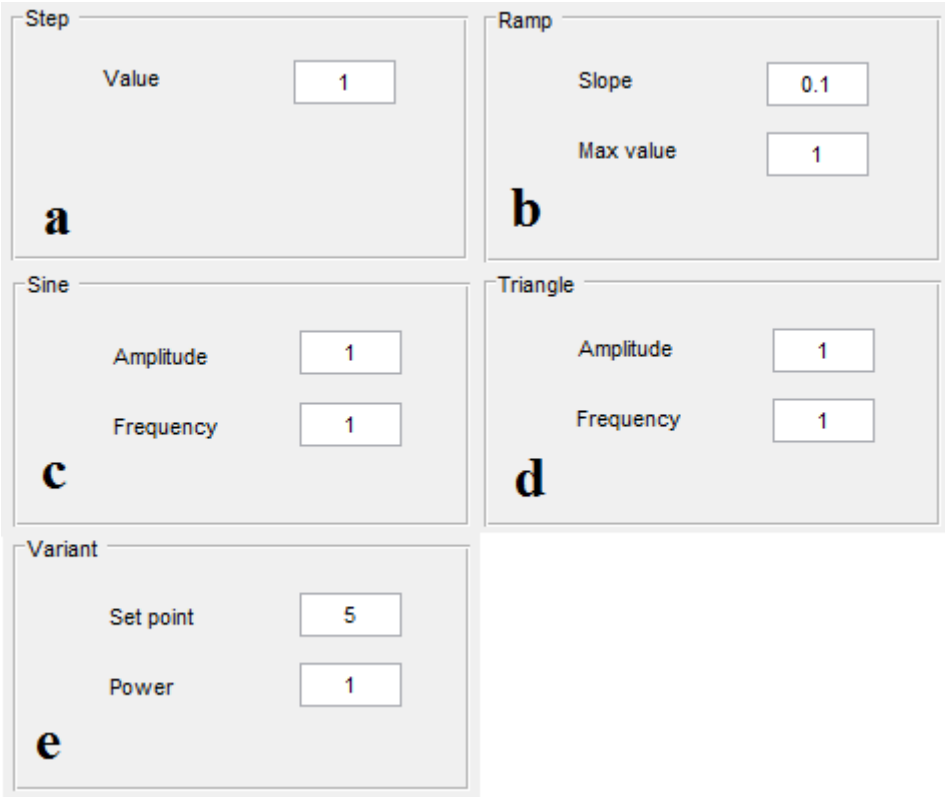
Następną częścią GUI jest wybranie rodzaju sterowania podanego na obiekt o zadanej transmitancji. Opcje te dostępne są w bloku *Control*. Składa się on z rozwijanego menu przedstawionego na rysunku 3.4.




**Rysunek 3.4** Rozwijane menu z panelu *Control*

Po wybraniu sterowania wyświetla się odpowiedni panel z parametrami dotyczącymi konkretnej struktury sterowania (rysunek 3.5). Możliwe są następujące rodzaje sterowania:

- *Step* – skok jednostkowy. Parametrem, który należy ustawić dla tego rodzaju sterowania jest wartość (*Value*) tego skoku (rysunek 3.5a), domyślną wartością jest 1. Wszystkie wartości domyślne dla kolejnych rodzajów sterowań pokazane są na rysunku 3.5.
- *Ramp* – sterowanie narastające liniowo z nasyceniem. Dla tego sterowania są ustawiane dwa parametry: nachylenie rampy - *Slope* (wzrost wartości sterowania w ciągu sekundy) oraz wartość nasycenia – *Max value* (rysunek 3.5b).
- *Sine* – sterowanie sinusoidalne, sinus przesunięty w fazie o $\pi$. Ustawiane parametry to amplituda (*Amplitude*) i częstotliwość (*Frequency*) dla tego sinusa (rysunek 3.5c).
- *Triangle* – funkcja trójkątna. Parametrami tego sterowania są amplituda (*Amplitude*) i częstotliwość (*Frequency*) (rysunek 3.5d).
- *Variant* – funkcja krążąca wokół zadanej wartości w sposób nieregularny. Możliwymi do ustawienia parametrami są: wartość wokół której będzie krążyć wartość funkcji (*Set point*) oraz amplituda odchyleń (*Power*). Amplituda ta będzie maksymalną wartością, o którą może wartość funkcji różnić się od zadanej wartości. Parametry są pokazane na rysunku 3.5e.

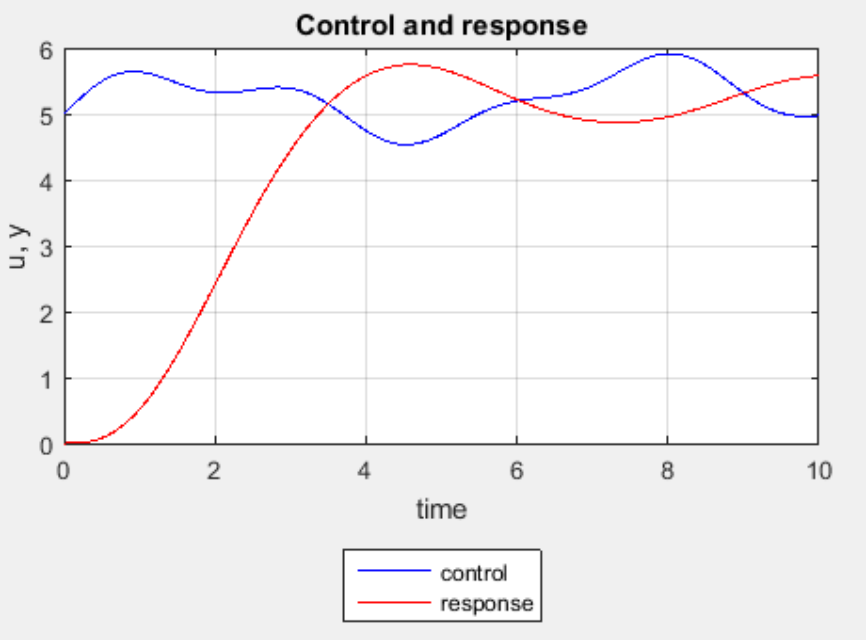



**Rysunek 3.5** Parametry różnych rodzajów sterowania (a – skok jednostkowy, b – rampa z nasyceniem, c – sinus, d – trójkąt, e – sterowanie krążące wokół zadanej wartości)

Interfejs graficzny umożliwia także możliwość dodania zakłóceń sterowania do modelu (panel *Noise*). Są to zakłócenia będące szumem białym. Aby dodać zakłócenia należy zaznaczyć pole *Enable white noise*. Dopóki pole to nie będzie zaznaczone pole *Amplitude* będzie nieaktywne. W tym miejscu ustawiana jest moc zakłóceń. Moc zakłóceń jest parametrem, który nie może zostać ustawiony zbyt

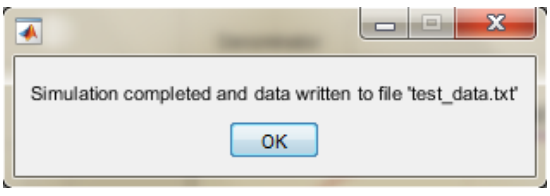
duży (w odniesieniu do wartości funkcji sterującej), ponieważ wtedy podczas późniejszego obliczania charakterystyk impulsowej i skokowej występują znaczne błędy numeryczne, które powodują generowanie nieprawidłowych danych dążących do nieskończoności.

Ostatnim blokiem z parametrami możliwymi do ustawienia w interfejsie jest panel *Plot*. Dotyczy on wykresu, który będzie się pojawiał w prawym dolnym rogu okna po wygenerowaniu danych. Pola zaznaczone w tym bloku w żaden sposób nie wpływają na wygenerowane dane, dotyczy on tylko danych wyświetlanych na wykresie. Przykładowy wykres pokazany jest na rysunku 3.6. Pokazane są na nim dwie funkcje: sterowanie (*Control*) oraz odpowiedź (*Response*). Panel *Plot* umożliwia zrezygnowanie z wyświetlenia którejś z funkcji, co może być przydatne w przypadku dużych dysproporcji w wartościach funkcji. Wyświetlone zostaną tylko wykresy, które będą zaznaczone w bloku *Plot*.



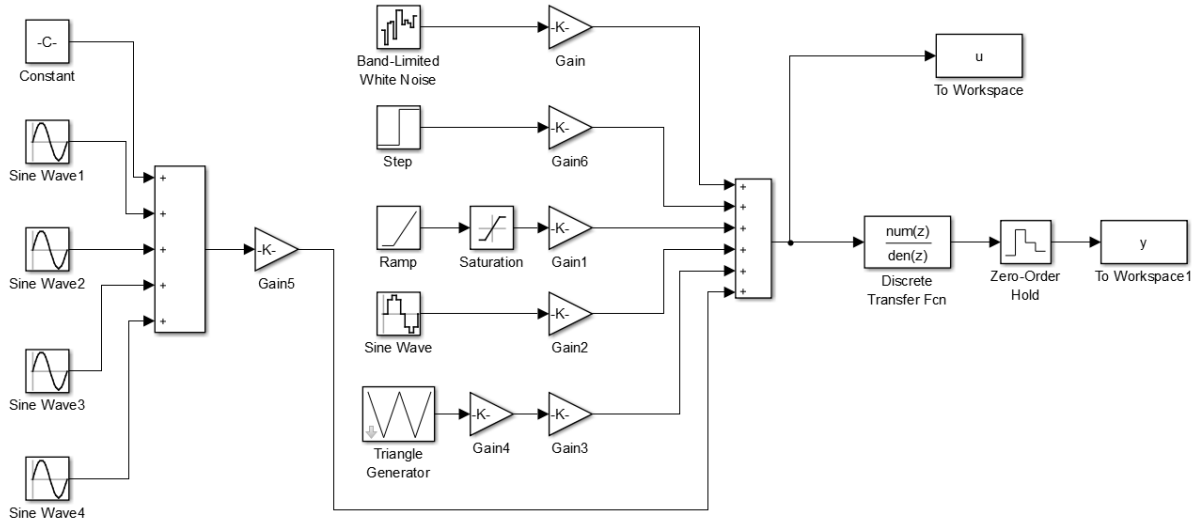

**Rysunek 3.6** Przykładowy wykres wyświetlany w interfejsie graficznym do generowania danych

Po ustawieniu wszystkich parametrów można wygenerować dane wejściowe do właściwego programu z algorytmem. Dodatkową opcją jest możliwość wyświetlenia wykresów bez zapisywania danych do pliku, co wykonuje się za pomocą przycisku *View Results*. Przycisk *Save to file* generuje dane, wyświetla je na wykresie i zapisuje do pliku o nazwie *test_data.txt*, gdzie *test* jest nazwą podaną w polu *Test name*. Format pliku jest taki jak opisano na początku tego rozdziału. Po skończonej symulacji zostanie wyświetlony komunikat jak pokazano na rysunku 3.7 (tylko w przypadku zapisywania danych do pliku).




**Rysunek 3.7** Komunikat ukazujący się po generacji danych (za pomocą przycisku *Save to file*)

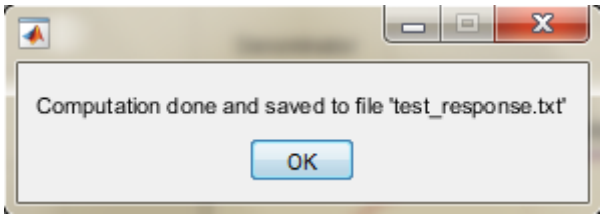
Dane sterujące i wyjściowe dla obiektu wybranego za pomocą interfejsu są generowane przy użyciu modelu w środowisku Simulink (*generate_model.slx*). W modelu tym znajdują się wszystkie możliwe do wybrania sterowania podane razem z szumem na sumator. Sygnały te są mnożone przez współczynnik 0 lub 1 w zależności od tego, które sterowanie jest wybrane. W danej symulacji tylko przy jednym sterowaniu może być współczynnik mnożący równy 1. Wyjątkiem są zakłócenia, których współczynnik nie zależy od wybranego sterowania. Następnie sygnał ten wchodzi do bloczka transmitancji dyskretnej. Sygnał zapisywany jest na wejściu i wyjściu z transmitancji (uzupełniony bloczkiem *Zero-Order Hold*). Model środowiska Simulink jest przedstawiony na rysunku 3.8.




**Rysunek 3.8** Model symulacyjny zapisany w pliku *generate_model.slx*

W przypadku uruchomienia symulacji z zapisaniem danych do pliku (*Save to file*) tworzony jest dodatkowo plik *test_tf.mat* z zapisaną transmitancją obiektu. Transmitancja ta nie jest znana właściwemu programowi z algorytmem, używana jest pomocniczo w trzeciej części oprogramowania, czyli drugim interfejsie graficznym napisanym w środowisku MATLAB (rozdział 3.3). Wykorzystana jest ona do sprawdzenia poprawności otrzymanych wyników.

Ostatnim przyciskiem w interfejsie jest funkcja *Compute*. Służy ona do uruchomienia programu napisanego w języku C++, który zawiera właściwy algorytm do wyznaczania pożądanych charakterystyk. Program ten jest uruchamiany z pliku wykonywalnego *responses.exe* za pomocą komendy systemowej. Po wykonaniu programu zostaje wyświetlony komunikat pokazany na rysunku 3.9. Jest w nim napisana nazwa pliku, do którego zostały zapisane wyniki obliczeń w postaci wektorów czasu oraz kolejno charakterystyki impulsowej i skokowej. Struktura pliku opisana została w rozdziale 3.2. Plik ten następnie będzie można wczytać w kolejnym interfejsie graficznym *compareData.m*, który jest opisany w rozdziale 3.3.




**Rysunek 3.9** Komunikat wyświetlany po zakończeniu działania programu generującego charakterystyki impulsową i skokową

## *3.2 Aplikacja obliczająca charakterystyki impulsową i skokową*

Druga, główna część programu napisana w języku C++ odpowiada za obliczenie charakterystyk impulsowej i skokowej z wykorzystaniem metody opisanej w rozdziale 2. Kod źródłowy aplikacji jest podzielony na pięć plików:

- *main.cpp* – główny plik programu
- *get_impulse_response.h* i *get_impulse_response.cpp* – obliczanie charakterystyki impulsowej
- *get_step_response.h* i *get_step_response.cpp* – obliczanie charakterystyki skokowej

W aplikacji stworzonej na potrzeby projektu wykorzystywany jest plik wykonywalny o nazwie *responses.exe* wygenerowany podczas kompilacji ww. plików źródłowych. Pliki te są również dostępne dla użytkownika i może on je modyfikować i kompilować własny plik wykonywalny (na przykład zmieniając wygląd pliku wejściowego lub wyjściowego, lub sposób podawania ścieżki tego pliku).

Do napisania funkcji wykorzystano standardową bibliotekę STL (Standard Template Library)[2], a w szczególności klasę *list* będącą kontenerem danych znacznie ułatwiającym operacje na danych w postaci ciągów próbek [3]. Wykorzystanie tej klasy pozwala na swobodne podawanie dużej ilości danych jako parametry wejściowe do funkcji, jak również znacznie ułatwia wczytywanie danych z pliku tekstowego (głównie przez już zaimplementowane mechanizmy dynamicznego przydzielania pamięci dla wprowadzanych danych).

**Pliki nagłówkowe *get_impulse_response.h* i *get_step_response.h*:**

Pliki nagłówkowe zawierają deklaracje funkcji *get_impulse_response()* i get_*step_response()*. Należy je dołączyć w pliku main.cpp:

```
#include "get_impulse_response.h"
#include "get_step_response.h"
```

**Plik *get_impulse_response.cpp:***

Zawiera ciało funkcji *get_impulse_response()*. Funkcja służy do obliczania charakterystyki impulsowej z wykorzystaniem wzoru rekurencyjnego (2.3).

Argumentami wejściowymi funkcji są obiekty klasy *list* zawierające ciągi próbek sterowań i odpowiadających im wyjść systemu.

Funkcja zwraca obiekt klasy *list* zawierający obliczoną charakterystykę impulsową.

**Plik *get_step_response.cpp:***

Zawiera ciało funkcji *get_step_response()*. Funkcja służy do obliczania charakterystyki skokowej z wykorzystaniem wzoru (2.4).

Argumentem wejściowym funkcji jest obiekt klasy *list* zawierający ciąg próbek charakterystyki impulsowej systemu. W przypadku niniejszej aplikacji, jako argument wejściowy jest podawany obiekt klasy *list* powstały po wcześniejszym wykonaniu funkcji *get_impulse_response()*.

Funkcja zwraca obiekt klasy *list* zawierający obliczoną charakterystykę skokową.

**Plik *main.cpp***

Zawiera główne ciało aplikacji. Można wyróżnić w nim cztery sekcje:

1. Wczytywanie danych z pliku tekstowego
2. Wstępne przetwarzanie danych (preprocessing)
3. Obliczenie charakterystyk impulsowej i skokowej
4. Zapis obliczonych charakterystyk do pliku tekstowego

W przypadku, gdy użytkownik chce sam skompilować kod źródłowy i otrzymać plik wykonywalny, konieczne jest ustawienie sposobu podawania ścieżek pliku wejściowego i wyjściowego. Zostały przygotowane dwa sposoby określania ścieżek dla plików tekstowych i przed kompilacją należy niechciany sposób ująć w komentarz.

Pierwszy ze sposobów pozwala na wczytywanie pliku tekstowego zawsze o tej samej nazwie i z tej samej lokalizacji na dysku. Również plik wyjściowy będzie się znajdował w stałej lokalizacji. Takie podejście jest polecane jedynie podczas testowania programu po kompilacji, kiedy użytkownik pracuje na jednym zestawie danych testowych. Wykorzystanie tego podejścia do innych celów jest bardzo uciążliwe, gdyż trzeba by dla każdego eksperymentu umieszczać dane wejściowe zawsze w tym samym pliku.

Drugi ze sposobów pozwala na podawanie lokalizacji pliku wejściowego jako argumentu przy uruchamianiu pliku wykonywalnego. Jest to domyślnie ustawiony sposób wczytywania pliku wejściowego ze względu na jego uniwersalność. Przykładowo wywołanie aplikacji dla pliku wejściowego o ścieżce dostępu *C:\test1\test_data.txt* wygląda następująco w konsoli Windows:

```
> responses.exe C:\test1\test_data.txt
```

lub w linii komend programu Matlab:

```
>> pathName = 'C:\test1\test_data.txt'
>> system(['responses.exe ' pathName]);
```

Plik wejściowy do programu powinien mieć postać pokazaną w podrozdziale 3.1, to znaczy wektorów czasu, sterowania i wyjścia w kolumnach rozdzielonych spacjami. Jeżeli plik nie istnieje lub nastąpią

problemy z odczytem danych, aplikacja zostanie zamknięta i zostanie zwrócony odpowiedni komunikat o błędzie.

Dane z pliku wejściowego są wczytywane do trzech obiektów klasy *list*. Po zakończeniu wczytywania danych, w konsoli wyświetlany jest komunikat o ilości próbek we wczytanych wektorach. Następnie dane są wstępnie przetwarzane, aby spełniały założenia metody obliczeniowej. By uchronić program przed dzieleniem przez zero, sprawdzana jest pierwsza wartość wektora sterowania ($u_0$ we wzorze 2.3 znajduje się w mianowniku, więc nie może być równe zeru). W przypadku zerowych wartości sterowania na początku wektora usuwane są wszystkie kolejne próbki począwszy od pierwszej aż do osiągnięcia niezerowej wartości w pierwszej próbce wektora. Jeżeli sterowanie było stałe i równe zero dla całego wektora, oznacza to brak danych do obliczenia odpowiedzi impulsowej i skokowej – aplikacja kończy swoje działanie i wyświetla odpowiedni komunikat o błędzie. Kolejnym krokiem jest dostosowanie wektora czasu tak, aby pierwsza chwila czasu miała wartość zero (wektor czasu jest potrzebny w głównej mierze do wyświetlania wykresów). Jeżeli nie nastąpiło zatrzymanie aplikacji z powodu wymienionych błędów, następuje wyświetlenie komunikatu o tym ile próbek zostało usuniętych w preprocessingu (za wyjątkiem sytuacji gdy żadne nie zostały usunięte – wtedy komunikat nie wyświetla się).

Kolejnym etapem jest obliczenie charakterystyk impulsowej i skokowej poprzez wywołanie funkcji *get_impulse_response()* i get_*step_response()*. Następnie charakterystyka impulsowa jest dodatkowo przemnażana przez czas próbkowania uzyskany z wektora czasu, aby odpowiadała rzeczywistej charakterystyce obiektu dyskretnego o określonym czasie próbkowania.

Ostatnia sekcja programu odpowiada za zapis obliczonych odpowiedzi skokowej i impulsowej do pliku wyjściowego. W zależności od wybranej metody wczytywania pliku podczas generowania pliku wykonywalnego, plik wyjściowy będzie:

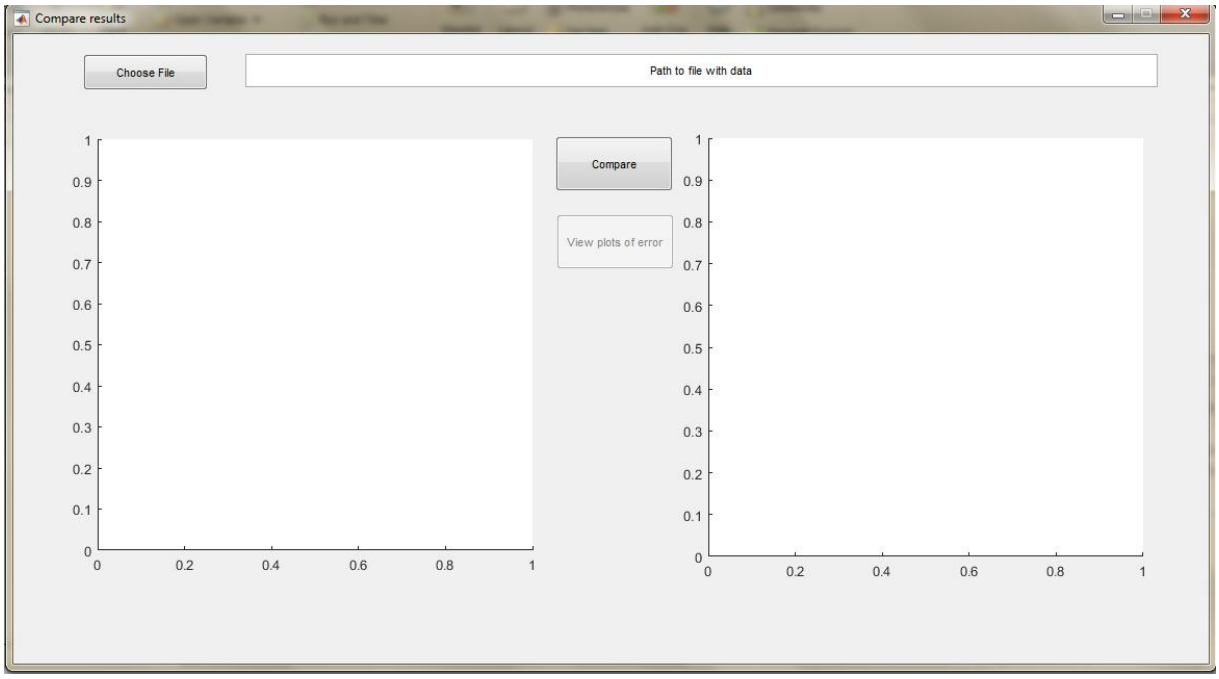
- miał nazwę i lokalizację określoną na stałe w przypadku pierwszego sposobu podawania danych wejściowych
- miał nazwę uzupełnioną o *_response* i będzie się znajdował w tej samej lokalizacji co plik wejściowy. Przykładowo jeżeli plik wejściowy znajduje się w lokalizacji *C:\test1\test_data.txt* to zostanie usunięte 5 znaków nazwy (*_data*) i nazwa zostanie uzupełniona, czyli w efekcie plik wyjściowy będzie miał lokalizację *C:\test1\test_response.txt*

Plik wyjściowy zawiera wektory: czasu, obliczonej charakterystyki impulsowej i obliczonej charakterystyki skokowej w postaci kolumn rozdzielonych przecinkiem i spacją. Umożliwia to łatwe importowanie pliku w tej postaci do programów Excel lub Matlab w celu narysowania wykresu lub dalszej obróbki danych. Przykładowy fragment danych wyjściowych pokazany jest poniżej:

```
0, 0, 0
0.5, 0.180408, 0.090204
1, 0.348074, 0.264241
1.5, 0.355867, 0.442175
2, 0.303639, 0.593994
2.5, 0.237417, 0.712703
…
```

## *3.3 Interfejs graficzny do weryfikacji wyników*

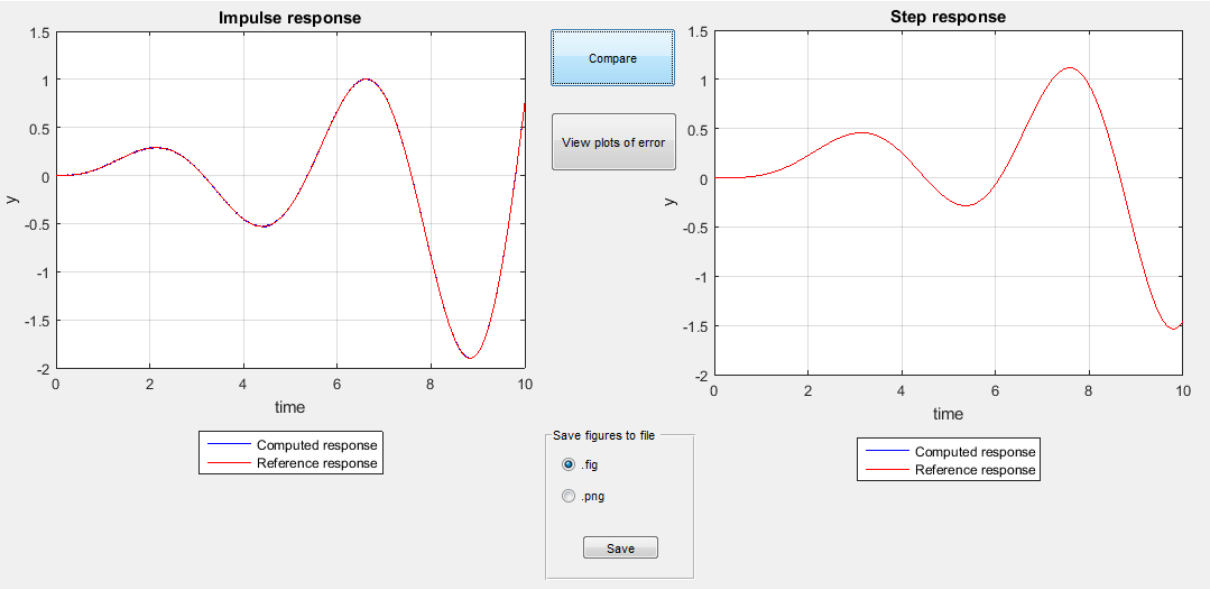
Kolejną część oprogramowania, czyli interfejs graficzny do weryfikacji uzyskanych wyników, można uruchomić przez wpisanie komendy *compareData* w linii komend w środowisku MATLAB. Weryfikacja ta polega na narysowaniu wykresów z danych uzyskanych z programu opisanego w rozdziale 3.2 oraz porównanie ich z wykresami narysowanymi przez środowisko MATLAB za pomocą funkcji *impulse* i *step*. Jest to możliwe dzięki zapisanej wcześniej do pliku *test_tf.mat* transmitancji, ponieważ funkcje *impulse* i *step* jej wymagają. Po uruchomieniu GUI zostanie wyświetlone okno pokazane na rysunku 3.10.




**Rysunek 3.10** Okno startowe GUI compareData

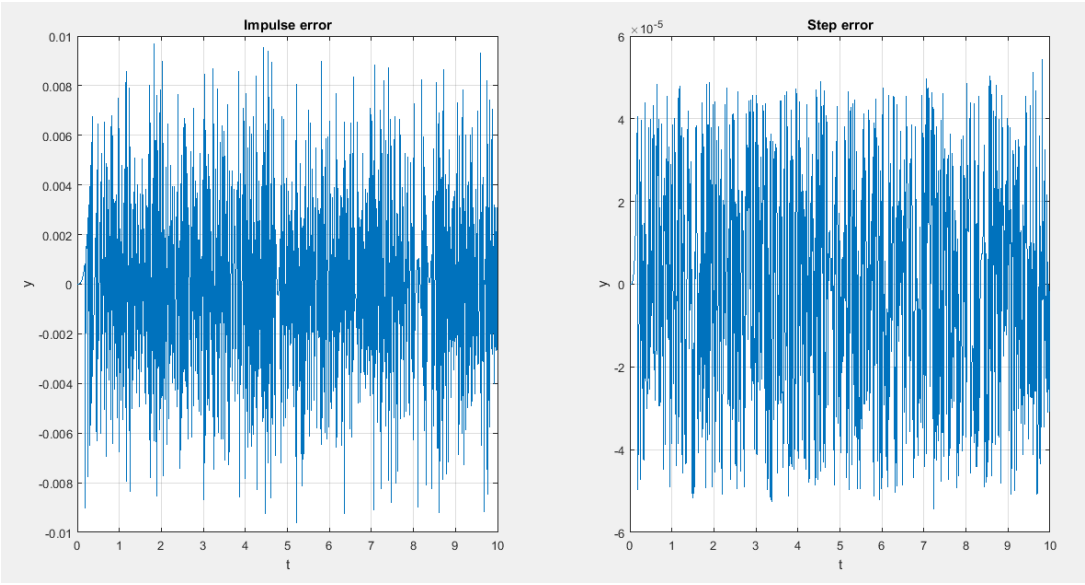
Pierwszym krokiem w przypadku tego GUI jest zawsze wybranie odpowiedniego pliku z danymi za pomocą przycisku *Choose File*. Należy pamiętać o tym, żeby wybrać plik, którego nazwa jest zakończona na *_response.txt*, w przeciwnym wypadku zostanie wyświetlony komunikat z błędem. Jest to plik z danymi wygenerowany przez program opisany w rozdziale 3.2. Po jego wybraniu w polu edycji zostanie wyświetlona ścieżka do tego pliku.

Następnie, aby otrzymać wykresy wyjściowe należy nacisnąć przycisk Compare. Zostaną wtedy wyświetlone wykresy charakterystyk impulsowej (na lewym wykresie) i skokowej (na prawym wykresie). Na każdym z tych wykresów pojawią się dwa przebiegi: w kolorze niebieskim dane obliczone oraz dane referencyjne w kolorze czerwonym. W przypadku braku pliku z transmitancją obiektu (na przykład w sytuacji gdy dane zbierane były z obiektu rzeczywistego) zostanie wyświetlony tylko wykres danych z pliku. Rysunek 3.11 przedstawia przykładowy fragment GUI po porównaniu wyników. Można na nim zauważyć, że po zakończeniu porównania został aktywowany przycisk *View plots of error* oraz panel *Save figures to file*.

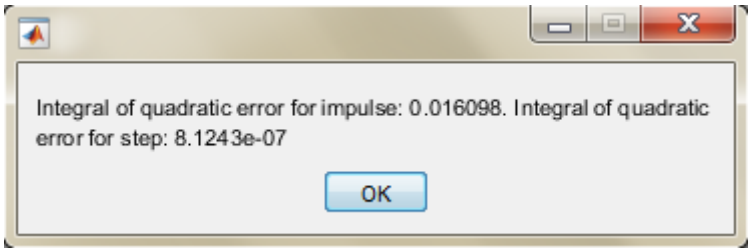



**Rysunek 3.11** Fragment GUI z wyświetlonymi wykresami

Na rysunku 3.11 widać, że wykres niebieski, czyli dane odczytane z pliku jest prawie niewidoczny pod wykresem czerwonym. Jest tak dla zdecydowanej większości danych wejściowych, ponieważ zastosowany algorytm wyznaczania charakterystyki impulsowej z wektorów sterowania i wyjścia obiektu jest bardzo dokładny. Z tego powodu wprowadzono opcję View plots of error. Po wciśnięciu przycisku zostanie wyświetlony wykres pokazany na rysunku 3.12 oraz komunikat pokazany na rysunku 3.13. Na wykresach wyświetlony jest błąd pomiędzy charakterystyką obliczoną i referencyjną odpowiednio jak w głównym oknie GUI: po lewej stronie dla charakterystyki impulsowej, a po prawej stronie dla charakterystyki skokowej. Wykresy te są bardzo szybko zmienne dlatego ich głównym celem jest pokazanie rzędu wartości błędu, a nie dokładnego przebiegu. W komunikacie (rysunek 3.13) jest wyświetlona informacja o wartości całki z kwadratu błędu, co może być przydatne w przypadku porównania różnych zestawów danych wejściowych ze sobą.




**Rysunek 3.12** Przykładowe wykresy błędu




**Rysunek 3.13** Komunikat informujący o całce z kwadratu błędu pomiędzy wykresami obliczonymi i referencyjnymi

Panel *Save figures to file* służy do zapisania otrzymanych wykresu do pliku .fig lub .png, w zależności od pola wybranego w bloku. Opcja ta została dodana, ponieważ w interfejsie nie ma możliwości przybliżenia wykresu, w razie takiej konieczności można zapisać go jako osobny wykres, który będzie można potem dowolnie przybliżać. Wykresy zostaną zapisane w dwóch plikach o nazwach *test_impulse* i *test_step* z odpowiednim rozszerzeniem.

# 4. Eksperymenty

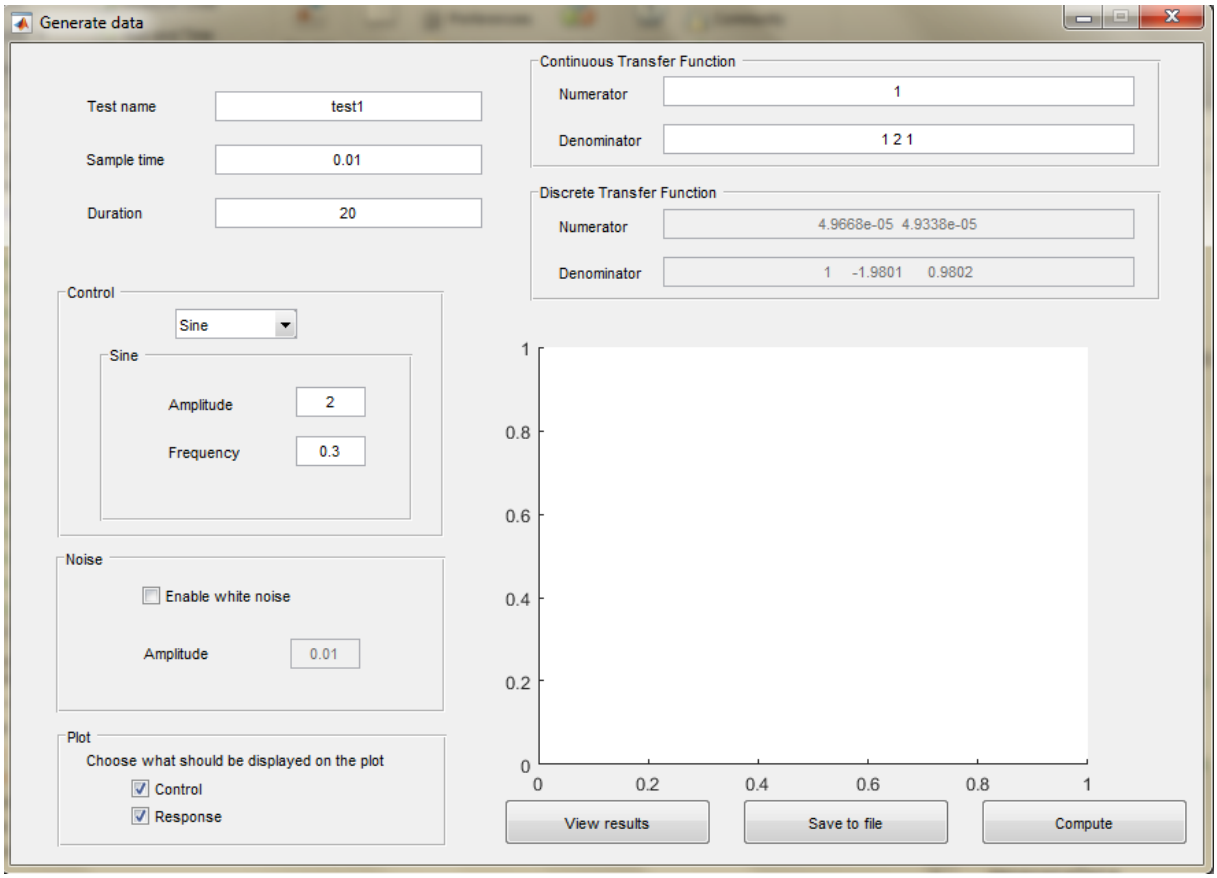
W tym rozdziale zostaną opisane przeprowadzone eksperymenty dla różnych obiektów i sterowań podanych na te obiekty.

Eksperyment 1

Pierwszym wykonanym eksperymentem jest podanie sterowania sinusoidalnego bez zakłóceń o amplitudzie 2 i częstotliwości 0.3 w czasie 20s na obiekt o transmitancji

$$G(s) = \frac{1}{s^2 + 2s + 1}$$

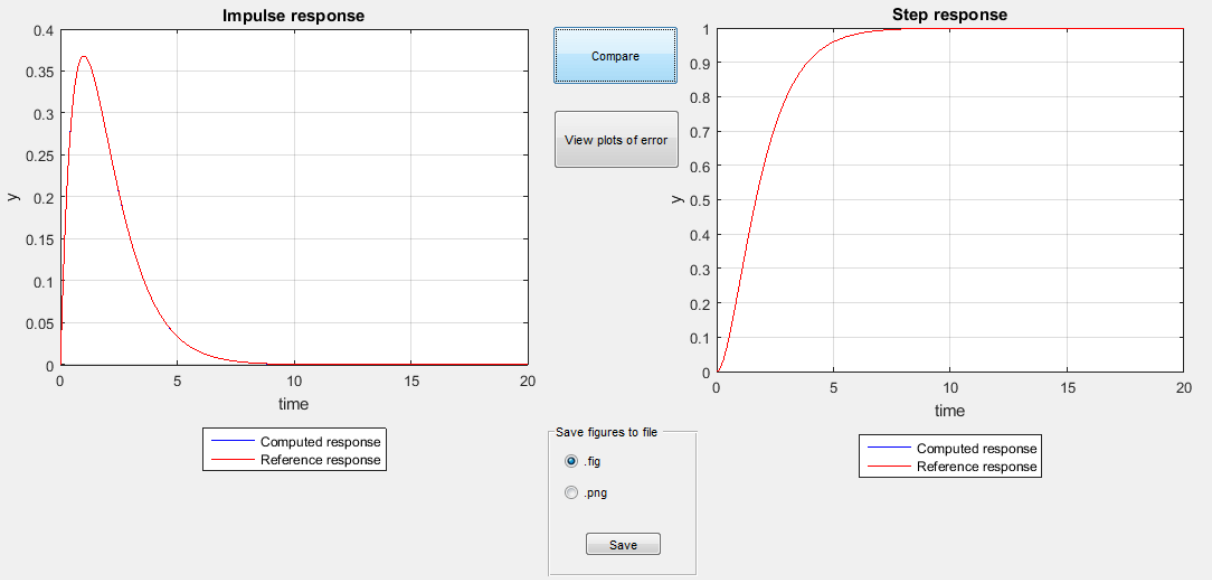
Test został wykonany wielokrotnie z różnym czasem próbkowania w celu zbadania jego wpływu na błędy obliczenia zadanych charakterystyk. Parametry ustawione do symulacji pokazane są na rysunku 4.1.



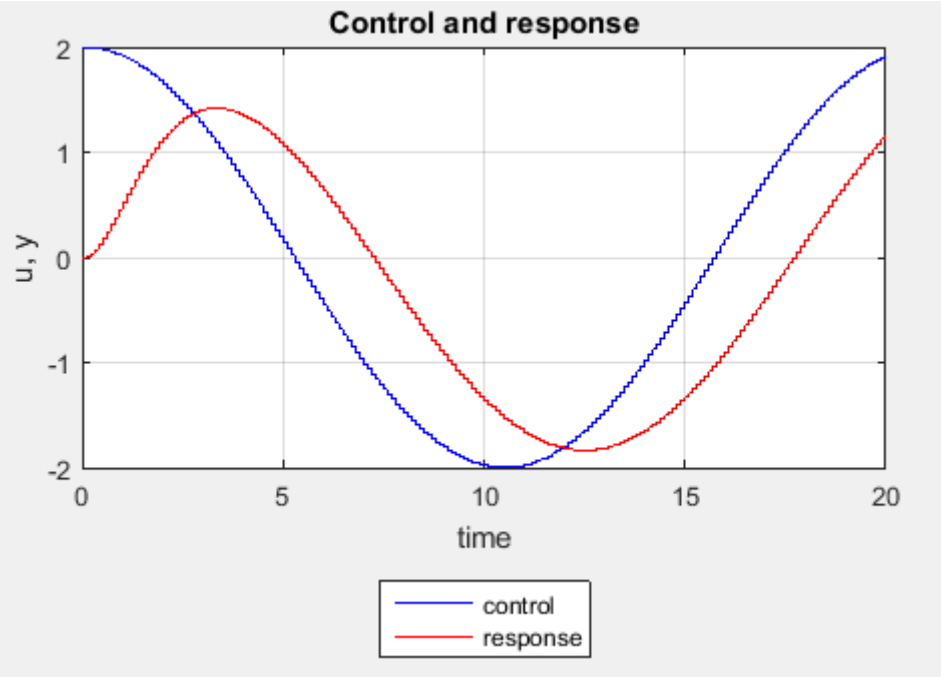

**Rysunek 4.1** Parametry testu pierwszego

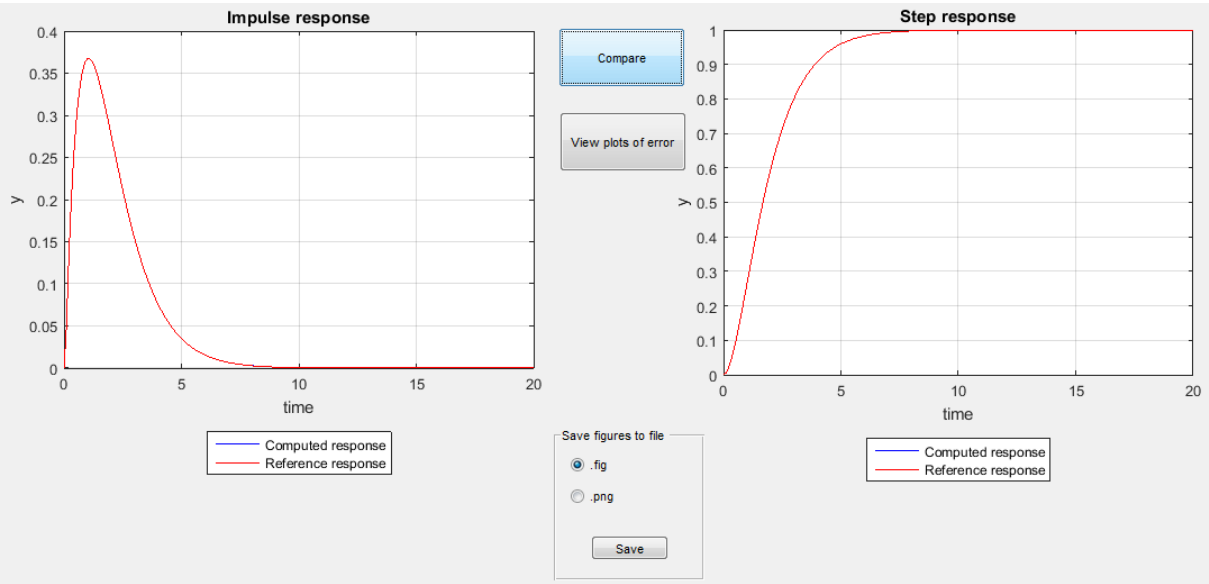
Eksperyment pierwszy składał się z następujących testów:

- test1 – czas próbkowania 0.01s (rysunek 4.2)
- test2 – czas próbkowania 0.1s (rysunek 4.3)
- test3 – czas próbkowania 0.5s (rysunek 4.4)
- test3 – czas próbkowania 3s (rysunek 4.5)

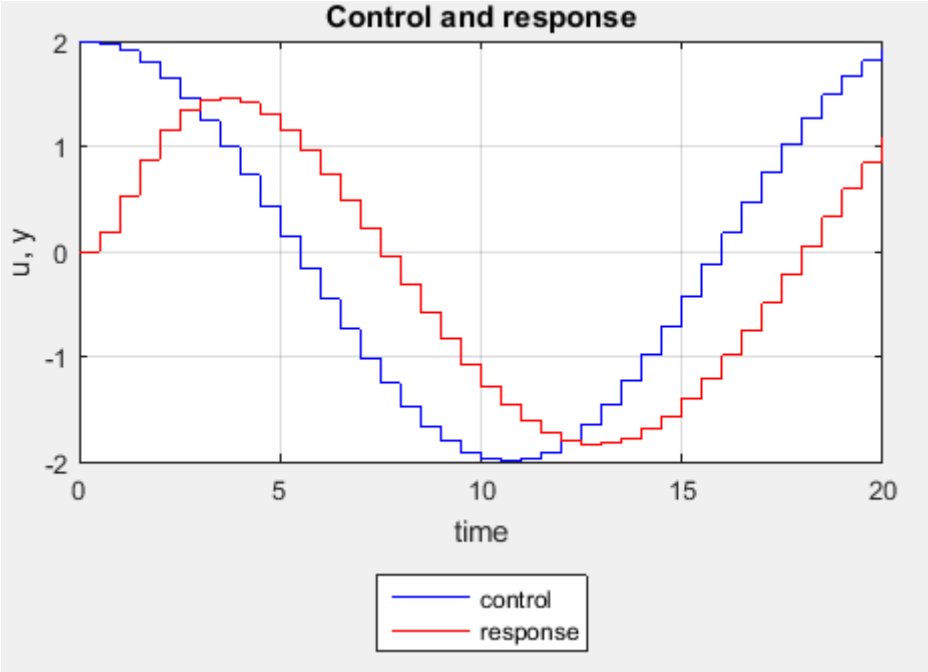



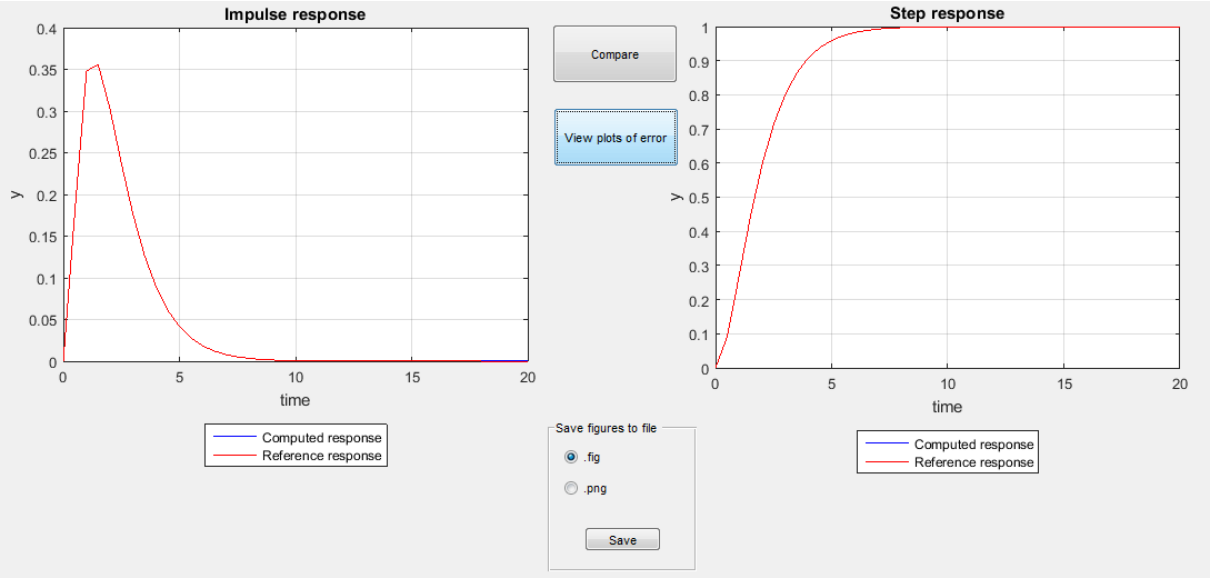

**Rysunek 4.2** Dane wejściowe i wyjściowe oprogramowania – test1



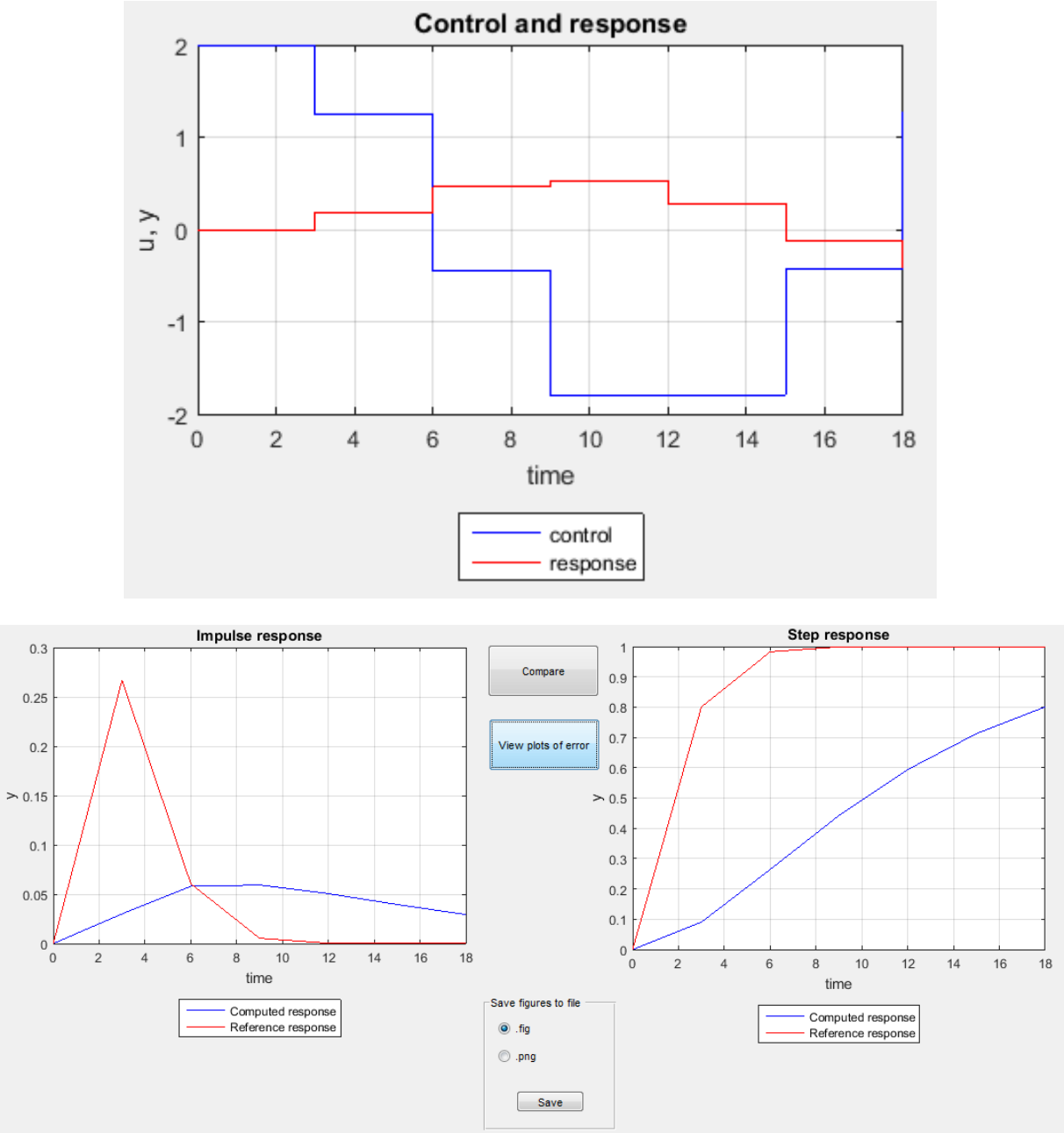




**Rysunek 4.3** Dane wejściowe i wyjściowe oprogramowania – test2




**Rysunek 4.4** Dane wejściowe i wyjściowe oprogramowania – test3




**Rysunek 4.5** Dane wejściowe i wyjściowe oprogramowania – test4

Podsumowanie

Dla podanych testów wskaźnik jakości (całka z kwadratu błędu) jest następujący:

- test1 – charakterystyka impulsowa: 7.9041e-07, charakterystyka skokowa: 1.9036e-10
- test2 – charakterystyka impulsowa: 9.0181e-10, charakterystyka skokowa: 2.4057e-11
- test3 – charakterystyka impulsowa: 6.1793e-12, charakterystyka skokowa: 3.4959e-12
- test4 – charakterystyka impulsowa: 0.063979, charakterystyka skokowa: 1.6179

Na rysunkach 4.2 – 4.4 widać, że niezależnie od czasu próbkowania (0.01s, 0.1s lub 0.5s) wykresy wyjściowe prawie idealnie pokrywają się z wykresami referencyjnymi. Patrząc na wskaźnik jakości można nawet stwierdzić, że poprawia się on wraz ze wzrostem czasu próbkowania. Może to być jednak skutek tego, że próbek jest dla tych testów coraz mniej, więc mniej też jest składników sumy w liczeniu wskaźnika jakości co powoduje mniejszą wartość tej sumy. Porównując wykresy błędów widać, że były one porównywalne dla tych trzech testów.

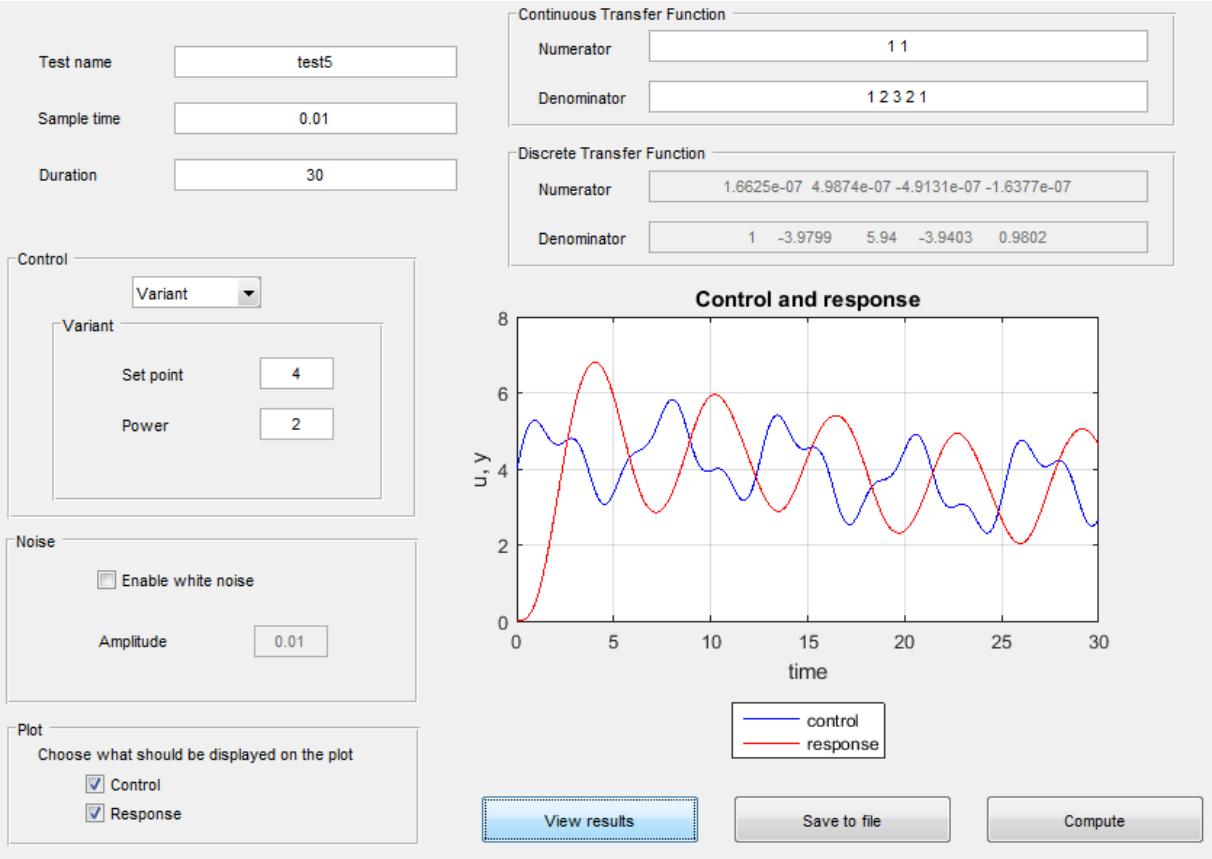
Inny jest wykres dla czasu próbkowania równego 3 sekundy (rysunek 4.5). Różni się on bardzo od wykresu referencyjnego, co jest wynikiem bardzo małej liczby próbek, z których ciężko jest odwzorować rzeczywiste sterowanie i odpowiedź obiektu. Na wykresie sterowania i odpowiedzi trudne jest zauważenie jaki kształt sterowania był zadany. Wykonywanie testów dla zbyt dużego okresu próbkowania nie jest wskazane ze względu na duże błędu obliczeń, jednak jest wykonalne w przypadku potrzeby przeprowadzenia takiego eksperymentu.

Eksperyment 2

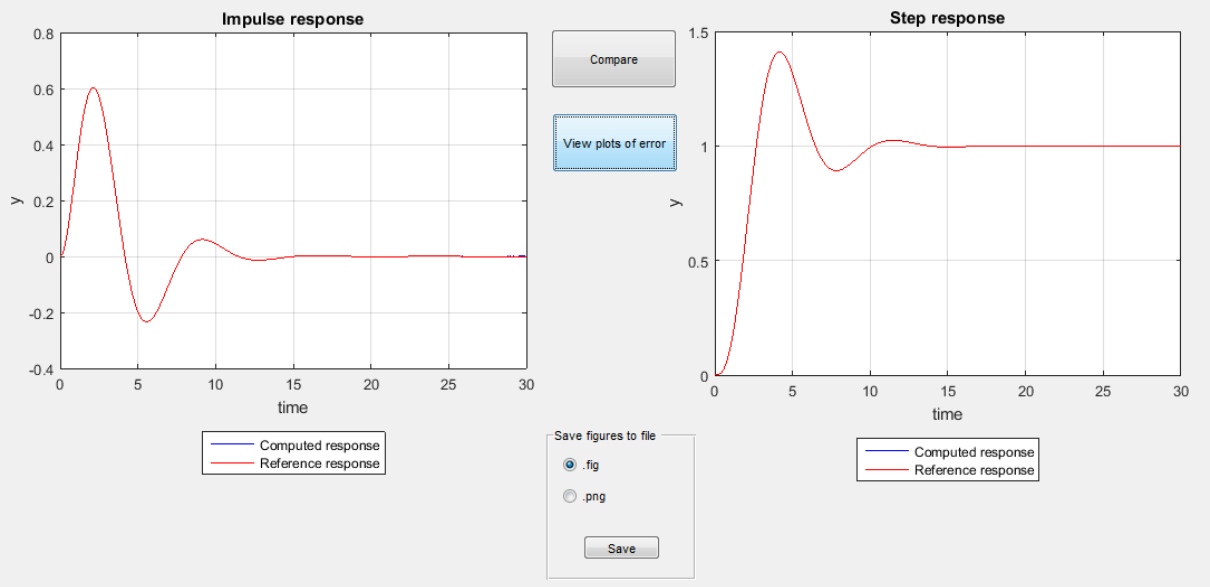
Kolejny eksperyment został przeprowadzony dla sterowania typu *Variant* o wartości zadanej równej 4 i amplitudzie równej 2. Transmitancja ciągła obiektu to:

$$G(s) = \frac{s+1}{s^4 + 2s^3 + 3s^2 + 2s + 1}$$

Te i pozostałe parametry pokazane są na rysunku 4.6. Rysunek 4.6 pokazuje także dane wejściowe w postaci sterowania i odpowiedzi. Wyniki działania oprogramowania przedstawione są na rysunku 4.7.




**Rysunek 4.6** Parametry i dane wejściowe eksperymentu drugiego – test5




**Rysunek 4.7** Wyniki eksperymentu drugiego

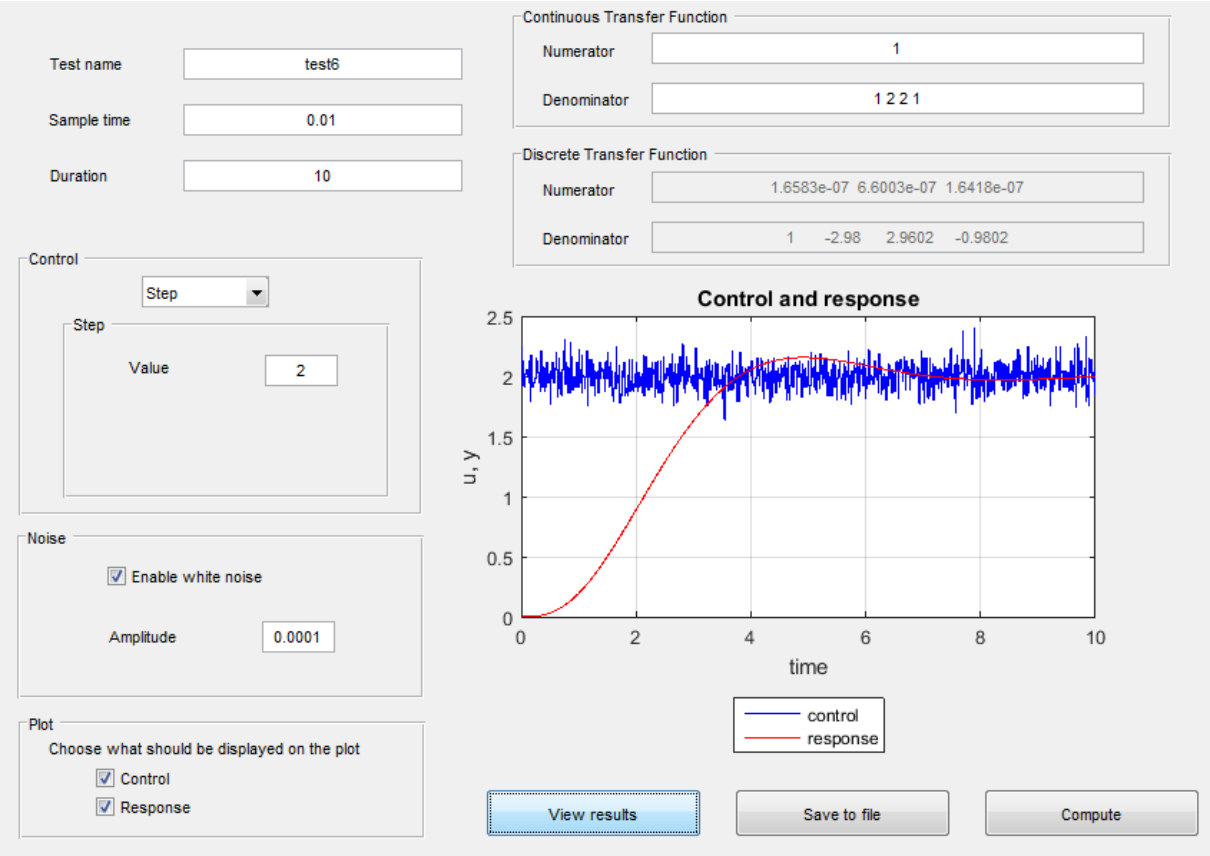
Na rysunku 4.7 widać, że także dla obiektu większego rzędu algorytm działa bardzo dobrze (błędy rzędu $10^{-7}$).

Eksperyment 3

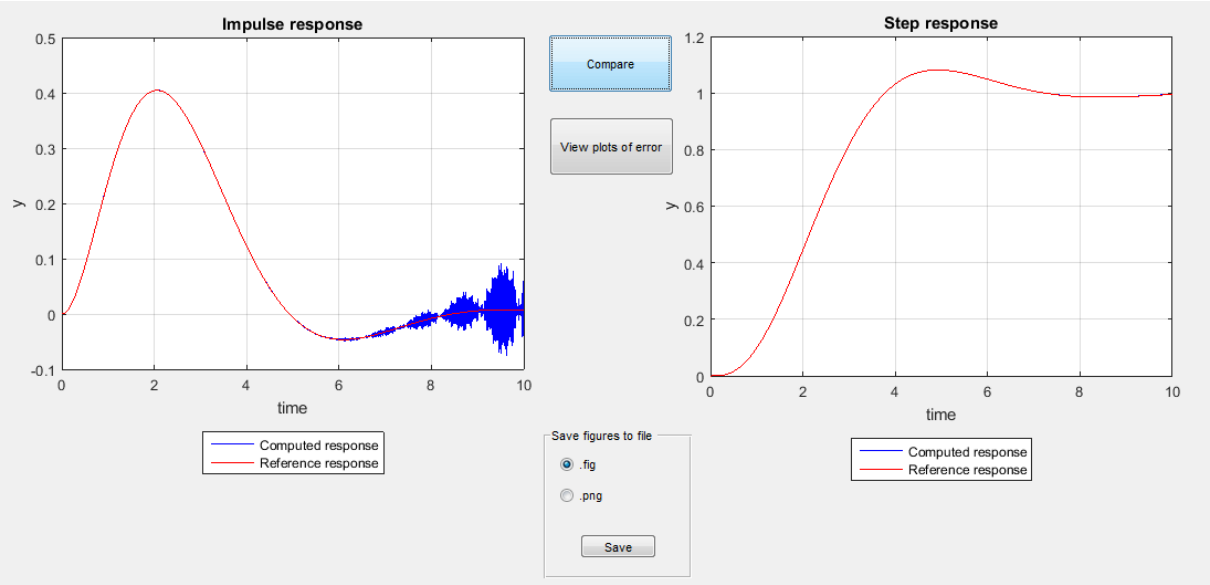
Ostatni eksperyment został przeprowadzony dla sterowania zakłóconego. Na obiekt o transmitancji

$$G(s) = \frac{1}{s^3 + 2s^2 + 2s + 1}$$

podano skok jednostkowy zakłócony szumem białym. Wszystkie parametry oraz dane wejściowe pokazane są na rysunku 4.8. Rysunek 4.9 przedstawia wyniki symulacji.




**Rysunek 4.8** Parametry i dane wejściowe eksperymentu trzeciego – test6




**Rysunek 4.9** Wyniki eksperymentu trzeciego

Przedstawiony eksperyment różni się od poprzednich, co jest skutkiem wprowadzenia zakłóceń do modelu. Można zauważyć, że przez pierwsze 6 sekund testu wynik pokrywa się prawie idealnie z wykresem referencyjnym, jednak wraz z upływem czasu błędy charakterystyki impulsowej narastają. Są to błędy numeryczne wynikające z obecności zakłóceń w modelu. Po przeprowadzeniu tego eksperymentu i kilku innych, które nie zostały zamieszczone w sprawozdaniu stwierdzono, że błędy numeryczne są większe wraz z upływem czasu oraz gdy charakterystyka ma wartości w okolicy zera.

# 5. Podsumowanie i wnioski

Zastosowany w oprogramowaniu algorytm działa bardzo dobrze, błędy pomiędzy obliczoną impulsową funkcją przejścia a charakterystyką referencyjną były bardzo małe, w większości przypadków rzędu $10^{-7}$. Dzięki zastosowaniu wzorów rekurencyjnych metoda nie jest obarczona dużą ilością obliczeń dlatego nawet dla dużej ilości próbek program wykonuje się bardzo szybko.

W początkowej fazie projektu system został zamodelowany jako układ ciągły, a nie dyskretny jak to było w założeniu. Jednak mimo tej pomyłki układ działał dobrze w większości testowanych przypadków. Jest tak dlatego, że wyjście z transmitancji ciągłej i odpowiadającej transmitancji dyskretnej jest takie samo dla takiego samego sterowania. Różnice pojawiały się przy dużej wartości czasu próbkowania w stosunku do czasu symulacji. W tym przypadku zastosowany algorytm do układu ciągłego nie był w stanie poprawnie obliczyć odpowiedzi, błędy numeryczne narastały do nieskończoności. Po poprawieniu modelu na układ dyskretny problem ten przestał występować.

Wadą oprogramowania jest, że nie zawsze dobrze radzi sobie z zakłóceniami podanego sterowania. Jeśli są one niewielkie w porównaniu do wartości sterowania to nie wpływają prawie wcale na jakość otrzymanych wyników. Jednak jeśli amplituda zakłóceń jest duża, to powodują one narastanie błędów numerycznych, które w efekcie sumują się do nieskończoności.

Efektem działania programu jest tworzenie dużej liczby różnych plików w folderze z programem, co niekiedy może być cechą niepożądaną. Pliki, które powstają po zakończeniu działania pojedynczego eksperymentu (o nazwie *test*):

- *test_data.txt* – plik z danymi wejściowymi do programu
- *test_tf.mat* – plik z transmitancją
- *test_response.txt* - wyniki działania programu
- *test_impulse.fig*/*test_impulse.png* – wykres z porównaniem obliczonej odpowiedzi impulsowej z referencyjną w formacie .fig lub .png
- *test_step.fig*/*test_step.png* – wykres z porównaniem obliczonej odpowiedzi skokowej z referencyjną w formacie .fig lub .png

Należy jednak zauważyć, że wszystkie pliki mają taką samą nazwę z odpowiednim dodatkiem na końcu, co umożliwia łatwe wydzielenie lub usunięcie wszystkich plików dotyczących jednego eksperymentu.

# Literatura


1. Witold Byrski, *Obserwacja i sterowanie w systemach dynamicznych*, Wydawnictwa AGH, Kraków 2007
2. https://www.sgi.com/tech/stl/
3. http://www.cplusplus.com/reference/list/list/