

Wydział Elektroniki i Technik Informacyjnych  
Politechnika Warszawska

Projektowanie układów sterowania  
(projekt grupowy)

Sprawozdanie z projektu i ćwiczenia laboratoryjnego  
nr 1, zadanie nr 1

Stanislau Stankevich, Rafał Bednarz, Ostrysz Jakub

Warszawa, 2021

# Spis treści

<b>1. Sprawdzenie poprawności podanych wartości</b>	2
<b>2. Wzory matematyczne</b>	3
2.1. Stałe i zmienne, indeksowanie	3
2.2. Wektory	3
2.3. Macierze	4
2.4. Większe wyrażenia matematyczne	4
<b>3. Tabele</b>	5
<b>4. Rysunki</b>	7
4.1. Schematy blokowe	7
4.2. Kolory	8
4.3. Funkcje statyczne	8
4.4. Wyniki symulacji i eksperymentów	11
4.4.1. Procesy jednowymiarowe	11
4.4.2. Procesy wielowymiarowe	15
4.5. Zapis wielu rysunków do jednego pliku pdf	20
4.6. Prosty eksport wykresów: skrypt matlab2tikz	21
4.6.1. MATLAB i dobór kolorów	23
4.7. Lokalizacja rysunków i tabel	26
<b>5. Listingi programów</b>	27
<b>6. Uwagi do wykonywania sprawozdań</b>	28
6.1. Kodowanie znaków	28
6.2. Niemerytoryczne powody niepowodzenia	28
6.3. Styl językowy sprawozdania	28
6.4. Zgodność z polskimi normami dotyczącymi przygotowywania publikacji	28
6.5. Używanie zwrotów w cudzysłowie	29
6.6. Separator dziesiętny	29
6.7. Odwołania do rysunków i tabel	29
6.8. Jednostki	30
6.9. Wstawianie kodu programu i jego zasadność	30
6.10. Warunki stosowania plot i stairs	30
6.11. Dodatkowe uwagi dotyczące wykresów	31
6.12. Przeprowadzanie eksperymentów	31
6.13. Krytyczne podejście do omówienia wyników	32
6.14. Cykliczna archiwizacja i analiza wyników po laboratorium	32
6.15. Omówienie wyników a dobieranie parametrów	32
<b>Bibliografia</b>	33

# 1. Sprawdzenie poprawności podanych wartości

Żeby sprawdzić poprawność podanych wartości podajemy na wejście wartości  $U_{PP}$  i patrzymy na jakiej wartości się ustali  $Y_{PP}$ .

Jeżeli dostępne są rysunki w formacie **pdf**, najwygodniej do przetworzenia dokumentu użyć polecenia **pdflatex**, które bezpośrednio generuje dokument w formacie **pdf**. Polecenie **latex** wymaga rysunków w formacie **ps** lub **eps** i generuje dokument w formacie **dvi**, który następnie można przekształcić do formatu **eps** lub **pdf**. Nie używamy rysunków zapisanych w plikach bitmapowych (**bmp**, **jpg**, **png**). Jedynym wyjątkiem są zdjęcia.

Istnieje wiele podręczników do nauki zasad składania dokumentów w  $\text{\LaTeX}$ u, np. doskonała praca zbiorowa [2] lub ew. podręcznik Wikibooks [5]. Do edycji dokumentów można wykorzystać np. program **T<sub>E</sub>XnicCenter**, dostępny pod adresem <http://www.texniccenter.org>. W przypadku problemów warto poszukać rozwiązania na forum <http://tex.stackexchange.com>.

W dalszej części dokumentu podano najważniejsze wymagania dotyczące wzorów matematycznych, tabeli i rysunków. Najszybszą metodą prowadzącą do otrzymania dokumentu jest modyfikacja niniejszego szablonu.

## 2. Wzory matematyczne

Stosujemy przecinek dziesiętny, a nie kropkę dziesiętną. Aby uniknąć dodatkowego odstępu, stosujemy zapis `\num{1,2345}` lub `\num{1.2345}`, co prowadzi do 1,2345, a nie `$1,2345$`, co prowadzi do 1, 2345. Stosujemy zapis  $1,2345 \cdot 10^{10}$ , a nie  $1,2345 \times 10^{10}$ . Powyższy zapis można stosować również w trybie matematycznym, np. `$\num{1.2345e10}$` skompiluje się do  $1,2345 \cdot 10^{10}$ .

### 2.1. Stałe i zmienne, indeksowanie

Skalarne stałe i zmienne zapisujemy w trybie matematycznym, np.  $x, y, z$ . Stosujemy indeksy dolne, np.  $x_i$ , górne, np.  $x^j$ , lub oba, np.  $x_i^j$ . Można również zastosować indeksy w nawiasach, np.  $y(k)$ . Jeżeli indeks zapisany jest czcionką pochyłą, spodziewamy się, że przyjmuje on wartość liczbową (liczby naturalne), np.  $x_i$  dla  $i = 1, \dots, 10$ . Jeżeli natomiast zastosujemy oznaczenie  $x_i$ , to wówczas indeks  $i$  nie przyjmuje żadnej wartości, jest on integralną częścią zmiennej lub stałej. Dlatego oznaczając horyzont sterowania stosujemy symbol  $N_u$ , a nie  $N_u$ , co by sugerowało, że indeks  $u$  przyjmuje pewne wartości z zakresu liczb naturalnych. Analogicznie, stała czasowa całkowania oznaczana jest jako  $T_i$ , a nie jako  $T_i$ , stała czasowa różniczkowania to  $T_d$ , a nie  $T_d$ . Sygnał wartości zadanej oznaczamy przez  $y^{zad}$ , a nie przez  $y^{zad}$ .

Nie należy stosować czcionki pochyłej również do tekstów, które uzupełniają wyrażenia matematyczne, np. zamiast błędnej postaci

$$y(x) = \begin{cases} x^2 & \text{gdy } x \leq 0 \\ x^3 & \text{gdy } x > 0 \end{cases}$$

powinno być

$$y(x) = \begin{cases} x^2 & \text{gdy } x \leq 0 \\ x^3 & \text{gdy } x > 0 \end{cases}$$

Odstępy w trybie matematycznym wymuszamy za pomocą instrukcji `\`, `\quad`, `\qquad` itd.

### 2.2. Wektory

Do oznaczenia wektorów najczęściej stosujemy symbole pogrubione, np.  $\mathbf{x}$ ,  $\Delta \mathbf{u}(k)$ . Pamiętajmy, że w matematyce wektory zawsze są pionowe. Wektory, których elementami są skalary, zapisujemy więc jako

$$\Delta \mathbf{u}(k) = [\Delta u(k|k) \ \dots \ \Delta u(k + N_u - 1|k)]^T \quad (2.1)$$

lub w postaci

$$\Delta \mathbf{u}(k) = \begin{bmatrix} \Delta u(k|k) \\ \vdots \\ \Delta u(k + N_u - 1|k) \end{bmatrix} \quad (2.2)$$

Jeżeli używamy wektorów, których elementami składowymi są inne wektory, najwygodniej zapisać je pionowo. Np. elementami wektora (2.2) są podwektory

$$\Delta u(k + p|k) = \begin{bmatrix} \Delta u_1(k + p|k) \\ \vdots \\ \Delta u_{n_u}(k + p|k) \end{bmatrix} \quad (2.3)$$

gdzie  $p = 1, \dots, N_u$ . A więc każdy z wektorów (2.3) ma długość  $n_u$ , wektor (2.2) ma długość  $n_u N_u$ .

### 2.3. Macierze

Do oznaczenia macierzy najczęściej stosujemy symbole pogrubione, np. macierz dynamiczna w algorytmie DMC dla procesu o jednym wejściu i jednym wyjściu ma wymiar  $N \times N_u$  i strukturę

$$\mathbf{G} = \begin{bmatrix} s_1 & 0 & \dots & 0 \\ s_2 & s_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ s_N & s_{N-1} & \dots & s_{N-N_u+1} \end{bmatrix} \quad (2.4)$$

W przypadku procesu o  $n_u$  wejściach i  $n_y$  wyjściach ma ona wymiar  $N \times N_u$  i postać

$$\mathbf{G} = \begin{bmatrix} \mathbf{S}_1 & \mathbf{0}_{n_y \times n_u} & \dots & \mathbf{0}_{n_y \times n_u} \\ \mathbf{S}_2 & \mathbf{S}_1 & \dots & \mathbf{0}_{n_y \times n_u} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_N & \mathbf{S}_{N-1} & \dots & \mathbf{S}_{N-N_u+1} \end{bmatrix} \quad (2.5)$$

gdzie każda z macierzy składowych ma wymiar  $n_y \times n_u$

$$\mathbf{S}_p = \begin{bmatrix} s_p^{1,1} & \dots & s_p^{1,n_u} \\ \vdots & \ddots & \vdots \\ s_p^{n_y,1} & \dots & s_p^{n_y,n_u} \end{bmatrix} \quad (2.6)$$

gdzie  $p = 1, \dots, N$ . A więc macierz (2.5) ma wymiar  $n_y N \times n_u N_u$ .

### 2.4. Większe wyrażenia matematyczne

W przypadku długich wzorów nie należy korzystać z otoczenia `equation`, ponieważ wzór taki zwykle nie mieści się na stronie o przyjętej szerokości, np.

$$y(k) = b_1 u(k-1) + b_2 u(k-2) + b_3 u(k-3) + b_4 u(k-4) + b_5 u(k-5) - a_1 y(k-1) - a_2 y(k-2) - a_3 y(k-3) - a_4 y(k-4) - a_5 y(k-5) \quad (2.7)$$

Należy zastosować otoczenie `align`, co prowadzi do wzoru

$$\begin{aligned} y(k) = & b_1 u(k-1) + b_2 u(k-2) + b_3 u(k-3) + b_4 u(k-4) + b_5 u(k-5) \\ & - a_1 y(k-1) - a_2 y(k-2) - a_3 y(k-3) - a_4 y(k-4) - a_5 y(k-5) \end{aligned} \quad (2.8)$$

Nie stosujemy otoczenia `split` z powodu błędnego centrowania. Numer wzoru złożonego z wielu wierszy umieszczamy tylko w ostatnim wierszu.

### 3. Tabele

W praktyce bardzo często należy wyrównać liczby względem cyfr znaczących w poszczególnych kolumnach (czyli przecinek dziesiętny ma być we wszystkich wierszach tabeli umieszczony w tym samym miejscu w pionie). Do wyrównania liczb należy wykorzystać pakiet `siunitx` (pakiety `rccol` oraz `dcolumn` mają mniejsze możliwości). Wszystkie przykłady podane w niniejszym rozdziale wykorzystują pakiet `siunitx`. Zwróćmy uwagę, że tytuły znajdujące się w pierwszym wierszu wszystkich tabel są wyśrodkowane (w obrębie kolejnych komórek).

Jeżeli standardowa szerokość kolumn jest za mała, należy w dowolnym wierszu wstawić z obu stron zawartości komórki polecenia `\hspace{odległość}`, które zapewniają odpowiednią szerokość. Modyfikację taką zastosowano w drugiej kolumnie tab. 3.3.

Jeżeli tabela jest szersza niż szerokość strony, należy zastosować otoczenie `sidewaystable` z pakietu `rotating`, co wykorzystano w tab. 3.4.

W zamieszczonych tabelkach wykorzystano polecenie `\rule` do wstawienia linii o zerowej szerokości do wierszy tabel, które są zbyt wąskie.

Tab. 3.1. Porównanie liczby parametrów (LP) i dokładności (SSE) modeli

Model	LP	$SSE_{ucz}$	$SSE_{wer}$	$SSE_{test}$
Liniowy	4	90,1815	70,7787	—
Neuronowy, $K = 1$	7	10,1649	10,3895	—
Neuronowy, $K = 2$	13	0,3282	0,3257	—
Neuronowy, $K = 3$	19	0,2014	0,1827	0,1468
Neuronowy, $K = 4$	25	0,1987	0,1906	—
Neuronowy, $K = 5$	31	0,1364	0,1971	—
Neuronowy, $K = 6$	37	0,1340	0,2044	—

Tab. 3.2. Porównanie liczby parametrów (LP) i dokładności (SSE) modeli

Model	LP	$SSE_{ucz}$	$SSE_{wer}$	$SSE_{test}$
Liniowy	4	$9,1815 \cdot 10^1$	$7,7787 \cdot 10^1$	—
Neuronowy, $K = 1$	7	$1,1649 \cdot 10^1$	$1,3895 \cdot 10^1$	—
Neuronowy, $K = 2$	13	$3,2821 \cdot 10^{-1}$	$3,2568 \cdot 10^{-1}$	—
Neuronowy, $K = 3$	19	$2,0137 \cdot 10^{-1}$	$1,8273 \cdot 10^{-1}$	$1,4682 \cdot 10^{-1}$
Neuronowy, $K = 4$	25	$1,9868 \cdot 10^{-1}$	$1,9063 \cdot 10^{-1}$	—
Neuronowy, $K = 5$	31	$1,3642 \cdot 10^{-1}$	$1,9712 \cdot 10^{-1}$	—
Neuronowy, $K = 6$	37	$1,3404 \cdot 10^{-1}$	$2,0440 \cdot 10^{-1}$	—

Tab. 3.3. Porównanie złożoności obliczeniowej

Algorytm	$N$	$N_u = 1$	$N_u = 2$	$N_u = 3$	$N_u = 4$	$N_u = 5$	$N_u = 10$
NPL	5	0,40	0,53	0,85	1,29	1,92	—
NO	5	2,61	5,04	8,00	12,65	18,37	—
NO <sub>apr</sub>	5	2,47	4,32	7,98	15,25	26,53	—
NPL	10	0,63	0,79	1,14	1,62	2,31	9,13
NO	10	5,20	9,04	13,56	19,17	26,26	76,50
NO <sub>apr</sub>	10	4,38	7,58	12,63	20,09	31,77	154,15

Tab. 3.4. Porównanie złożoności obliczeniowej

Algorytm	$N$	$N_u = 1$	$N_u = 2$	$N_u = 3$	$N_u = 4$	$N_u = 5$	$N_u = 10$	$N_u = 15$	$N_u = 20$	$N_u = 30$
NPL	5	0,40	0,53	0,85	1,29	1,92	—	—	—	—
NO	5	2,61	5,04	8,00	12,65	18,37	—	—	—	—
NO <sub>apr</sub>	5	2,47	4,32	7,98	15,25	26,53	—	—	—	—

## 4. Rysunki

Wszystkie elementy dokumentu opracowanego w systemie  $\text{\LaTeX}$  powinny wyglądać jednolicie. Do wykonywania rysunków korzystamy więc z mechanizmów oferowanych przez dodatkowe pakiety  $\text{\LaTeX}$ a, nie dołączamy rysunków wykonanych jakościowo różnych, np. wykonanych w programie Word. Nie używamy rysunków zapisanych w plikach bitmapowych, lecz w plikach wektorowych (pdf, ew. ps lub eps). Jedynym wyjątkiem są zdjęcia.

### 4.1. Schematy blokowe

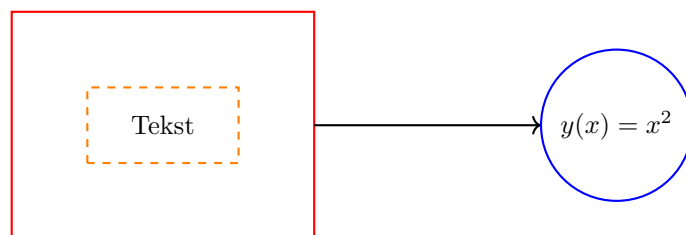
Do opracowania schematów blokowych najlepiej wykorzystać język opisu rysunków TikZ/PGF [3]. Przy wykonywaniu prostych rysunków po prostu opisujemy je za pomocą poleceń dodających kolejne elementy, tzn. prostokąty, okręgi, linie. Na przykład, ciąg poleceń:

```
\begin{figure}[b]
\centering
\begin{tikzpicture}
\draw [red, thick] (0,0) rectangle (4,3);
\draw [orange, thick,dashed] (1,1) rectangle (3,2);
\draw [->,thick] (4,1.5) -- (7,1.5);
\draw [blue, thick] (8,1.5) circle [radius=1];
\node at (2,1.5) {Tekst};
\node at (8,1.5) {$y(x)=x^2$};
\end{tikzpicture}
\end{figure}
```

pozwała narysować figury geometryczne przedstawione na rys. 4.1. Zwróćmy uwagę, że napis oraz wzór są złożone aktualnie wykorzystywaną czcionką, jej wielkość jest taka sama jak w całym dokumencie.

Przy większych rysunkach można wykorzystać programy umożliwiające ich przygotowanie przy wykorzystaniu środowiska graficznego, np. TikzEdt (<http://www.tikzedt.org/>), TpX (<http://tpx.sourceforge.net/>), ktikz (<https://www.linux-apps.com/p/1126914/>), GraTeX (<https://sourceforge.net/projects/gratex/>).

Można również wykorzystać starsze, klasyczne pakiety `picture`, `epic`, `eepic`. Również w tych przypadkach można „ręcznie” opisywać poszczególne elementy graficzne lub skorzystać ze środowiska graficznego, np. LaTeXPiX (<http://latexpixmap.informer.com/>), które znacznie przyspiesza pracę. Inne narzędzia, umożliwiające opracowanie rysunków wysokiej jakości, to METAPOST oraz PSTricks.



Rys. 4.1. Przykładowy rysunek wykonany w języku TikZ/PGF



## 4.2. Kolory

Przy umieszczaniu kilku wykresów na tym samym rysunku, np. wyników eksperymentów dla różnych wartości parametrów, należy zastosować kolory różniące się od siebie w znacznym stopniu, nie można stosować kolorów podobnych, np. kilku odcieni tego samego koloru. Warto zastosować standardową paletę kolorów o nazwie `lines`, użytą w pakiecie MATLAB, która została przedstawiona w tab. 4.1. Definicja tych kolorów w systemie L<sup>A</sup>T<sub>E</sub>X jest następująca:

```
\definecolor{niebieski}{rgb}{0,0.447,0.741}
\definecolor{czerwony}{rgb}{0.85,0.325,0.098}
\definecolor{zolty}{rgb}{0.929,0.694,0.125}
\definecolor{fioletowy}{rgb}{0.494,0.184,0.556}
\definecolor{zielony}{rgb}{0.466,0.674,0.188}
\definecolor{jasnoniebieski}{rgb}{0.301,0.745,0.933}
\definecolor{ciemnioczerwony}{rgb}{0.635,0.078,0.184}
```

Tab. 4.1. Paleta 7 kolorów

niebieski	
czerwony	
zolty	
fioletowy	
zielony	
jasnoniebieski	
ciemnioczerwony	

## 4.3. Funkcje statyczne

Do wykonywania wykresów prezentujących wyniki symulacji i eksperymentów stosuje się pakiet PGFPLOTS [1]. Załóżmy, że w katalogu `rysunki/dane_stat` znajduje się plik `dane_fx.txt` zawierający w pierwszej kolumnie wartości argumentu  $x$ , natomiast w drugiej kolumnie wartości funkcji  $f(x)$ :

```
-10.0000 -808.7350
-9.0000 -696.4791
-8.0000 -539.7850
-7.0000 -386.1268
-6.0000 -291.5881
-5.0000 -267.6436
-4.0000 -268.9551
-3.0000 -233.3995
-2.0000 -137.5303
-1.0000 -16.4788
0 70.0000
1.0000 94.1212
2.0000 87.2697
3.0000 112.8005
4.0000 209.4449
5.0000 357.3564
6.0000 498.0119
7.0000 589.6732
8.0000 647.4150
9.0000 730.9209
10.0000 891.2650
```

oraz podobny plik `dane_gx.txt`, definiujący funkcję  $g(x)$ . Aby narysować te funkcje stosuje się polecenia:

```

\newcommand{\szer}{8cm}
\newcommand{\wys}{7cm}

\begin{figure}[t]
\centering
\begin{tikzpicture}
\begin{axis}[
    width=\szer,
    height=\wys,
    xmin=-10,xmax=10,ymin=-1000,ymax=1000,
    xlabel={\textit{x}},
    ylabel={\textit{f(x)}, \textit{g(x)}},
    xtick={-10,-5,0,5,10},
    ytick={-1000,-500,0,500,1000},
    legend pos=south east,
]

\definecolor{niebieski}{rgb}{0,0.447,0.741}
\definecolor{czerwony}{rgb}{0.85,0.325,0.098}

\addplot[niebieski,thick]
    file {rysunki/dane_stat/dane_fx.txt};
\addplot[czerwony,thick,densely dashed]
    file {rysunki/dane_stat/dane_gx.txt};
\legend{\textit{f(x)},\textit{g(x)}}
\end{axis}
\end{tikzpicture}
\caption{Przykładowy rysunek funkcji  $f(x)$  i  $g(x)$  wykonany
w języku \texttt{PGFPLOTS}}
\label{r_pgfpLOTS_funkcje}
\end{figure}

```

Otrzymany rezultat przedstawiono na rys. 4.2.

Dla wykorzystanej palety kolorów ładnie wyglądają rysunki o pogrubionych liniach (styl `thick`).

Istnieje możliwość ustawienia wielkości czcionek liczb umieszczonych: na osiach (`tick label style`), w oznaczeniach osi (`label style`), w legendzie (`legend style`) oraz w tytule rysunku (`title style`). Przykładowa konfiguracja zmieniająca wielkość czcionek jest następująca:

```

\pgfplotsset{
    tick label style={font=\tiny},
    label style={font=\footnotesize},
    legend style={font=\footnotesize},
    title style={font=\footnotesize},
    /pgf/number format/.cd, use comma, 1000 sep={}
}

```

Dodatkowo, sekwencja `/pgf/number format/.cd, use comma, 1000 sep={}` ustawia przecinek jako separator dziesiętny (zamiast domyślnej kropki) oraz kasuje separator, oddzielający tysiące od setek (domyślenie jest to przecinek).

Opisany sposób implementacji rysunków jest poprawny, ale ma poważną wadę, ponieważ  $\text{\LaTeX}$  potrzebuje dość dużo czasu na ich przetworzenie. Okazuje się to dużym mankamentem szczególnie wówczas, gdy w dokumencie znajduje się dużo skomplikowanych rysunków. Skutecznym rozwiązaniem jest przygotowanie rysunków i zapis ich do plików `pdf`, a następnie dołączenie ich do głównego dokumentu poleceniem `\includegraphics`. W katalogu `rysunki/zapisz_pdf` podano kilka przykładów takich rysunków. Plik źródłowy `funkcje.tex` ma postać:

```

\documentclass{standalone}
\usepackage{pgfplots}

```

```

\pgfplotsset{
    tick label style={font=\tiny},
    label style={font=\footnotesize},
    legend style={font=\footnotesize},
    title style={font=\footnotesize},
    /pgf/number format/.cd, use comma, 1000 sep={}
}

\newcommand{\szer}{8cm}
\newcommand{\wys}{7cm}

\definecolor{niebieski}{rgb}{0,0.447,0.741}
\definecolor{czerwony}{rgb}{0.85,0.325,0.098}

\begin{document}

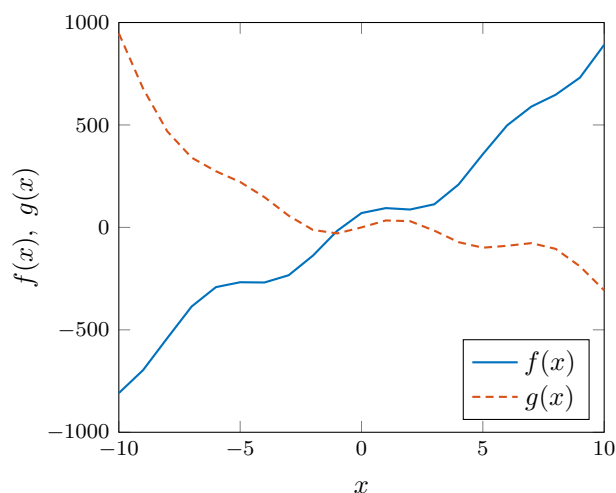
\begin{tikzpicture}
\begin{axis}[
    width=\szer,
    height=\wys,
    xmin=-10,xmax=10,ymin=-1000,ymax=1000,
    xlabel={x},
    ylabel={f(x), g(x)},
    xtick={-10,-5,0,5,10},
    ytick={-1000,-500,0,500,1000},
    legend pos=south east,
]

\addplot[niebieski,thick]
    file {../dane_stat/dane_fx.txt};
\addplot[czerwony,thick,densely dashed]
    file {../dane_stat/dane_gx.txt};
\legend{f(x),g(x)}
\end{axis}
\end{tikzpicture}

\end{document}

```

Zwróćmy uwagę, że dzięki zastosowaniu klasy `standalone`, wygenerowany zostanie rysunek o za-



Rys. 4.2. Przykładowy rysunek funkcji  $f(x)$  i  $g(x)$  wykonany w języku PGFPLOTS (rysunek jest generowany przy każdym przetworzeniu pliku źródłowego)

danych wymiarach (bez niepotrzebnych marginesów), plik **pdf** nie będzie formatu A4. Następujące polecenie

```
pdflatex funkcje.tex
```

zapisuje plik **funkcje.pdf**. Wygenerowany rysunek dołącza się do dokumentu ciągiem instrukcji:

```
\begin{figure}[tb]
\centering
\includegraphics[scale=1]{pgfplots_pdf/funkcje}
\caption{Przykładowy rysunek funkcji  $f(x)$  i  $g(x)$  wykonany
w języku \texttt{PGFPLOTS}}
\label{r_pgfplots_funkcje}
\end{figure}
```

Otrzymany rezultat przedstawiono na rys. 4.3. Oczywiście, rysunki 4.2 i 4.3 są bardzo podobne, inna jest tylko wielkość czcionek na osiach.

Przy dużych zbiorach danych  $\text{\LaTeX}$  zgłasza błąd pamięci. Należy wówczas zastosować system składu  $\text{Lua}\text{\LaTeX}$ .

Nie należy skalować rysunku przy wykorzystaniu argumentów polecenia `\includegraphics`, gdyż zmieni to wielkość zastosowanej czcionki. Jeżeli zachodzi konieczność zmiany wielkości rysunku, należy zmodyfikować plik źródłowy generujący rysunek.

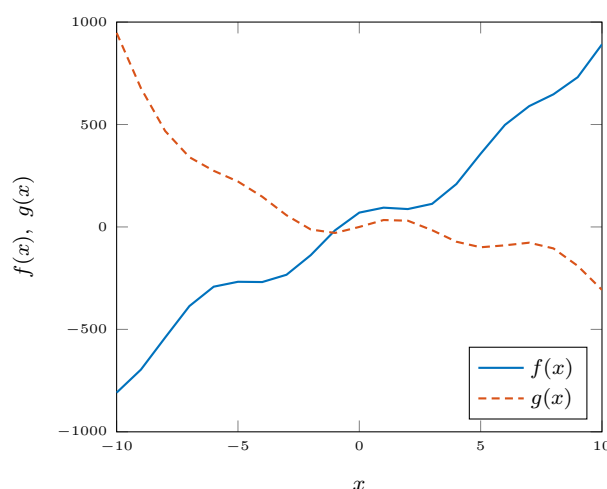
Jeżeli rysunek jest znacznie szerszy niż szerokość strony (oraz gdy nie możemy lub nie chcemy redukować szerokości rysunku), należy zastosować otoczenie `sidewaysfigure` z pakietu `rotating`, które działa analogicznie jak otoczenie `sidewaystable`.

W dokumentacji pakietu `PGFPLOTS` omówiono sposób przygotowania rysunków trójwymiarowych [1].

## 4.4. Wyniki symulacji i eksperymentów

### 4.4.1. Procesy jednowymiarowe

Załóżmy, że w katalogu `rysunki/symulacje11` znajduje się plik `yzad.txt` zawierający w pierwszej kolumnie pomiary czasu  $t$  (w sekundach), natomiast w drugiej kolumnie próbki sygnału wartości zadanej  $y^{\text{zad}}$ . Wykonano symulacje algorytmu regulacji GPC przy pięciu różnych wartościach parametru  $\lambda$ : 0,1, 0,2, 0,5, 1 i 2. Przebiegi sygnału sterującego  $u$  zapisano w plikach



Rys. 4.3. Przykładowy rysunek funkcji  $f(x)$  i  $g(x)$  wykonany w języku `PGFPLOTS` i zapisany w pliku **funkcje.pdf**

u\_lambda\_0\_1.txt, u\_lambda\_0\_2.txt, u\_lambda\_0\_5.txt, u\_lambda\_1.txt i u\_lambda\_2.txt, natomiast przebiegi sygnału wyjściowego procesu  $y$  zapisano w plikach y\_lambda\_0\_1.txt, y\_lambda\_0\_2.txt, y\_lambda\_0\_5.txt, y\_lambda\_1.txt i y\_lambda\_2.txt. Przygotowano plik symulacje11.tex, umożliwiający zapisanie rysunku do pliku symulacje11.pdf. Znajduje się on w katalogu rysunki/zapisz\_pdf i ma następującą postać:

```
\documentclass{standalone}
\usepackage{pgfplots}
\usepackage{siunitx}
\sisetup{detect-weight,exponent-product=\cdot,output-decimal-marker={,},
per-mode=symbol,binary-units=true,range-phrase={-},range-units=single}
\usetikzlibrary{pgfplots.groupplots}

\pgfplotsset{
    tick label style={font=\footnotesize},
    label style={font=\footnotesize},
    legend style={font=\footnotesize},
    title style={font=\footnotesize},
    /pgf/number format/.cd, use comma, 1000 sep={}
}

\newcommand{\szer}{16cm}
\newcommand{\wys}{5.6cm}
\newcommand{\odstepionowy}{1.2cm}

\definecolor{niebieski}{rgb}{0,0.447,0.741}
\definecolor{czerwony}{rgb}{0.85,0.325,0.098}
\definecolor{zolty}{rgb}{0.929,0.694,0.125}
\definecolor{fioletowy}{rgb}{0.494,0.184,0.556}
\definecolor{zielony}{rgb}{0.466,0.674,0.188}
\definecolor{jasnoniebieski}{rgb}{0.301,0.745,0.933}
\definecolor{ciemnioczerwony}{rgb}{0.635,0.078,0.184}

\begin{document}

\begin{tikzpicture}
\begin{groupplot}[group style={group size=1 by 2,vertical sep=\odstepionowy},
width=\szer,height=\wys]
%%1
\nextgroupplot
[xmin=0,xmax=1,ymin=-4.5,ymax=8.5,
xtick={0,0.2,0.4,0.6,0.8,1},ytick={-4,-2,0,2,4,8},
xlabel=\mathrm{s},ylabel=\$u$,legend cell align=left,
legend pos=north east]
\addplot[const plot,color=niebieski,thick]
file {../symulacje11/u_lambda_0_1.txt};
\addplot[const plot,color=czerwony,thick,densely dashed]
file {../symulacje11/u_lambda_0_2.txt};
\addplot[const plot,color=zolty,thick,densely dashdotted]
file {../symulacje11/u_lambda_0_5.txt};
\addplot[const plot,color=fioletowy,thick,densely dashdotdotted]
file {../symulacje11/u_lambda_1.txt};
\addplot[const plot,color=zielony,thick,densely dotted]
file {../symulacje11/u_lambda_2.txt};
\legend{\lambda=\num{0.1}\$, \lambda=\num{0.2}\$, \lambda=\num{0.5}\$,
\lambda=\num{1}\$, \lambda=\num{2}\$}
%%2
\nextgroupplot
[xmin=0,xmax=1,ymin=-1.25,ymax=2.25,
xtick={0,0.2,0.4,0.6,0.8,1},ytick={-1,-0.5,0,0.5,1,1.5,2},
xlabel=\mathrm{s},ylabel={\$y^{\mathrm{zad}}}, \ y\$},
legend cell align=left,legend style={at={(axis cs:0.6,-0.3)},
```

```

        anchor=south west}}
\addplot[const plot,color=gray,thick]
    file {../symulacje11/yzad.txt};
\addplot[color=niebieski,thick]
    file {../symulacje11/y_lambda_0_1.txt};
\addplot[const plot,color=czerwony,thick,densely dashed]
    file {../symulacje11/y_lambda_0_2.txt};
\addplot[const plot,color=zolty,thick,densely dashdotted]
    file {../symulacje11/y_lambda_0_5.txt};
\addplot[const plot,color=fioletowy,thick,densely dashdotdotted]
    file {../symulacje11/y_lambda_1.txt};
\addplot[const plot,color=zielony,thick,densely dotted]
    file {../symulacje11/y_lambda_2.txt};
\legend{$y^{\mathrm{zad}}$, $\lambda=\mathrm{num}\{0.1\}$, $\lambda=\mathrm{num}\{0.2\}$,
        $\lambda=\mathrm{num}\{0.5\}$, $\lambda=\mathrm{num}\{1\}$, $\lambda=\mathrm{num}\{2\}$}
\end{groupplot}
\end{tikzpicture}

\end{document}

```

Na początku powyższego pliku zostaje wczytany i skonfigurowany pakiet `siunitx`. Między innymi, ustawiamy przecinek jako separator dziesiętny. Liczby z częścią ułamkową zapisujemy w postaci `\num{0.1}`. W zależności od konfiguracji, otrzymamy 0.1 lub 0,1. Polecenie

```
pdflatex symulacje11.tex
```

zapisuje plik `symulacje11.pdf`. Rezultat przedstawiono na rys. 4.4. Rysunek dołącza się do dokumentu instrukcją `\includegraphics`. Zwróćmy uwagę, że do narysowania wyników symulacji dla kolejnych wartości parametru  $\lambda$  zastosowano różne kolory oraz różne style linii (linia ciągła, linia przerywana, itd.). Umożliwia to łatwe rozróżnienie dokumentów na czarno-białym wydruku. Przy wydruku kolorowym oraz dokumentach elektronicznych można zrezygnować ze stosowania różnych stylów linii, do ich rozróżnienia wystarczające są kolory, pod warunkiem jednak, że kolejne krzywe nie są położone bardzo blisko siebie.

Czasami, ze względu na ograniczoną objętość dokumentu, należy zmniejszyć rysunki. Aby zmniejszyć objętość można zastosować ułożenie poziome dwóch rysunków, prezentujących wyniki symulacji procesu. Przygotowano plik `symulacje11_wersja2.tex`, umożliwiający zapisanie rysunku do pliku `symulacje11_wersja2.pdf`. Znajduje się on w katalogu `rysunki/zapisz_pdf` i ma następującą postać:

```

\documentclass{standalone}
\usepackage{pgfplots}
\usepackage{siunitx}
\sisetup{detect-weight,exponent-product=\cdot,output-decimal-marker={,},
    per-mode=symbol,binary-units=true,range-phrase={-},range-units=single}
\usetikzlibrary{pgfplots.groupplots}

\pgfplotsset{
    tick label style={font=\footnotesize},
    label style={font=\footnotesize},
    legend style={font=\footnotesize},
    title style={font=\footnotesize},
    /pgf/number format/.cd, use comma, 1000 sep={}
}

\newcommand{\szer}{8cm}
\newcommand{\wys}{5.6cm}
\newcommand{\odstepoziomy}{1.9cm}

\definecolor{niebieski}{rgb}{0,0.447,0.741}
\definecolor{czerwony}{rgb}{0.85,0.325,0.098}
\definecolor{zolty}{rgb}{0.929,0.694,0.125}

```

```

\definecolor{fioletowy}{rgb}{0.494,0.184,0.556}
\definecolor{zielony}{rgb}{0.466,0.674,0.188}
\definecolor{jasnoniebieski}{rgb}{0.301,0.745,0.933}
\definecolor{ciemnioczerwony}{rgb}{0.635,0.078,0.184}

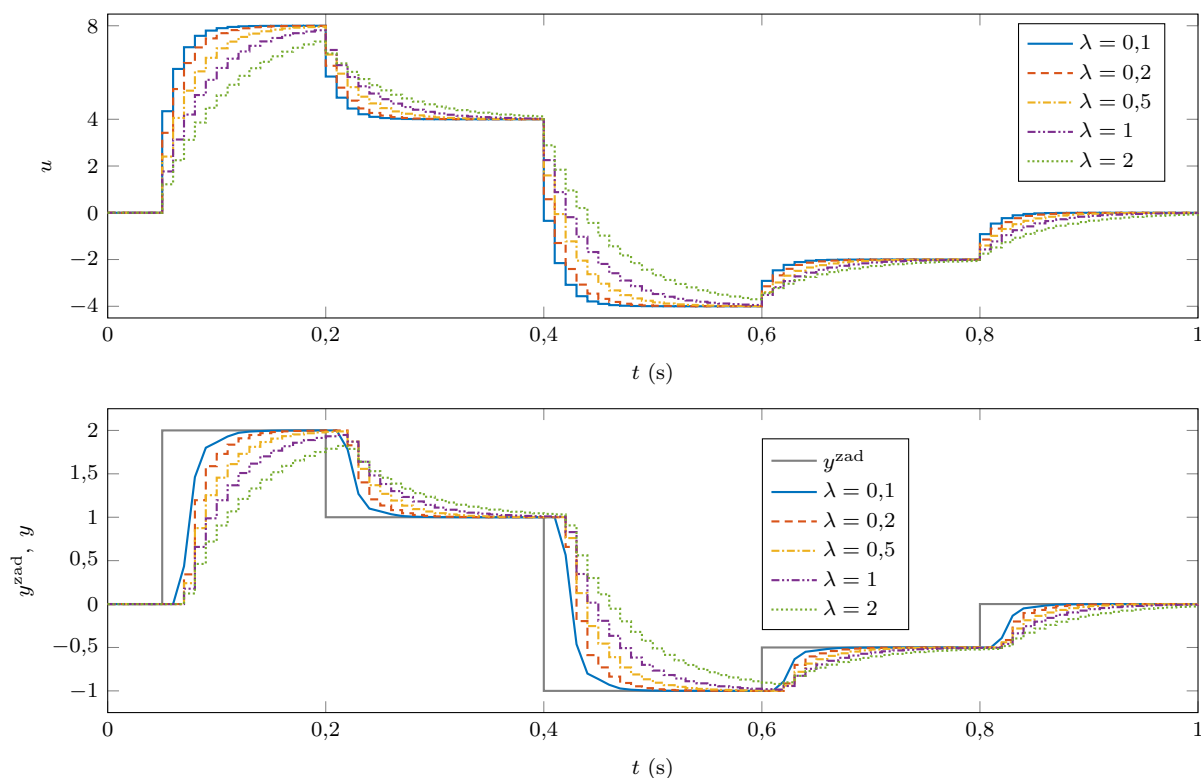
\begin{document}

\begin{tikzpicture}
\begin{groupplot}[group style={group size=2 by 1,horizontal sep=\odstepoziomy},
width=\szer,height=\wys]
%%1
\nextgroupplot
[xmin=0,xmax=1,ymin=-4.5,ymax=8.5,
xtick={0,0.2,0.4,0.6,0.8,1},ytick={-4,-2,0,2,4,6,8},
xlabel=$t \ (\mathrm{s})$,ylabel=$u$,
legend style={at={(0,1.15)},anchor=west},
legend columns=6,
legend style={/tikz/every even column/.append style={column sep=1.5cm}}]
\addplot[const plot,color=niebieski,thick]
file {../symulacje11/u_lambda_0_1.txt};
\addplot[const plot,color=czerwony,thick,densely dashed]
file {../symulacje11/u_lambda_0_2.txt};
\addplot[const plot,color=zolty,thick,densely dashdotted]
file {../symulacje11/u_lambda_0_5.txt};
\addplot[const plot,color=fioletowy,thick,densely dashdotdotted]
file {../symulacje11/u_lambda_1.txt};
\addplot[const plot,color=zielony,thick,densely dotted]
file {../symulacje11/u_lambda_2.txt};
\legend{$\lambda=\num{0.1}$,$\lambda=\num{0.2}$,$\lambda=\num{0.5}$,$\lambda=\num{1}$,$\lambda=\num{2}$}
%%2
\nextgroupplot
[xmin=0,xmax=1,ymin=-1.25,ymax=2.25,
xtick={0,0.2,0.4,0.6,0.8,1},ytick={-1,-0.5,0,0.5,1,1.5,2},
xlabel=$t \ (\mathrm{s})$,ylabel={$y^{\mathrm{zad}}$, \ y$},
legend cell align=left]
\addplot[const plot,color=gray,thick]
file {../symulacje11/yzad.txt};
\addplot[color=niebieski,thick]
file {../symulacje11/y_lambda_0_1.txt};
\addplot[const plot,color=czerwony,thick,densely dashed]
file {../symulacje11/y_lambda_0_2.txt};
\addplot[const plot,color=zolty,thick,densely dashdotted]
file {../symulacje11/y_lambda_0_5.txt};
\addplot[const plot,color=fioletowy,thick,densely dashdotdotted]
file {../symulacje11/y_lambda_1.txt};
\addplot[const plot,color=zielony,thick,densely dotted]
file {../symulacje11/y_lambda_2.txt};
\legend{$y^{\mathrm{zad}}$}
\end{groupplot}
\end{tikzpicture}

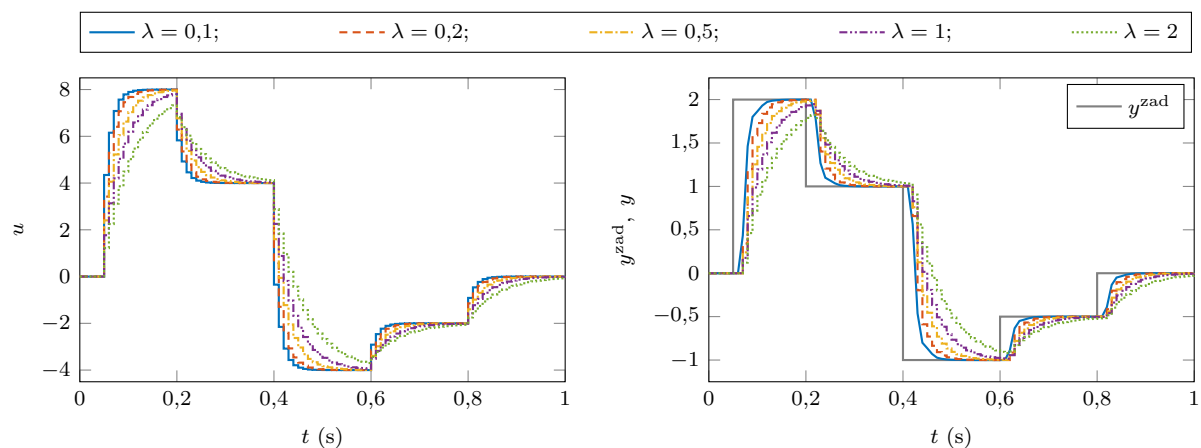
\end{document}

```

Rezultat przedstawiono na rys. 4.5.



Rys. 4.4. Przykładowy rysunek wyników symulacji procesu jednowymiarowego wykonany w języku PGFPLOTS i zapisany w pliku `symulacje11.pdf`



Rys. 4.5. Przykładowy rysunek wyników symulacji procesu jednowymiarowego wykonany w języku PGFPLOTS i zapisany w pliku `symulacje11_wersja2.pdf`

#### 4.4.2. Procesy wielowymiarowe

ZałóŜmy, Ŝe w katalogu `rysunki/symulacje22` znajduj się wyniki symulacji procesu o dwóch wejściach i dwóch wyjściach zapisane w plikach: `yzad1.txt`, `yzad2.txt`, `u1.txt`, `u2.txt`, `y1.txt`, `y2.txt`. W pierwszej kolumnie tych plików podano czas  $t$  (w sekundach), natomiast w drugiej kolumnie wartoć odpowiedniej zmiennej. Plik `symulacje22.tex`, umoŝliwiaj zapisanie rysunku do pliku `symulacje22.pdf` znajduje się w katalogu `rysunki/zapisz_pdf` i ma następuj postać:

```
\documentclass{standalone}
```



```

\usepackage{pgfplots}
\usepackage{siunitx}
\sisetup{detect-weight,exponent-product=\cdot,output-decimal-marker={,},
         per-mode=symbol,binary-units=true,range-phrase={-},range-units=single}
\usetikzlibrary{pgfplots.groupplots}

\pgfplotsset{
    tick label style={font=\footnotesize},
    label style={font=\footnotesize},
    legend style={font=\footnotesize},
    title style={font=\footnotesize},
    /pgf/number format/.cd, use comma, 1000 sep={}
}

\newcommand{\szer}{16cm}
\newcommand{\wys}{6cm}
\newcommand{\odstepionowy}{1.2cm}

\definecolor{niebieski}{rgb}{0,0.447,0.741}

\begin{document}

\begin{tikzpicture}
\begin{groupplot}[group style={group size=1 by 4,vertical sep=\odstepionowy},
width=\szer,height=\wys]
%%1
\nextgroupplot
[xmin=0,xmax=1,ymin=-0.1,ymax=3,
xtick={0,0.2,0.4,0.6,0.8,1},ytick={0,1,2,3},
xlabel=\(t \ (\mathrm{s)}),ylabel=\(u_1),legend cell align=left,
legend pos=north east]
\addplot[const plot,color=niebieski,thick]
file {../symulacje22/u1.txt};
%%2
\nextgroupplot
[xmin=0,xmax=1,ymin=-0.5,ymax=1.5,
xtick={0,0.2,0.4,0.6,0.8,1},ytick={-0.5,0,0.5,1,1.5},
xlabel=\(t \ (\mathrm{s)}),ylabel=\(u_2),legend cell align=left,
legend pos=north east]
\addplot[const plot,color=niebieski,thick]
file {../symulacje22/u2.txt};
%%3
\nextgroupplot
[xmin=0,xmax=1,ymin=-0.25,ymax=1.5,
xtick={0,0.2,0.4,0.6,0.8,1},ytick={0,0.5,1,1.5},
xlabel=\(t \ (\mathrm{s)}),ylabel=\(y_1^{\mathrm{zad}}), \ y_1},
legend cell align=left,legend pos=north east]
\addplot[const plot,color=gray,thick]
file {../symulacje22/yzad1.txt};
\addplot[color=niebieski,thick]
file {../symulacje22/y1.txt};
\legend{\(y_1^{\mathrm{zad}}), \(y_1}
%%4
\nextgroupplot
[xmin=0,xmax=1,ymin=-0.25,ymax=1.5,
xtick={0,0.2,0.4,0.6,0.8,1},ytick={0,0.5,1,1.5},
xlabel=\(t \ (\mathrm{s)}),ylabel=\(y_2^{\mathrm{zad}}), \ y_2},
legend cell align=left,legend pos=north east]
\addplot[const plot,color=gray,thick]
file {../symulacje22/yzad2.txt};
\addplot[color=niebieski,thick]
file {../symulacje22/y2.txt};
\legend{\(y_2^{\mathrm{zad}}), \(y_2}

```

```

\end{groupplot}
\end{tikzpicture}

\end{document}

```

Rezultat przedstawiono na rys. 4.6.

Aby zmniejszyć objętość można nieco inaczej ułożyć 4 rysunki, prezentujące wyniki symulacji procesu dwuwymiarowego. Przygotowano plik `symulacje22_wersja2.tex`, umożliwiający zapisanie rysunku do pliku `symulacje22_wersja2.pdf`. Znajduje się on w katalogu `rysunki/zapisz_pdf` i ma następującą postać:

```

\documentclass{standalone}
\usepackage{pgfplots}
\usepackage{siunitx}
\sisetup{detect-weight,exponent-product=\cdot,output-decimal-marker={,},
per-mode=symbol,binary-units=true,range-phrase={-},range-units=single}
\usetikzlibrary{pgfplots.groupplots}

\pgfplotsset{
    tick label style={font=\footnotesize},
    label style={font=\footnotesize},
    legend style={font=\footnotesize},
    title style={font=\footnotesize},
    /pgf/number format/.cd, use comma, 1000 sep={}
}

\newcommand{\szer}{8cm}
\newcommand{\wys}{5.6cm}
\newcommand{\odstepionowy}{1.2cm}
\newcommand{\odstepoziomy}{1.9cm}

\definecolor{niebieski}{rgb}{0,0.447,0.741}

\begin{document}

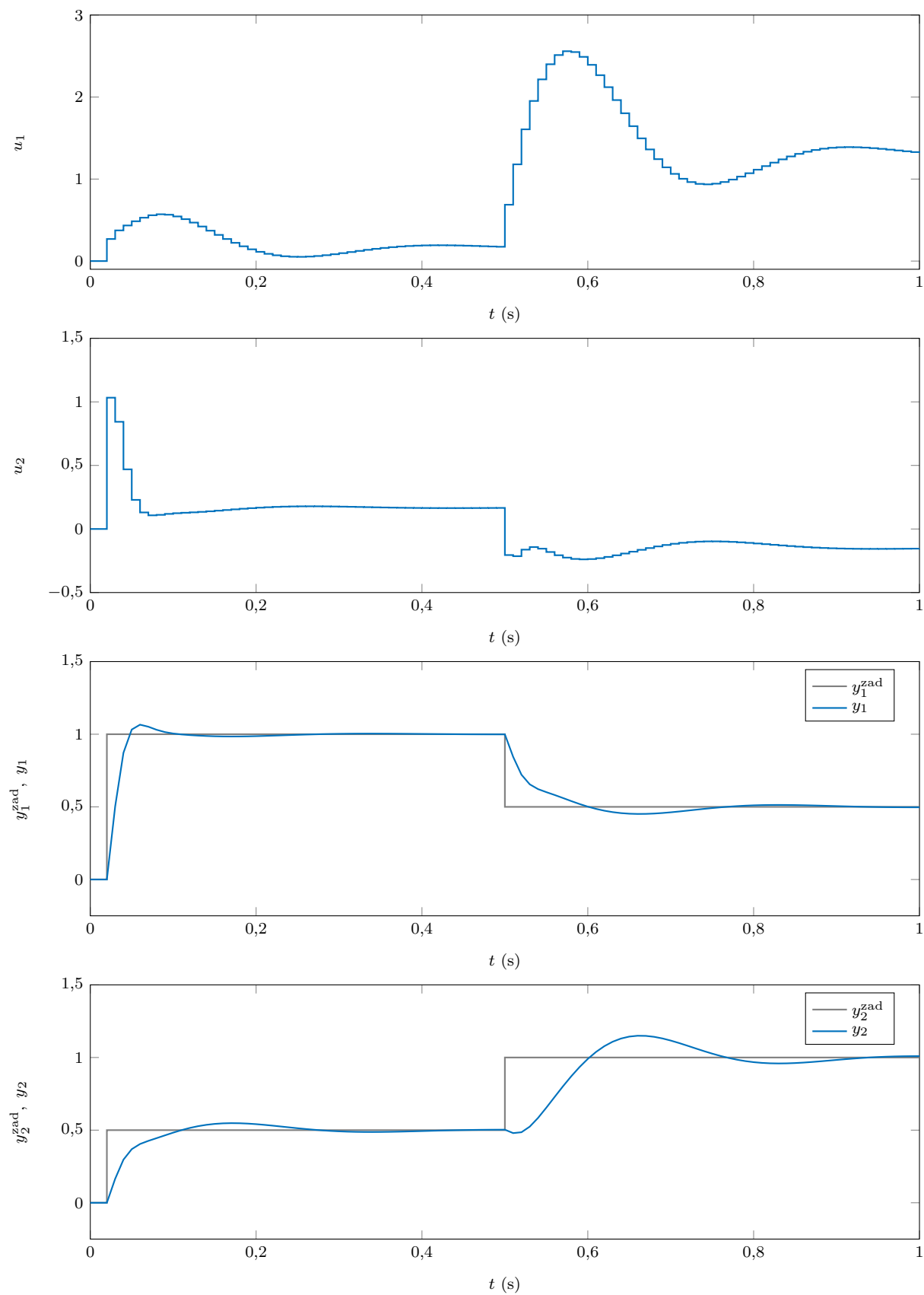
\begin{tikzpicture}
\begin{groupplot}[group style={group size=2 by 2,horizontal sep=\odstepoziomy,
vertical sep=\odstepionowy},width=\szer,height=\wys]
%%1
\nextgroupplot
[xmin=0,xmax=1,ymin=-0.1,ymax=3,
xtick={0,0.2,0.4,0.6,0.8,1},ytick={0,1,2,3},
xlabel=$t \ (\mathrm{s})$,ylabel=$u_1$,legend cell align=left,
legend pos=north east]
\addplot[const plot,color=niebieski,thick]
file {../symulacje22/u1.txt};
%%2
\nextgroupplot
[xmin=0,xmax=1,ymin=-0.25,ymax=1.5,
xtick={0,0.2,0.4,0.6,0.8,1},ytick={0,0.5,1,1.5},
xlabel=$t \ (\mathrm{s})$,ylabel={$y_1^{\mathrm{zad}}$, \ $y_1$},
legend cell align=left,legend pos=south east]
\addplot[const plot,color=gray,thick]
file {../symulacje22/yzad1.txt};
\addplot[color=niebieski,thick]
file {../symulacje22/y1.txt};
\legend{$y_1^{\mathrm{zad}}$, $y_1$}
%%3
\nextgroupplot
[xmin=0,xmax=1,ymin=-0.5,ymax=1.5,
xtick={0,0.2,0.4,0.6,0.8,1},ytick={-0.5,0,0.5,1,1.5},
xlabel=$t \ (\mathrm{s})$,ylabel=$u_2$,legend cell align=left,
legend pos=north east]

```

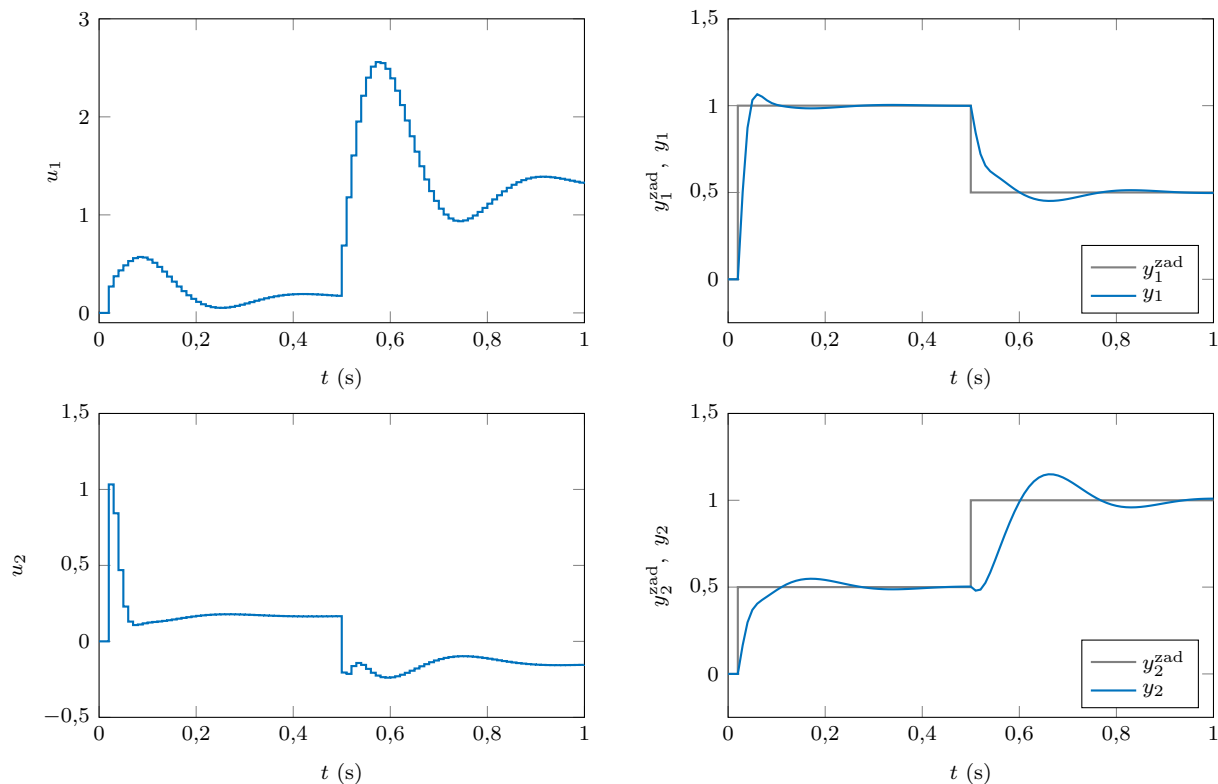
```
\addplot[const plot,color=niebieski,thick]
    file {../symulacje22/u2.txt};
%%4
\nextgroupplot
[xmin=0,xmax=1,ymin=-0.25,ymax=1.5,
    xtick={0,0.2,0.4,0.6,0.8,1},ytick={0,0.5,1,1.5},
    xlabel=$t \ (\mathrm{s})$,ylabel={$y_2^{\mathrm{zad}}$, \ y_2$},
    legend cell align=left,legend pos=south east]
\addplot[const plot,color=gray,thick]
    file {../symulacje22/yzad2.txt};
\addplot[color=niebieski,thick]
    file {../symulacje22/y2.txt};
\legend{$y_2^{\mathrm{zad}}$, $y_2$}
\end{groupplot}
\end{tikzpicture}

\end{document}
```

Rezultat przedstawiono na rys. 4.7.



Rys. 4.6. Przykładowy rysunek wyników symulacji procesu dwuwymiarowego wykonany w języku PGFPLOTS i zapisany w pliku `symulacje22.pdf`



Rys. 4.7. Przykładowy rysunek wyników symulacji procesu dwuwymiarowego wykonany w języku PGFPLOTS i zapisany w pliku `symulacje22_wersja2.pdf`

#### 4.5. Zapis wielu rysunków do jednego pliku pdf

Aktywacja klasy `standalone` z opcją `tikz` umożliwia zapis wielu rysunków do jednego pliku pdf. Rozważmy przykładowy plik `funkcje2.tex` z katalogu `rysunki/zapisz_pdf`:

```
\documentclass[tikz]{standalone}
\usepackage{pgfplots}

\pgfplotsset{
    tick label style={font=\tiny},
    label style={font=\footnotesize},
    legend style={font=\footnotesize},
    title style={font=\footnotesize},
    /pgf/number format/.cd, use comma, 1000 sep={}
}

\newcommand{\szer}{8cm}
\newcommand{\wys}{7cm}

\definecolor{niebieski}{rgb}{0,0.447,0.741}
\definecolor{czerwony}{rgb}{0.85,0.325,0.098}
\definecolor{zolty}{rgb}{0.929,0.694,0.125}
\definecolor{zielony}{rgb}{0.466,0.674,0.188}

\begin{document}

\begin{tikzpicture}
\begin{axis}[
    width=\szer,
    height=\wys,
```

```

        xmin=-10,xmax=10,ymin=-1000,ymax=1000,
        xlabel={x},
        ylabel={f(x), \ g(x)},
        xtick={-10,-5,0,5,10},
        ytick={-1000,-500,0,500,1000},
        legend pos=south east,
    ]

\addplot[niebieski,thick]
    file {../dane_stat/dane_fx.txt};
\addplot[czerwony,thick,densely dashed]
    file {../dane_stat/dane_gx.txt};
\legend{f(x),g(x)}
\end{axis}
\end{tikzpicture}

\begin{tikzpicture}
\begin{axis}[
    width=\szer,
    height=\wys,
    xmin=-10,xmax=10,ymin=-1000,ymax=1000,
    xlabel={x},
    ylabel={f(x), \ g(x)},
    xtick={-10,-5,0,5,10},
    ytick={-1000,-500,0,500,1000},
    legend pos=south east,
]

\addplot[zolty,thick]
    file {../dane_stat/dane_fx.txt};
\addplot[zielony,thick,densely dashed]
    file {../dane_stat/dane_gx.txt};
\legend{f(x),g(x)}
\end{axis}
\end{tikzpicture}

\end{document}

```

Każdy rysunek wygenerowany przez otoczenie `tikzpicture` znajdzie się na osobnej stronie. Co ważne, jeżeli każdy rysunek będzie miał inne wymiary, dzięki klasie `standalone` każda strona jest przycinana do zawartości. Gotowe rysunki wstawiamy do dokumentu poleceniami (oczywiście w otoczeniu `figure`):

```
\includegraphics[page=1]{funkcje2.pdf}
```

oraz

```
\includegraphics[page=2]{funkcje2.pdf}
```

Opisany sposób zapisu rysunków pozwala na ograniczenie liczby plików oraz umożliwia grupowanie treści.

#### 4.6. Prosty eksport wykresów: skrypt `matlab2tikz`

Oprócz opisanego sposobu przygotowywania rysunków z wykorzystaniem pakietu `PGFPLOTS`, można wykorzystać skrypt `matlab2tikz`, który generuje kod źródłowy (TikZ) rysunku wykonanego w programie MATLAB. Skrypt `matlab2tikz` dostępny jest pod adresem: <https://www.mathworks.com/matlabcentral/fileexchange/22022-matlab2tikz-matlab2tikz>. Rozważmy przykładowy plik `wykresy_z_matlaba.m` z katalogu `rysunki/matlab2tikz`:

```
x = 0:.001:pi;
y = sin(2*x.^4);

set(0,'defaulttextinterpreter','latex');

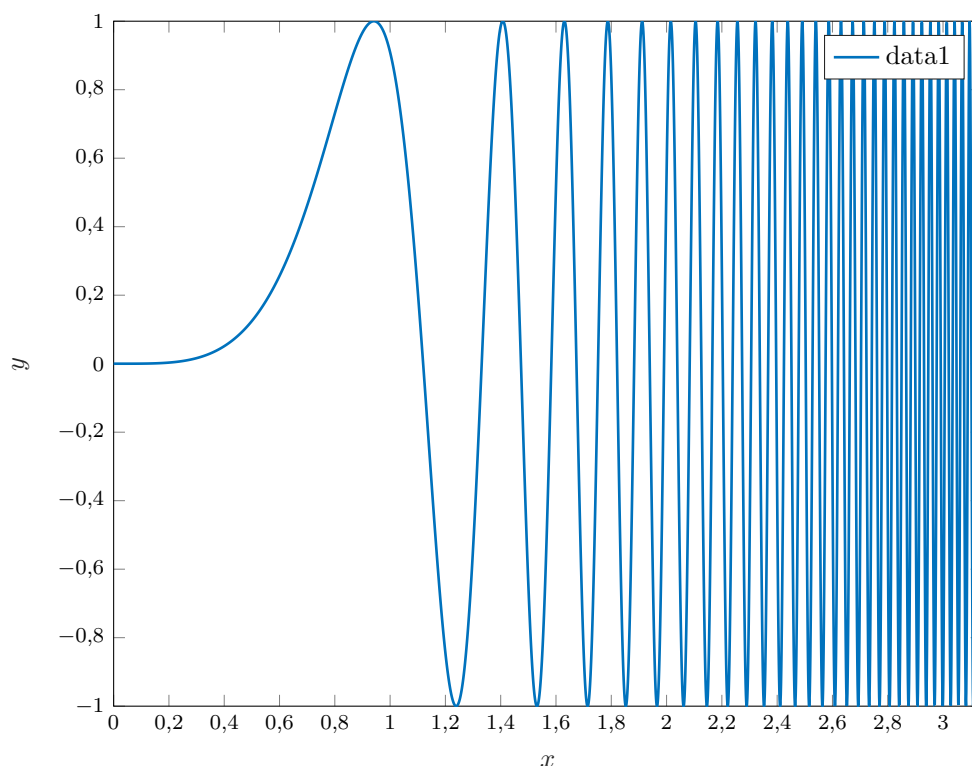
plot(x,y,'linewidth',1);
xlim([min(x) max(x)]);
xlabel('$x$');
ylabel('$y$');
matlab2tikz('wykres_tex.tex','showInfo', false);
```

Wykres wstawiamy do dokumentu w następujący sposób:

```
\begin{figure}[H]
\centering
\input{./rysunki/matlab2tikz/wykres_tex.tex}
\caption{Przykładowy rysunek wykonany w~MATLABie
i~wygenerowany skryptem \texttt{matlab2tikz}}
\label{r_matlab2tikz}
\end{figure}
```

Otrzymany rezultat przedstawiono na rys. 4.8. Oczywiście, czasami konieczna jest ingerencja i drobna modyfikacja plików wygenerowanych przez skrypt `matlab2tikz`, np. w celu zmiany wielkości czcionki lub całego rysunku.

Rysunek jest generowany na podstawie kodu źródłowego przy każdym przetworzeniu dokumentu przez system  $\text{\LaTeX}$ , a więc przy większej liczbie rysunków celowe jest oddzielne przygotowanie poszczególnych rysunków w plikach `pdf`, a następnie ich dołączenie do dokumentu przy użyciu polecenia `\includegraphics`.



Rys. 4.8. Przykładowy rysunek wykonany w MATLABie i wygenerowany skryptem `matlab2tikz`

Pomimo tego, że w MATLAB-ie znakiem oddzielającym część całkowitą od części ułamkowej jest kropka, to w sprawozdaniu zastąpiona ona została przecinkiem. Jest to wynikiem działania jednej z linii znajdujących się w preambule, które na stałe konfiguruje pakiet PGFPlots tak, aby zawsze liczby formatował zgodnie z polskimi standardami.

Zwróćmy uwagę, że rysunek zapisany w pliku źródłowym plik `wykres_tex.tex` jest generowany przy każdym przetworzeniu dokumentu przez system L<sup>A</sup>T<sub>E</sub>X. Aby przyspieszyć działanie systemu, warto osobno wygenerować wszystkie rysunki, a następnie wstawić do dokumentu gotowe pliki pdf. Służy do tego plik `wykres_tex_standalone.tex`, umożliwiając zapisanie rysunku do pliku `wykres_tex_standalone.pdf`. Znajduje się on w katalogu `rysunki/matlab2tikz` i ma następującą postać:

```
\documentclass{standalone}
\usepackage{pgfplots}

\pgfplotsset{
    tick label style={font=\footnotesize},
    label style={font=\footnotesize},
    legend style={font=\footnotesize},
    title style={font=\footnotesize},
    /pgf/number format/.cd, use comma, 1000 sep={}
}

\begin{document}

\centering
\input{wykres_tex.tex}

\end{document}
```

W celu efektywnej pracy ze skryptem `matlab2tikz` należy przenieść wszystkie potrzebne pliki do katalogu, w którym MATLAB poszukuje wywoływanych funkcji. W tym celu należy utworzyć katalog `<MATLAB_ROOT>/toolbox/matlab2tikz` (gdzie `<MATLAB_ROOT>` jest ścieżką do katalogu gdzie zainstalowany jest MATLAB), skopiować doń zawartość katalogu `src`, a następnie dodać powyższy katalog do listy ścieżek MATLAB-a (np. klikając zakładkę `HOME` i przycisk `SetPath` w MATLAB-ie).


**UWAGA:** MATLAB 2017b automatycznie dodaje do wykresów legendę (MATLAB 2017a tego nie robi). Dla zainteresowanych rozwiązaniem tego problemu, link <https://github.com/matlab2tikz/matlab2tikz/issues/1006>

#### 4.6.1. MATLAB i dobór kolorów

Uwaga: nie stosujemy starej palety kolorów pakietu MATLAB, przedstawionej w tab. 4.2.

Tab. 4.2. Stara paleta 7 kolorów pakietu MATLAB (nie używać)

'r'	
'g'	
'b'	
'c'	
'm'	
'y'	
'k'	



Aktualne wersje pakietu MATLAB automatycznie stosują kolejne kolory dla kolejnych poleceń `plot` lub `stairs` z palety `lines`, przedstawione w tab. 4.1. Nie musimy stosować żadnych identyfikatorów kolorów. Co ważne, nie stosujemy identyfikatorów kolorów postaci `'r'`, `'g'`, `'b'`, `'c'`, `'m'`, `'y'`, `'k'`. Rozważmy plik `kolory_matlab.m` z katalogu `rysunki/matlab2tikz`:



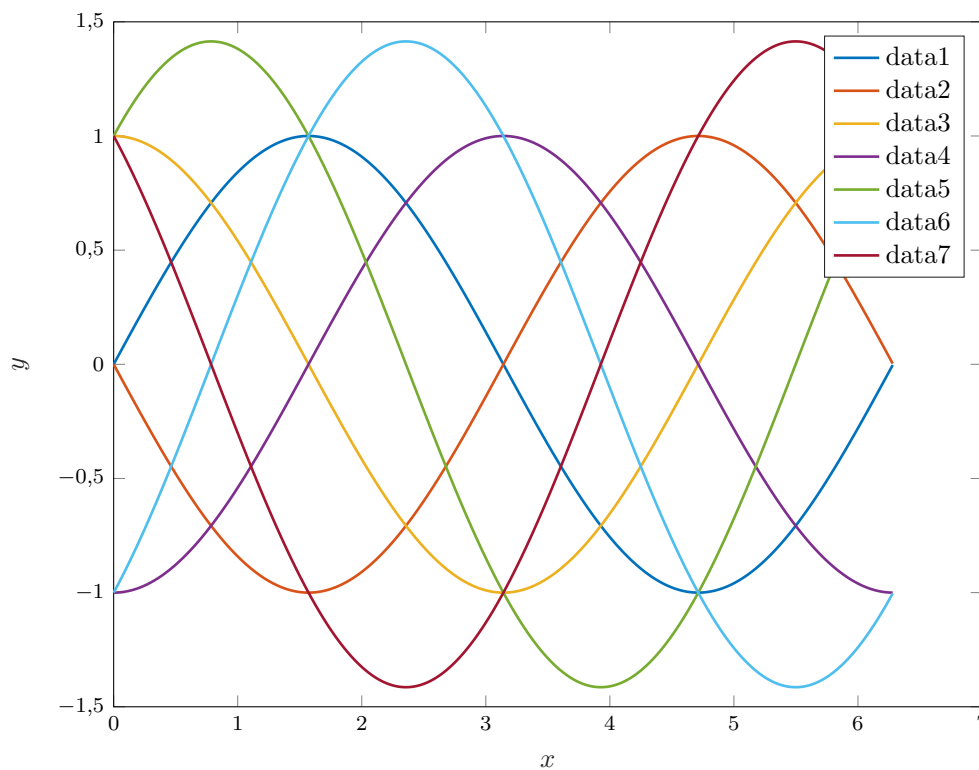
```
x = 0:.01:2*pi;
y1 = sin(x);
y2 = -sin(x);
y3 = cos(x);
y4 = -cos(x);
y5 = sin(x)+cos(x);
y6 = sin(x)-cos(x);
y7 = cos(x)-sin(x);

set(0, 'defaulttextinterpreter', 'latex');
set(0, 'DefaultLineLineWidth', 1);
set(0, 'DefaultStairLineWidth', 1);

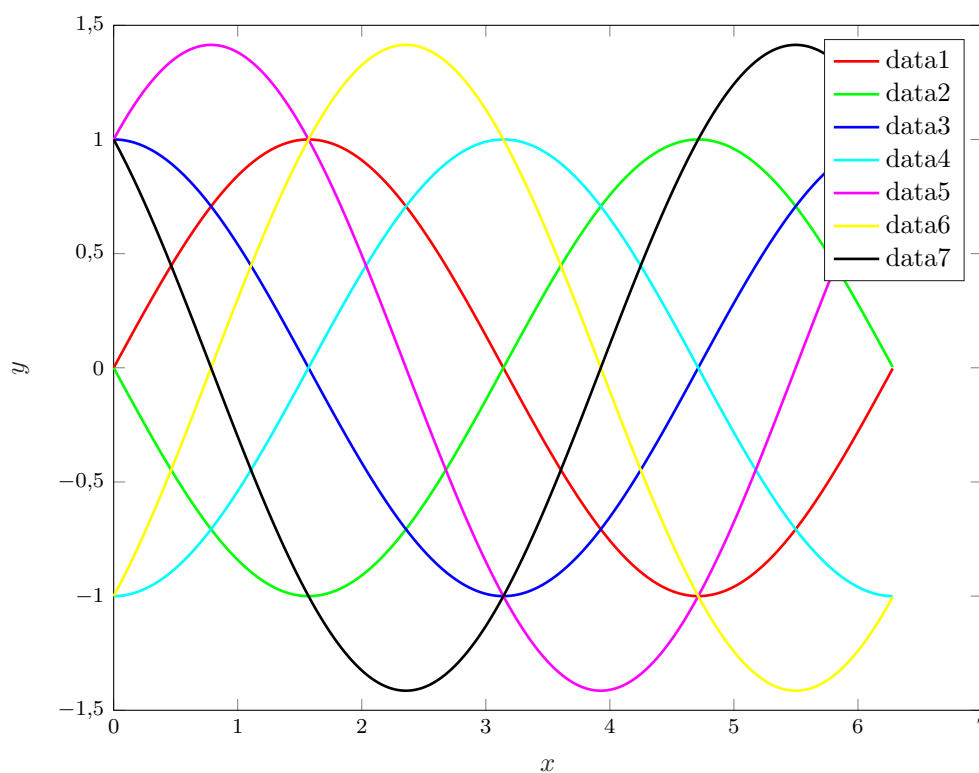
figure;
plot(x, y1);
hold on;
plot(x, y2);
plot(x, y3);
plot(x, y4);
plot(x, y5);
plot(x, y6);
plot(x, y7);
xlabel('$x$');
ylabel('$y$');
matlab2tikz('wykres_kolory1.tex', 'showInfo', false);

figure;
plot(x, y1, 'r');
hold on;
plot(x, y2, 'g');
plot(x, y3, 'b');
plot(x, y4, 'c');
plot(x, y5, 'm');
plot(x, y6, 'y');
plot(x, y7, 'k');
xlabel('$x$');
ylabel('$y$');
matlab2tikz('wykres_kolory2.tex', 'showInfo', false);
```

Powyższy skrypt generuje dwa rysunki, przy czym na pierwszym z nich kolejne kolory są dobierane automatycznie, co osiągamy nie definiując żadnych identyfikatorów kolorów, na drugim rysunku używamy historycznej palety kolorów. Otrzymane rezultaty zamieszczono na rys. 4.9 i rys. 4.10. Kolory z historycznej palety są zbyt jaskrawe, szczególnie zielony i żółty. Reasumując, stosujemy automatyczny przydział kolorów (paleta `lines`).



Rys. 4.9. Przykładowy rysunek wykonany w MATLABie i wygenerowany skryptem `matlab2tikz`: prawidłowy dobór kolorów



Rys. 4.10. Przykładowy rysunek wykonany w MATLABie i wygenerowany skryptem `matlab2tikz`: nieprawidłowy dobór kolorów

#### 4.7. Lokalizacja rysunków i tabel

Zgodnie z filozofia systemu L<sup>A</sup>T<sub>E</sub>X, wszystkie rysunki i tabele powinniśmy umieszczać jako „pływające”, a więc w otoczeniu `figure` oraz `table`, możliwe pozycje: `[t]` – na górze strony, `[b]` – na dole strony, `[p]` – na stronie rysunków/tabel (możliwe są również kombinacje tych opcji). Można również zastosować opcję `[h]` (jeżeli się da, to w tym miejscu). Wymienione opcje działają dobrze, pod warunkiem jednak, że w dokumencie znajduje się „odpowiednia” ilość tekstu, wystarczająca do ładnego ułożenia razem z rysunkami i tabelami. Niestety, jeżeli rysunków i tabel jest znacznie więcej niż tekstu, mogą się znajdować daleko od miejsca, w którym są omawiane. Należy wówczas zastosować opcję `[H]` z pakietu `float` (dokładnie w tym miejscu). Wadą takiego rozwiązania jest prawdopodobieństwo wystąpienia pustych fragmentów stron, ale rysunki i tabele są dokładnie w tym miejscu, w którym wskażemy. Aby uniknąć umieszczenia rysunków w tekście kolejnego rozdziału można również zastosować polecenie `\FloatBarrier` z pakietu `placeins`.

## 5. Listingi programów

Do zamieszczenia programów można zastosować otoczenie `verbatim`, ale znacznie większe możliwości oferuje pakiet `listings`. Przykładowy program w języku C ma postać:

```
//Hello World in C
#include <stdio.h>
int main (void)
{
    puts ("Hello World!");
    return 0;
}
```

natomiast przykładowy program w języku MATLAB jest następujący:

```
%Hello World in MATLAB
clear all;

disp('Hello World!');
```

## 6. Uwagi do wykonywania sprawozdań

### 6.1. Kodowanie znaków

Aktualnie kodowanie znaków ustawione jest na cp1250 (czasem oznaczane jako windows-1250). Wiele programów dokonuje automatycznej analizy dokumentu otwieranego i na podstawie jego zawartości odgaduje jakie kodowanie powinno zostać zastosowane dla danego pliku. O ile, przykładowo, TeXstudio (dobre IDE na początek), nie ma problemów z automatycznym określeniem kodowania dla pliku `sprawozdanie_szablon.tex`, ponieważ rozpoznaje np. zawartą w nim linijkę `\usepackage[cp1250]{inputenc}`, o tyle pozostałe pliki tej linijki nie zawierają, co powoduje często błędnie odgadnięte kodowanie. Aby wskazać dla danego pliku jakie kodowanie ma zostać użyte, można wstawić na początku tego pliku linijkę o treści

```
% !TEX encoding = cp1250
```

która spowoduje wymuszenie użycia tego właśnie kodowania w tym pliku.

Oczywiście ewentualne przejście na kodowanie UTF-8 wymagać będzie:

- zmiany linijki `\usepackage[cp1250]{inputenc}` na `\usepackage[utf8]{inputenc}`,
- zmiany kodowania w poszczególnych plikach z kodowania cp1250 na UTF-8,
- ewentualną zmianę linijek `% !TEX encoding = cp1250` na `% !TEX encoding = utf8`.

Warto na koniec przekompilować wszelkie pliki, aby nie było problemów ze spójnością, a więc w szczególności skasować wszystkie pliki o rozszerzeniu `.aux` i `.pdf`.

### 6.2. Niemerytoryczne powody niepowodzenia

Należy unikać zrzucania winy za niewykonanie zadań na:

- ograniczony czas trwania laboratorium,
- błędy we własnym kodzie, których znalezienie trwało zbyt długo,
- zbyt dużą liczbę zadań do wykonania,
- niedziałający sprzęt,
- zaskakujące działanie programu, którego nikt się nie spodziewał,
- itp.

### 6.3. Styl językowy sprawozdania

Sprawozdanie należy pisać poprawną polszczyzną, bez stosowania żargonu i anglicyzmów. Najlepiej stosować formę bezosobową, tzn. sformułowania typu: zaimplementowano algorytm..., zwiększono/zmniejszono parametr..., na rys. ... przedstawiono..., tabela ... przedstawia ... itp.

### 6.4. Zgodność z polskimi normami dotyczącymi przygotowywania publikacji

Należy koniecznie stosować klasę `mwrep` oraz pakiet `polski`. Umożliwi to przygotowanie dokumentu zgodnie z normami stosowanymi w Polsce. Automatycznie zostaną wówczas reguły dotyczące składu tytułów, akapitów itd. Oczywiście, nawet zalecana klasa i pakiet nie zapobiegą

celowym błędem. Na przykład, nie stosujemy amerykańskiej formy cudzysłowu, tzn. sekwencji znaków “ oraz ”.

Należy unikać samotnych pojedynczych znaków (i, a, o, w) na końcu wiersza. Aby tego uniknąć, stosujemy znak tyldy, który zabrania przejścia do nowej linii. Zamiast w algorytmie, piszemy w~algorytmie.

## 6.5. Używanie zwrotów w cudzysłowie

Potrzeba użycia wyrażenia otoczonego znakami „ oraz ”, sugeruje, że piszący nie zna poprawnego określenia opisywanego zjawiska. W takim wypadku należy zgłębić kłopotliwy temat lub w szczególności poprosić o pomoc prowadzącego.

Jeśli mają Państwo jakieś problemy z oprogramowaniem/sprzętem, to należy to zgłosić prowadzącemu laboratorium. Nie mniej sprzęt jest testowany przed zajęciami – w tym aspekcie problemów być nie powinno.

Odwiedzanie prowadzącego w ramach konsultacji jest wysoce wskazane i zalecane. Warto korzystać także z kontaktu mailowego – wiele prostych problemów można rozwiązać tą drogą.

## 6.6. Separator dziesiętny

Stosowanie języka polskiego wymusza na autorze sprawozdania przestrzeganie podstawowych zasad. Jedną z często ignorowanych reguł jest stosowanie przecinka zamiast kropki do oddzielenia części całkowitej liczby rzeczywistej od jej części ułamkowej. Ma to duży wpływ na estetykę sprawozdania. Przecinki stosujemy konsekwentnie na rysunkach, w tabelach oraz w tekście sprawozdania. Najłatwiej zadbać o poprawne stosowanie separatorów dziesiętnych wykorzystując pakiet `siunitx` oraz polecenie `\num`. Przykłady złego i dobrego zastosowania separatora dziesiętnego przedstawiono w tab. 6.1.

Tab. 6.1. Przykłady złego i dobrego zastosowania separatora dziesiętnego

zapis w L <sup>A</sup> T <sub>E</sub> X	efekt	czy poprawne?
<code>\$12.345\$</code>	12.345	nie
<code>\$12,345\$</code>	12,345	nie
<code>\$\num{12.345}\$</code>	12,345	<b>tak</b>
<code>\$\num{12,345}\$</code>	12,345	<b>tak</b>

Nawyk pisania kropek zamiast przecinków w liczbach rzeczywistych często wywodzi się ze stylu narzuconego przez język programowania i dlatego można go zachować tylko w listingach – w sprawozdaniu i na wykresie koniecznie należy stosować przecinki.

Ciekawe efekty pojawiają się przy wymienianiu różnych wartości parametrów w jednym zdaniu, np. „Do testowania zastosowane zostały okresy próbkowania 0,01, 0,1, 1, 10 oraz 100 sekund.”

## 6.7. Odwołania do rysunków i tabel

Wszystkie rysunki oraz tabele powinny być ponumerowane oraz być odpowiednio podpisane. W tekście sprawozdania powołujemy się na rysunki i tabele podając odpowiednie numery (stosujemy do tego odnośniki, dostępne w systemie L<sup>A</sup>T<sub>E</sub>X). Nie stosujemy określeń typu: na rysunku poniżej/powyżej/3 strony później.

## 6.8. Jednostki

Przykłady złego i dobrego zastosowania jednostek podano w tab. 6.2.

Tab. 6.2. Przykłady złego i dobrego zapisu jednostek

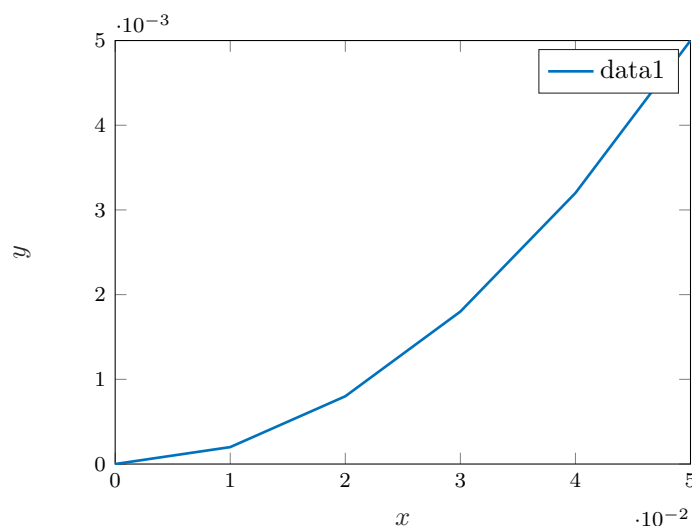
zapis w L <sup>A</sup> T <sub>E</sub> X	efekt	czy poprawne?
<code>\$\num{12.345}\mu s\$</code>	12,345 $\mu s$	nie
<code>\$\SI{12.345}{\micro s}\$</code>	12,345 $\mu s$	<b>tak</b>
<code>\$\num{12.345}\ \mu s\$</code>	12,345 $\mu s$	nie

## 6.9. Wstawianie kodu programu i jego zasadność

Kod programu będącego implementacją algorytmu wykorzystanego do realizacji zadań laboratoryjnych lub projektowych musi być dobrze opisany. Opis kodu może być zawarty w komentarzach będących częścią listingu lub zostać dodany jako osobny fragment tekstu. Podstawą oceny jest sprawozdanie, w związku z czym kod dołączony do sprawozdania w postaci osobnych plików często nie jest czytany – jego treść służy najczęściej do ustalenia samodzielności pracy.

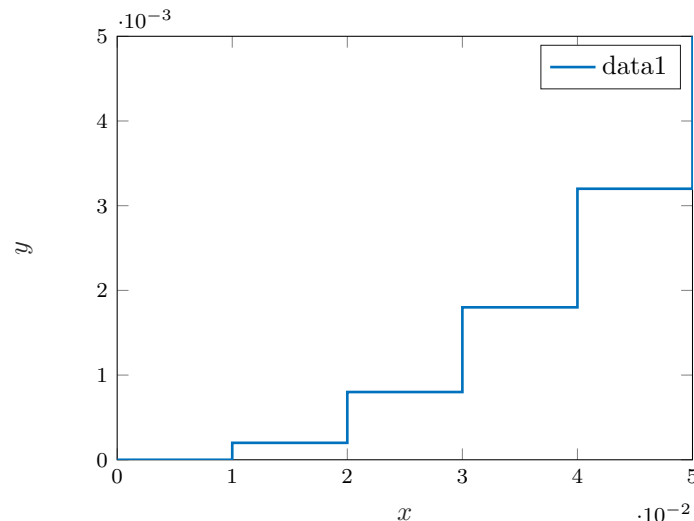
## 6.10. Warunki stosowania `plot` i `stairs`

Polecenie `plot` łączy kolejne punkty odcinkiem, przykładowy wykres podano na rys. 6.1. Wykorzystuje się przy prezentacji wyników, których pomiar dokonywany jest w dyskretnych chwilach, lecz wartości sygnału zmieniają się w sposób ciągły, a nie skokowy. Przykład: wykres przebiegu pozycji obiektu w czasie (ogólnie: wyjścia procesu).



Rys. 6.1. Przykładowy rysunek uzyskany poleceniem `plot`

Polecenie `stairs` łączy kolejne punkty odcinkami równoległymi do osi poziomej, przykładowy wykres podano na rys. 6.2. Wykorzystuje się przy prezentacji wyników, których pomiar dokonywany jest w dyskretnych chwilach, lecz wartości sygnału zmieniają się w sposób skokowy, a nie ciągły. Przykład: wykres sygnału okresowo zmieniającego się generowanego przez sterownik programowalny (ogólnie: sygnały wartości zadanych wyjść oraz sygnały sterujące).

Rys. 6.2. Przykładowy rysunek uzyskany poleceniem `stairs`

### 6.11. Dodatkowe uwagi dotyczące wykresów

Umiejętność prezentacji wyników rzutuje na to jak sprawozdanie zostanie odebrane i w szczególności zrozumiane. W związku z tym:

- aby porównać przebiegi czasowe można wyświetlić wiele przebiegów na jednym wykresie – pod warunkiem, że dotyczą tego samego sygnału lub sygnałów porównywalnych ze sobą (np. różne przebiegi sygnału wyjściowego w zależności od zastosowanego parametru),
- aby porównać przebiegi czasowe można wyświetlić je jeden nad drugim na różnych wykresach – pod warunkiem, że skala czasu jest identyczna, a skala osi pionowej jest różna na obu wykresach.

Kombinacje oraz inne podejścia są oczywiście również dopuszczalne – należy jednak mieć na uwadze, że najistotniejsza jest tutaj czytelność. Czytelnik nie może domyślać się intencji autora – należy założyć, że zawsze domyśli się ich niepoprawnie.

Proszę nie obawiać się, że sprawozdanie będzie zbyt długie – nie są one drukowane, a sprawdzający nie studiuje każdego rysunku z osobna. Zamieszczenie nadmiarowych rysunków często jest bardziej opłacalne niż zaprezentowanie zbyt małej ich liczby – skutkuje to brakiem istotnych informacji (wykresów) w sprawozdaniu. Także jeśli wykresy są do siebie podobne, to warto je zaprezentować – wskazanie skrajnych podobieństw wykresów jest często bardzo istotnym wnioskiem.

### 6.12. Przeprowadzanie eksperymentów

Eksperymenty przed ich przeprowadzeniem powinny zostać zaplanowane. Jeśli etap planowania zostanie pominięty, może zabraknąć czasu na realizację wszystkich zadań w ramach laboratorium. Plan eksperymentu powinien także uwzględniać określenie oczekiwanych rezultatów lub celów, np.:

**Przebieg eksperymentu:** Wykonanie pojedynczej skokowej zmiany wartości sterowania.

**Oczekiwany rezultat:** Pomiary sygnału wyjściowego będące podstawą do wyznaczenia odpowiedzi skokowej. Pomiary powinny ustabilizować się po skończonej liczbie pomiarów.

W sprawozdaniu nie ma potrzeby wypisywania eksperymentów tak jak wyżej, lecz dla usprawnienia własnej pracy warto przygotować sobie takie uproszczone opisy.



### 6.13. Krytyczne podejście do omówienia wyników

Podstawowym celem tych zajęć jest implementacja teoretycznie wspaniałych algorytmów regulacji do rzeczywistych obiektów. Spodziewać się więc należy, że teoria nie zawsze będzie doskonale pokrywała się z praktyką. W takich sytuacjach należy wskazać rozbieżność i wyciągnąć wnioski z czego ona wynika. Ignorowanie zaskakujących rezultatów jest niedopuszczalne.

Wszelkie zadania zawierające słowo „zaimplementuj” wymagają omówienia implementacji, a nie zdania „Zostało zaimplementowane.” Wszelkie zadania zawierające słowo „dobierz” wymagają omówienia sposobu doboru wartości w szczególności ich oceny pozwalającej na porównania ich z pozostałymi wartościami.

### 6.14. Cykliczna archiwizacja i analiza wyników po laboratorium

Warto mieć świadomość z istnienia w MATLAB-ie funkcji o nazwie `save`, która zapisuje wszystkie istniejące w chwili zapisu zmienne do pliku MATLAB-owego. Pozwala to przede wszystkim na lepszą organizację pracy – na laboratorium należy przeprowadzić eksperymenty i zebrać wyniki, a ich analizę warto odłożyć na czas po laboratorium. W szczególności, warto sprawdzić czy uzyskane wyniki pokrywają się z rezultatami osiągniętymi w ramach projektu – oczywiście mowa o zależnościach i relacjach, a nie o dokładnych wartościach.

### 6.15. Omówienie wyników a dobieranie parametrów

Proszę nie poświęcać przesadnie dużo czasu na poszukiwanie idealnych parametrów algorytmu regulacji. Pierwszym powodem jest to, że takich najczęściej nie ma, a drugim jest fakt, iż nie jest tak bardzo istotna jakość regulacji osiągnięta w ramach Państwa poszukiwań, a raczej zastosowana metodologia i ogólnie rozumiane podejście. Bardziej wartościowe jest omówienie ewentualnych dalszych kroków mających na celu znalezienie lepszych parametrów regulacji niż faktyczne ich poszukiwanie. Proszę jednak rozsądnie dobierać liczbę eksperymentów, po której Państwo przerwą poszukiwania – oczywistym jest, że jeden eksperyment to za mało, żeby mówić o jakiegokolwiek tendencji, na podstawie której możnaby wyciągać wnioski o dalszych krokach, natomiast 100 eksperymentów to zdecydowanie za wiele.

## Bibliografia

- [1] C. Feuersänger: Manual for package PGFPLOTS. <ftp://ftp.gust.org.pl/TeX/graphics/pgf/contrib/pgfplots/doc/pgfplots.pdf>, 2016.
- [2] T. Oetiker, H. Partl, I. Hyna, E. Schlegl: Nie za krótkie wprowadzenie do systemu  $\text{\LaTeX} 2_{\epsilon}$ . <ftp://ftp.gust.org.pl/pub/CTAN/info/lshort/polish/lshort2e.pdf>, 2007.
- [3] T. Tantau: The TikZ and PGF Packages Manual for version 3.0.1a. <ftp://ftp.gust.org.pl/TeX/graphics/pgf/base/doc/pgfmanual.pdf>, 2015.
- [4] M. Woliński: Moje Własne CLaSy dokumentów dla  $\text{\LaTeX}$ a 2e <http://marcinwolinski.pl/mwcls.html>, 2013.
- [5] Wikibooks:  $\text{\LaTeX}$ . <https://en.wikibooks.org/wiki/LaTeX>, 2017.