Autorzy: Jaromir Przybyło
Wersja: 13.10.2016

Ćwiczenie A2
Modelowanie i symulacja w programie MATLAB cz.2

Cele laboratorium:

1. Zaznajomienie się ze środowiskiem MATLAB [1]
2. Ćwiczenie umiejętności:
    a. tworzenia wykresów
    b. automatyzacji pracy (m-skrypty) i programowania w środowisku MATLAB.
    c. typy danych środowiska MATLAB

Informacje ogólne

Środowisko MATLAB stało się faktycznym standardem we współczesnych obliczeniach naukowo-technicznych, pozwalającym na przyspieszenie rozwiązywania różnorodnych problemów badawczych oraz inżynierskich. Przyspieszenie to jest osiągane przede wszystkim przez automatyzację rutynowych czynności obejmującą wszystkie fazy rozwiązywania danego zadania: od zbierania danych, poprzez ich analizę i rozwijanie algorytmów, do prezentacji wyników i wdrażania aplikacji. W ćwiczeniu przedstawione zostały wybrane zagadnienia związane z posługiwaniem się środowiskiem MATLAB.

Gdzie szukać dodatkowych informacji:

- Dokumentacja MATLABa
- Internet – słowa kluczowe: MATLAB, http://www.mathworks.com/academia/
- Tadeusiewicz R., Jaworek J., Kańtoch E., Miller J., Pięciak T., Przybyło J., „Wprowadzenie do modelowania systemów biologicznych oraz ich symulacji w środowisku MATLAB”, UMCS Lublin, 2012 http://informatyka.umcs.lublin.pl/files/skrypty2012/tadeusiewicz_wprowadzenie_do_modelowania_systemow_biologicznych.pdf

Przygotowanie

W katalogu wskazanym przez prowadzącego utwórz podkatalog, w którym będą zapisywane wszystkie dane podczas ćwiczenia. Nazwa katalogu powinna zawierać: **<nr ćwiczenia>_<data>_<nazwisko1>_<nazwisko2>**

Zapisz i rozpakuj pliki ćwiczenia (pobrane z MOODLE) do utworzonego katalogu.

[1] *Niniejsza instrukcja opracowana została do wersji R2014b MATLABa. W innych wersjach niektóre elementy mogą się w niewielkim stopniu różnić.*

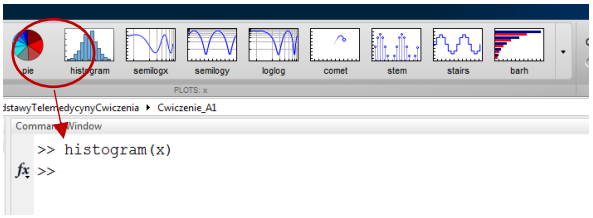
Autorzy: Jaromir Przybyło
Wersja: 13.10.2016

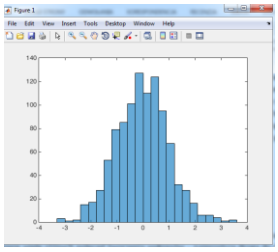
Część I – MATLAB jako środowisko programistyczne (przykłady)

Należy zapoznać się z informacjami z tej części powtarzając podane przykłady

Wizualizacja

Środowisko MATLAB zawiera wiele narzędzi pozwalających na szybkie tworzenie i testowanie algorytmów oraz automatyzację pracy. Niewątpliwie dużą pomocą służą możliwości środowiska ukierunkowane na tworzenie wykresów. W poprzednim ćwiczeniu realizowaliśmy wykresy używając narzędzi MATLABa w sposób interaktywny (tzn. wybierając odpowiednie polecenia z katalogu wykresów). Wykresy mogą być również tworzone w sposób programowy przy pomocy odpowiednich poleceń. MATLAB ułatwia użytkownikowi pracę poprzez różnego rodzaju podpowiedzi – np. wybierając typ wykresu z paska narzędzi automatycznie w oknie poleceń pojawia się odpowiednie polecenie.





Tworzenie wykresów za pomocą poleceń jest wygodniejsze dla zaawansowanych użytkowników. Istnieje wiele różnych poleceń tworzenia wykresów – najprostszą drogą do ich poznania jest użycie katalogu wykresów zawierającego szczegółowe opisy. Jednym z najczęściej używanych poleceń jest komenda `plot`. Ma ona wiele odmian składni:

```
plot(y)        % gdzie y to nazwa zmiennej której
               % wykres chcemy wyświetlić

plot(x,y)      % j.w. ale dla danych x i y
```

W przykładach podanych powyżej pokazano możliwość zaopatrywania poleceń dla MATLABa w **komentarze**. Do tworzenia komentarzy używa się znaku %. Jeśli się go umieści w jakiejś linii, to cały dalszy tekst od tego znaku do końca linii jest przez MATLAB ignorowany, użytkownik może więc umieścić tam dowolne wyjaśnienia i komentarze, które nie będą wpływały na zachowanie komputera, ale będą pozwalały na umieszczenie dowolnych objaśnień i notatek czytelnych tylko dla ludzkiego oka.

Jeśli mówimy o komendzie `plot` to szczególnie przydatna jest jej składnia pozwalająca na zdefiniowanie rodzaju linii, markera oraz koloru dla tworzonego wykresu.

```
plot(x,y,'r') % dodatkowy parametr 'r' określający
  % rodzaj linii i markera oraz kolor
```

Parametr `'r'` definiuje się tekstowo jako złożenie kilku symboli. Symbole te opisane są szczegółowo w dokumentacji dla polecenia `plot`. Jako przykład można utworzyć wykres utworzonej wcześniej zmiennej:

Autorzy: Jaromir Przybyło
Wersja: 13.10.2016

Przykład - programowe tworzenie wykresów oraz adnotacji

```
>> clear all              % czyszczenie przestrzeni
                          % roboczej

>> x=0:pi/180:2*pi;       % tworzenie wektora x
>> y=sin(x)+2*cos(2*x);   % tworzenie wektora y
>> plot(x,y,'-bo')        % tworzenie wykresu


>> xlabel('oś x')         % opis wykresu
>> title('tytuł')
>> legend('wykres 1')
>> ylabel('oś \alpha')
```

W powyższym przykładzie (3.4), parametr `'-bo'` określa linię ciągłą (`-`), kolor niebieski (`b`) oraz marker w kształcie kółka (`o`). Po utworzeniu wykresu istnieje możliwość dostosowania jego wyglądu za pomocą narzędzia "*Plot Tools*" (patrz poniższy rysunek) lub w sposób programowy. W celu programowego dostosowania wykresu należy skorzystać z dokumentacji - często używane polecenia to: `xlabel, ylabel` (opisy osi x i y), `title` (tytuł osi wykresu), `legend` (dodanie legendy). Ciekawą opcją jest możliwość stosowania notacji LATEX w opisach wykresów.

Dostęp do narzędzia „Plot tools" z menu okna wykresu

Bardzo często zachodzi potrzeba aby na jednym oknie wykresu umieścić kilka wykresów. MATLAB oferuje kilka sposób osiągnięcia tego celu. Pierwszym z nich są tzw. „*subplots*", czyli możliwość umieszczenia na jednym oknie wykresu kilku niezależnych osi współrzędnych. W sposób programowy „*subplots*" tworzy się poleceniem:

```
subplot(m,n,k)
```

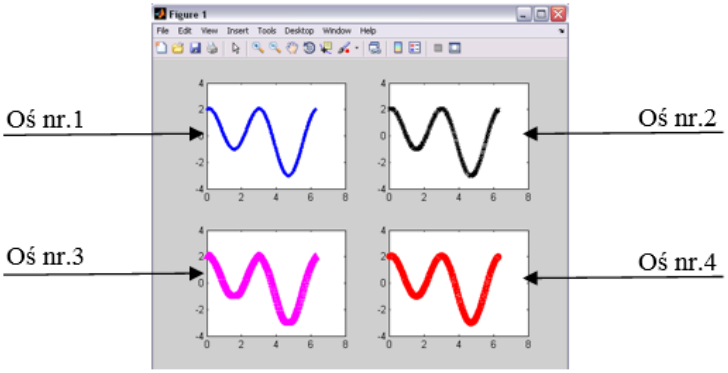
Polecenie to ma 3 argumenty wejściowe:

- `m, n` – oznaczają ilość części na jaką ma być dzielone okno w pionie i poziomie,
- `k` – oznacza numer osi która ma być uaktywniona.

Przykład – tworzenie wykresów (`subplot`)

```
>> x=0:pi/180:2*pi;       % tworzenie wektora x
>> y=sin(x)+2*cos(2*x);   % tworzenie wektora y
>> subplot(2,2,1)         % oś wykresu nr.1
>> plot(x,y,'-b.')
>> subplot(2,2,4)         % oś wykresu nr.4
>> plot(x,y,'-ro')
>> subplot(2,2,2)         % oś wykresu nr.2
>> plot(x,y,'-kx')
>> subplot(2,2,3)         % oś wykresu nr.3
>> plot(x,y,'-m^')
```

Autorzy: Jaromir Przybyło
Wersja: 13.10.2016



Jeżeli zachodzi potrzeba umieszczenia kilku wykresów na tej samej osi, konieczne jest zastosowanie polecenia `hold`. Polecenie to informuje środowisko MATLABa o tym, aby kolejne rezultaty poleceń graficznych były umieszczane na tym samym wykresie. Domyślnie MATLAB usuwa poprzedni wykres.

Przykład – tworzenie wykresów (`hold`)

```
>> figure                % tworzenie okna wykresu
>> x=0:pi/180:2*pi;      % tworzenie wektora x
>> y1=sin(x)+2*cos(2*x); % tworzenie wektora y1
>> y2=sin(2*x)-2*cos(x); % tworzenie wektora y2
>> plot(x,y1,'-b.')
>> plot(x,y2,'-rx')      % nadpisanie wykresu innym

>> figure                % tworzenie okna wykresu
>> plot(x,y1,'-b.')
>> hold on            % blokada nadpisania  wykresu
>> plot(x,y2,'-rx')   % nadpisanie wykresu innym
>> hold off           % wyłączenie blokady
```

Prezentacja wyników na wykresach (bądź też rezultatów numerycznych w linii poleceń) nie kończy pracy z danymi. Bardzo często zachodzi potrzeba wyeksportowania rezultatów i wykresów poza środowisko MATLABa w celu ich dalszej obróbki przy użyciu innych programów. Eksport danych może być zrealizowany poprzez odpowiednie funkcje zapisu danych (`doc fileformats`). Najprostszą z nich jest funkcja `save`, pozwalająca na zapis danych z przestrzeni roboczej środowiska do pliku MAT (binarny format danych MATLABa).

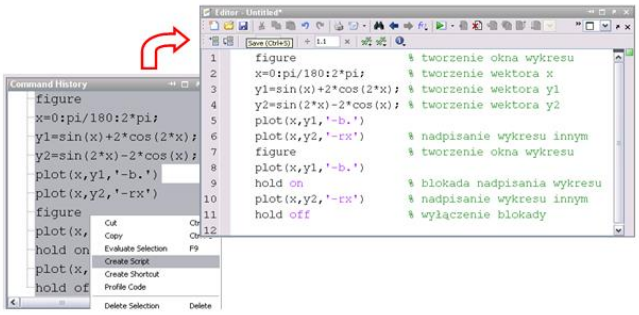
Przykład – różne sposoby programowego eksportu danych

```
>> clear all             % czyszczenie przestrzeni
                         % roboczej

>> x=0:pi/180:2*pi;      % tworzenie wektora x
>> y=sin(x)+2*cos(2*x); % tworzenie wektora y

>> save mojedane x y     % zapis zmiennych do pliku MAT
>> clear all
>> plot(x,y)             % błąd - brak zmiennych
Undefined function or variable 'x'.

>> load mojedane         % wczytanie danych z pliku MAT
>> plot(x,y)

% programowy eksport wykresu do pliku
>> saveas(gcf,'przyklad.tiff','tiffn')
```

Wykresy można również kopiować poprzez schowek lub eksportować przy pomocy narzędzi eksportu (z menu „File" okna wykresu wybieramy polecenie „*Export setup…*" lub „*Save as…*", a następnie wskazujemy określony format pliku).

Programowanie

Użytkownicy środowiska MATLAB bardzo często wymieniają się nie tylko wynikami obliczeń, ale również tzw. M-plikami. M-pliki są to pliki tekstowe z rozszerzeniem `*.m`, zawierające zapisane polecenia MATLABa, pozwalające odtworzyć wykonywane wcześniej obliczenia. Jednym ze sposobów tworzenia M-skryptu jest wykorzystanie historii poleceń („*Command History*"). Po wpisaniu i wykonaniu komend, są one automatycznie zapamiętywane w historii poleceń. Zakładka ta pozwala m.in. na szybkie powtarzanie komend (poprzez zaznaczanie i przeciąganie ich do okna poleceń) oraz na błyskawiczne skopiowanie zaznaczenia do edytora MATLABa (prawy klawisz myszki i wybór z okna opcji *Create Script*). Edytor MATLABa można również otworzyć korzystając z paska narzędzi (*HOME>New>Script*).



Po skopiowaniu lub wpisaniu poleceń w edytorze należy go zapisać na dysku z wybraną nazwą (np. z nazwą `przyklad1.m`). Powstaje wtedy w aktualnym katalogu, tzw. m-skrypt. M-skrypt to inaczej zapamiętane komendy MATLABa wraz z kolejnością ich wykonania, zapisane w pliku tekstowym. Utworzony w ten sposób m-skrypt może zostać uruchomiony poleceniem wskazującym jego nazwę (`>> przyklad1`). Wszystkie zawarte w nim polecenia będę wtedy kolejno wykonywane.

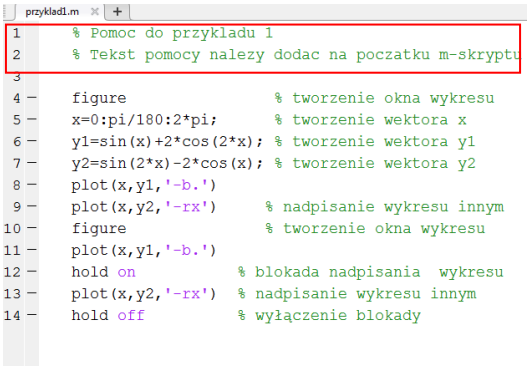
Tworzenie m-skryptów pozwala na automatyzację pracy oraz na wymianę wiedzy między użytkownikami. W skryptach można używać komentarzy. Komentarze zaczynają się od znaku `%` lub `%%`. MATLAB automatycznie koloruje składnię podkreślając komentarze na zielono. Dodawanie takich opisów jest bardzo istotne. Dzięki temu tworzona jest od razu prosta dokumentacja m-skryptu, pozwalająca na jego zrozumienie innym osobom (a także samemu twórcy jeśli zagląda do m-skryptu raz na kilka miesięcy). Szczególne znaczenie mają komentarze w pierwszych liniach M-pliku.

Autorzy: Jaromir Przybyło
Wersja: 13.10.2016

Przykład – tworzenie dokumentacji do m-skryptu

```
>> help przyklad1     % przed dodaniem opisu

No help found for przyklad1.m.
```

Uzupełnienie m-skryptu komentarzem



```
1     % Pomoc do przykladu 1
2     % Tekst pomocy nalezy dodac na poczatku m-skryptu
3
4 -   figure                % tworzenie okna wykresu
5 -   x=0:pi/180:2*pi;      % tworzenie wektora x
6 -   y1=sin(x)+2*cos(2*x); % tworzenie wektora y1
7 -   y2=sin(2*x)-2*cos(x); % tworzenie wektora y2
8 -   plot(x,y1,'-b.')
9 -   plot(x,y2,'-rx')     % nadpisanie wykresu innym
10 -  figure                % tworzenie okna wykresu
11 -  plot(x,y1,'-b.')
12 -  hold on           % blokada nadpisania  wykresu
13 -  plot(x,y2,'-rx')  % nadpisanie wykresu innym
14 -  hold off          % wyłączenie blokady
```

```
>> help przyklad1     % po dodaniu komentarza
  Pomoc do przykładu 1
  Tekst pomocy nalezy dodac na początku m-skryptu
```

Dodawanie komentarzy do m-skryptu ma również inne zastosowanie. Są one wykorzystywane przez narzędzie do generacji raportów. Komentarz oznaczony podwójnym znakiem procentu (`%%`) ma specjalne znaczenie – tworzy tzw. **sekcje kodu** umożliwiające wykonywanie fragmentów m-skryptu. Naciśnięcie przycisku „*Publish*" spowoduje automatyczne uruchomienie m-skryptu, zebranie wszystkich wyników i wygenerowanie raportu w domyślnym formacie (HTML). Możliwe są również inne formaty (DOC, PPT, TeX, PDF…).

Oprócz możliwości automatyzacji pracy przy pomocy m-skryptów, MATLAB pozwala na tworzenie całych aplikacji. Jest to możliwe dzięki efektywnemu językowi programowania osadzonemu w tym środowisku, licznym modułom rozszerzającym oraz wielu narzędziom pomocniczym. Język programowania MATLABa jest językiem wysokiego poziomu, zapewniającym między innymi automatyczne zarządzanie dynamiczną alokacją pamięci dla zmiennych (ang. *garbage collection*). Mimo iż powszechnie uważa się że jest to język interpretowany, to faktycznie środowisko zapewnia automatyczną analizę składni kodu w tle. Przyspiesza to w dużym stopniu wykonywanie M-kodu.
Szczegółowe omówienie składni i poleceń języka MATLABa znajduje się w dokumentacji oprogramowania zawierającej obszerne przykłady. Przydatne będzie zapoznanie się z instrukcjami warunkowymi (`if else`, `switch case`), pętlami (`for`, `while`), poleceniami wyświetlania danych (`disp`, `sprintf`) oraz modularnymi okienkami dialogowymi.

Przykład – zapoznanie się z wybranymi poleceniami MATLABa

```
>> doc switch          % zwróć uwagę na działanie tej
                       % instrukcji (np. w odróżnieniu od
                       % języka C, po spełnieniu danego
                       % warunku, kolejne nie są
                       % sprawdzane)
```

```
>> doc sprintf          % zwróć uwagę na składnię polecenia
                        % określającego sposób formatowania
                        % wyświetlanych liczb np. %025d

>> doc predefined dialog boxes % okienka dialogowe, zwróć uwagę na
                               % uigetfile
```

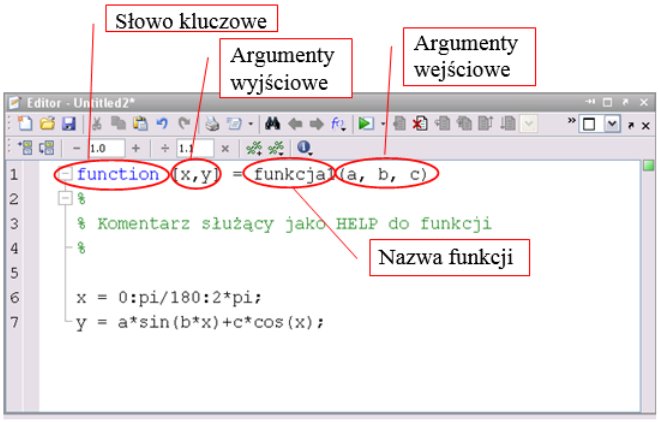
Do tworzenia aplikacji przydatnym elementem środowiska MATLAB jest możliwość tworzenia m-funkcji. Co to są m-funkcje? Podobnie jak m-skrypty, są to pliki tekstowe zawierające polecenia MATLABa. To co odróżnia m-funkcje od m-skryptów to m.in.: składania oraz sposób zapamiętywania i dostępu do zmiennych w pamięci. Podczas uruchamiania m-skryptów, wszystkie zmienne są zapamiętywane w tzw. głównej przestrzeni roboczej MATLABa. Może to w pewnych przypadkach rodzić problemy, związane z nadpisywaniem zmiennych innymi wartościami Problem ten ilustruje poniższy przykład.

Przykład problemu nadpisywania zmiennej

```
>> clear all             % czyszczenie głównej
                         % przestrzeni roboczej

>> W([1 2 3])=[2 4 6]    % przypisanie wektora
W =

     2     4     6

>> W = 'abc'             % nadpisanie zmiennej
                         % innym typem danych
W =

abc

>> W([1 2 3])=[72 74 76] % ponowne przypisanie,
                         % wynik to
                         % nieoczekiwany rezultat
W =

HJL

>> class(W)
ans =

char
```

M-funkcje, w odróżnieniu od m-skryptów, posiadają osobną przestrzeń roboczą – niezależną od głównej przestrzeni roboczej. Wymiana danych między przestrzeniami roboczymi odbywa się poprzez argumenty wejściowe i wyjściowe funkcji. Stąd wynika inna składnia M-pliku funkcyjnego. Przy zapisywaniu pliku istotne jest aby nazwa M-pliku odpowiadała nazwie funkcji.
Utwórz przykładową funkcję (patrz poniższy rysunek) i zapisz ją do pliku `funkcja1.m`

Autorzy: Jaromir Przybyło
Wersja: 13.10.2016

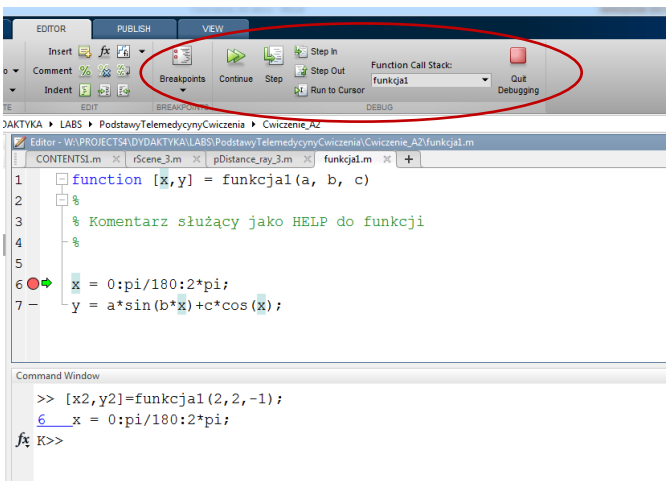


Uruchomienie m-funkcji odbywa się podobnie jak m-skryptu, poprzez wpisanie nazwy funkcji w oknie poleceń. Jedyną różnicą jest konieczność podania argumentów wejściowych i wyjściowych (zgodnie z definicją funkcji). Wszystkie zmienne używane wewnątrz funkcji są lokalne, tzn. istnieją tylko w czasie wywołania funkcji w jej lokalnej przestrzeni roboczej.

Przykład wywołania m-funkcji

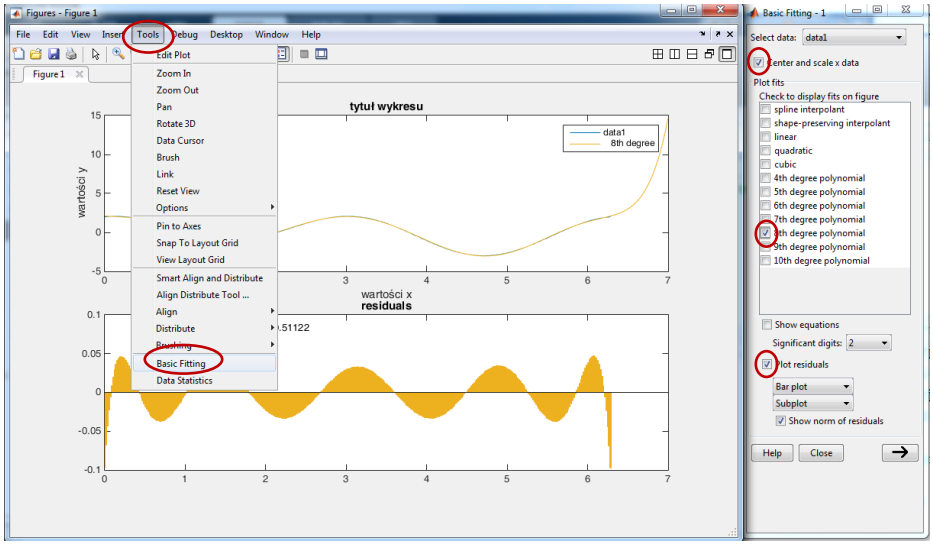
```
>> clear all
>> [x1,y1]=funkcja1(4,3,1);
>> subplot(2,1,1);plot(x1,y1)

>> [x2,y2]=funkcja1(2,2,-1);
>> subplot(2,1,2);plot(x2,y2)

>> x              % błąd - zmienna x nie istnieje
                  % w głównej przestrzeni roboczej
Undefined function or variable 'x'.
```

Jak każde środowisko programistyczne, MATLAB posiada narzędzia pozwalające na śledzenie wykonywania M-kodu (ang. *debugger*). Przy jego pomocy możliwe jest m.in. ustawianie pułapek (ang. *breakpoints*), krokowe wykonywanie kodu oraz podgląd i modyfikacja wartości zmiennych lokalnych w czasie wykonywania M-kodu. Okno debuggera zostało przedstawione na poniższym rysunku (uwaga – poniższe opcje debbuggera pojawiają się dopiero w momencie zaznaczenia pułapki i uruchomienia funkcji). Posługiwanie się tym narzędziem zostało dokładnie omówione w dokumentacji oprogramowania MATLAB.



Autorzy: Jaromir Przybyło
Wersja: 13.10.2016

Środowisko MATLAB wyposażone jest również w wiele innych narzędzi, wspomagających użytkownika w realizacji typowych czynności (import danych, wizualizacja, analiza…). Użytecznym narzędziem jest *Basic Fitting Tool* – pozwalający na dopasowanie krzywej parametrycznej (np. wielomian n-go stopnia) do danych wykresu. Narzędzie to jest dostępne z menu wykresu *Tools>Basic Fitting*. Proszę zwrócić uwagę, iż rezultatem takiego dopasowania jest zbiór parametrów (w tym przypadku parametry wielomianu N zmiennych), czyli inaczej mówiąc **model matematyczny** obserwowanego procesu lub obiektu. Mając taki model możemy dokonać jego **symulacji**.
Zapoznaj się z narzędziem tworząc pojedynczy wykres plot z poprzedniego przykładu, a następnie dopasowując do danych krzywą wielomianową (nie zapomnij o wyskalowaniu danych). Na osobnym wykresie `subplot` zwizualizuj błąd dopasowania.



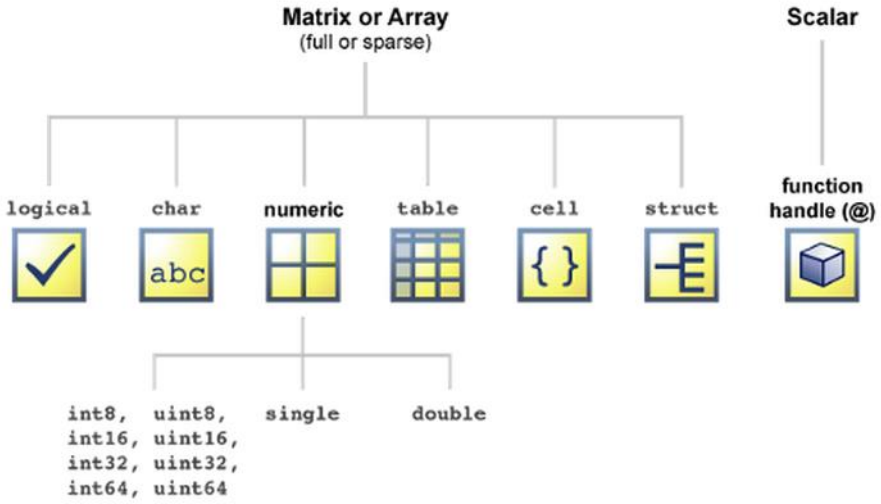
Środowisko MATLAB posiada możliwość automatycznej generacji m-kodu z narzędzi MATLABa (*import wizard, plot tools, basic fitting tool…*). Przyspiesza to przejście z pracy interaktywnej do programowania (tworzenie m-skryptów i m-funkcji). Jako przykład można wygenerować m-kod z wcześniej utworzonego dopasowania krzywej do danych. W tym celu w oknie wykresu należy wybrać File>Generate Code. Utworzona zostanie m-funkcja `createfigure` akceptująca (w tym przypadku) dwa argumenty wejściowe i odtwarzająca w sposób programowy całość analizy danych i tworzenia wykresu. W trakcie zapoznawania się z wygenerowanym kodem, należy zwrócić uwagę na następujące funkcje: `polyfit`, `polyval`.

Przykład uruchomienia wygenerowanej m-funkcji
`>> load mojedane` `>> createfigure(x, y)`

Autorzy: Jaromir Przybyło
Wersja: 13.10.2016

Część II – MATLAB typy danych

Na poniższym rysunku przedstawione są typy danych, występujące w środowisku MATLAB



Rys. II-1 Typy danych środowiska MATLAB (źródło: dokumentacja MATLABa)

Podstawowym typem danych jest typ danych numeryczny typu **double** (liczby zmiennoprzecinkowe podwójnej precyzji). Oprócz tego występują typy danych numerycznych **single** (pojedynczej precyzji) oraz typy stałoprzecinkowe (**uint**). Do zapamiętywania znaków i tekstów służy typ **char** – jest to tablica znakowa tzn. każdy element tablicy zawiera kod ASCII litery lub cyfry. Typ danych **logical** pozwala na zapamiętanie wartości logicznych i powstaje np. po wykonaniu operatorów porównania. Jest wykorzystywany m.in. do indeksowania logicznego.
Przydatnym typem danych są tablice komórkowe oraz struktury. Tablice komórkowe definiujemy podobnie jak zwykłe tablice, przy czym zamiast nawiasów prostokątnych **[]** stosujemy nawiasy klamrowe **{}**.

Przykład – używanie tablic komórkowych

```
>> C = {2 rand(3);pi 'abc'}

C =

  2×2 cell array

    [     2]    [3×3 double]
    [3.1416]    'abc'
>> C{1,2}

ans =

    0.7922    0.0357    0.6787
    0.9595    0.8491    0.7577
    0.6557    0.9340    0.7431
```

```
>> mojastruktura.pole1 = 'a'

mojastruktura =

  struct with fields:

    pole1: 'a'

>> mojastruktura.pole2 = 123

mojastruktura =

  struct with fields:

    pole1: 'a'
    pole2: 123
```

Innym typem danych są uchwyty do funkcji (*function handle*), pozwalające na zapamiętywanie referencji do funkcji. Przykład ich użycia znajduje się poniżej.

Przykład – uchwyty do funkcji

```
>> f1 = @(x) x.^2 - 2*x
f1 =

  function_handle with value:

    @(x)x.^2-2*x
>> f1([3 4 5])
ans =

     3     8    15
```

Oprócz standardowych typów danych, istnieją inne. Przykładem mogą być dane tabelaryczne (typ `table`). Uwaga – niektóre typy danych nie są dostępne w starszych wersjach MATLABa.

Przykład – dane tabelaryczne

```
>> Imie = {'Rafał';'Monika';'Paweł';'Elżbieta';'Mirek'};
>> Wiek = [38;43;38;40;49];
>> Wzrost = [71;69;64;67;64];
>> Waga = [176;163;131;133;119];

>> T = table(Wiek,Wzrost,Waga,'RowNames',Imie)

T =

                Wiek    Wzrost    Waga
                ____    ______    ____

    Rafał       38      71        176
    Monika      43      69        163
    Paweł       38      64        131
    Elżbieta    40      67        133
    Mirek       49      64        119
```

Oprócz tego MATLAB udostępnia możliwość tworzenia własnych typów danych, dzięki możliwościom programowania obiektowego. Więcej informacji na temat rodzajów danych można znaleźć w dokumentacji MATLABa.

Autorzy: Jaromir Przybyło
Wersja: 13.10.2016

Część III – MATLAB jako środowisko programistyczne (ćwiczenia)

Ćwiczenia należy zrealizować a ich rozwiązania umieścić w raporcie:

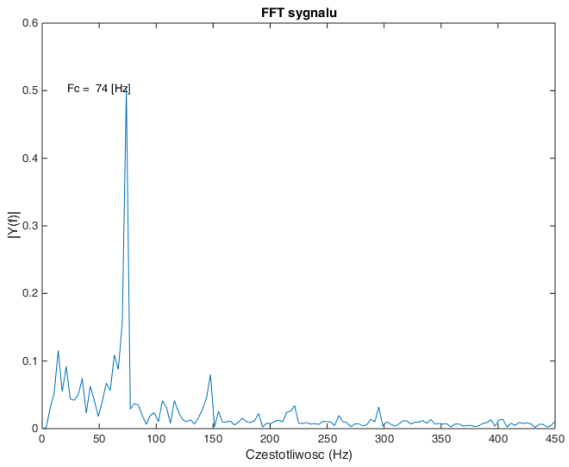
a) Utwórz m-skrypt wczytujący dane z pliku `daneP.csv` a następnie realizującego wizualizację danych na wykresie typu `plot` (każda zmienna na osobnym wykresie). Dodaj do każdego wykresu osobny tytuł.
   - Wskazówka: użyj *import wizarda* do importu danych, następnie wygeneruj m-funkcję do importu danych i wykorzystaj ją w swoim m-skrypcie

b) Z zaimportowanych danych wybierz taki fragment, na którym widoczny jest sygnał okresowy. Utwórz nową zmienną zawierającą wybrany fragment sygnału i zwizualizuj ją na osobnym oknie wykresu.
   - Wskazówka: Przyglądnij się wykresowi każdej składowej i wybierz taką, na której okresowość jest najlepiej widoczna. Następnie, przy pomocy indeksowania, wybierz fragment pomijając zakłócenia na początku i na końcu sygnału.

c) Z wybranego fragmentu sygnału, usuń trend poprzez dopasowanie krzywej wielomianowej. Dobierz stopień wielomianu jak najmniejszego stopnia przy zachowaniu jak najmniejszego błędu dopasowania (funkcja `norm`). Do wykresu z poprzedniego punktu dodaj linię dopasowanego trendu (inny kolor). Utwórz nowy wykres zawierający sygnał z usuniętym trendem.
   - Wskazówka – możesz zrealizować dopasowanie przy pomocy narzędzia *Basic Fitting Tool*, a następnie wygenerować m-kod i wybrać z niego potrzebne fragmenty analizy sygnału.

d) Znajdź częstotliwość charakterystyczną sygnału. Sformatuj wykres dodając do niego w sposób programowy: opisy osi x,y i tytuł wykresu. Dodaj do wykresu punkt w miejscu maksimum (częstotliwość charakterystyczna) oraz opis informujący o wartości częstotliwości tego maksimum.
   - Wskazówka: użyj funkcji `fft` (patrz przykład w dokumentacji) z częstotliwością próbkowania odczytaną z pliku `daneP.csv`
   - Aby dodać do wykresu opis wykorzystaj funkcję `text` oraz sformatuj tekst przy pomocy funkcji `sprintf`.

Gotowy rezultat powinien wyglądać podobnie jak wykres przedstawiony poniżej.



e) Uzupełnij m-plik o komentarze i dokumentacje. Wygeneruj raport w formacie HTML

Autorzy: Jaromir Przybyło
Wersja: 13.10.2016

Część IV – MATLAB typy danych (ćwiczenia)

Ćwiczenia należy zrealizować a ich rozwiązania umieścić w raporcie:

a) Przy pomocy polecenia `randn` wygeneruj tablicę 3x3 liczb pseudolosowych **R** o rozkładzie normalnym (średnia 0 i odchylenie standardowe). Następnie utwórz zmienną **A** jako typ UINT32, zawierającą liczbę 100. Pomnóż zmienną **R** przez **A**, odpowiednio dostosowując typy danych. Rezultat (zmienna **B**) powinna być typu UINT32. Zwróć uwagę czy rezultaty mnożenia są poprawne ! W sprawozdaniu zanotuj liczbę bajtów potrzebną do zapamiętania jednej liczby typu **double** oraz jednej liczby typu UINT32 (wskazówka – skorzystaj z polecenie `whos`).

b) Utwórz dwie tablice znakowe zawierające teksty: „ćwiczenie 2” oraz „laboratorium 1”. Połącz te dwie tablice tak aby tablica wynikowa zawierała tekst jak poniżej (wskazówka – skorzystaj z polecenia `strvcat`)

```
str3 =

ćwiczenie 2
laboratorium 1
```

c) Utwórz tablicę znakową `str1` zawierającą tekst „*Krasnoludy przeszły przez rzekę w bród, nie zamoczywszy swych bród i do tego zmywszy ze swych nóg brud*”. Znajdź indeksy słów zaczynających się na literę „b”, kończących na literę „d” i nie zawierających litery „u”. Wskazówka – skorzystaj z wyrażeń regularnych – dokumentacja do polecenia `regexp`.

d) Utwórz tablicę komórkową o rozmiarze 2x2 zawierającą następujące dane jak na rysunku poniżej. Wybierz z tablicy komórkowej, tablicę liczb pseudolosowych znajdującą się w komórce 2-wiersz, 1-kolumna, dodaj do niej wartość 100, a rezultat zapisz w to samo miejsce do tablicy komórkowej.

Liczba 123	Tekst ‘abcd’
Tablica liczb losowych 3x3	Liczba 0.1

e) Oblicz całkę oznaczoną w przedziale $x \in (-2,2)$ z funkcji $f(x) = x^2 - 2 \cdot x + 4$ i narysuj jej wykres dla tego przedziału. Wskazówka – zdefiniuj funkcję przy pomocy uchwytu do funkcji, wykorzystaj funkcję `quad` oraz `fplot`.

f) Utwórz typ danych tabelaryczny (`table`) zawierający dane jak na rysunku poniżej. Wyeksportuj dane z tabeli do pliku CSV. Wskazówka – sprawdź w systemie pomocy jak definiować nazwy kolumn oraz wierszy. Liczby do tabeli wygeneruj losowo. Skorzystaj z polecenia `writetable`

```
T =
```

	Matematyka	Fizyka	Chemia
Rafał	36	65	30
Monika	83	74	75
Paweł	2	65	19
Elżbieta	5	46	69
Mirek	17	55	19

Autorzy: Jaromir Przybyło
Wersja: 13.10.2016

Raport z ćwiczenia nr A2

Data:
Imię i nazwisko:
Imię i nazwisko:

Zamieść w raporcie polecenia z realizacji z części III i IV