Informatyka

Wykład 2

Witold Dyrka
witold.dyrka@pwr.wroc.pl

5/3/2012

Program wykładów

0. Informatyka. Wprowadzenie do Matlaba (13.02.12)
1. Matlab dla programistów C/C++ (20.02.12)
2. **Optymalizacja obliczeń. Grafika** (05.03.12)
3. Programowanie zorientowane obiektowo (19.03.12)
4. Programowanie zorientowane obiektowe w praktyce (02.04.12)
5. Graficzny interfejs użytkownika (16.04.12)
6. Obliczenia numeryczne (30.04.12)
7. Kolokwium (14.05.12)

Dzisiejszy wykład w oparciu o...

- MATLAB Product Documentation. Techniques for Improving Performance
  http://www.mathworks.com/help/techdoc/matlab_prog/f8-784135.html
- S. McGarrity. Programming Patterns: Maximizing Code Performance by Optimizing Memory Access. The MathWorks News & Notes - June 2007
  http://www.mathworks.com/company/newsletters/news_notes/june07/patterns.html
- MATLAB Product Support. 1109 - Code Vectorization Guide.
  http://www.mathworks.com/support/tech-notes/1100/1109.html
- The Mathworks. Accelerating MATLAB (*dot. wersji 6.5*)
  http://www.ee.columbia.edu/~marios/matlab/accel_matlab.pdf
- P. Getreuer. Writing Fast Matlab Code (*dot. wersji R2008b*)
  http://www.sal.ufl.edu/NewComers/matlab_optimization_2.pdf
- B. Mrozek, Z. Mrozek. MATLAB i Simulink. Poradnik użytkownika. Wydanie III. Helion 2010. Rozdział 4.
- M. Kotulska. Informatyka – wykład 3: Wykresy i grafika w Matlabie. PWr.
  www.if.pwr.wroc.pl/~kotulska/informatyka/info3.ppt

Program na dziś (1)

- Optymalizacja kodu
  - prealokacja tablic
  - indeksowanie
  - wektoryzacja
  - akcelerator *Just-In-Time*

- Grafika

Prealokacja tablic

- Matlab potrafi automatycznie powiększać tablice

```
>> X = 1
X =     1

>> X(2,10) = 2
X =  1     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     2
```

  - jeśli zarezerwowany na tablicę obszar był za mały
    - musiał zarezerwować większy obszar i przenieść tam dane
      to zajmuje czas!
  - jeśli sytuacja powtarza się w pętli
    - staje się to krytyczne!!

- Lepiej od razu zarezerwować całą pamięć
  - funkcja `zeros`

Prealokacja tablic – test

```
t=zeros(1,1000);            % tablica przechowująca kolejne czasy obliczeń
for k=1:1000                % 1000 powtórzeń eksperymentu
    clear b                 % usuwamy b z przestrzeni roboczej
    tic                     % czas START!
    for i=1:30000           % testowany kod: pętla bez prealokacji
        b(i) = i^2;
    end
    t(k) = toc;             % czas STOP!
end
mean(t)                     % średni czas wykonania pętli wewnętrznej
                            % R2007b: 600ms    R2011b: 11.0ms

t2=zeros(1,1000);           % tablica przechowująca kolejne czasy obliczeń
for k=1:1000                % 1000 powtórzeń eksperymentu
    clear b                 % usuwamy b z przestrzeni roboczej
    tic                     % czas START!
    b=zeros(1,30000);       % testowany kod: pętla z prealokacją
    for i=1:30000
        b(i) = i^2;
    end
    t2(k) = toc;            % czas STOP!
end
mean(t2)                    % średni czas wykonania pętli wewnętrznej
                            % R2007b: 0.250ms  R2011b: 0.225ms

mean(t)/mean(t2)            % przyspieszenie
                            % R2007b: 2400x    R2011b: 49x
```

Dostęp do kolumn jest szybszy

- Nowoczesne procesory przechowują ostatnio używane fragmenty pamięci RAM w szybszej pamięci cache
  - MATLAB umieszcza elementy w kolumnach tablic w ciągłym bloku pamięci
    - przy pierwszym dostępie do elementu kolumny, wszystkie elementy mogą znaleźć się w cache'u
  - **Wniosek**
    - iterację po tablicy wielowymiarowej zaczynaj od kolumn

```
N=2e3;
x=randn(N);
y=zeros(N);
for w=1:N        % wiersze
    for k=1:N    % kolumny
        if x(w,k)>=0
            y(w,k)=x(w,k);
        end
    end
end
```

0.19 sek.

```
N=2e3;
x=randn(N);
y=zeros(N);
for k=1:N        % kolumny
    for w=1:N    % wiersze
        if x(w,k)>=0
            y(w,k)=x(w,k);
        end
    end
end
```

0.14 sek.

Indeksowanie tablic (1)

- **wiersz-kolumna** (*subscripts*)
  - parametrem jest adres elementu/ów, np. `A(2,3), B(6,:,2:5)`

```
A =  11    14    17
     12    15    18
     13    16    19

>> A(2,3)       % wiersz nr 2, kolumna nr 3
ans = 18

>> A(2:3,1:2)
ans =   12    15
        13    16
```

Indeksowanie tablic (2)

- **liniowe**
  - parametrem jest numer kolejny elementu, np. `A(8)`

```
A =  11   14   17
     12   15   18
     13   16   19

>> A(8)          % element nr 8
ans = 18

>> A(1:3)
ans = 11    12    13

>> A([4;5;6])
ans = 14        % kształt zwracanego wektora wartości
      15        % zależy od kształtu wektora indeksów
      16
```

Indeksowanie tablic (3)

- **logiczne**
  - parametrem jest tablica zero-jedynkowa
    - o tym samym wymiarze co tablica indeksowana

```
A =  11    14    17
     12    15    18
     13    16    19

>> ind = mod(A,2)==0        % tworzy macierz logiczną
ind =  0     1     0        % true (1) - gdy A%2==0
       1     0     1        % false(0) - gdy A%2!=0
       0     1     0        %

>> A(ind)                   % zwraca elementy A,
ans =  12                   % na miejscu których
       14                   % w ind jest 1
       16                   % (parzyste)
       18
```

Użyteczne techniki

- Usuwanie wierszy i kolumn

```
A =  11    14    17
     12    15    18
     13    16    19
```

```
>> A(:,2) = []
A =  11    17
     12    18
     13    19
>> A(1:2,:)
A =  13    19
```

- Replikacja skalara / wektora (tzw. trik Tony'ego)

```
>> c = 5
c = 5
>> c = c(ones(2,3))
c =  5     5     5
     5     5     5
```

```
>> d=(1:3)'
d =  1
     2
     3
>> d = d(:,ones(6,1))
d =  1 1 1 1 1 1
     2 2 2 2 2 2
     3 3 3 3 3 3
```

- Replikacja macierzy

```
>> B = repmat(eye(2),2,4)
B =  1     0     1     0     1     0     1     0
     0     1     0     1     0     1     0     1
     1     0     1     0     1     0     1     0
     0     1     0     1     0     1     0     1
```

Wektoryzacja

- MATLAB został napisany do wykonywania działań na macierzach
  - posiada wbudowane funkcje obliczeń na macierzach
    - są szybkie, bo napisane w Fortranie i C oraz skompilowane
  - w przeciwieństwie do funkcji wbudowanych, M-funkcje (takie jakie piszemy na laboratoriach)
    - są wolne, bo parsowane i interpretowane instrukcja po instrukcji
- **Wektoryzacja**
  - to zamienienie kodu wykonywanego w pętli na skalarach
  - na kod wykonywany przez f-cje wbudowane na macierzach

Wektoryzacja obliczeń – test

```
t2=zeros(1,1000);           % tablica przechowująca kolejne czasy obliczeń
for k=1:1000                % 1000 powtórzeń eksperymentu
    clear b                 % usuwamy b z przestrzeni roboczej
    tic                     % czas START!
    b=zeros(1,30000);       % testowany kod: pętla z prealokacją
    for i=1:30000
        b(i) = i^2;
    end
    t2(k) = toc;            % czas STOP!
end
median(t2)                  % średni czas wykonania pętli wewnętrznej
                            % R2007b: 0.226ms

t3=zeros(1,1000);           % tablica przechowująca kolejne czasy obliczeń
for k=1:1000                % 1000 powtórzeń eksperymentu
    clear b                 % usuwamy b z przestrzeni roboczej
    tic                     % czas START!
    b = (1:30000).^2;       % testowany kod - obliczenia na wektorze
    t3(k) = toc;            % czas STOP!
end
median(t3)                  % średni czas wykonania pętli wewnętrznej
                            % R2007b: 0.092ms

median(t2)/median(t3)       % przyspieszenie
                            % R2007b: 2.45x
```

Wektoryzacja – przykład 2

```
% Średnia geometryczna elementów dwóch macierzy,
% w drugiej każdy element pomniejszony o 1

% Metoda klasyczna:
A = magic(100);
B = pascal(100);
for m = 1:100
    for n = 1:100;
        X(m,n) = sqrt(A(m,n)) * (B(m,n) - 1);
    end
end

% Metoda wektorowa:
A = magic(100);
B = pascal(100);
X = sqrt(A).*(B-1);
```

- Kod zwektoryzowany jest zwięzły i przejrzysty

http://www.mathworks.com/support/tech-notes/1100/1109.html

Wektoryzacja – przykład 3

```
% Wartość elementu zależy od poprzedniego elementu
% z wykorzystaniem dodawania i mnożenia

% Metoda klasyczna:

n=10000;
V_B  = 100*ones(1,n);          % prealokacja
V_B2 = 100*ones(1,n);          %
ScaleFactor=rand(1,n-1);

for i = 2:n
    V_B(i) = V_B(i-1)*(1+ScaleFactor(i-1));
end

for i=2:n
   V_B2(i) = V_B2(i-1)+3;
end

% Metoda wektorowa:
n=10000;
V_A  = 100*ones(1,n);          % prealokacja
V_A2 = 100*ones(1,n);          %
ScaleFactor=rand(1,n-1);
V_A  = cumprod([100 1+ScaleFactor]);
V_A2 = cumsum([100 3*ones(1,n-1)]);
```

http://www.mathworks.com/support/tech-notes/1100/1109.html

Wektoryzacja obliczania wartości funkcji dwóch zmiennych

- Niech $F(x,y) = x \cdot \exp(-x\text{^}2 - y\text{^}2)$
  - chcemy znać wartość F dla każdego punktu (x, y),
    $x \in$ **x**, a $y \in$ **y**, np.

    x = [*x1 x2 x3 x4*]
    y = [*y1 y2 y3*]
  - utworzymy siatki obliczeniowe: `[X, Y] = meshgrid(x,y)`

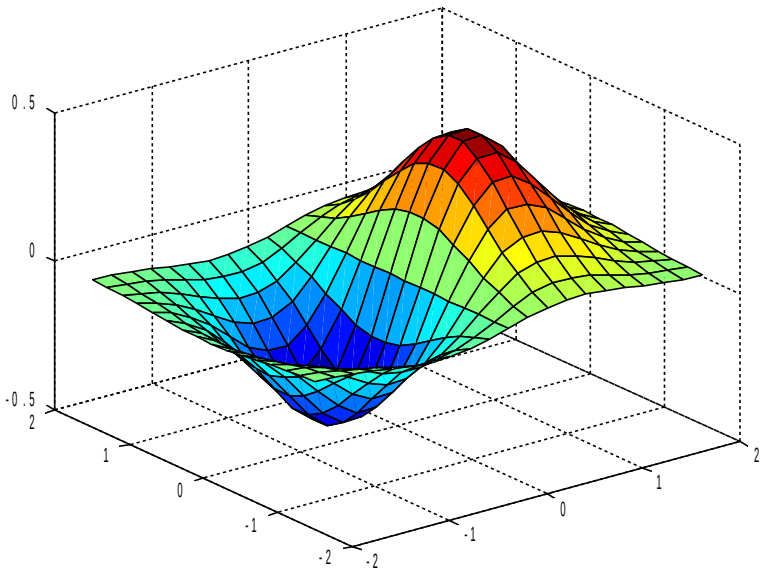
$$\mathbf{X} = \begin{bmatrix} x1 & x2 & x3 & x4 \\ x1 & x2 & x3 & x4 \\ x1 & x2 & x3 & x4 \end{bmatrix} \qquad \mathbf{Y} = \begin{bmatrix} y1 & y1 & y1 & y1 \\ y2 & y2 & y2 & y2 \\ y3 & y3 & y3 & y3 \end{bmatrix}$$

  - `size(X) = size(Y) = length(y) x length(x)`
  - wykonamy obliczenia na siatce:

```
F = X .* exp(-X.^2 - Y.^2)
```

Wektoryzacja obliczania wartości funkcji dwóch zmiennych (2)

```
x = (-2:.2:2);
y = (-1.5:.2:1.5)';
[X,Y] = meshgrid(x, y);
F = X .* exp(-X.^2 - Y.^2);
```



http://www.mathworks.com/support/tech-notes/1100/1109.html

Wektoryzacja sterowania

- Wektoryzować można również
  - porównania – wynik: tablice indeksów logicznych
  - instrukcje warunkowe – na podstawie indeksów logicznych

```
% Przypisanie co drugiemu elementowi myVector wartości jego cotangensa

% pętla for:
myVector = 1:100000;
for x=1:length(myVector)
    if(mod(myVector(x),2))                    % mod - modulo
        myVector(x) = cot(myVector(x));       % cot - cotangens
    end
end

% po wektoryzacji:
myVector = 1:100000;                  %   1    2    3    4    5    6 … 100000
indexTrue = mod(myVector,2)==1;       %   1    0    1    0    1    0 …      0
myVector(indexTrue) = cot(myVector(indexTrue));
                                      % 1.4  2.0 -1.1  4.0 -3.3  6.0   100000
```

Wektoryzacja sterowania (2)

- **Niech** `param_funk` – macierz o wymiarze 2181 x 11
  - w wierszach – modele kanałów białkowych (2181)
  - w kolumnach – parametry charakterystyki I-V (11), np.
    - kolumny 3,7 – prąd jonowy przy -100mV / +100mV
    - kolumny 4,8 – selektywność przy -100mV / +100mV
    - kolumna 9 – rektyfikacja (stosunek prądu) przy +100/-100mV

- **Zadanie:** znaleźć średnią rektyfikację modeli, dla których
  - prąd jonowy (wartość bezwzględna) > 1pA
  - selektywność > 10:1

```
>> ind_prad = param_funk(:,3) < -1 & param_funk(:,7) > 1;   % 1230
>> ind_sel  = param_funk(:,4) > 10 & param_funk(:,8) >10;   %  475
>> mean(param_funk(ind_prad & ind_sel, 9))          % wynik: 0.9792
```

indeksy wybranych modeli

indeks parametru rektyfikacji

Jak przyspieszyć interpreter?

- MATLAB<6.5 przetwarzał program w dwóch krokach
  - Parsowanie **m-kodu** na p-kod
  - Wykonanie **p-kodu** przez interpreter
    - instrukcja po instrukcji
    - narzut czasowy interpretera
      - w stosunku do kodu skompilowanego
      - szczególnie dotkliwe w pętlach na skalarach

  - Najlepszym rozwiązaniem była wektoryzacja
    - nie zawsze możliwa
    - czasem nieczytelny kod

http://www.ee.columbia.edu/~marios/matlab/accel_matlab.pdf

Kompilacja Just-In-Time

- Rozwiązaniem jest generacja kodu *Just-In-Time (JIT)*
  - Jeśli kod ma być wykonany wiele razy
    - interpreterowi może opłacać się skompilować go
  - Największe przyspieszenie uzyskano w pętlach `for` i `while`
    - także w instrukcjach warunkowych `if, elseif, switch`
    - pod warunkiem, że dane są typu
      `double, logical, char, int8-32, uint8-32`
    - nie więcej niż 3-wymiarowe (Matlab R2008b; 2-wym. – Matlab 6.5)
    - kod jest niezwektoryzowany
  - Akcelerator można włączyć(domyślnie) / wyłączyć:

```
>> feature accel on/off
```

http://www.sal.ufl.edu/NewComers/matlab_optimization_2.pdf

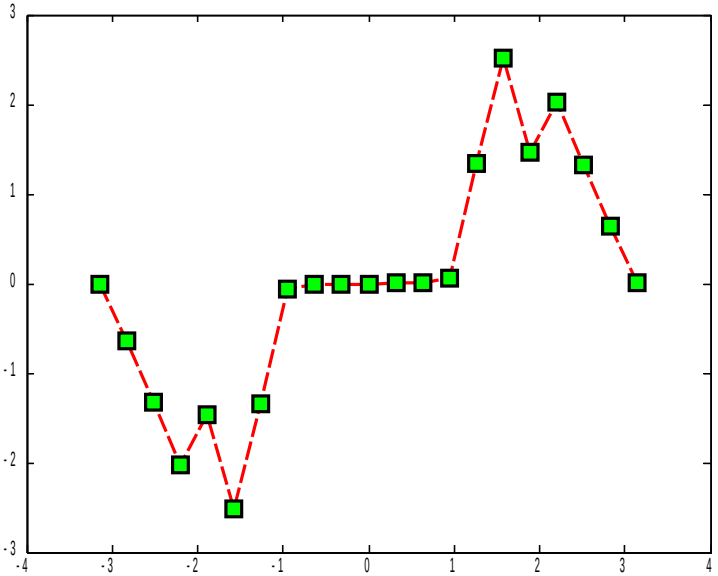
Program na dziś (2)

- Optymalizacja kodu
- **Grafika**
  - wykresy 2-wymiarowe
    - punktowe i liniowe: `plot`
      - formatowanie wykresów (`title, axes, xlabel, legend`)
    - wizualizacji danych (`area, bar, hist, stairs, stem`)
    - powierzchniowe i konturowe (`pcolor, contour`)
  - wykresy 3-wymiarowe
    - siatkowe i powierzchniowe (`mesh, surf`)
    - wizualizacja obiektów 3-wymiarowych (`slice, isosurface`)
  - zapis rysunku do pliku (`print, saveas`)
  - animacje (`getframe, movie`)

Wykresy 2-wymiarowe: plot

```
plot(X,Y)    % wykres punktów (x,y) należących do X i Y
plot(Y)      % X = indeks Y
plot(X,Y,S)  % S - format wykresu
plot(X,Y,S,'nazwa_param','wart_param',...)
plot(X1,Y1,S1, X2,Y2,S2, X3,Y3,S3,...) % wiele wykresów
                                       % w jednym
```

```
% Przykład:
x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));
plot(x,y,'--rs','LineWidth',2,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',10)
```

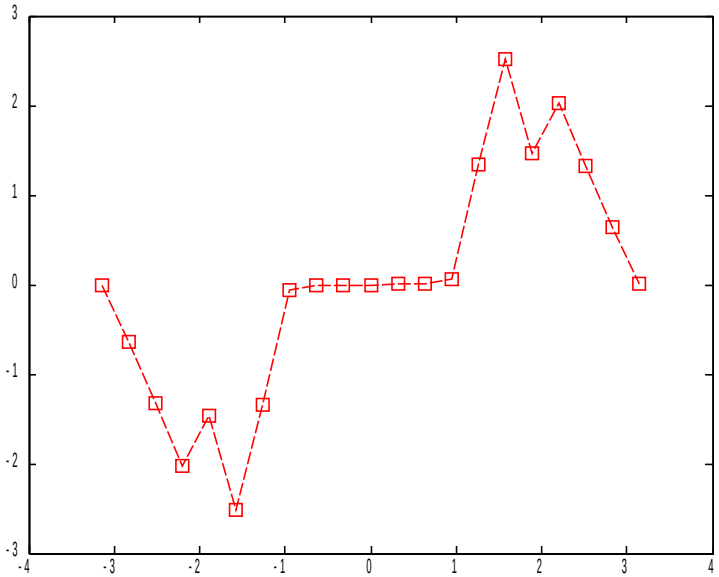


Formatowanie wykresu

```
>> help plot
```

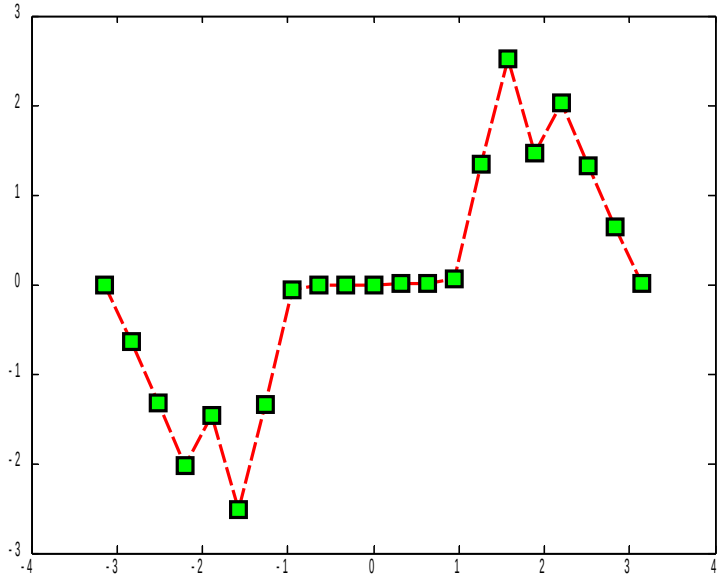
KOLOR		MARKER PUNKTU		STYL LINII	
b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
w	white	v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

```
>> plot(x,y,'rs--')  % kolor: red
                     % punkt: square (□)
                     % linia: dashed (--)
```



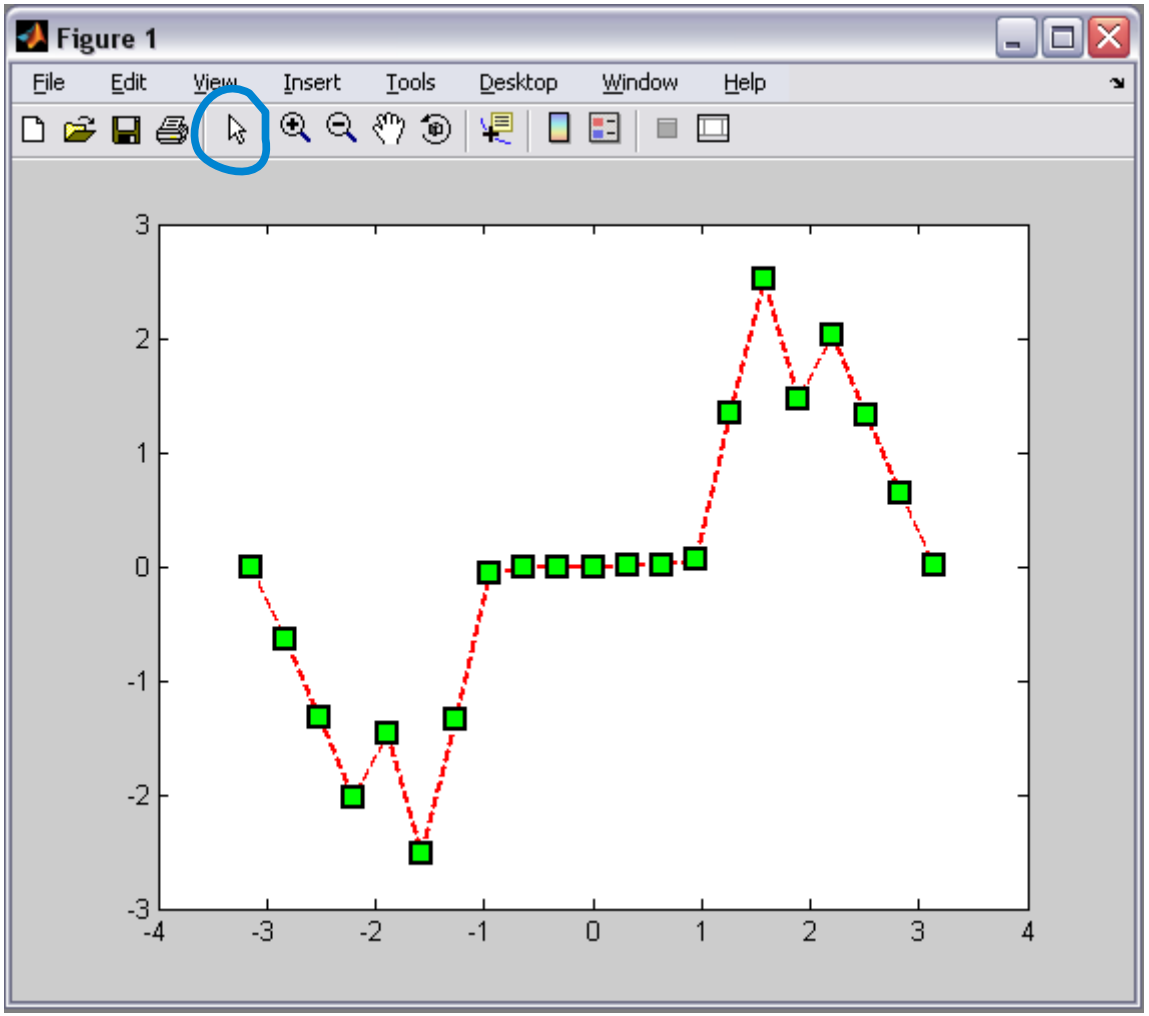
Parametry dodatkowe

```
x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));
plot(x,y,'--rs','LineWidth',2,...                % grubość linii
                'MarkerEdgeColor','k',...        % kolor obwódki markera pktu
                'MarkerFaceColor','g',...        % kolor obszaru markera pktu
                'MarkerSize',10)                 % rozmiar markera
```



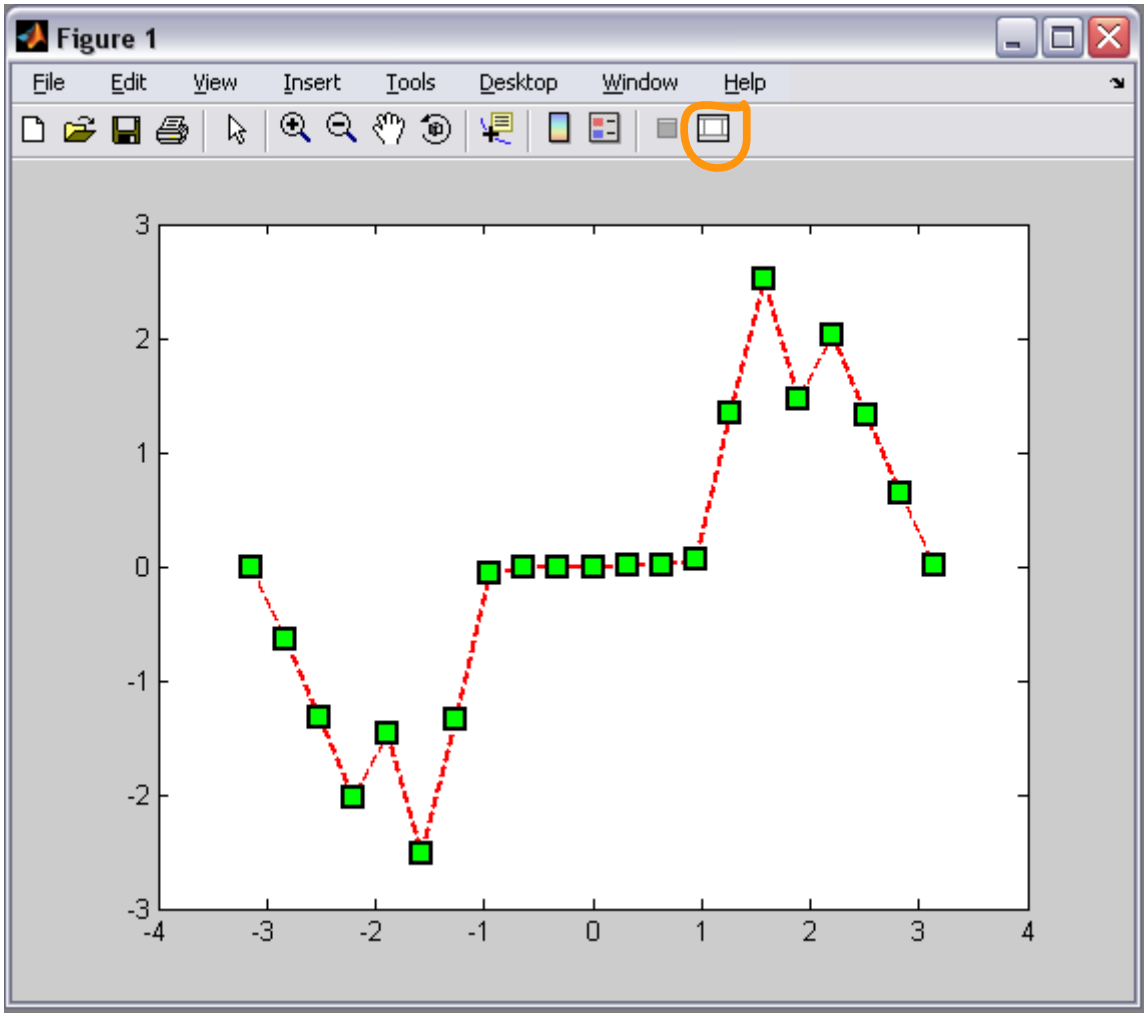
Jak sprawdzić zestaw wszystkich parametrów?

Interaktywny dostęp do parametrów



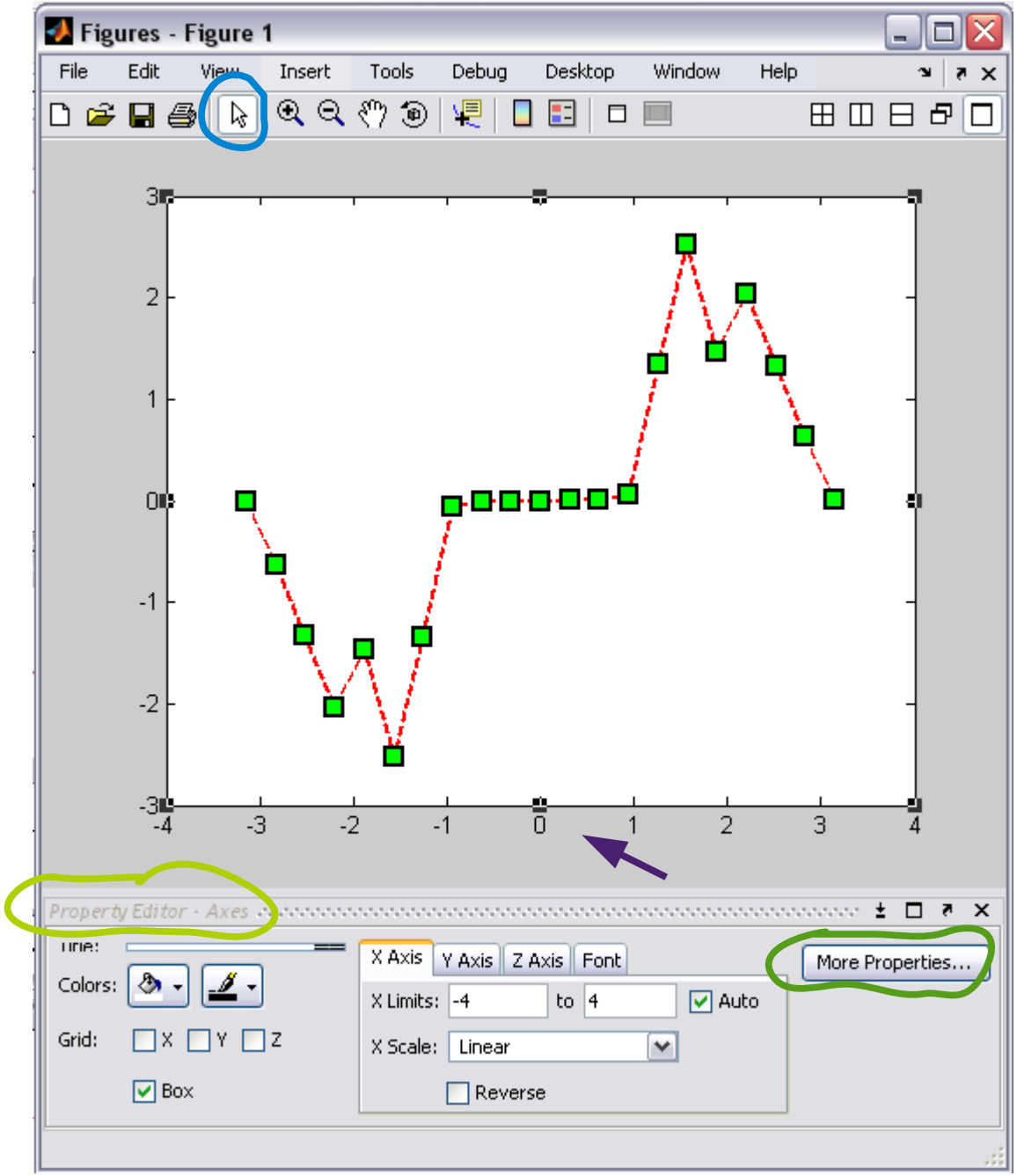
- Kliknij na strzałkę
  - aby przejść w tryb edycji
- Kliknij na elemencie wykresu
  - aby go zaznaczyć
- Kliknij prawym klawiszem
  - aby otworzyć menu kontekstowe
- Kliknij dwukrotnie na elemencie wykresu,
  - aby otworzyć rozszerzone okno narzędzi (*Plot Tools*)

Interaktywny dostęp do parametrów (2)



- Kliknij na ostatnią ikonkę
  - aby otworzyć rozszerzone okno narzędzi (*Plot Tools*)
  - i zadokować w nim rysunek

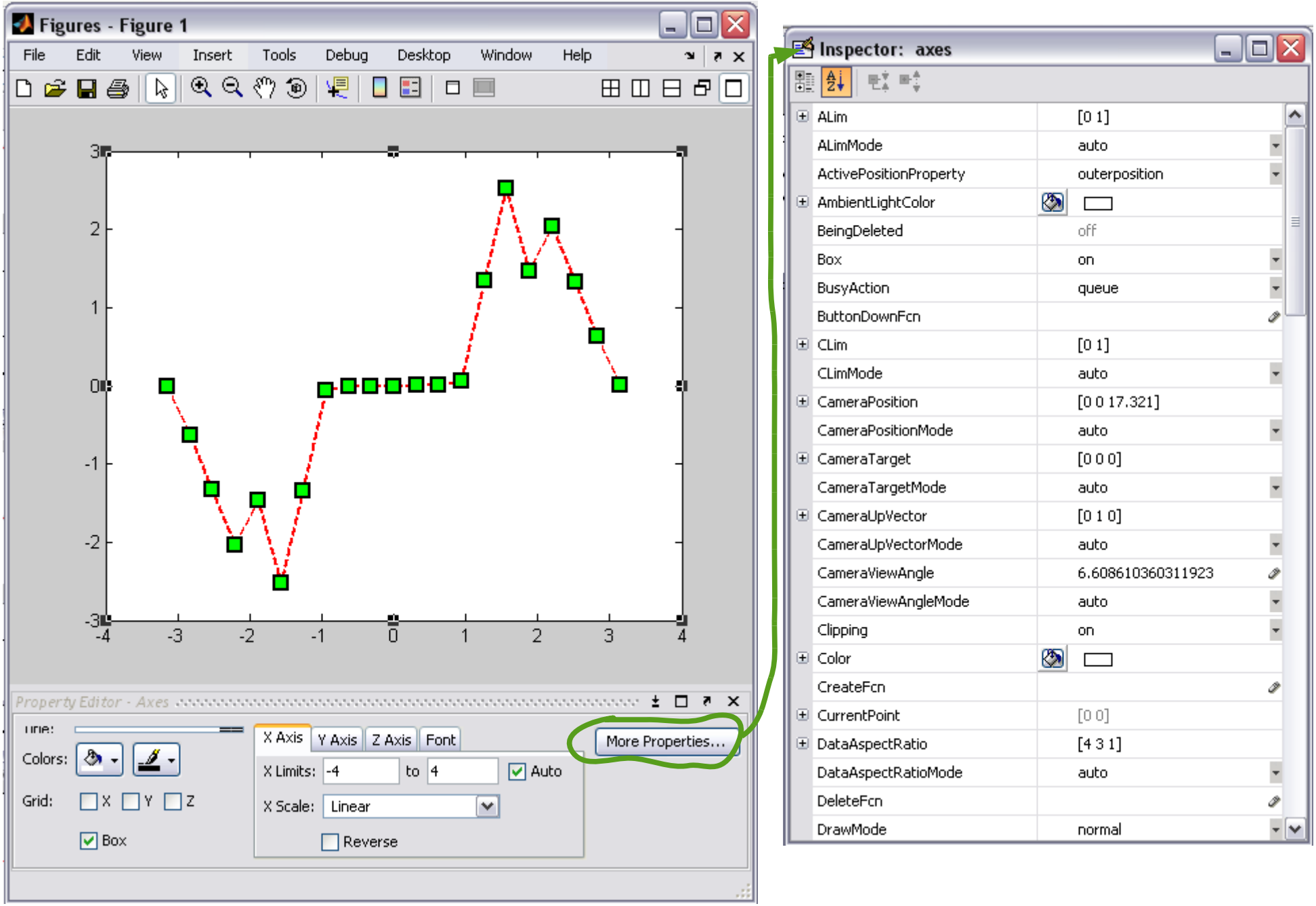
Interaktywny dostęp do parametrów (3)



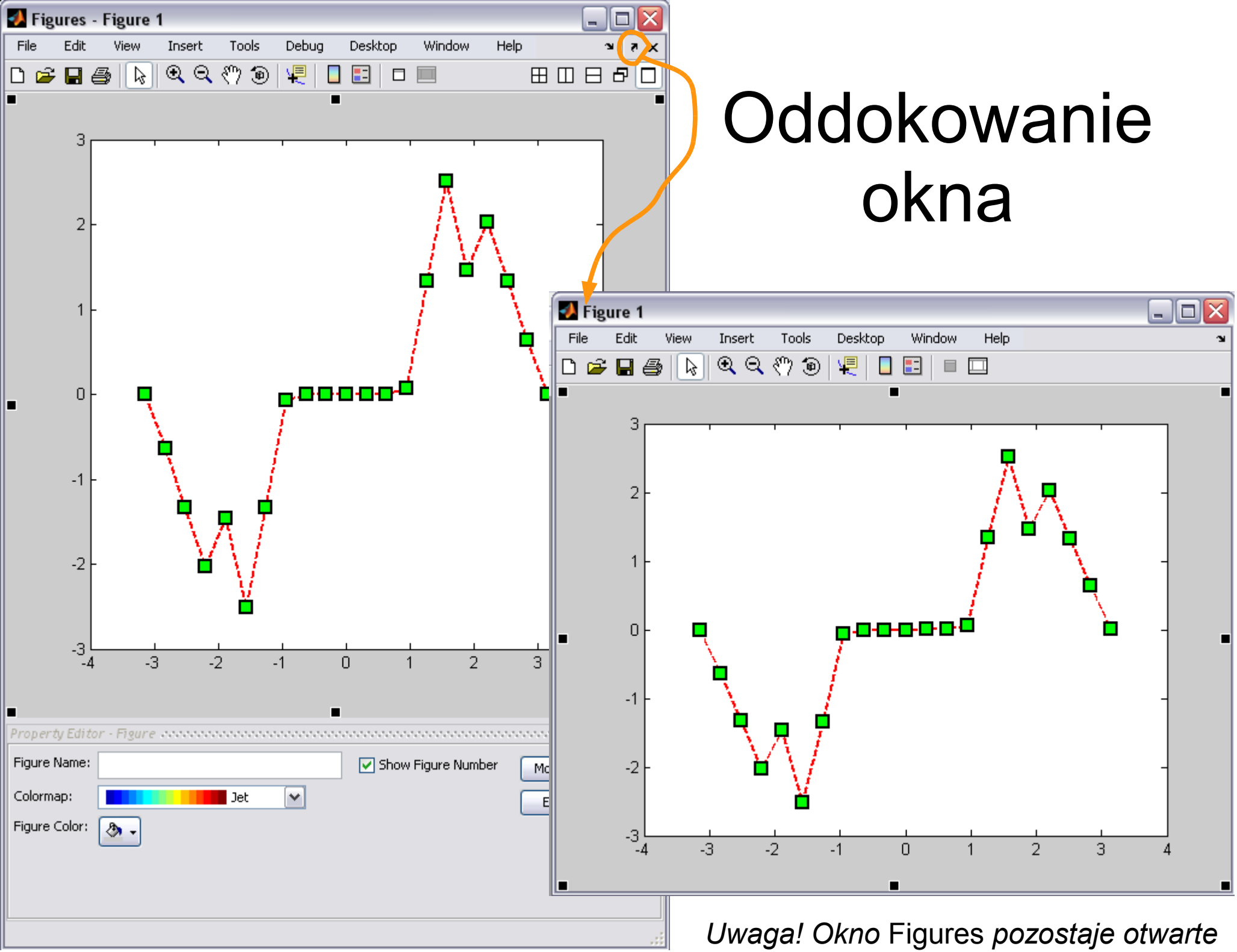
- Kliknij na strzałkę
  - aby przejść w tryb edycji
- Kliknij na elemencie wykresu
  - aby go zaznaczyć

    np. oś współrzędnych (axes)
- W edytorze właściwości (*Property Editor*)
  - ustaw parametry elementu
- Kliknij *More Properties...*
  - jeśli potrzebujesz bardziej szczegółowych parametrów otworzy się *Inspector*

Inspektor właściwości



Oddokowanie okna



Uwaga! Okno Figures *pozostaje otwarte*

Dalsza obróbka wykresu

```
>> plot(x,abs(cos(x)),'bs-',x,cos(x).^2,'go:')
```

- Osie
  - zakres

    ```
    >> axis([-pi pi 0 1])
    ```

  - nazwy

    ```
    >> xlabel('x'),ylabel('y')
    ```

- Siatka

  ```
  >> grid on
  ```

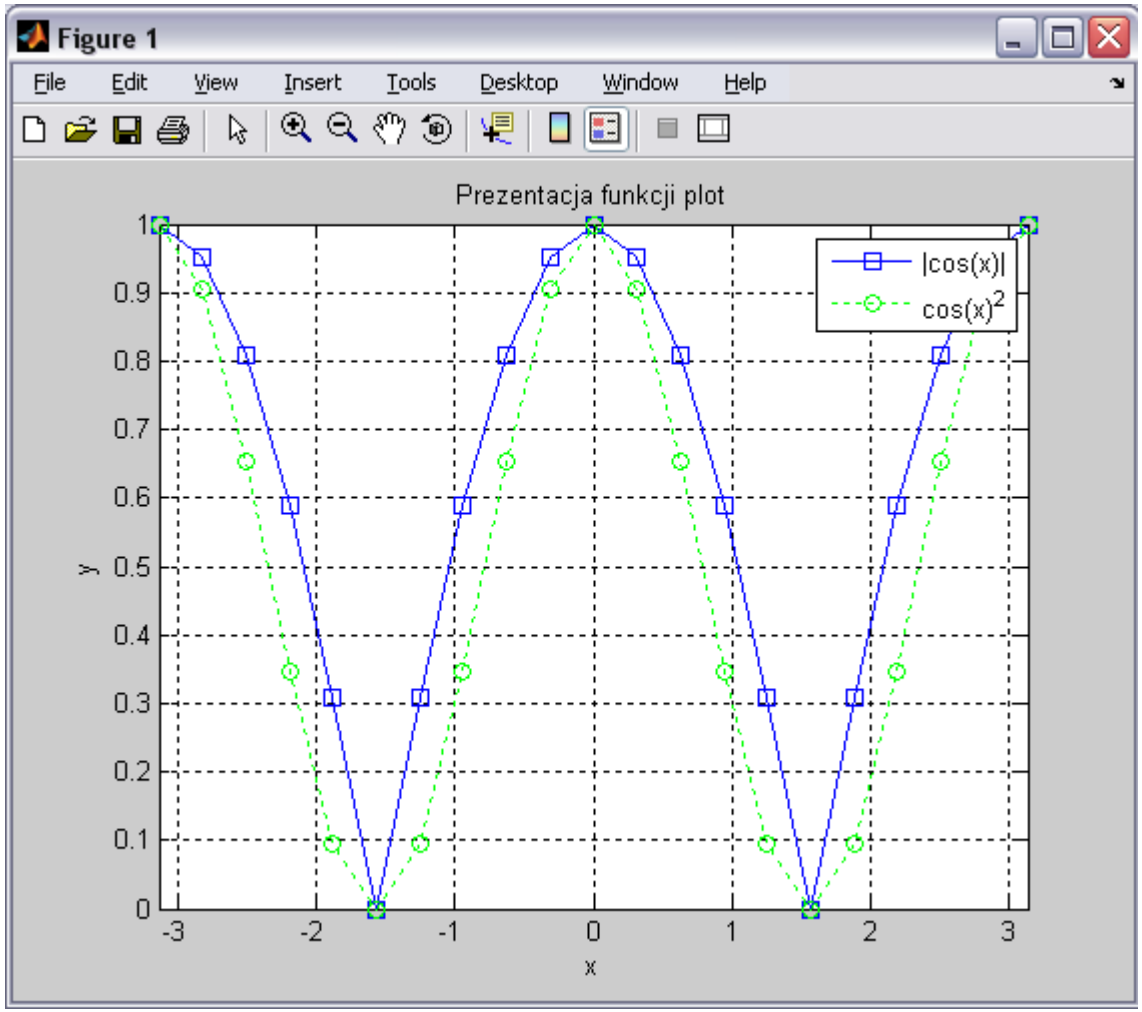
- Tytuł

  ```
  >> title('Prezentacja...')
  ```

- Legenda

  ```
  >> legend('|cos(x)|',...
            'cos(x)^2')
  ```

  Uwaga! formatowanie wzorów wzorowane na `LaTex`*-ie*



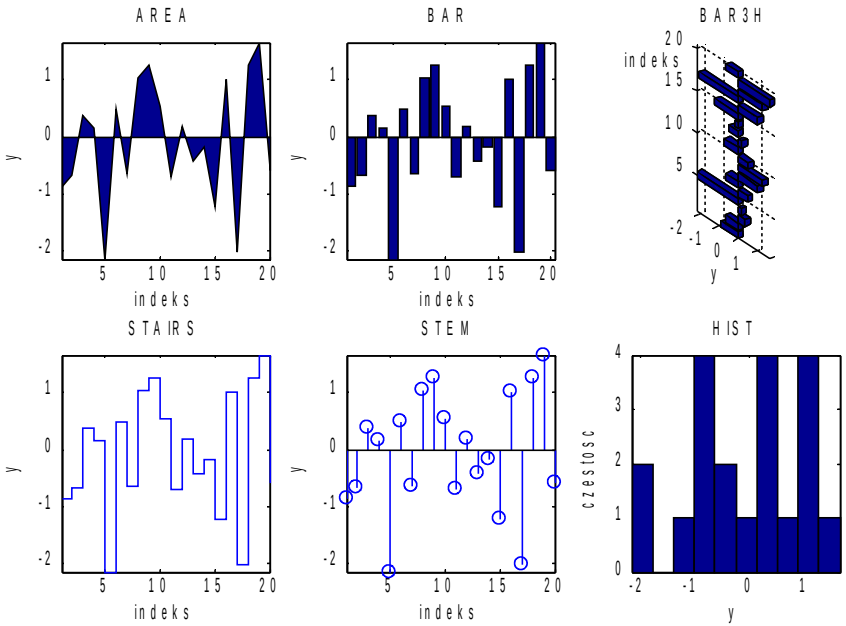
Proste wykresy 2-wymiarowe

- **Punktowe i liniowe**
  - `plot` - wykres liniowy 2-wym. we współrzędnych kartezjańskich
  - `plotyy` - wykres liniowy 2-wym. z dwoma osiami wartości
  - `semilogx` - wykres liniowy 2-wym. z osią X w skali logarytmicznej
  - `semilogy` - wykres liniowy 2-wym. z osią Y w skali logarytmicznej
  - `loglog` - wykres liniowy 2-wym. z osiami X i Y w skali logarytmicznej
  - `line` - dodaje linię do wykresu liniowego
  - `scatter` - wykres punktowy (np. wizualizacja korelacji)
  - `polar` - wykres we współrzędnych polarnych

Wykresy 2-wymiarowe

- **Wizualizacja danych**
  - `area` - wykres powierzchniowy
  - `bar, barh` - wykres słupkowy wertykalny i horyzontalny
  - `bar3, bar3h` - wykresy słupkowe przestrzenne
  - `pie, pie3` - wykres kołowy płaski i przestrzenny
  - `ribbon` - wykres wstążkowy (liniowy przestrzenny)
  - `hist` - histogram
  - `stairs` - wykres schodkowy dla danych dyskretnych
  - `stem` - wykres łodyga liście dla danych dyskretnych

Wykresy 2-wymiarowe (2)



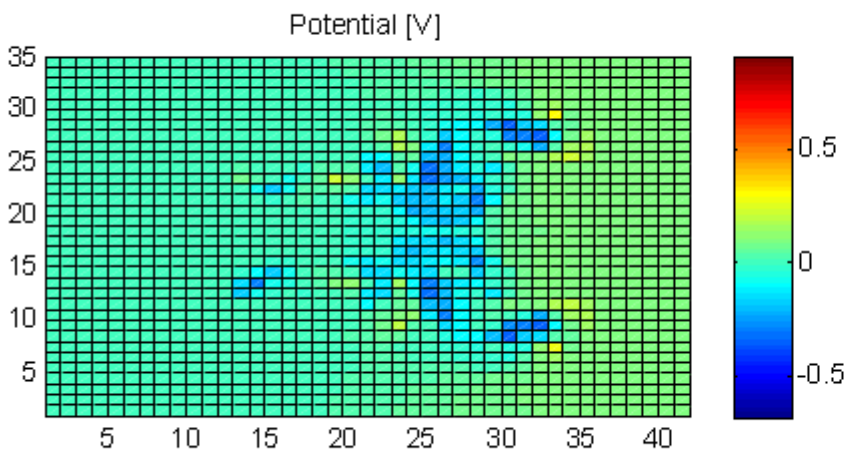

```
>> y = randn(1,20);
>> subplot(2,3,1), area(y), axis tight, title('AREA'),      % subplot(N,M,K)
   xlabel('indeks'), ylabel('y');                            % - dzieli rysunek
>> subplot(2,3,2), bar(y), axis tight, title('BAR'),         %   na NxM części
   xlabel('indeks'), ylabel('y');                            % - i przełącza
>> subplot(2,3,3), bar3h(y), axis tight, title('BAR3H'),     %   do K-tej części
   xlabel('indeks'), ylabel('y');
>> subplot(2,3,4), stairs(y), axis tight, title('STAIRS'),   % axis tight
   xlabel('indeks'), ylabel('y');                            % zakresy osi „ciasno”
>> subplot(2,3,5), stem(y), axis tight, title('STEM'),       % opasują dane
   xlabel('indeks'), ylabel('y');
>> subplot(2,3,6), hist(y), axis tight, title('HIST'),
   xlabel('y'), ylabel('czestosc');
```

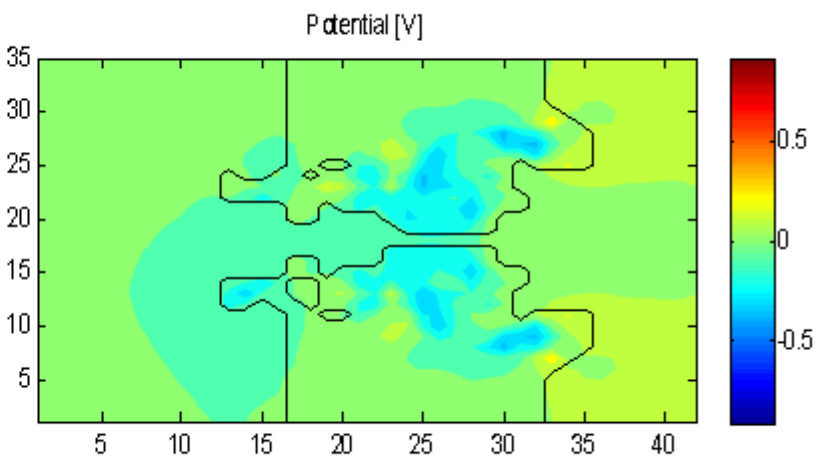
Wykresy izoliniowe (konturowe) i powierzchniowe

- Przekrój przez wolumetryczną mapę potencjału elektrostatycznego w kanale jonowym (białko)

`pcolor` – pseudokolorowa wizualizacja macierzy

`contour` – kontur
`contourf` – kontur wypełniony



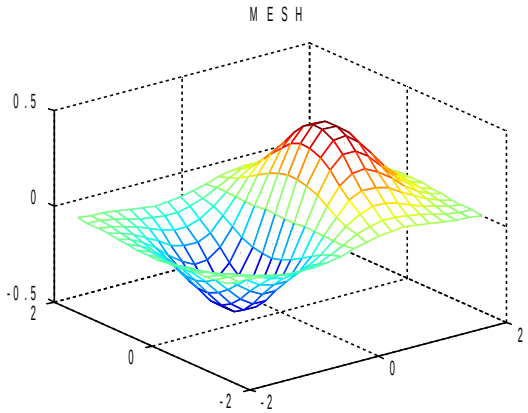


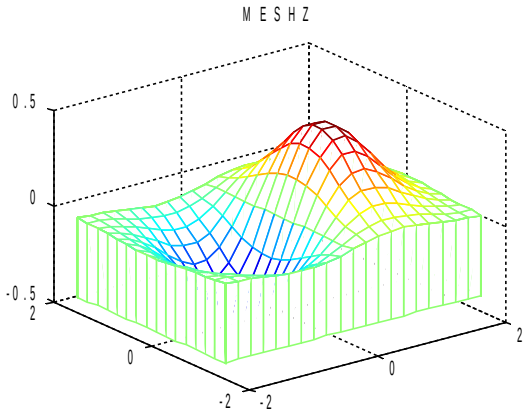
Wykresy 3-wymiarowe

- **Wersje wykresów 2-wymiarowych**
  - `plot3` - wykres liniowy 3-wym. we współrzędnych kartezjańskich
  - `scatter3` - wykres punktowy w przestrzeni 3-wym.
  - `stem3` - wykres łodyga liście dla danych dyskretnych (3-wym.)
  - `contour3` - wykres izoliniowy w przestrzeni 3-wym.

- **Wykresy siatkowe i powierzchniowe**
  - `mesh, meshz` - wykres siatkowy i siatkowy z kurtyną
  - `meshc` - wykres siatkowo-konturowy
  - `surf` - wykres powierzchniowy
  - `surfc` - wykres powierzchniowo-konturowy

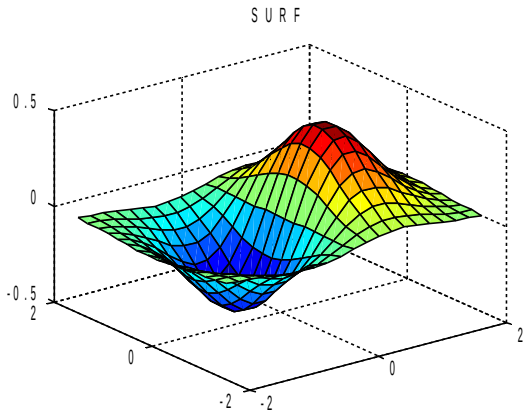
Wykresy siatkowe i powierzchniowe

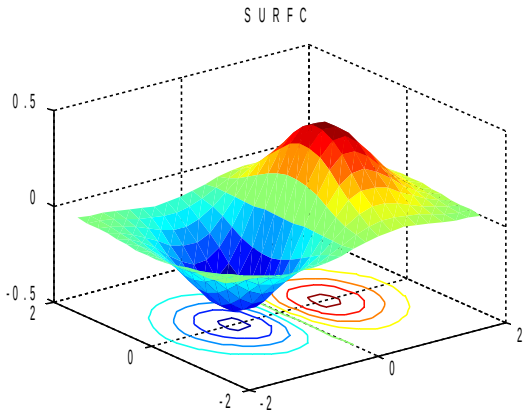





```
x = (-2:.2:2);
y = (-1.5:.2:1.5)';
[X,Y] = meshgrid(x, y);
F = X .* exp(-X.^2 - Y.^2);
```

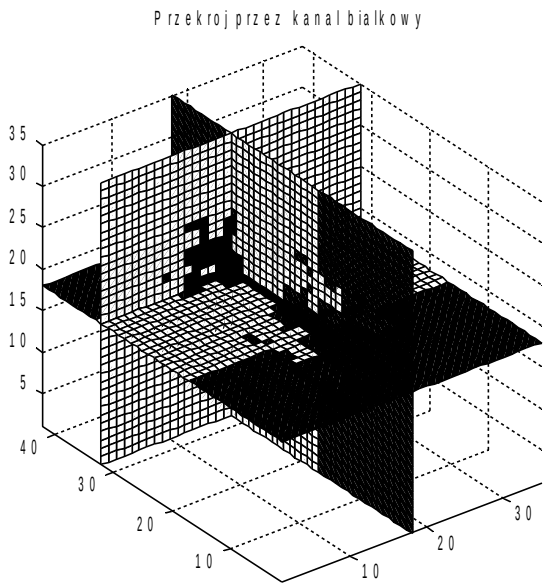
```
mesh(X,Y,F)
meshz(X,Y,F)
```

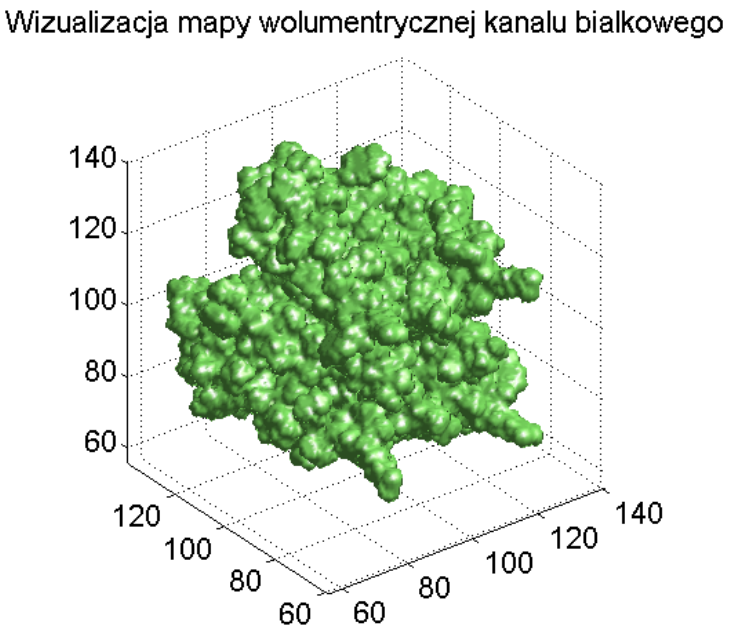





```
surf(X,Y,F)
surfc(X,Y,F), shading flat

% shading flat
% - usuwa linie siatki
```

Wizualizacja obiektów 3-wymiarowych






```
% data - mapa wolumetryczna
%        kanału białkowego
%        w błonie komórkowej

slice(data,18,32,18,'nearest')
axis equal
title('Przekroj przez kanal ...
bialkowy')
```

```
% data - mapa wolumetryczna
%        kanału białkowego

isosurface(data);
axis equal
grid on
title('Wizualizacja mapy ...
wolumentrycznej kanalu bialkowego')
```

Zapis rysunków

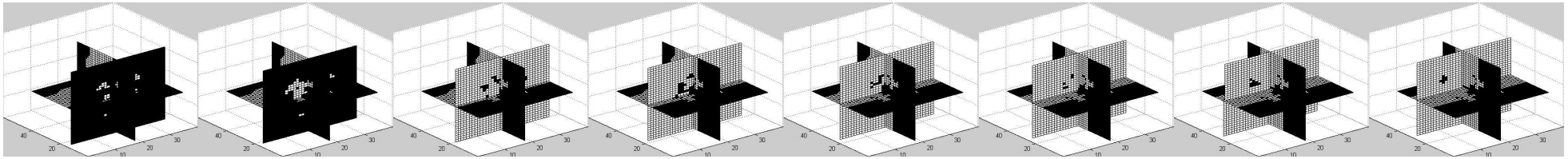
- Z linii poleceń lub funkcji

```
print -urządzenie -opcje nazwa_pliku

% urządzenie - drukarka albo plik, np.
%   -dbmp   - bitmapa 24bit
%   -djpeg  - obraz JPEG
%   -dpdf   - dokument PDF

% opcje - m.in. rozdzielczość, np. -r300 oznacza 300 dpi

>> print -djpeg -r600 moj_obraz

saveas(rysunek,'nazwa_pliku')

% rysunek - uchwyt do rysunku (będziemy o tym mówić później:-)
%           np. uchwyt aktualnego rysunku zwraca funkcja gcf

>> saveas(gcf, 'moj_dokument.pdf');
```

- Interaktywnie – przez menu rysunku
  - **File / Save as...**
  - **File / Export Setup... / Export...** (dodatkowe opcje)

Tworzenie animacji

```
% data - mapa wolumetryczna kanału białkowego w błonie
% Możesz stworzyć własny uproszczony model kanału:
% n = 21;
% [X, Y] = meshgrid(linspace(-1,1,n));
% R2 = X.^2 + Y.^2
% rmax = 0.5; rmin = 0.05; m = 100;
% A = linspace(-rmax+rmin,rmax-rmin,m).^2 + rmin.^2;
% R2A = repmat(R2,[1 1 length(A)]);
% R2A = shiftdim(R2A,2);
% AR2 = repmat(A',[1 size(R2,1) size(R2,2)]);
% data = double(R2A>AR2);

for k=1:size(data,1)
    slice(data,size(data,2)/2,k,size(data,3)/2,'nearest')
    animacja(k) = getframe;    % tworzy ramkę z wykresu/rysunku
end
movie(animacja)                % wyświetla animację (sprawdź opcje)
```



Dziś najważniejsze było...

- Aby program był prostszy i działał szybciej
  - prealokuj tablice
  - poznaj i stosuj różne metody indeksowania
  - wektoryzuj, jeśli potrafisz

Warto wiedzieć jak Matlab działa aby pisać lepsze programy!

- Wypróbuj bardzo duże możliwości grafiki Matlaba
  - pamiętaj o opisywaniu wykresów :-)

A za 2 tygodnie...

- Złożone typy danych
  - tablice komórkowe (*cell*)
  - struktury (*struct*)
- Programowanie zorientowane obiektowo