Informatyka

Wykład 6

Witold Dyrka
witold.dyrka@pwr.wroc.pl

30/04/2012

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. Złożone typy danych. Programowanie zorientowane obiektowo (OOP) (19.03.12)
4. OOP część 2 (02.04.12)
5. Graficzny interfejs użytkownika (16.04.12)
6. **Obliczenia numeryczne** **(30.04.12)**
7. Kolokwium (14.05.12)

Materiały do wykładu

- B. Mrozek, Z. Mrozek. MATLAB i Simulink. Poradnik użytkownika. Wydanie III. Helion 2010. Rozdział 8.
- J.L. Cornette and R.A. Ackerman. Calculus for the Life Sciences: A Modeling Approach. Część I-II. http://cornette.public.iastate.edu/CLS.html
- M. Kotulska. Informatyka. Wykład 4 i 6. PWr WPPT IB lato 2010/11
  www.if.pwr.wroc.pl/~kotulska/informatyka/info4.ppt, www.if.pwr.wroc.pl/~kotulska/informatyka/info6.ppt
- MATLAB Product Documentation. Mathematics.
  - Linear Algebra. Systems of Linear Equations
    http://www.mathworks.com/help/techdoc/math/f4-983672.html
  - Calculus. Ordinary Differential Equations
    http://www.mathworks.com/help/techdoc/math/f1-662913.html

Program na dziś

- Obliczenia numeryczne
  - **algebra liniowa**
    - rozwiązywanie układów równań liniowych
  - aproksymacja i interpolacja
  - uchwyty funkcji
  - całkowanie i różniczkowanie
  - równania różniczkowe zwyczajne

Algebra liniowa

- Układ równań liniowych

$$\mathrm{U}: \begin{cases} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = b_1, \\ a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = b_2, \\ \vdots & & \vdots & & \ddots & & \vdots & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n & = b_m. \end{cases}$$

- Zapis macierzowy:

$$\begin{bmatrix} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n \\ a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n \\ \vdots & & \vdots & & \ddots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix},$$

  - równoważne iloczynowi $\mathbf{Ax} = \mathbf{b}$

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}}_{\mathbf{b}}.$$

http://pl.wikipedia.org/wiki/Uk%C5%82ad_r%C3%B3wna%C5%84_liniowych

Układ równań liniowych – przykład

$$\begin{cases} -x + 3y + 2z = 0 \\ x \quad - z = 1 \\ 2x - 3y - z = 3 \end{cases}$$

Układ oznaczony

Liczba liniowo niezależnych, niesprzecznych równań
=
liczbie zmiennych

```
>> A = [ -1 3 2; 1 0 -1; 2 -3 -1];
>> b = [0 1 3]';

   % A = -1  3  2      b = 0
   %      1  0 -1          1
   %      2 -3 -1          3
```

- Jak znaleźć rozwiązać?
  - jeśli macierz `A` jest kwadratowa i odwracalna

```
A*x = b
inv(A)*A*x = inv(A)*b   % mnożenie obustronne przez
                        % macierz odwrotną inv(A)
x = inv(A)*b            % bo: inv(A)*A = I
```

Przykład wg Mrozek&Mrozek, Matlab i Simulink. Rozdział 2.2.9. Helion 2010

Układ równań liniowych – przykł. (2)

```
% A = -1  3  2      b = 0
%      1  0 -1          1
%      2 -3 -1          3
```

- Jak znaleźć rozwiązać?

  - jeśli macierz `A` jest kwadratowa i odwracalna

```
>> x = inv(A)*b              % wolniejsze i mniej dokładne
```

lub

```
>> x = A\b                   % szybsze i bardziej dokładne
```

    - w obydwu przypadkach:

```
x =  2
     0
     1
```

Przykład wg Mrozek&Mrozek, Matlab i Simulink. Rozdział 2.2.9. Helion 2010

Jak działa operator dzielenia macierzowego?

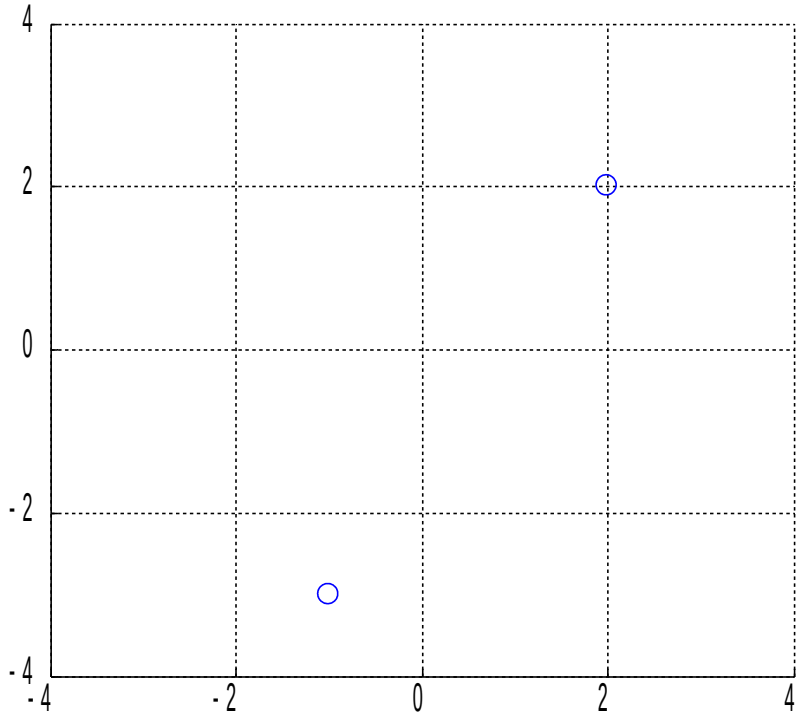
- Automatyczny wybór najlepszej metody
  - macierz trójkątna
    - metoda podstawiania wstecz
  - macierz symetryczna dodatnio określona
    - rozkład Choleskiego lub LDL'
  - macierz Hessenberga górna
    - eliminacja Gaussa
  - inne macierze kwadratowe
    - rozkład LU z częściowym wyborem elementu podstawowego

- Uwaga!
  - jeśli problem w postaci: `A*x = b`
    - używamy operatora lewostronnego `x = A\b`
  - jeśli problem w postaci: `x*A = b`
    - używamy operatora prawostronnego `x = A/b` `% nb. M/N=(M'\N')'`

http://www.mathworks.com/help/techdoc/math/f4-983672.html

Interpretacja graficzna

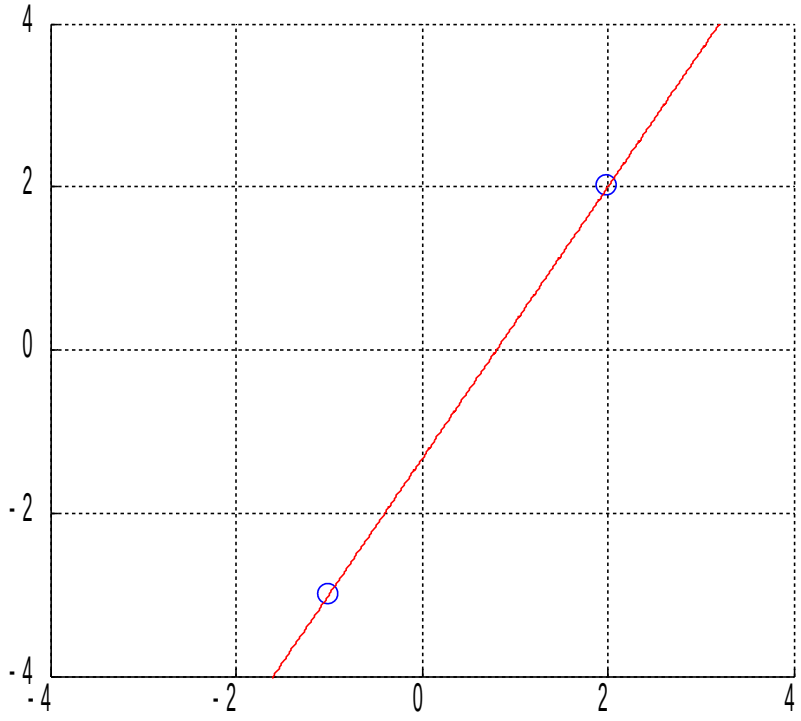
- Układ oznaczony




```
a1*x1 + a0 = y1
a1*x2 + a0 = y2

a1*-1 + a0 = -3
a1* 2 + a0 =  2

X = -1 1   y = -3
     2 1        2

>> a = X\y

a = 1.6667
   -1.3333
```

Interpretacja graficzna (2)

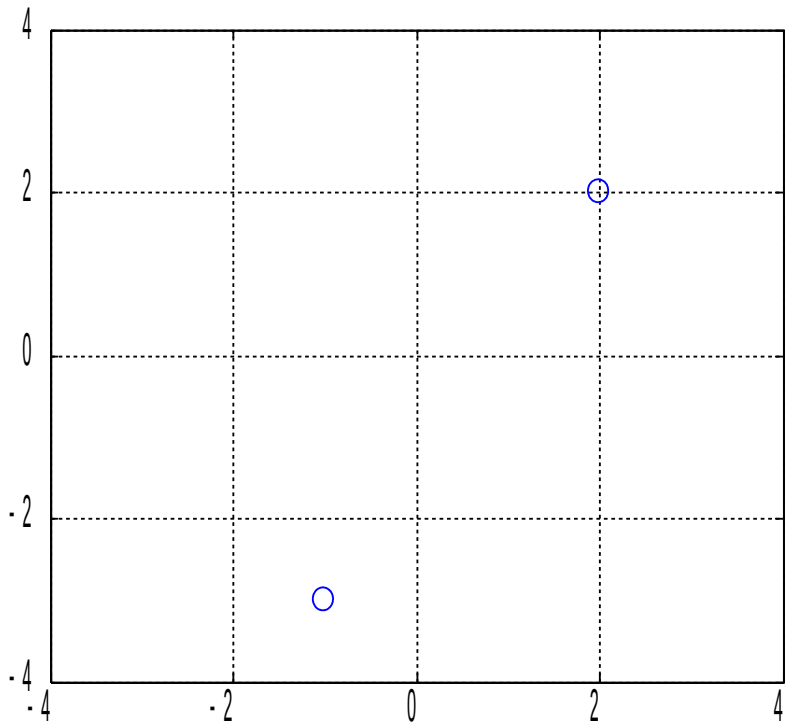
- Układ oznaczony




```
a1*x1 + a0 = y1
a1*x2 + a0 = y2

a1*-1 + a0 = -3
a1* 2 + a0 =  2

X = -1 1   y = -3
     2 1        2

>> a = X\y

a = 1.6667
   -1.3333

y = 1.6667*x - 1.3333

>> t = -4:0.01:4;

>> plot(t,a(1)*t+a(2));
```

Układ niedookreślony

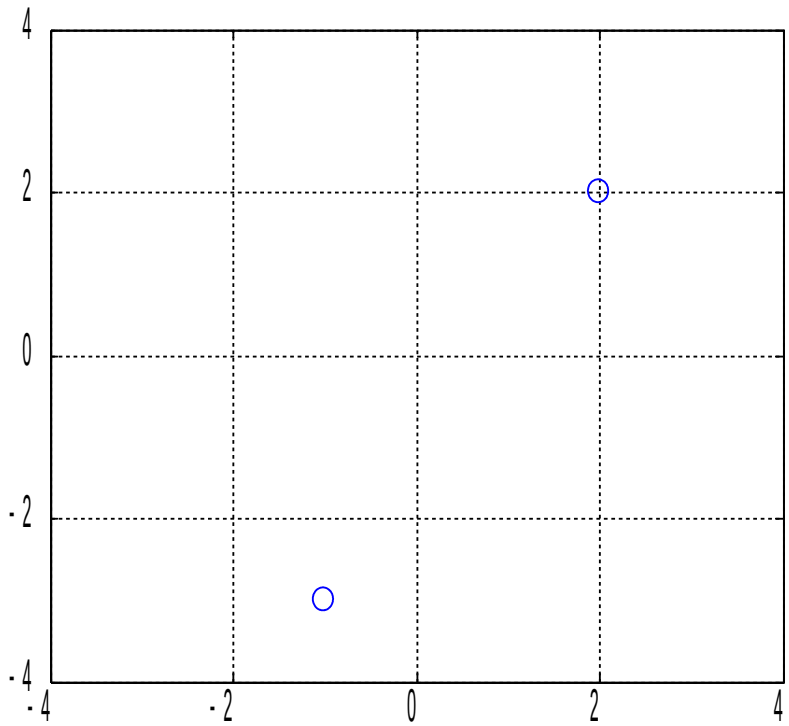
- Układ niedookreślony




```
a2*x1^2 + a1*x1 + a0 =y1
a2*x2^2 + a1*x2 + a0 =y2

%  x^2 x 1
X = 1 -1 1    y = -3
    4  2 1         2
```

Układ niedookreślony

- Układ niedookreślony

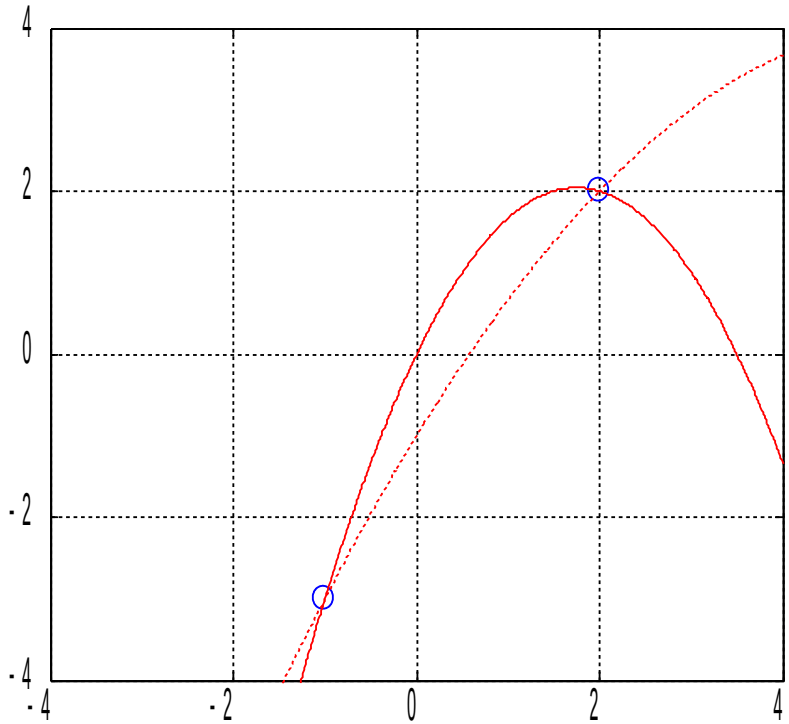



```
a2*x1^2 + a1*x1 + a0 =y1
a2*x2^2 + a1*x2 + a0 =y2

%  x^2 x 1
X = 1 -1 1    y = -3
    4  2 1        2
```

- Do wyznaczenia paraboli potrzebne są 3 punkty
  - tutaj mamy 2 punkty: układ **niedookreślony**
  - istnieje nieskończenie wiele rozwiązań

Interpretacja graficzna

- Układ niedookreślony

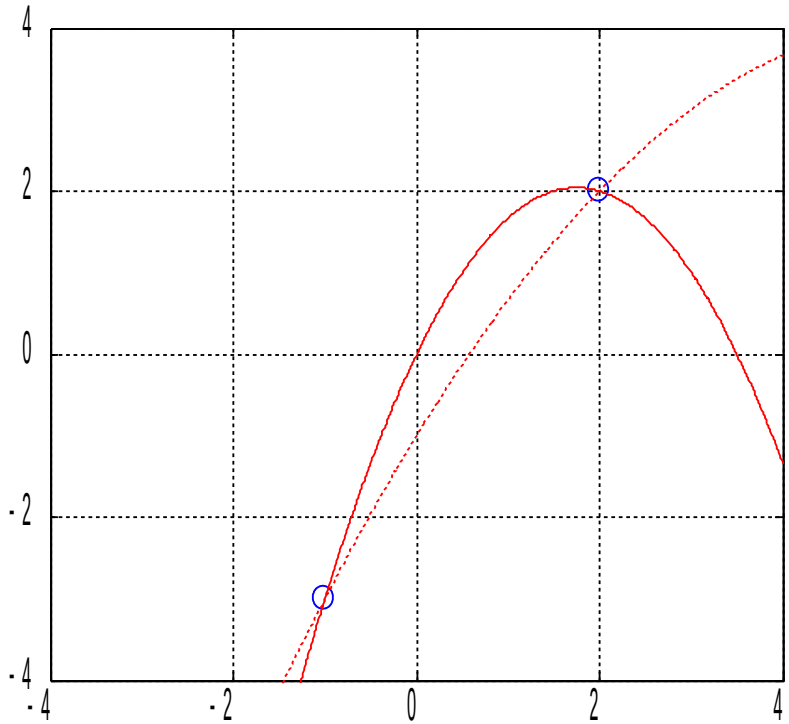



```
a2*x1^2 + a1*x1 + a0 =y1
a2*x2^2 + a1*x2 + a0 =y2

%  x^2  x  1
X = 1  -1  1    y = -3
    4   2  1        2
```

- Do wyznaczenia paraboli potrzebne są 3 punkty
  - tutaj mamy 2 punkty: układ **niedookreślony**
  - istnieje nieskończenie wiele rozwiązań

Interpretacja graficzna

- Układ niedookreślony

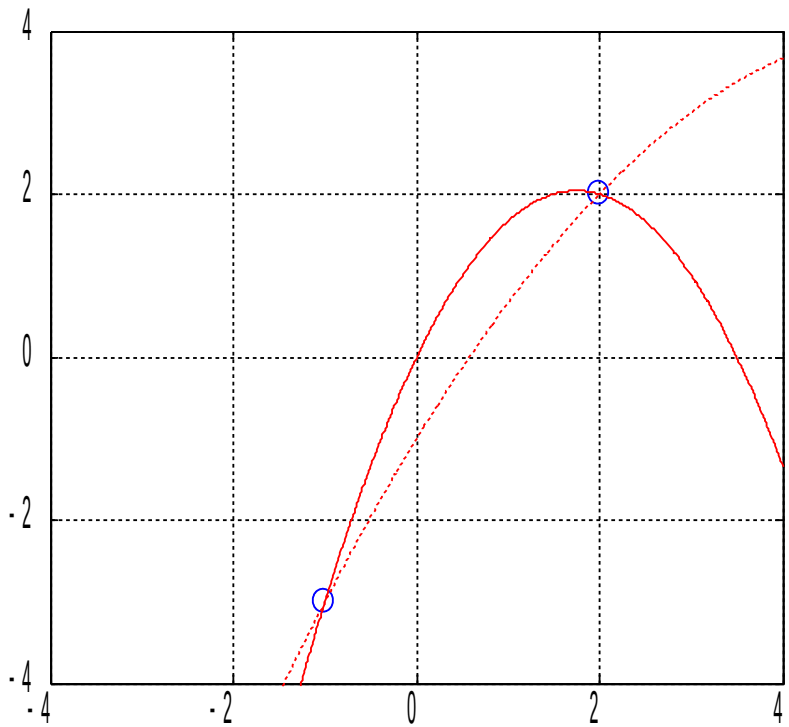



```
a2*x1^2 + a1*x1 + a0 =y1
a2*x2^2 + a1*x2 + a0 =y2

%  x^2  x  1
X = 1  -1  1    y = -3
    4   2  1        2
```

- Do wyznaczenia paraboli potrzebne są 3 punkty
  - tutaj mamy 2 punkty: układ **niedookreślony**
  - istnieje nieskończenie wiele rozwiązań

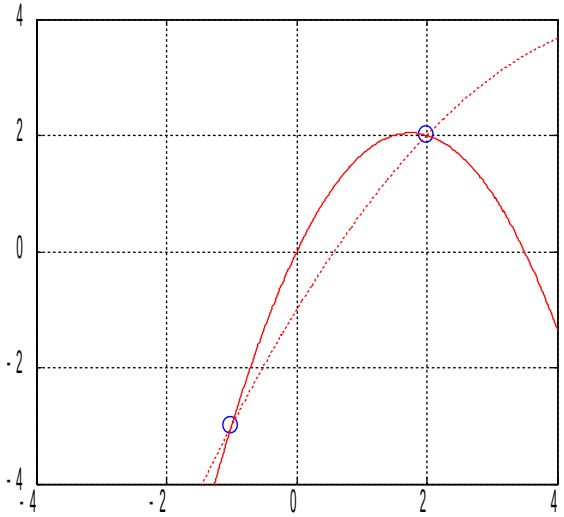
Układ niedookreślony

- Układ niedookreślony




```
a2*x1^2 + a1*x1 + a0 =y1
a2*x2^2 + a1*x2 + a0 =y2

%  x^2 x 1
X = 1 -1 1    y = -3
    4  2 1         2

% Dwa z możliwych rozw.:
>> a = X\y
a = -0.6667
     2.3333
          0

>> b = pinv(X)*y
b = -0.1667
     1.8333
    -1.0000

>> t = -4:0.01:4;
>>  plot(t,...
a(1)*t.^2+a(2)*t+a(3),'r')
>> plot(t,...
b(1)*t.^2+b(2)*t+b(3),'r:')
```

Układ niedookreślony



- Rozwiązanie układu niedookreślonego

```
Xa = y
% X - macierz m x n, tj. m punktów
```

  - MATLAB używa metody najmniejszych kwadratów

```
a = X\y
```

    - najwyżej **m** niezerowych współczynników **a**
    - operator dzielenia (`mldivide`) stosuje rozkład QR

lub

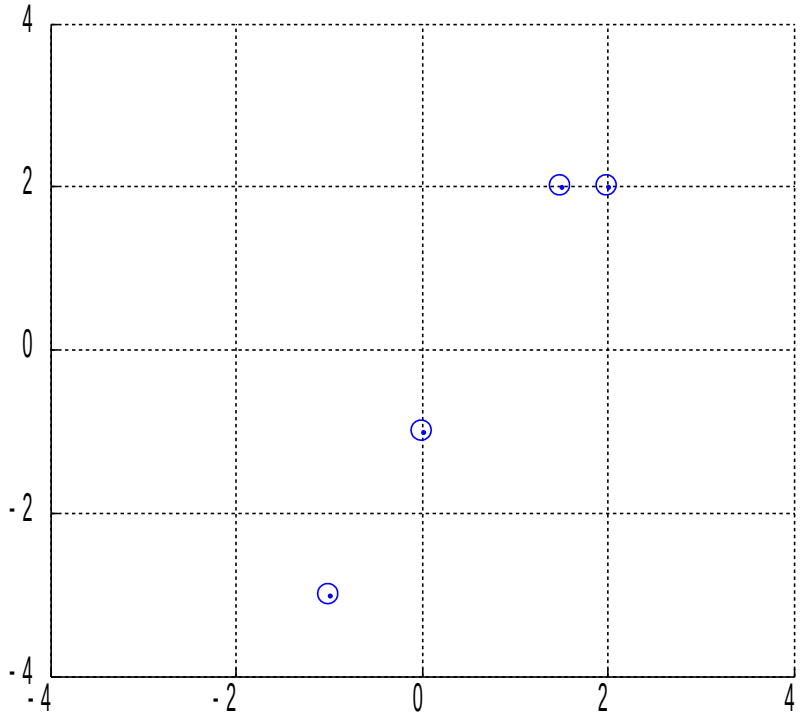
```
a = pinv(X)*y
```

    - najmniejsza norma wektora współczynników **a**
    - `pinv` – macierz pseudoodwrotna Penrose-Moore'a

http://www.mathworks.com/help/techdoc/ref/pinv.html

Układ nadokreślony

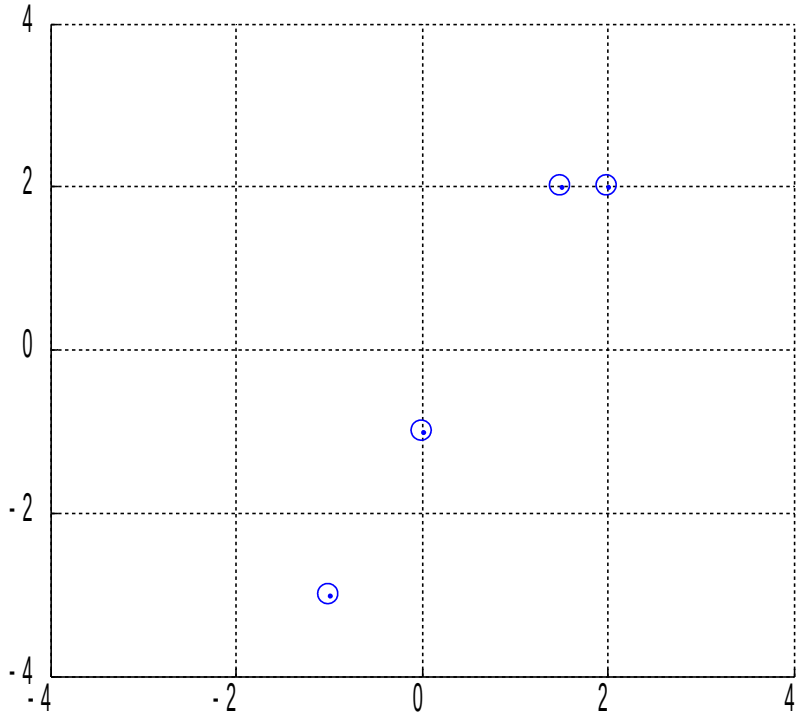
- Układ nadokreślony




```
a1*x1 + a0 = y1
a1*x2 + a0 = y2
a1*x3 + a0 = y3
a1*x4 + a0 = y4

X = -1 1   y = -3
     2 1        2
   1.5 1        2
     0 1       -1
```

Układ nadokreślony

- Układ nadokreślony

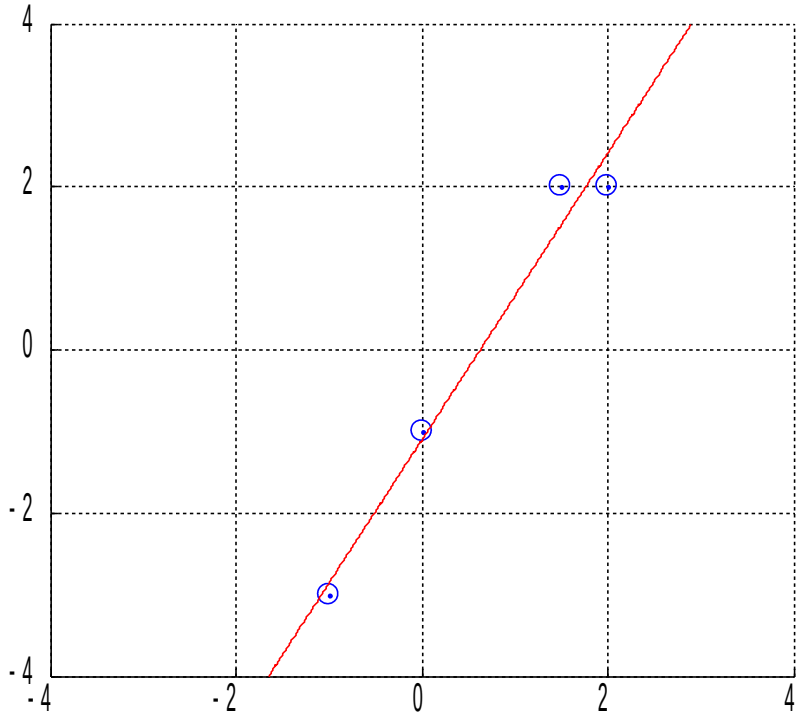



```
a1*x1 + a0 = y1
a1*x2 + a0 = y2
a1*x3 + a0 = y3
a1*x4 + a0 = y4

X = -1 1    y = -3
     2 1         2
   1.5 1         2
     0 1        -1
```

- Do wyznaczenia prostej wystarczą 2 punkty
  - tutaj mamy 4 punkty: układ **nadokreślony**
  - nie ma jednej prostej przechodzącej przez wszystkie punkty

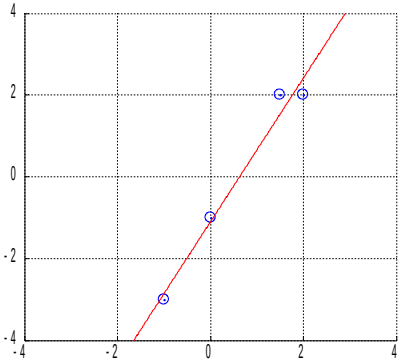
Interpretacja graficzna

- Układ nadokreślony




```
a1*x1 + a0 = y1
a1*x2 + a0 = y2
a1*x3 + a0 = y3
a1*x4 + a0 = y4

X = -1 1    y = -3
     2 1         2
   1.5 1         2
     0 1        -1

>> a = X\y

a = 1.7582
   -1.0989
```

Układ nadokreślony



- Rozwiązanie układu nadokreślonego
  - dopasowanie krzywej do danych
    metodą najmniejszych kwadratów

    `a = X\y`

    lub

    `a = pinv(X)*y`

    - `pinv` – macierz pseudoodwrotna Penrose-Moore'a

Uwaga! Metody zwracają różne wyniki

- *jeśli problem najmniejszych kwadratów nie ma unikalnego rozwiązania*

http://www.mathworks.com/help/techdoc/math/f4-983672.html#f4-2064
http://www.mathworks.com/help/techdoc/ref/pinv.html

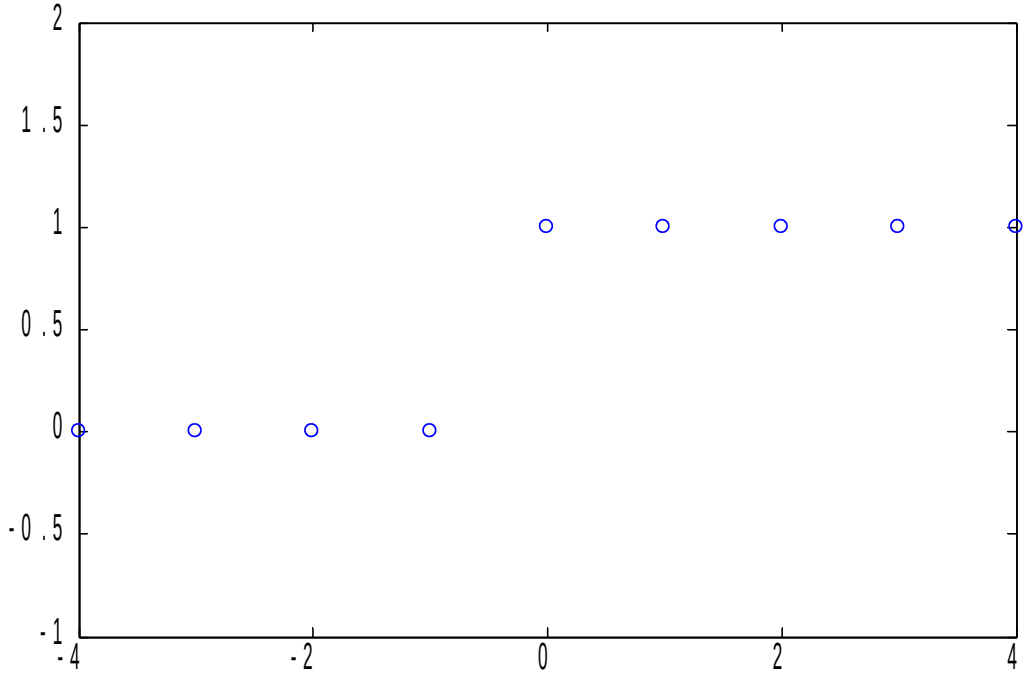
Program na dziś

- Obliczenia numeryczne
  - algebra liniowa
  - **aproksymacja i interpolacja**
    - dopasowanie wielomianem
    - interpolacja w Matlabie
  - uchwyty funkcji
  - całkowanie i różniczkowanie
  - równania różniczkowe zwyczajne

Dopasowanie wielomianem

- Funkcja schodkowa

```
>> x = -4:1:4;
>> y = [0 0 0 0 1 1 1 1 1];

>> stairs(x,y,'o:')
>> axis([-4 4 -1 2])
```



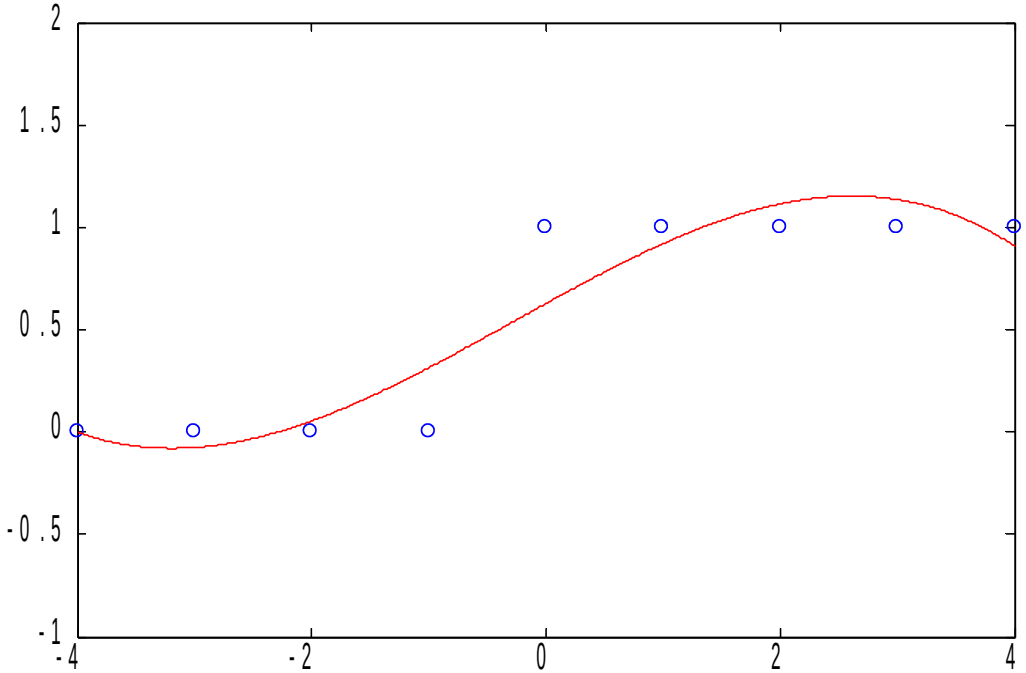
Aproksymacja wielomianem

- Wielomian 3 stopnia

```
>> p=polyfit(x,y,3)
p = -0.0126 -0.0108 0.3157 0.6277   % wsp. wiel. 3-st.

>> xx=-4:.01:4;
>> yy=polyval(p,xx);                % wartości wiel. p dla xx

>> plot(xx,yy,'r')
>> axis([-4 4 -1 2])
```

Aproksymacja

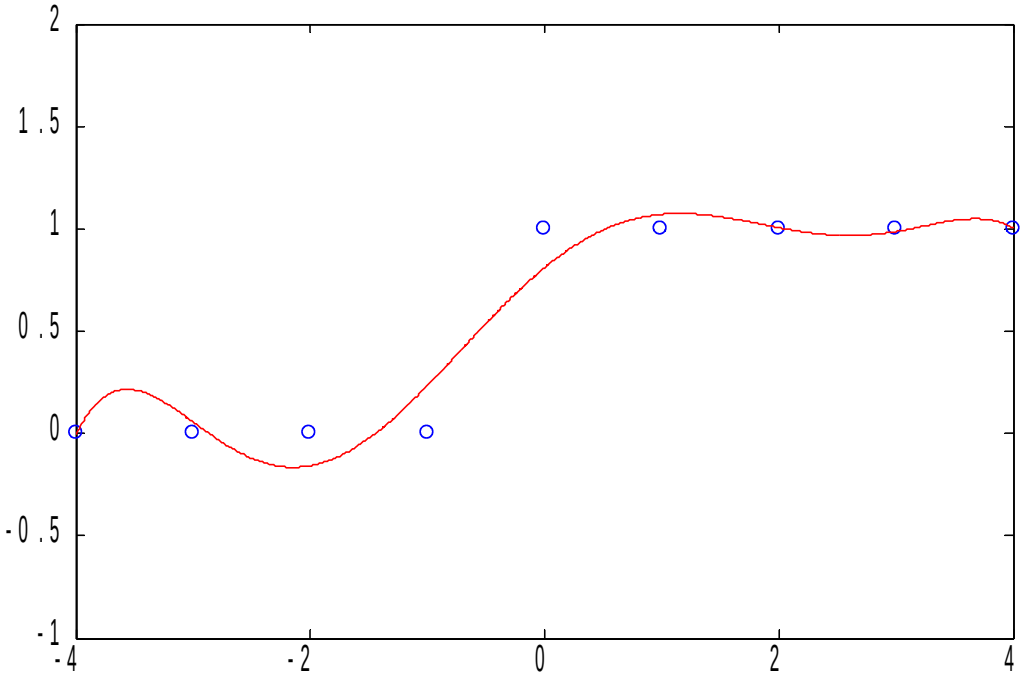
przybliżenie funkcji skomplikowanej prostą



Aproksymacja wielomianem (2)

- Wielomian 6 stopnia

```
>> p=polyfit(x,y,6);

>> xx=-4:.01:4;
>> yy=polyval(p,xx);          % wartości wiel. p dla xx

>> plot(xx,yy)
>> axis([-4 4 -1 2])
```



Interpolacja wielomianowa

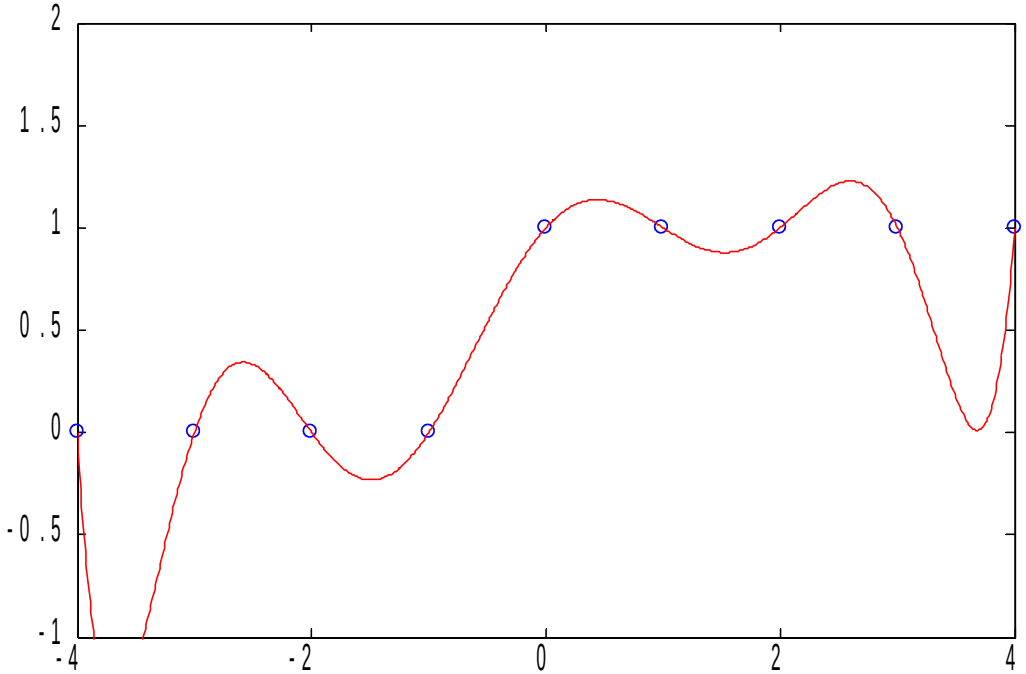
- Wielomian 8 stopnia

```
>> p=polyfit(x,y,8);
```

- Uwaga! Wielomian stopnia **n** przechodzi przez **n+1** punktów

Interpolacja

funkcja dopasowująca przechodzi przez wszystkie zadane punkty

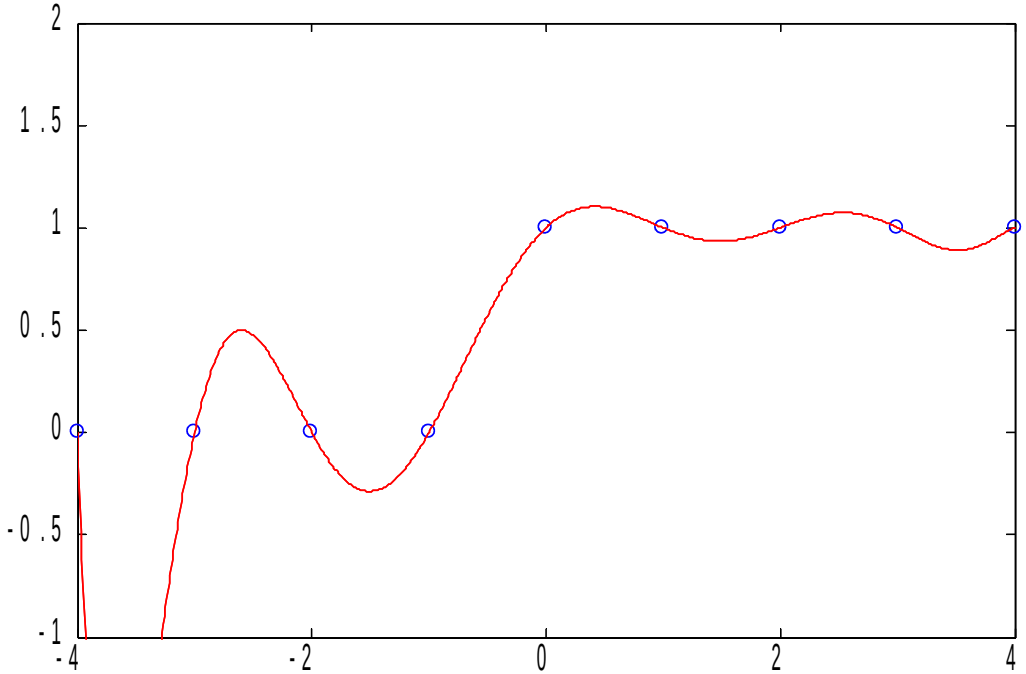


Interpolacja wielomianowa

- Wielomian 9 stopnia

```
>> p=polyfit(x,y,9);
Warning: Polynomial is not unique; degree >= number of
data points.
```

  - istnieje wiele wielomianów 9 stopnia przechodzących przez 9 punktów

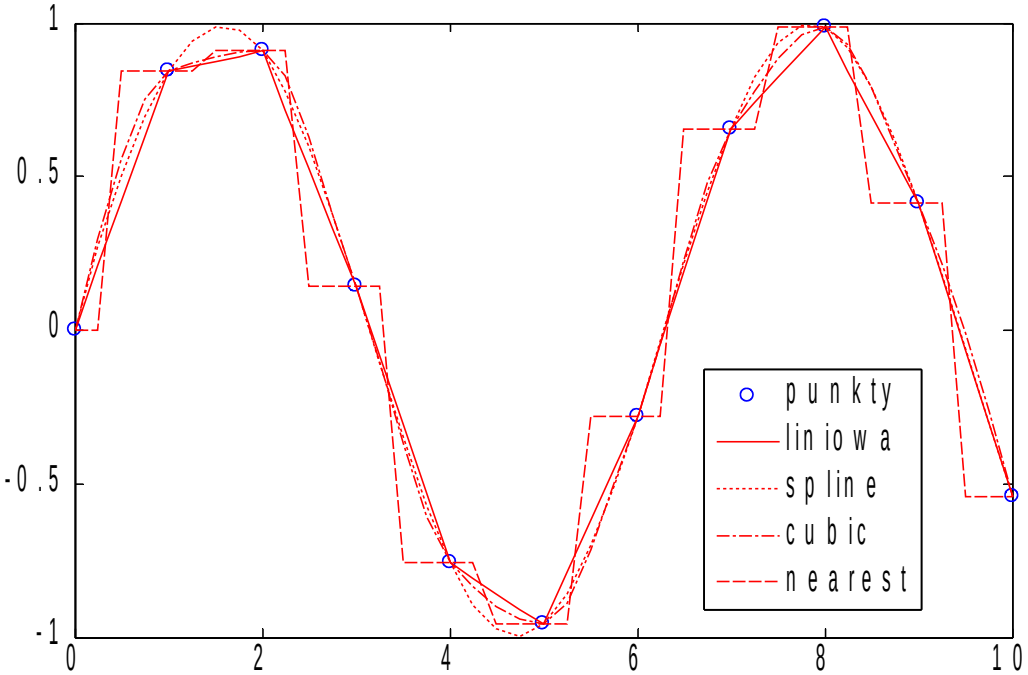


Aproksymacja, interpolacja i ekstrapolacja

- Aproksymacja
  - przybliżanie przebiegu funkcji za pomocą prostszych funkcji
- Interpolacja
  - wyznaczanie wartości funkcji (interpolującej) pomiędzy zadanymi punktami (węzłami interpolacji)
- Ekstrapolacja
  - prognozowanie wartości funkcji poza znanym zakresem

Interpolacja w Matlabie

- Interpolacja danych 1-D
  - funkcja `interp1`

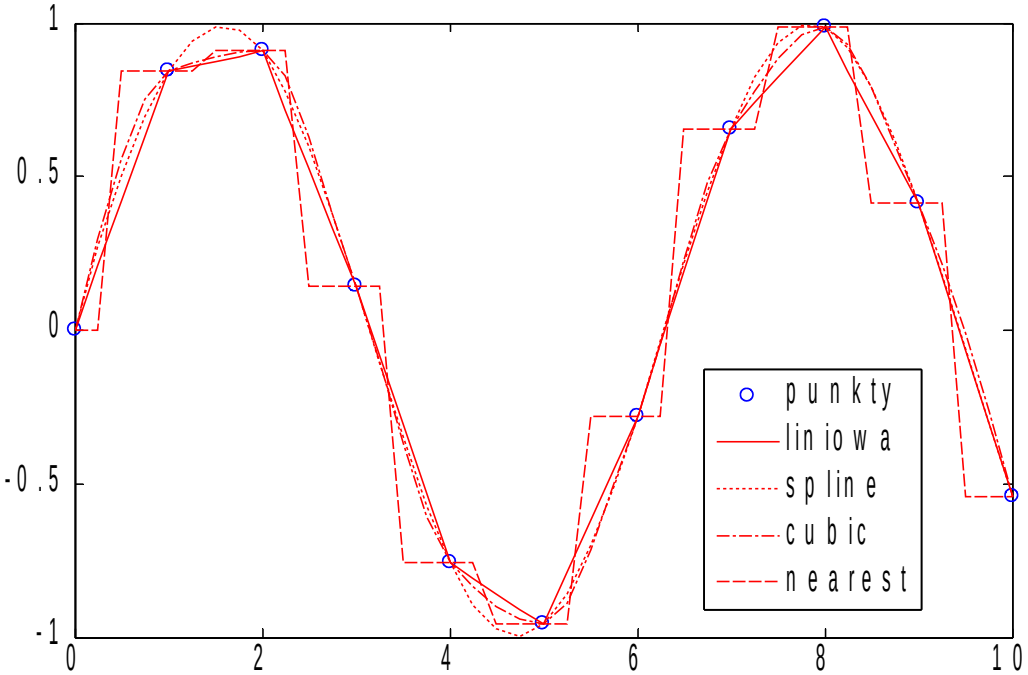



```
x=0:10;
y=sin(x);
xi=0:0.25:10;
yi=interp1(x,y,xi,'linear');
yi2=interp1(x,y,xi,'spline');
yi3=interp1(x,y,xi,'cubic');
yi4=interp1(x,y,xi,'nearest');
plot(x,y,'o',xi,yi,'r-', xi,yi2,'r:',xi,yi3,'r-.',xi,yi4,'r--')
legend('punkty','liniowa','spline','cubic','nearest')
```

Przykład za M. Kotulska, Informatyka – wykład 4 (PWr WPPT IB 2010/11)

Interpolacja w Matlabie

- Interpolacja danych 1-D
  - funkcja `interp1`



<u>Metody interpolacji</u>

- `linear` – liniowa
- `spline`
  - sklejenie wielomianów 3-st. gładkie, podatne na oscylacje
- `cubic`
  - sklejenie wielomianów 3-st. mniej gładkie, mniej oscylacji
- `nearest`
  - najbliższy sąsiad (schodki)

Użyteczne funkcje

<u>Interpolacja</u>

- `interp1, interp2` – 1- i 2-wymiarowa
- `interp3, interpN` – 3- i N-wymiarowa
- `interpft` – 1-wymiarowa z użyciem FFT
- `griddata` – 2-wymiarowa na siatce (por. `meshgrid`)
- `griddata3, griddatan` – 3- i N-wymiarowa na siatce
- `spline` – 1-wym. funkcją sklejaną (por. `interp1`)
- `ppval` – wylicza wartości funkcji sklejanej

<u>Ekstrapolacja</u>

- powyższe funkcję mogą także służyć ekstrapolacji danych (ostrożnie!)

Program na dziś

- Obliczenia numeryczne
  - algebra liniowa
  - aproksymacja i interpolacja
  - **uchwyty funkcji**
    - funkcje anonimowe
    - generatory funkcji
  - całkowanie i różniczkowanie
  - równania różniczkowe zwyczajne

Uchwyt funkcji

- Tablica n-wymiarowa
  - **uchwyt funkcji** (`function_handle`)
    - odnośnik do funkcji
  - Zastosowanie
    - przekazywanie uchwytu jako parametru funkcji
    - tworzenie funkcji anonimowych
    - generowanie funkcji sparametryzowanych

Uchwyt nazwanej funkcji

- Tworzenie obiektu `function_handle`

```
>> pierwiastek = @sqrt
pierwiastek =  @sqrt
```

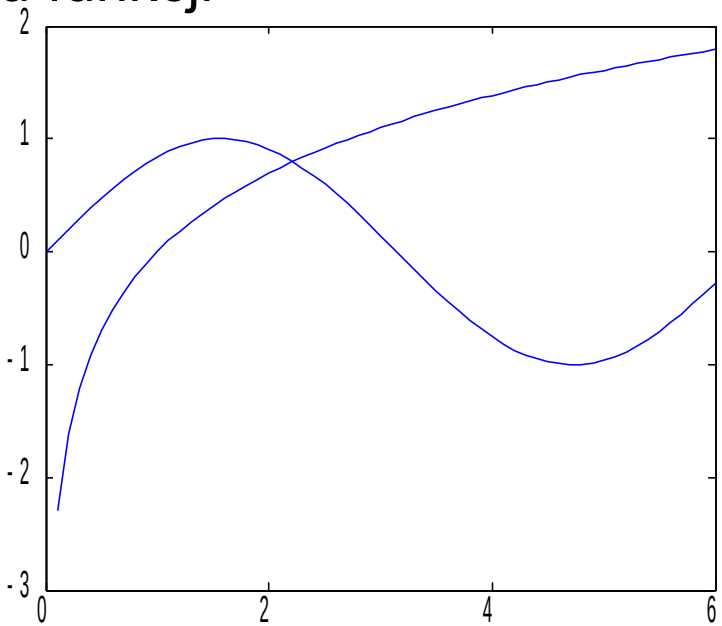
- Proste użycie

```
>> pierwiastek(9)
ans =   3
```

- Przekazanie uchwytu funkcji jako parametru funkcji

```
function wykres(fun,zakres)
%WYKRES rysuje wykres fun na zakres-ie
y = fun(zakres);
plot(zakres,y)
```

```
>> wykres(@sin,0:0.1:6), hold on
>> wykres(@log,0:0.1:6)
```



Uchwyt funkcji anonimowej

- Tworzenie funkcji anonimowej – sposób na nazywanie wyrażeń

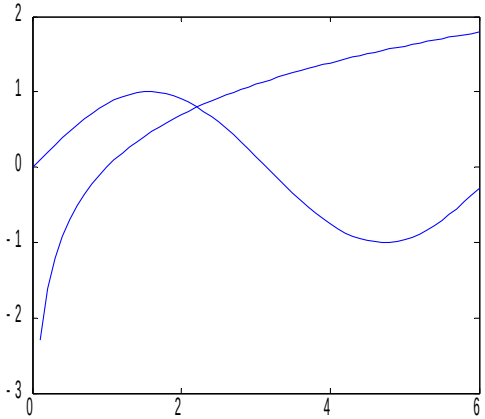
```
>> kwadrat = @(x)x.^2
kwadrat = @(x)x.^2
```

- Proste użycie

```
>> kwadrat(4)
ans = 16
```

- Funkcja wykres jako anonimowa

```
>> wykres = @(fun,zakres)plot(zakres,fun(zakres))
wykres = @(fun,zakres)plot(zakres,fun(zakres))

>> wykres(@sin,0:0.1:6), hold on
>> wykres(@log,0:0.1:6)
```



Generatory funkcji

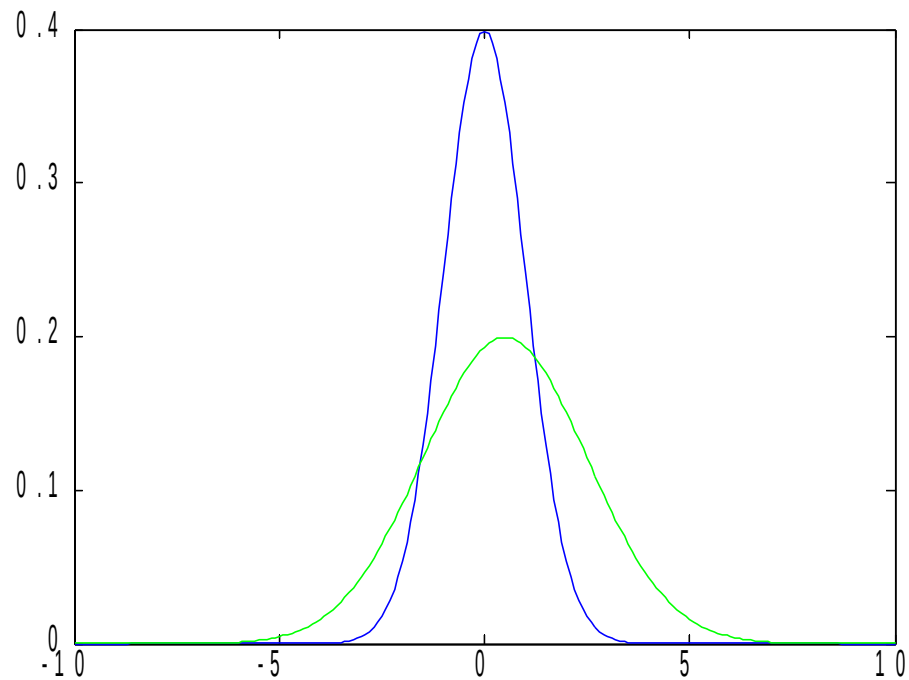
- Funkcja Gaussa (rozkład normalny)

$$\phi_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

```
function fg = utworzGaussa(mi,sigma)
%UTWORZGAUSSA tworzy funkcje gestosci rozkladu N(mi,sigma)

fg = @(x) 1/(sigma*sqrt(2*pi))*exp(-(x-mi).^2/(2*sigma^2));
```

```
>> N0_1 = utworzGaussa(0,1);
>> N05_2 = utworzGaussa(0.5,2);
>> x=-10:.1:10;
>> plot(x,N0_1(x),'b'), hold on
>> plot(x,N05_2(x),'g')
```

```
% N0_1 i N05_2 są uchwytami funkcji
%                        anonimowych
```



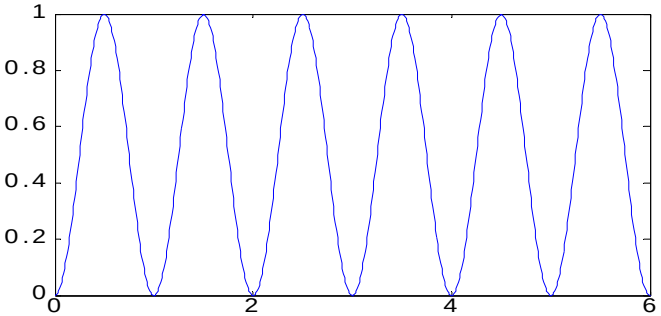
Program na dziś

- Obliczenia numeryczne
  - algebra liniowa
  - aproksymacja i interpolacja
  - uchwyty funkcji
  - **całkowanie i różniczkowanie numeryczne**
  - równania różniczkowe zwyczajne

Kumulacja glonów – model

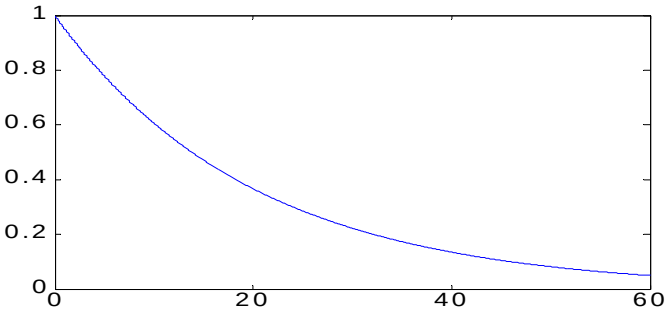
- Problem:
  - <u>kumulacja glonów w jeziorze</u>
  - model matematyczny:
    - dzienna oscylacja przyrostu:


```
sin(pi*t).^2
```



    - ubytek tlenu w wodzie:

```
exp(-0.05*t)
```



http://cornette.public.iastate.edu/V_I_13_fund_thm_appl.pdf

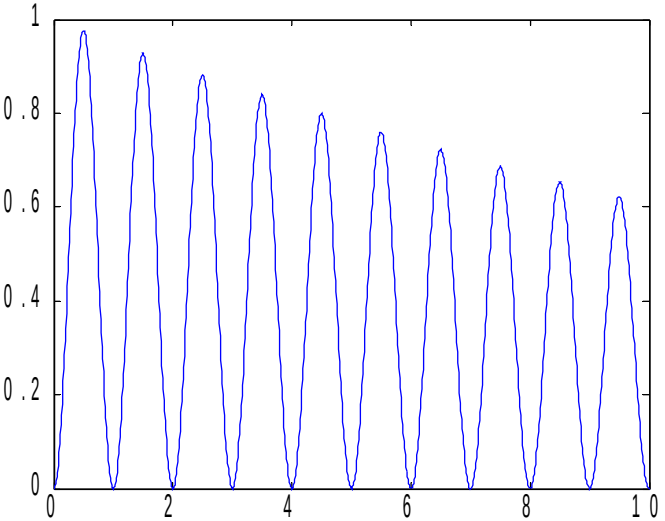
Kumulacja glonów – model (2)

- Problem:
  - <u>kumulacja glonów w jeziorze</u>
  - model matematyczny:
    - dzienna oscylacja przyrostu:

      `sin(pi*t).^2`
    - ubytek tlenu w wodzie:

      `exp(-0.05*t)`
    - **kompletny model:**

      `y = exp(-0.05*t).*sin(pi*t).^2`



http://cornette.public.iastate.edu/V_I_13_fund_thm_appl.pdf

Kumulacja glonów – problem

- Zadania
  - ilość glonów po *n* dniach kumulacji
  - czy i po jakim czasie populacja się ustabilizuje

- Rozwiązanie
  - policzenie całki
    - analityczne (na papierze, toolbox Symbolic Math)
    - **numeryczne**

Całkowanie numeryczne
Metoda trapezów

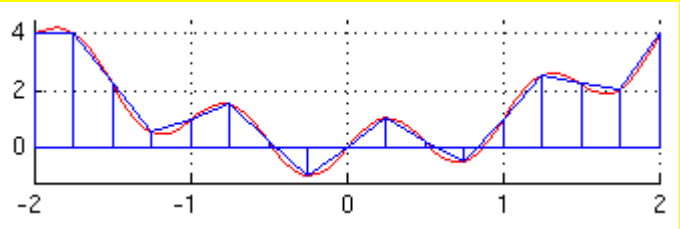
- Zadania
  - ilość glonów po *n* dniach kumulacji
    - po 7 dniach:

```
>> t=0:0.01:7;
>> y=exp(-0.05*t).*sin(pi*t).^2;
>> Y7 = trapz(t,y)
Y7 = 2.9529
```

    - po 30 dniach:

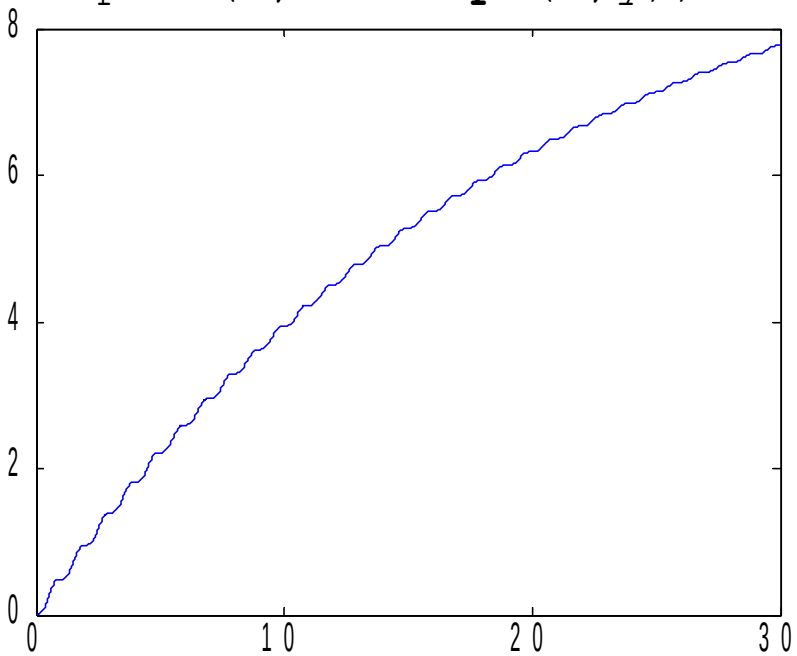
```
>> t=0:0.01:30;
>> y=exp(-0.05*t).*sin(pi*t).^2;
>> Y30 = trapz(t,y)
Y30 = 7.7682
```

`trapz, cumtrapz` – metoda trapezów



http://pl.wikipedia.org/wiki/Ca%C5%82kowanie_numeryczne#Metoda_trapez.C3.B3w

`plot(t,cumtrapz(t,y))`



Całkowanie numeryczne
Metoda adaptacyjna

- Dokładniejsze obliczanie całki oznaczonej
  - metoda adaptacyjna Simpsona

- Idea:
  - podajemy
    - całkowaną funkcję, np. `exp(-0.05*t).*sin(pi*t).^2`
    - podajemy zakres [a, b], np. `[0, 30]`
    - podajemy oczekiwany błąd przybliżenia całki, std: `1e-6`
  - przybliżamy całkowaną funkcję parabolami
  - obliczamy błąd przybliżenia
    - jeśli za duży dzielimy przedział całkowania na pół

Całkowanie numeryczne
Tolerancja błędu

```
Y = quad(FUN,A,B,TOL)

- FUN - uchwyt do funkcji
- A,B - zakres
- TOL - tolerancja błędu bezwzględnego (std: 1e-6)
```

```
>> wzrost_glonow = @(t) exp(-0.05*t).*sin(pi*t).^2
wzrost_glonow = @(t)exp(-0.05*t).*sin(pi*t).^2

>> quad(wzrost_glonow,0,30)
ans = 5.084959016383285

>> quad(wzrost_glonow,0,30,1e-7)
ans = 7.292598184528123

>> quad(wzrost_glonow,0,30,1e-8)
ans = 7.768206472952495
```

```
% >> trapz(t,wzrost_glonow(t))
% ans = 7.768206471062230
% Uwaga! dobra dokładność trapz
%  dzięki dt=0.01
```

```
>> quadl(wzrost_glonow,0,30)
ans = 7.768206471093651
```

`quadl` – metoda adaptacyjna Lobatto
bardziej efektywna dla f. gładkich

Całkowanie numeryczne
Granica w nieskończoności

- Czy przyrost glonów zakończy się?
  - granica całki oznaczonej w nieskończoności

```
>> quadl(wzrost_glonow,0,30)
ans =    7.7682

>> quadl(wzrost_glonow,0,90)
ans =    9.8883

>> quadl(wzrost_glonow,0,365)
ans =    9.9994

>> quadl(wzrost_glonow,0,1000)
ans =    9.9994

>> quadl(wzrost_glonow,0,10000)
ans =  1.7468e-017                    % zły wynik!

>> quadl(wzrost_glonow,0,Inf)
Warning: Minimum step size reached; singularity possible.
> In quadl at 104
ans =   NaN                           % jakiś problem

>> quadgk(wzrost_glonow,0,Inf)
ans =    9.9994        % ok!
```

`quadgk` – metoda Gaussa-Kronroda
- bardziej efektywna dla oscylacji
- działa na przedziałach nieskończonych

Różniczkowanie numeryczne

- Pochodna jest granicą ilorazu różnicowego

$$\lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$$

- Iloraz różnicowy opisuje
  - przyrost funkcji $\Delta y = f(x_0 + \Delta x) - f(x_0)$
  - na przedziale Δx

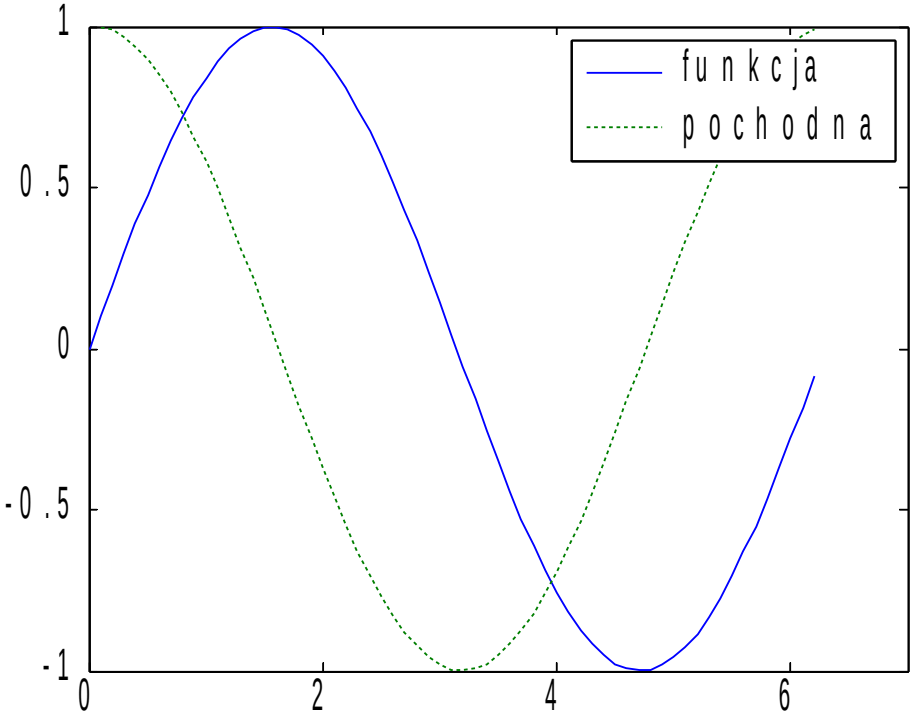
$$\frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} = \frac{\Delta y}{\Delta x}$$

- Mając wektor argumentów i wartości funkcji
  - łatwo policzymy iloraz różnicowy przybliżający pochodną

```
>> ypoch = diff(y)./diff(x)
```

Różniczkowanie numeryczne przykład

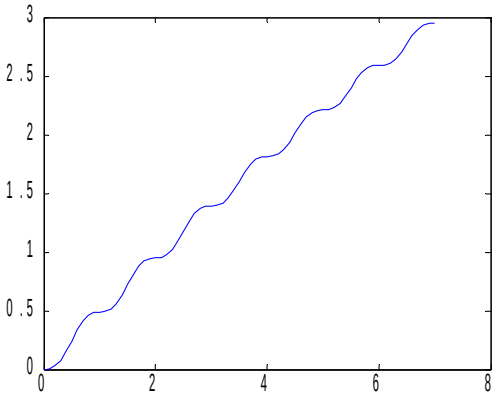
```
>> x=[0:0.1:2*pi];
>> funkcja=sin(x);

>> pochodna=diff(funkcja)./diff(x);

>> plot(x, funkcja, x(2:end),pochodna,':');
>> legend(' funkcja ', ' pochodna ');
```



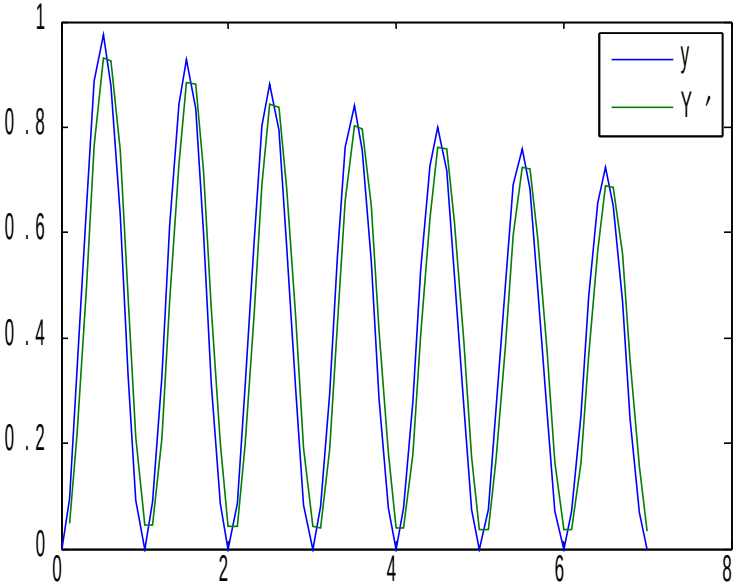
Przykład za www.if.pwr.wroc.pl/~kotulska/informatyka/info6.ppt

Różniczkowanie numeryczne przykład (2)

```
>> t= 0:0.1:7;
>> y = exp(-0.05*t).*sin(pi*t).^2;

>> Y = cumtrapz(t,y);

>> plot(t,Y);
```




```
>> Ypoch = diff(Y)./diff(t);

>> plot(t,y,t(2:end),Ypoch)
>> legend('y','Y\prime')
```



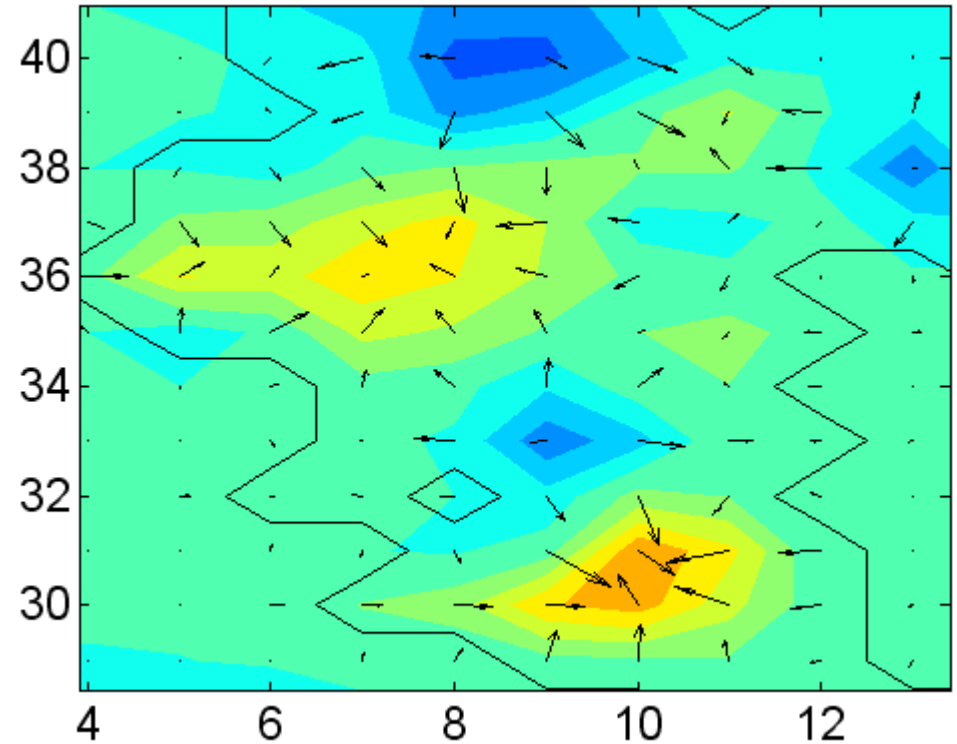
Gradient numeryczny

```
% F - wolumetryczna mapa potencjału wokół kanału białkowego

% Rysujemy kontury przekroju wzdłużnego białka
>> contour(n_mask(:,:,round(zMax/2)),1,'LineColor','k'); hold on;

% oraz wykres izoliniowy potencjału:
>> contourf(F(:,:,round(zMax/2)),[-1:.1:1]); shading flat;

% Liczymy gradient w przekroju przez środek (zMax/2):
>> [px,py] = gradient(F(:,:,round(zMax/2)));

% Rysujemy kierunki gradientu:
>> quiver(px,py,'k')
```

`gradient` – gradient numeryczny

pole wektorowe o składowych będących pochodnymi cząstkowymi w kierunku wersorów układu współrzędnych

$$\nabla F = \frac{\partial F}{\partial x}\hat{i} + \frac{\partial F}{\partial y}\hat{j} + \frac{\partial F}{\partial z}\hat{k} + \ldots$$



Użyteczne funkcje

<u>Całkowanie numeryczne</u>

- `quad` - metoda adaptacyjna Simpsona
  do niższych dokładności, f. niegładkie
- `quadl` - metoda adaptacyjna Lobatto
  do wyższych dokładności, f. gładkie
- `quadgk` - metoda Gaussa-Kronroda
  do wyższych dokładności, f. oscylujące,
  nieskończone przedziały całkowania
- `dblquad, triplequad` - całki podwójne i potrójne
- `trapz, cumtrapz` - metoda trapezów

<u>Różniczkowanie</u>

- `diff` - różnica elementów wektora
- `gradient` - przybliżony gradient

Program na dziś

- Obliczenia numeryczne
  - algebra liniowa
  - aproksymacja i interpolacja
  - uchwyty funkcji
  - całkowanie i różniczkowanie
  - **równania różniczkowe zwyczajne**
    - zagadnienie początkowe
    - równania wyższego rzędu

Zające i rysie

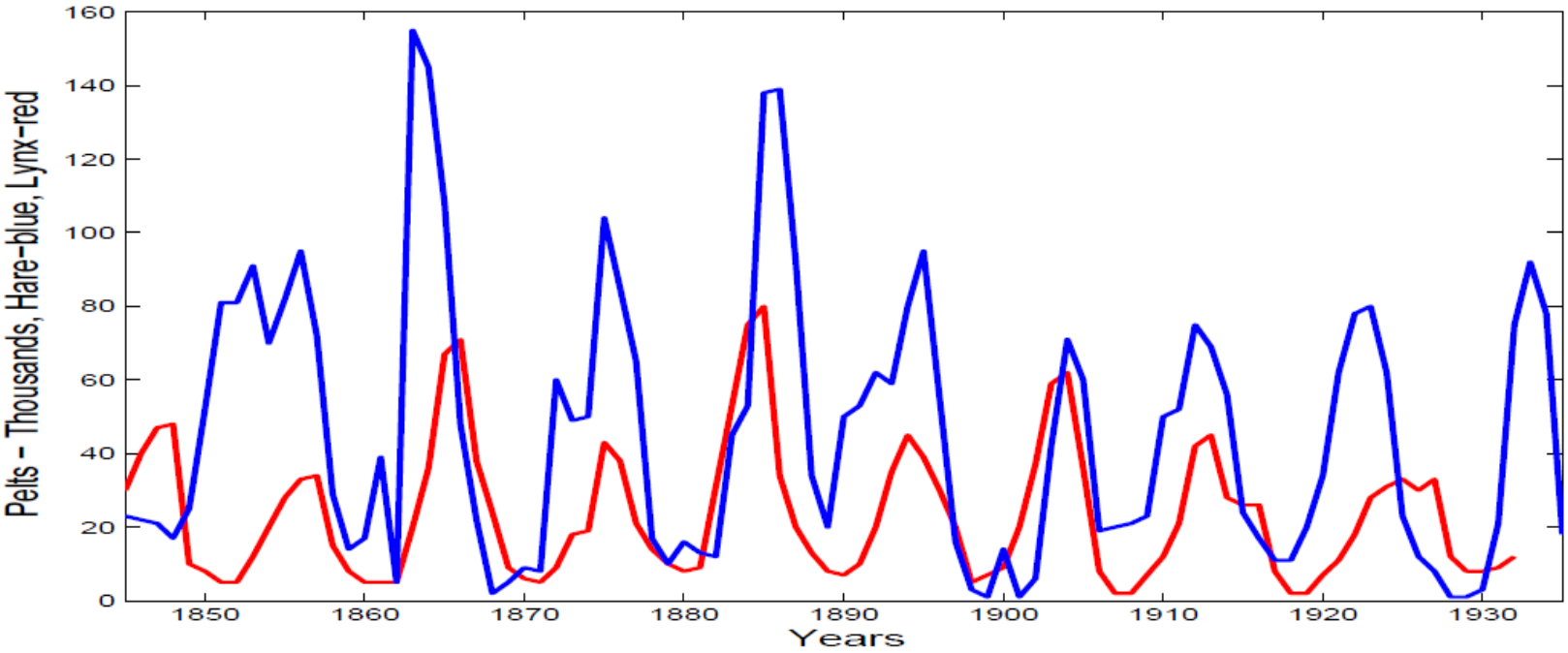
- Na przełomie XIX/XX wieku w Kanadzie
  - Hudson Bay Company skupowała futra zwierząt
    - m.in. zajęcy i rysi

http://en.wikipedia.org/wiki/Snowshoe_hare

fot. Michael Zahra
http://en.wikipedia.org/wiki/Canada_lynx

Zające i rysie (2)

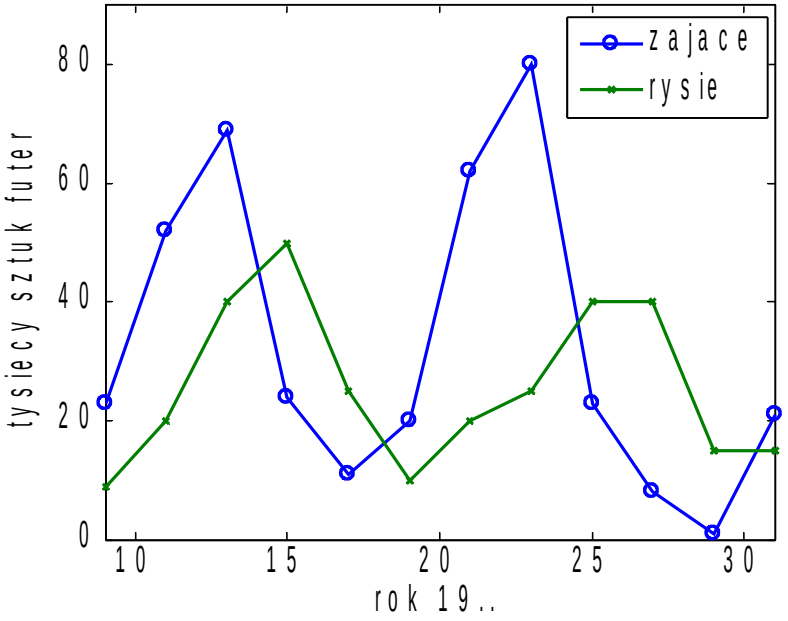
- Na przełomie XIX/XX wieku w Kanadzie
  - Hudson Bay Company skupowała futra zwierząt
  - skuteczność polowań zależała w dużej mierze od stanu populacji
    - na przestrzeni 90 lat zaobserwowano interesujące fluktuacje:

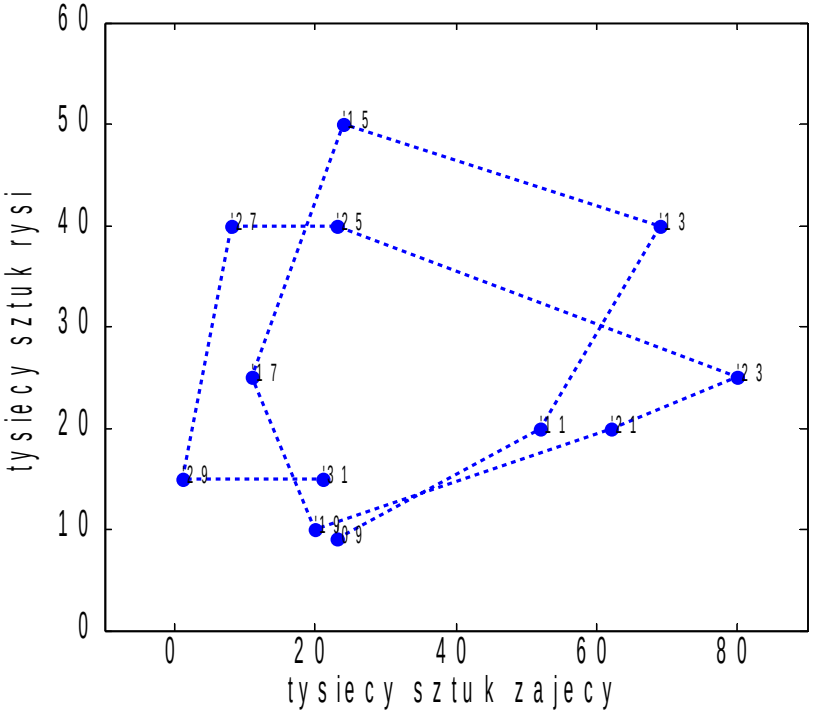


rys. Cornette&Ackermann: http://cornette.public.iastate.edu/V_I_07_der_trig.pdf

Dynamika populacji zajęcy i rysi

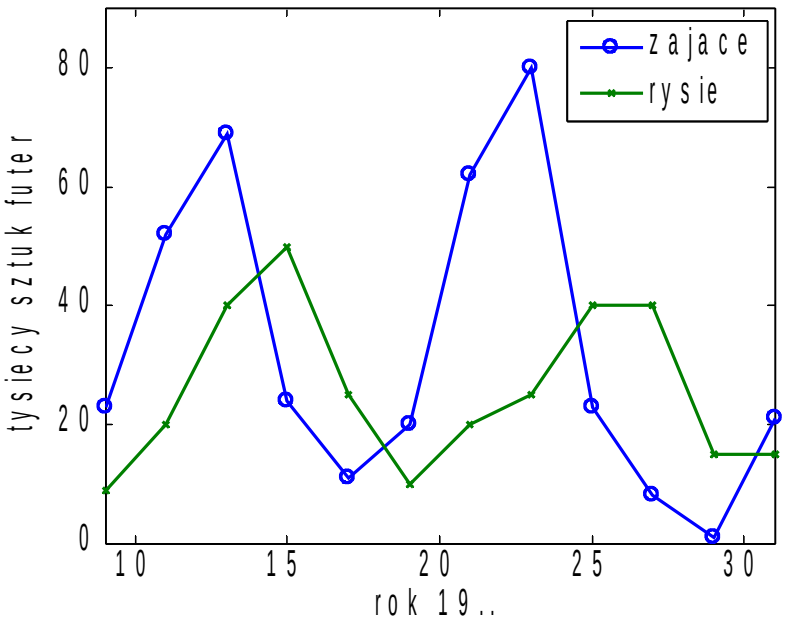
- Dane z lat 1909-1931
  - za D.A. McLulich, "Fluctuations in the numbers of the varying hare". Univ. Toronto Studies, Biol. Ser:, No. 43 (1937)

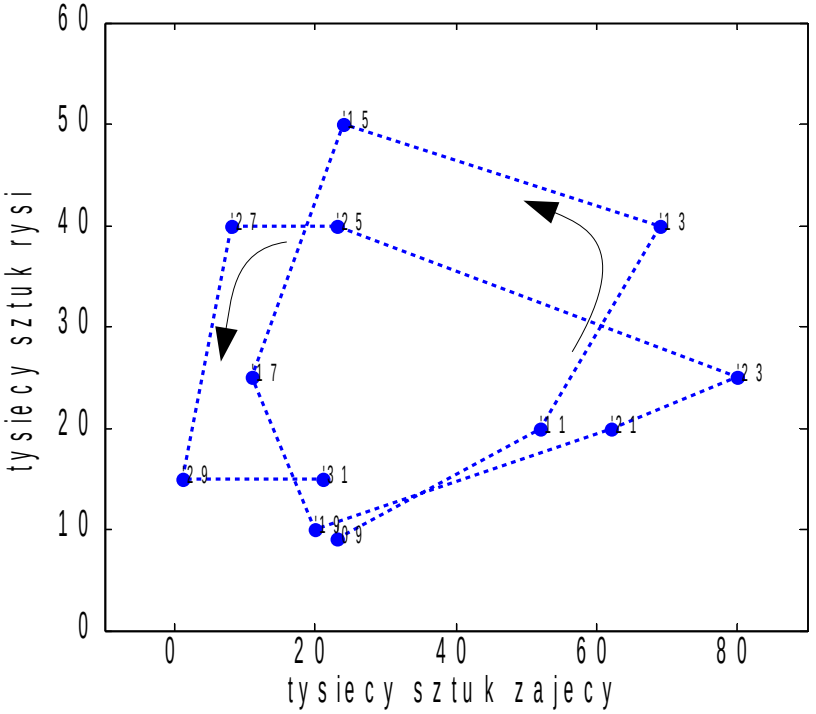




Dynamika populacji zajęcy i rysi

- Interpretacja:
  - wzrost populacji zajęcy (1909-13, 1919-23)
    - pociąga za sobą wzrost populacji rysiów (1909-15, 1919-25)
  - wzrost liczby rysi powoduje spadek populacji zajęcy (1913-17, 1923-29)
  - spadek populacji zajęcy powoduje spadek populacji rysi (1915-19, 1927-29)





Model dynamiki populacji drapieżnik-ofiara

- Dwa gatunki – **ofiara** (np. zając) i **drapieżnik** (np. ryś)

- Populacja **ofiar**
  - **rośnie** proporcjonalnie do liczby ofiar
    (im większa populacja, tym więcej rodzi się)
  - **spada** proporcjonalnie do
    - liczby ofiar (im większa populacja, tym więcej umiera)
    - liczby drapieżników (im większa populacja drapieżników, tym więcej ofiar ginie)

- Populacja **drapieżników**
  - **rośnie** proporcjonalnie do
    - liczby drapieżników (im większa populacja, tym więcej rodzi się)
    - liczby ofiar (im większa populacja ofiar, tym więcej drapieżników może się urodzić)
  - **spada** proporcjonalnie do liczby drapieżników
    (im większa populacja, tym więcej umiera)

Model Lotki-Volterry

- Populacja drapieżców – **v(t)**

- Populacja ofiar – **u(t)**

Lotka, A.J. (1920), "Analytical Note on Certain Rhythmic Relations in Organic Systems", Proc. Natl. Acad. Sci. U.S., 6, 410-415, http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1084562/pdf/pnas01916-0016.pdf
A.J.Lotka (1925), 'Elements of Physical Biology' reprinted by Dover in 1956 as Elements of Mathematical Biology. http://ia600307.us.archive.org/35/items/elementsofphysic017171mbp/elementsofphysic017171mbp.pdf
Volterra, V., "Variazioni e fluttuazioni del numero d'individui in specie animali conviventi", Mem. Acad. Lincei Roma, 2, 31-113, (1926)

Model Lotki-Volterry

- Populacja drapieżców – **v(t)**
  - rośnie proporcjonalnie do liczby ofiar i drapieżników: **v'(t) = c * u(t)*v(t)**
  - spada proporcjonalnie do liczby drapieżników: **u'(t) = -d * v(t)**
  - mamy więc:

$$v'(t) = c*u(t)*v(t) - d*v(t) = [c*u(t) - d] * v(t)$$

- Populacja ofiar – **u(t)**

Lotka, A.J. (1920), "Analytical Note on Certain Rhythmic Relations in Organic Systems", Proc. Natl. Acad. Sci. U.S., 6, 410-415, http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1084562/pdf/pnas01916-0016.pdf
A.J.Lotka (1925), 'Elements of Physical Biology' reprinted by Dover in 1956 as Elements of Mathematical Biology. http://ia600307.us.archive.org/35/items/elementsofphysic017171mbp/elementsofphysic017171mbp.pdf
Volterra, V., "Variazioni e fluttuazioni del numero d'individui in specie animali conviventi", Mem. Acad. Lincei Roma, 2, 31-113, (1926)

Model Lotki-Volterry

- Populacja drapieżców – **v(t)**
  - rośnie proporcjonalnie do liczby ofiar i drapieżników: $\mathbf{v'(t) = c * u(t)*v(t)}$
  - spada proporcjonalnie do liczby drapieżników: $\mathbf{u'(t) = -d * u(t)}$
  - mamy więc:

$$\mathbf{v'(t) = c*u(t)*v(t) - d*v(t) = [c*u(t) - d] * v(t)}$$

- Populacja ofiar – **u(t)**
  - rośnie proporcjonalnie do liczby ofiar: $\mathbf{u'(t) = a * u(t)}$
  - spada proporcjonalnie do liczby ofiar i drapieżników: $\mathbf{u'(t) = -b*u(t)*v(t)}$
  - mamy więc:

$$\mathbf{u'(t) = a*u(t) - b*u(t)*v(t) = [a - b*v(t)] * u(t)}$$

Lotka, A.J. (1920), "Analytical Note on Certain Rhythmic Relations in Organic Systems", Proc. Natl. Acad. Sci. U.S., 6, 410-415, http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1084562/pdf/pnas01916-0016.pdf
A.J.Lotka (1925), 'Elements of Physical Biology' reprinted by Dover in 1956 as Elements of Mathematical Biology. http://ia600307.us.archive.org/35/items/elementsofphysic017171mbp/elementsofphysic017171mbp.pdf
Volterra, V., "Variazioni e fluttuazioni del numero d'individui in specie animali conviventi", Mem. Acad. Lincei Roma, 2, 31-113, (1926)

Model Lotki-Volterry (2)

- Układ równań różniczkowych zwyczajnych

$$u'(t) = a*u(t) - b*u(t)*v(t) = [a - b*v(t)] * u(t)$$

$$v'(t) = c*u(t)*v(t) - d*v(t) = [c*u(t) - d] * v(t)$$

Model Lotki-Volterry (2)

- Układ równań różniczkowych zwyczajnych

$$u'(t) = a*u(t) - b*u(t)*v(t) = [a - b*v(t)] * u(t)$$

$$v'(t) = c*u(t)*v(t) - d*v(t) = [c*u(t) - d] * v(t)$$

<u>Przypomnienie:</u>

- równania różniczkowe (ang. *Differential Equations*)
  - występują <u>pochodne</u> szukanej funkcji
- równania różniczkowe zwyczajne (ang. *Ordinary Differential Equations – ODE*)
  - zależą od tylko jednej zmiennej niezależnej

Równania różniczkowe

- Układ równań różniczkowych
  - opis dynamiki systemu
  - baaardzo szerokie zastosowanie:
    - m.in. w fizyce, inżynierii, biologii, medycynie, ekonomii

- Zagadnienie początkowe (*initial value problem*)
  - jak wygląda ewolucja systemu opisanego
    - układem równań różniczkowych
    - wartościami początkowymi

Rozwiązanie zagadnienia początkowego

$$y'(t) = f(t,y(t))$$

- Analityczne
  - znaleźć funkcję: $y(t)$
    - która spełnia warunek początkowy: $y(t0) = y0$
  - czy rozwiązanie istnieje i jest jednoznaczne?

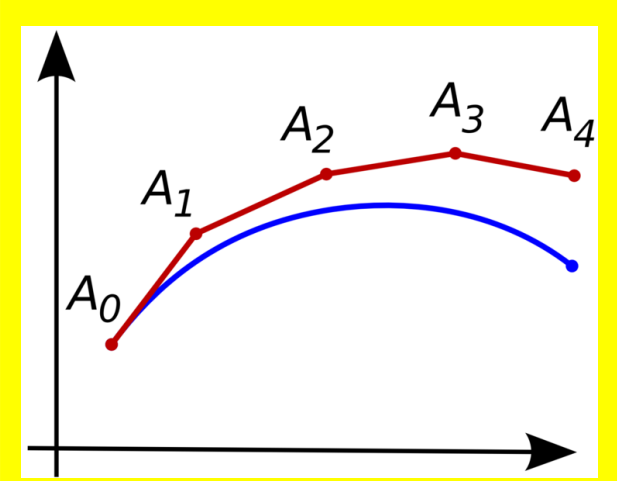
Rozwiązanie zagadnienia początkowego

$$y'(t) = f(t,y(t))$$

- Analityczne
  - znaleźć funkcję: $y(t)$
    - która spełnia warunek początkowy: $y(t0) = y0$
  - czy rozwiązanie istnieje i jest jednoznaczne?

- Numeryczne
  - aproksymacja kolejnych wartości $y(t)$
  - idea (metoda Eulera):

$$y(t+1) = y(t) + y'(t)$$



http://en.wikipedia.org/wiki/Euler_method

Rozwiązanie zagadnienia początkowego w MATLABIE

- Metoda Eulera nie jest zbyt dokładna ani stabilna
- Przeważnie stosuje się bardziej zaawansowane schematy
  - w MATLABie zostały zaimplementowane m.in. metody
    - Rungego-Kutty rzędu 4 i 5 (`ode45`) – standardowy, średnia dokł.
    - Rungego-Kutty rzędu 2 i 3 (`ode23`) – do rozw. niskiej dokładności
    - Adamsa-Bashforda-Moultona (`ode113`) – do rozwiązań gładkich

Rozwiązanie zagadnienia początkowego w MATLABIE

- Metoda Eulera nie jest zbyt dokładna ani stabilna
- Przeważnie stosuje się bardziej zaawansowane schematy
  - w MATLABie zostały zaimplementowane m.in. metody
    - Rungego-Kutty rzędu 4 i 5 (`ode45`) – standardowy, średnia dokł.
    - Rungego-Kutty rzędu 2 i 3 (`ode23`) – do rozw. niskiej dokładności
    - Adamsa-Bashforda-Moultona (`ode113`) – do rozwiązań gładkich

```
[tout,yout] = odeXX(odefun, tspan, y0, options, params)
```

- `odefun – uchwyt funkcji liczącej pochodną`
- `tspan – zakres zmiennej niezależnej [t0 t_koniec]`
- `y0 – wartości początkowe`
- `options – parametry algorytmu, m.in tolerancja błędu`
- `params – parametry przekazywane odefun`

odefun: Lotka-Volterra

- Układ równań różniczkowych zwyczajnych

$$u'(t) = [a - b*v(t)] * u(t)$$

$$v'(t) = [c*u(t) - d] * v(t)$$

```
function yp = lv(t,y,p)
%LV  Model drapieżnik-ofiara Lotki-Volterry

u = y(1);     % wektor y zawiera wartości u(t) i v(t)
v = y(2);

a = p(1);     % wektor p zawiera parametry układu
b = p(2);
c = p(3);
d = p(4);

up = (a-b*v)*u;      % liczymy pochodne
vp = (c*u-d)*v;

yp = [up; vp];   % zwracamy wektor yp z wartosciami u'(t) i v'(t)
```

odefun: Lotka-Volterra

- Układ równań różniczkowych zwyczajnych

$$u'(t) = [a - b*v(t)] * u(t)$$

$$v'(t) = [c*u(t) - d] * v(t)$$

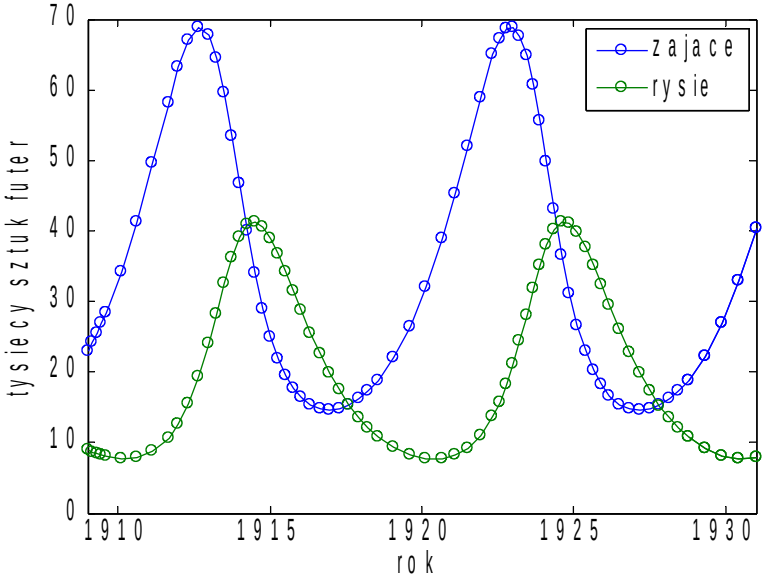
```
function yp = lv(t,y,p)
%LV  Model drapieżnik-ofiara Lotki-Volterry

% Zwięźle:
yp = [(p(1)-p(2)*y(2))*y(1); (p(3)*y(1)-p(4))*y(2);];
```

Solver ODE

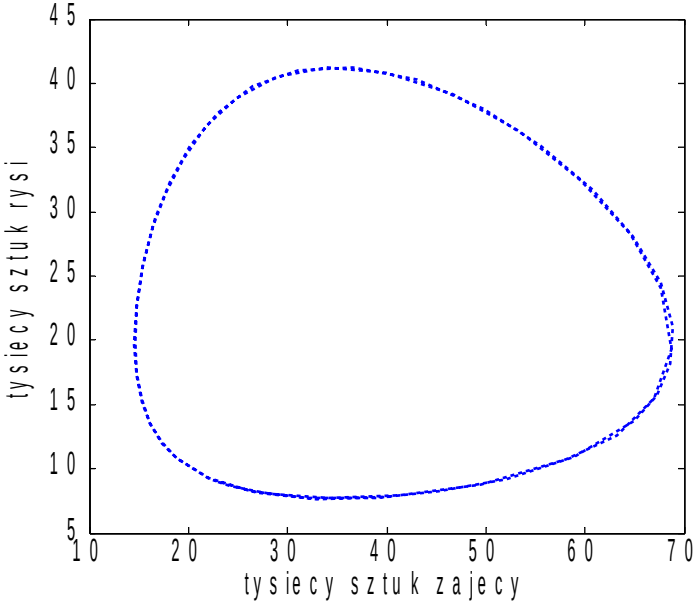
```
model - lv.m:
    u' =  [a - b*v] * u
    v' =  [c*u - d] * v

warunki początkowe: t0 = 1909, u(0) = 23, v(0) = 9;

współczynniki
- urodzeń ofiar: a = 0.6
- zgonów ofiar:  b = 0.03
- urodzeń drapieżników: c = 0.02
- zgonów drapieżników:  d = 0.7

% [tout,yout] = odeXX(odefun, tspan, y0, options, params)
>>[t,y]=ode45(@lv,[1909 1931],[23 9],[],[0.6 0.03 0.02 0.7]);
```

Rozwiązanie numeryczne

Populacje w czasie u(t), v(t)

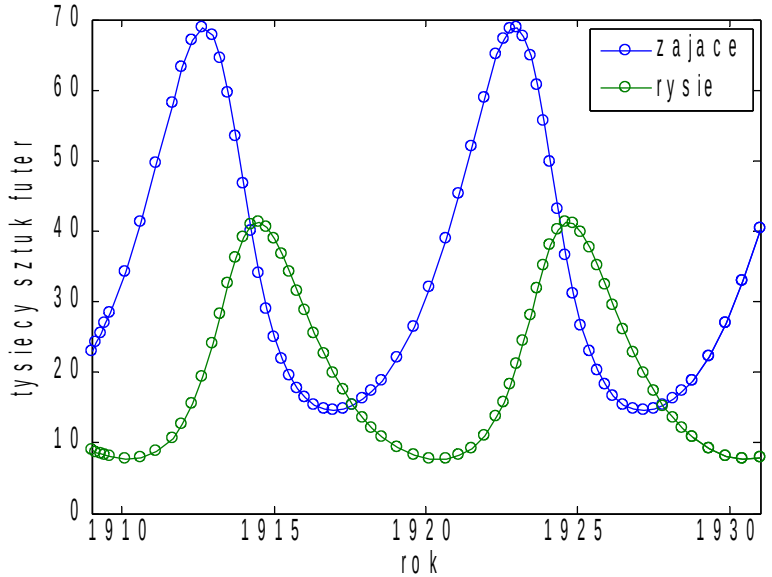


Wykres fazowy v(u)

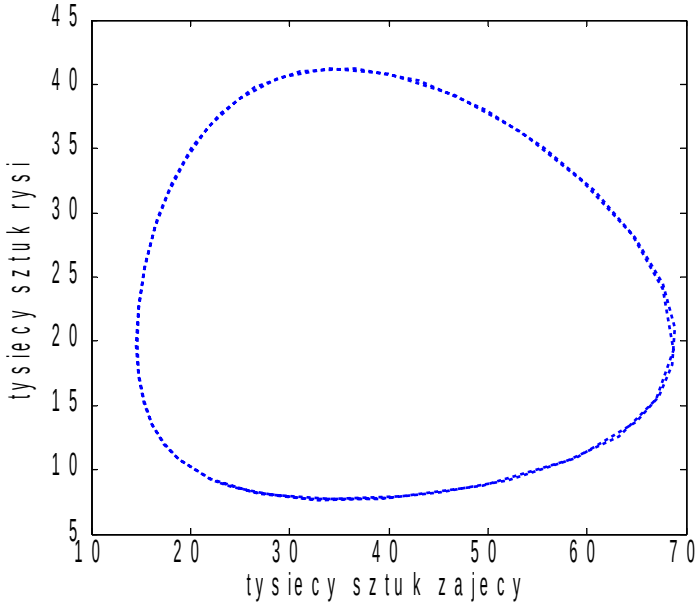


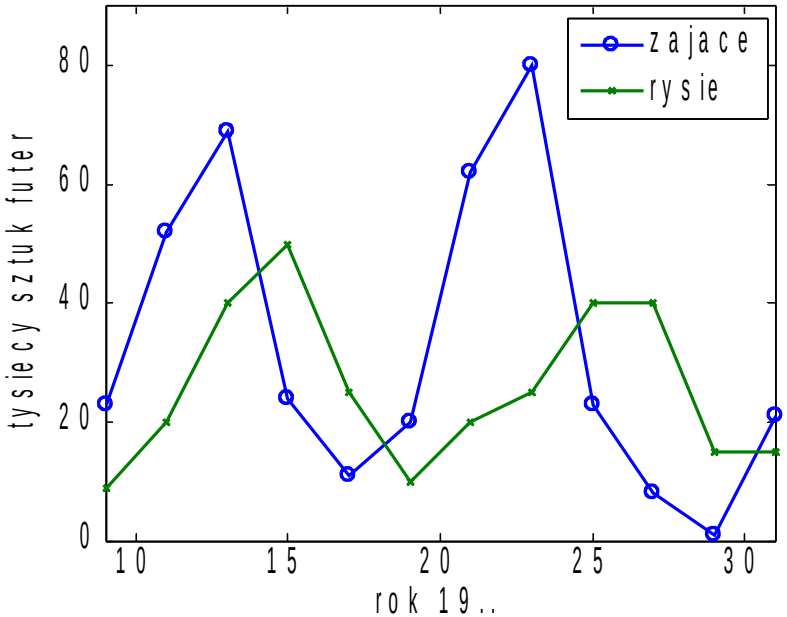
Rozwiązanie numeryczne

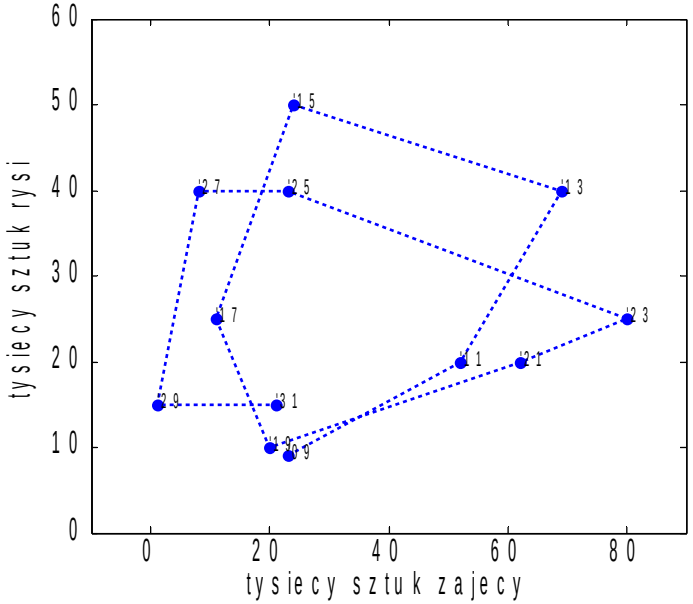
Populacje w czasie u(t), v(t)



Wykres fazowy v(u)





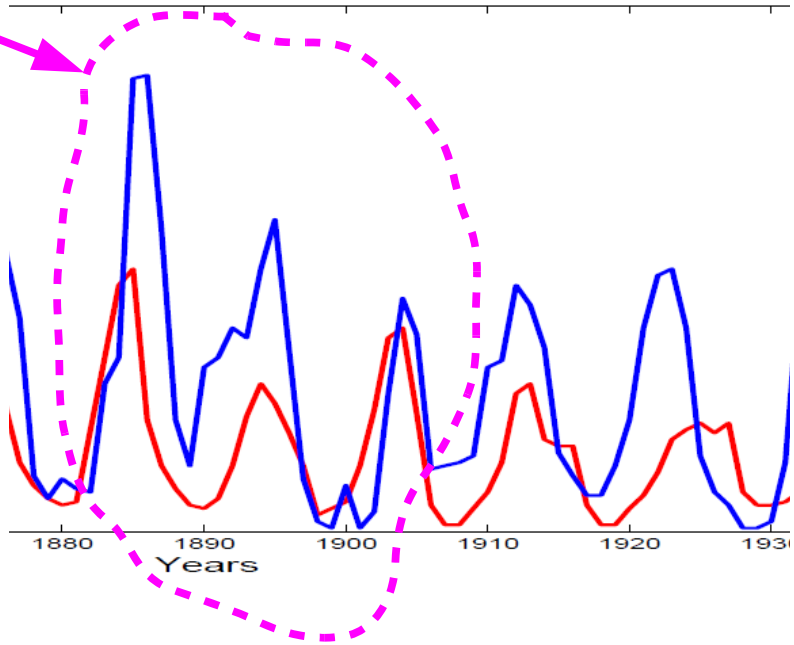


Uwaga!

- Porównanie danych
  - Hudson Bay Company
  - modelu Lotki-Volterry
- wypadło dość przekonująco

Uwaga!

- Porównanie danych
  - Hudson Bay Company
  - modelu Lotki-Volterry
- wypadło dość przekonująco, ale...
  - można znaleźć okresy, gdy dane niezgodne z modelem
  - sam model L-V bardzo uproszczony, m.in.
    - nie uwzględnia innych konsumentów zajęcy
    - brakuje ograniczenia na pojemność ekosystemu



Układ bezwymiarowy

- Często ważniejsza od przewidzenia dokładnych wartości składowych układu jest analiza dynamiki systemu
- W przypadku Lotki-Volterry będzie to łatwiejsze
  - gdy zredukujemy liczbę parametrów stosując <u>ubezwymiarowienie:</u>
  - niech

    s = at, x(s) = c/d*u(t), y(s) = b/a*v(t), alpha=d/a
  - wtedy

    x'(s) = [1 – y(s)] * x(s)

    y'(s) = [x(s) – 1] * alpha * y(s)
    - gdzie: **alpha** – jest jedynym parameterem

http://pl.wikipedia.org/wiki/R%C3%B3wnanie_Lotki-Volterry

Układ bezwymiarowy

- Często ważniejsza od przewidzenia dokładnych wartości składowych układu jest analiza dynamiki systemu
- W przypadku Lotki-Volterry będzie to łatwiejsze
  - gdy zredukujemy liczbę parametrów stosując <u>ubezwymiarowienie:</u>
  - niech

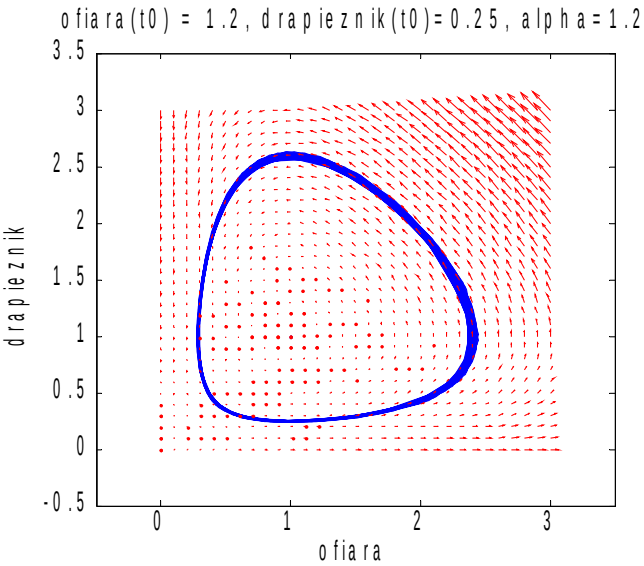
    s = at, x(s) = c/d*u(t), y(s) = b/a*v(t), alpha=d/a
  - wtedy

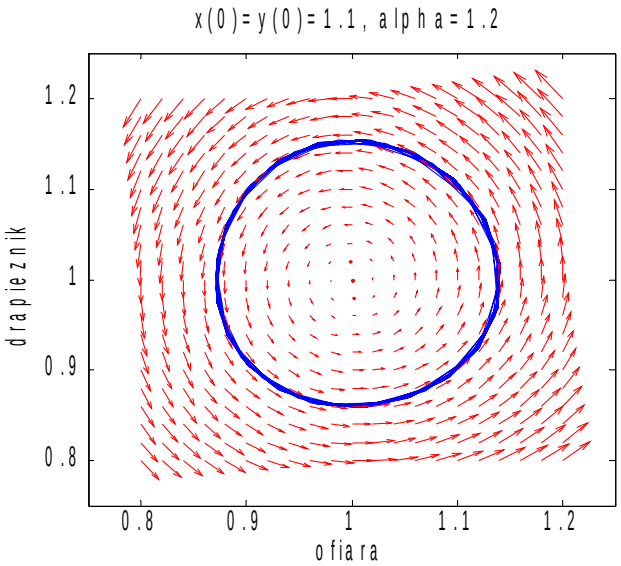
    x'(s) = [1 – y(s)] * x(s)

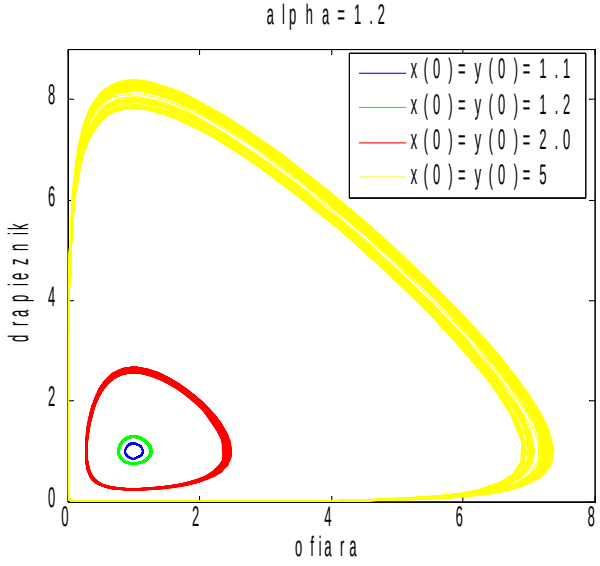
    y'(s) = [x(s) – 1] * alpha * y(s)
    - gdzie: **alpha** – jest jedynym parameterem

Trajektorie i portret fazowy

```
>> [t,y]=ode45(@lvbw2,[0 1000],[1.2 0.25],[],1.2);
>> plot(y(:,1),y(:,2));

>> [u, v] = meshgrid(0:1:20,0:0.5:10);
>> up = (1-v).*u;                          % pochodne, liczone macierzowo
>> vp = (u-1).*0.5.*v;
>> hold on; quiver(u,v,up,vp,1.5,'r')      % kierunek pola
```







Punkt krytyczny

x'(s) = [1 – y(s)] * x(s)

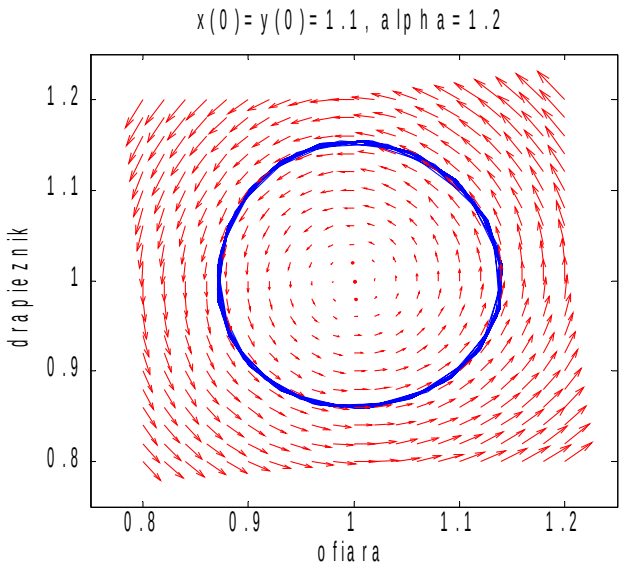
y'(s) = [x(s) – 1] * alpha * y(s)

- <u>Punkt krytyczny</u> gdy:

  x'(s) = y'(s) = 0

- czyli

  (x,y) = (1,1) jest punktem stacjonarnym



http://mathworld.wolfram.com/Lotka-VolterraEquations.html

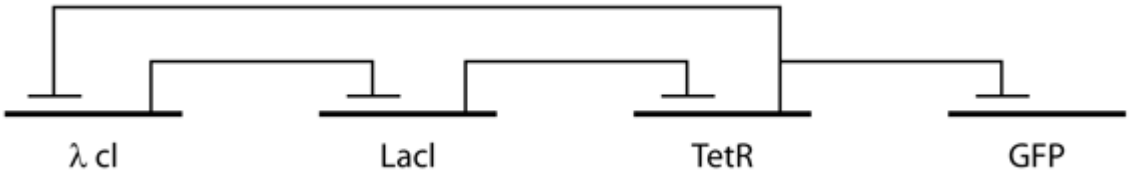
Po co uczyć się o zającach i rysiach?

- Modele różniczkowe są szeroko wykorzystywane w biomedycynie, np. w modelowaniu:
  - układu krążenia
  - cykli komórkowych
  - elektrofizjologii tkanek
  - regulacji ekspresji genów
  - transportu jonów

  oraz projektowaniu nowych funkcji komórek (biologia syntetyczna)
- Istnieją bazy modeli:
  - http://models.cellml.org/cellml
  - http://www.ebi.ac.uk/biomodels-main/

Oscylator Elowitza (represylator)

- Syntetyczna sieć regulacji ekspresji genów
  - bakterie *e.coli* zostały zaprogramowane do oscylacyjnego świecenia
  - pomysł opiera się na pętli sprzężenia zwrotnego
    - podobnej do obserwowanej pomiędzy zającami a rysiami



rys: http://en.wikipedia.org/wiki/Repressilator

A Synthetic Oscillatory Network of Transcriptional Regulators; Michael Elowitz and Stanislas Leibler; Nature. 2000 Jan 20;403(6767):335-8. http://www.elowitz.caltech.edu/publications/Repressilator.pdf

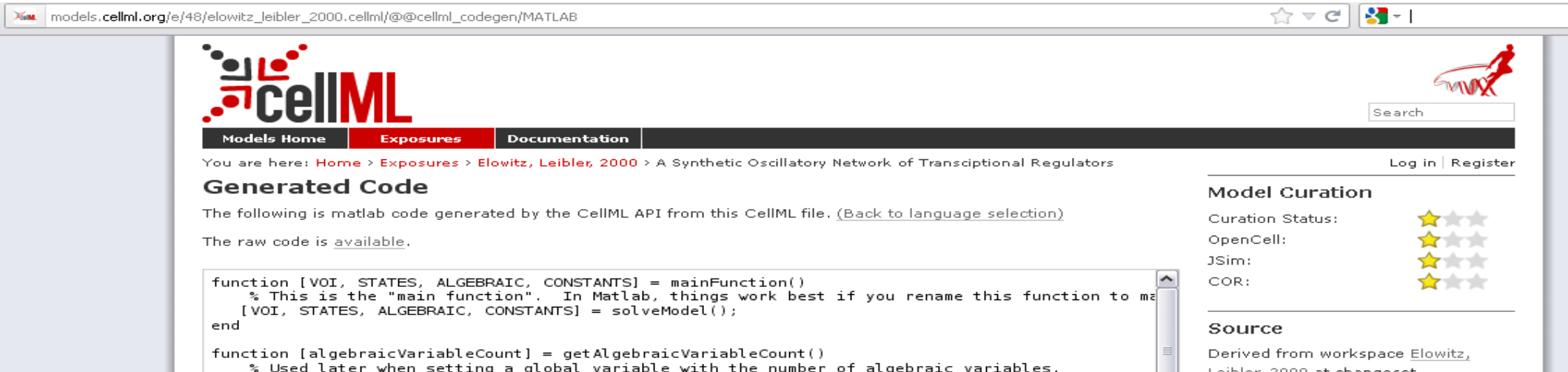
Oscylator Elowitza – model

- Układ sześciu równań różniczkowych
  - opisujących zmianę ilości białek lacI, tetR, cI i ich mRNA

$$\frac{dm_i}{dt} = -m_i + \frac{\alpha}{(1+p_j^n)} + \alpha_0 \qquad \begin{pmatrix} i = lacI, tetR, cI \\ j = cI, lacI, tetR \end{pmatrix}$$

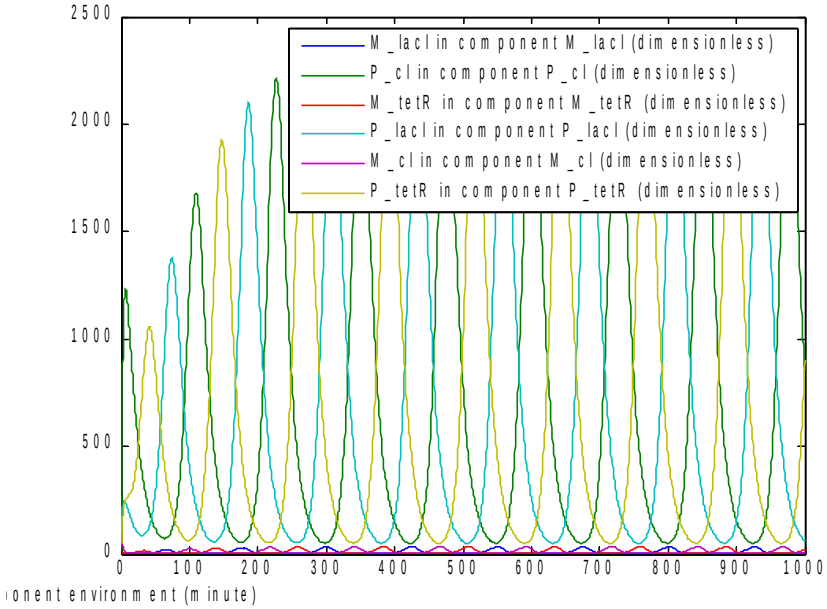
$$\frac{dp_i}{dt} = -\beta(p_i - m_i)$$

  - kod modelu m.in. w MATLABie na http://models.cellml.org/cellml



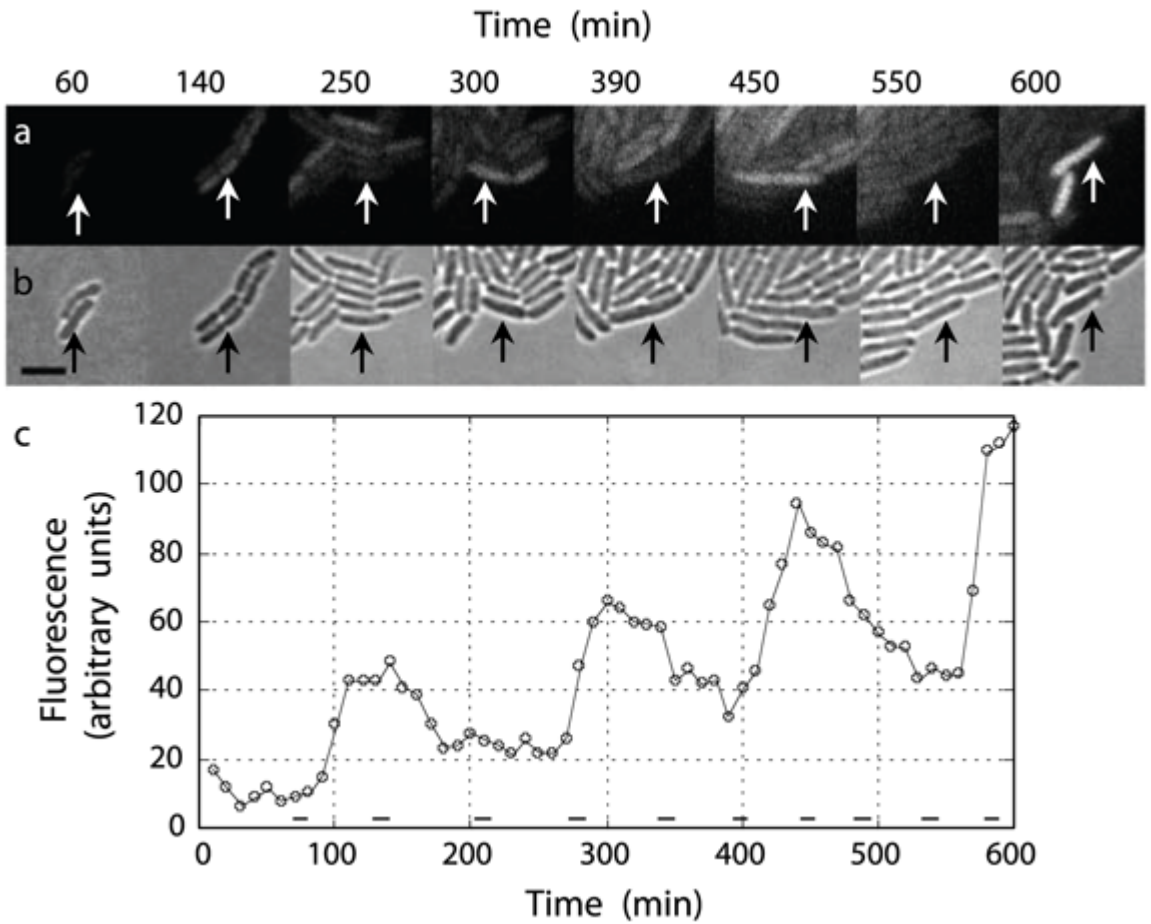
Uruchomienie modelu oscylatora

```
function [VOI, STATES, ALGEBRAIC, CONSTANTS] = solveModel()
    % Create ALGEBRAIC of correct size
    global algebraicVariableCount;  algebraicVariableCount = getAlgebraicVariableCount();
    % Initialise constants and state variables
    [INIT_STATES, CONSTANTS] = initConsts;

    % Set timespan to solve over
    tspan = [0, 1000];

    % Set numerical accuracy options for ODE solver
    options = odeset('RelTol', 1e-06, 'AbsTol', 1e-06, 'MaxStep', 1);

    % Solve model with ODE solver
    [VOI, STATES] = ode15s(@(VOI, STATES)computeRates(VOI, STATES, CONSTANTS),...
       tspan, INIT_STATES, options);

...

function [STATES, CONSTANTS] = initConsts()

    STATES(:,1) = 10;          % ilość mRNA i białek
    STATES(:,2) = 10;          % w chwili t0
    STATES(:,3) = 10;
    STATES(:,4) = 100;
    STATES(:,5) = 80;
    STATES(:,6) = 50;
```



Otrzymaliśmy syntetyczny oscylator!

Test w laboratorium



A Synthetic Oscillatory Network of Transcriptional Regulators; Michael Elowitz and Stanislas Leibler; Nature. 2000 Jan 20;403(6767):335-8. http://www.elowitz.caltech.edu/publications/Repressilator.pdf

Jeszcze o rozwiązywaniu ODE

- <u>Układy sztywne</u>
  - układ zawiera składowe szybkozmienne i wolnozmienne
  - metody rozwiązywania są niestabilne
    - mimo, że krok całkowania mały w stosunku do zmienności funkcji

  - w MATLABie zostały zaimplementowane metody
    - NDF/BDF (`ode15s`) – jeśli `ode45` nie znajduje rozw. lub wolny
    - Rosenbrocka rzędu 2 (`ode23s`) – do rozw. niskiej dokładności
    - trapezowa (`ode23t`) – do rozw. umiarkowanie sztywnych
    - TR-BDF2 (`ode23t`) – do rozw. niskiej dokładności

Równania różniczkowe wyższych rzędów

- <u>Wahadło matematyczne</u>

$$\frac{d^2\theta}{dt^2} + \frac{g}{\ell}\sin\theta = 0$$

  - θ – kąt wychylenia, g – przyspieszenie ziemskie, l – długość wahadła
  - Innymi słowy:

    θ''(t) = - g/l * sin(θ(t))

  - Podstawiamy

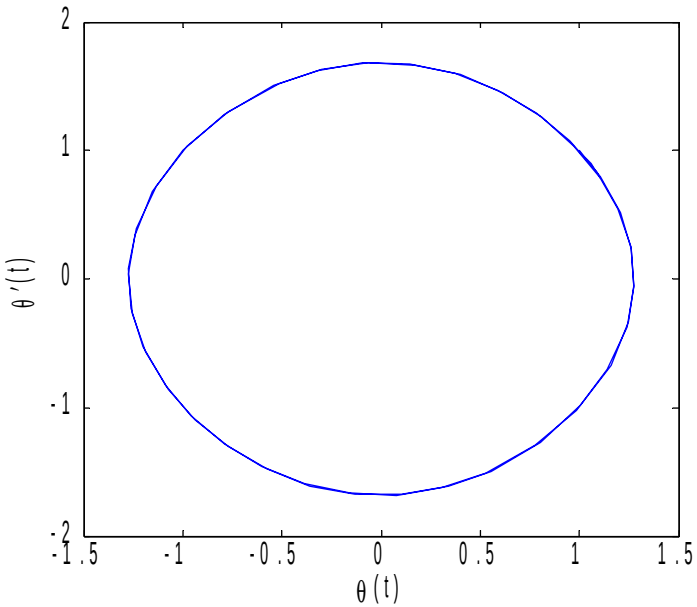
    y1(t) = θ(t)
    y2(t) = θ'(t)

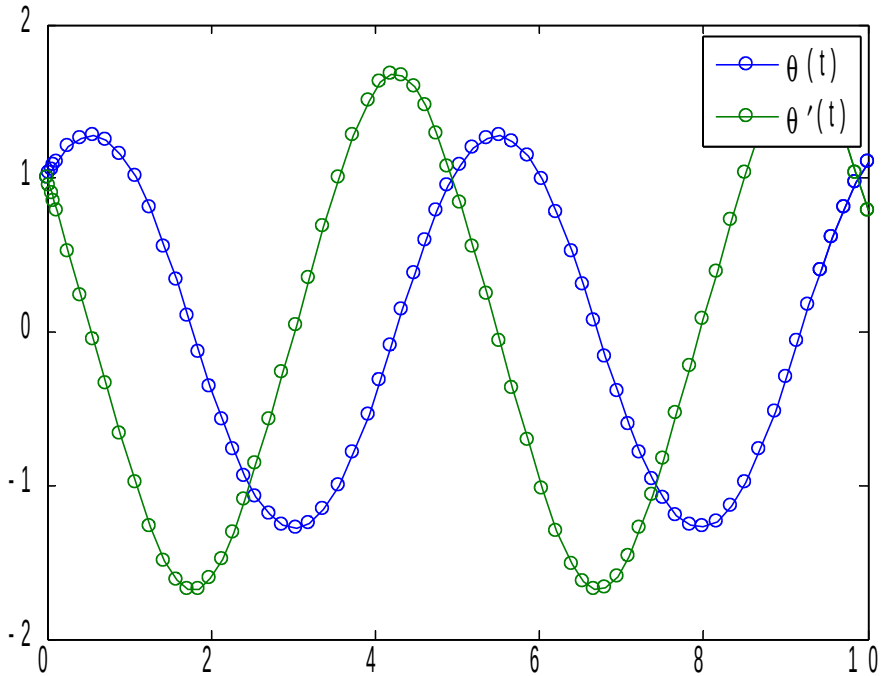
  - i otrzymujemy układ równań pierwszego rzędu

    y1'(t) = y2(t)
    y2'(t) = - g/l * sin(y1(t))

Równania różniczkowe wyższych rzędów (2)

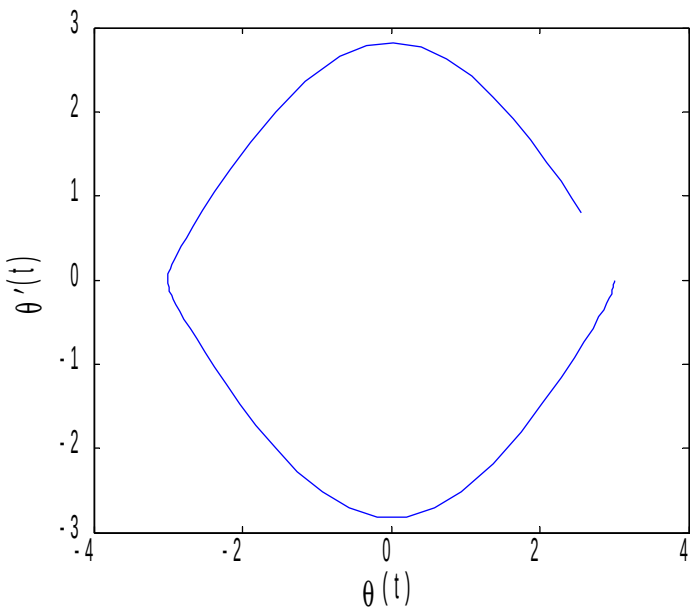
```
function yp = wah(t,y,p)
% WAH - pochodna rownania ruchu wahadla matematyczego
yp = [y(2);-p(1)/p(2)*sin(y(1))];

>> tspan=[0 10];        % czas: 0 - 10s
>> y0=[1; 1];           % wychylenie kątowe: 1rad, prędkość kątowa: 1rad/s

% parametry: przyspieszenie ziemskie 10m/s^2, długość wahadla: 5m
>> ode45(@wah,tspan,y0,[],[10 5]);
```

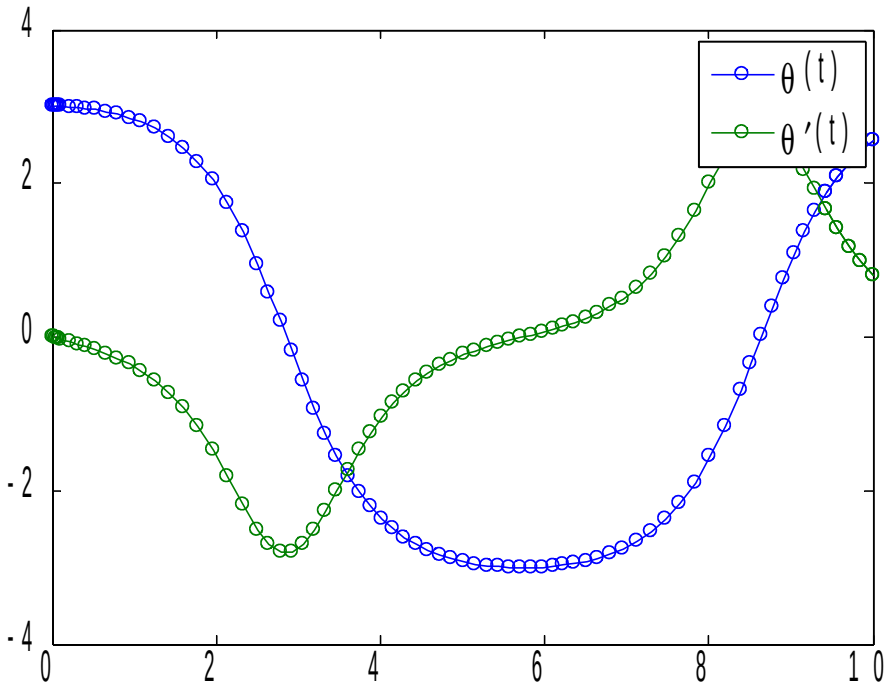




Wizualizacje przebiegów

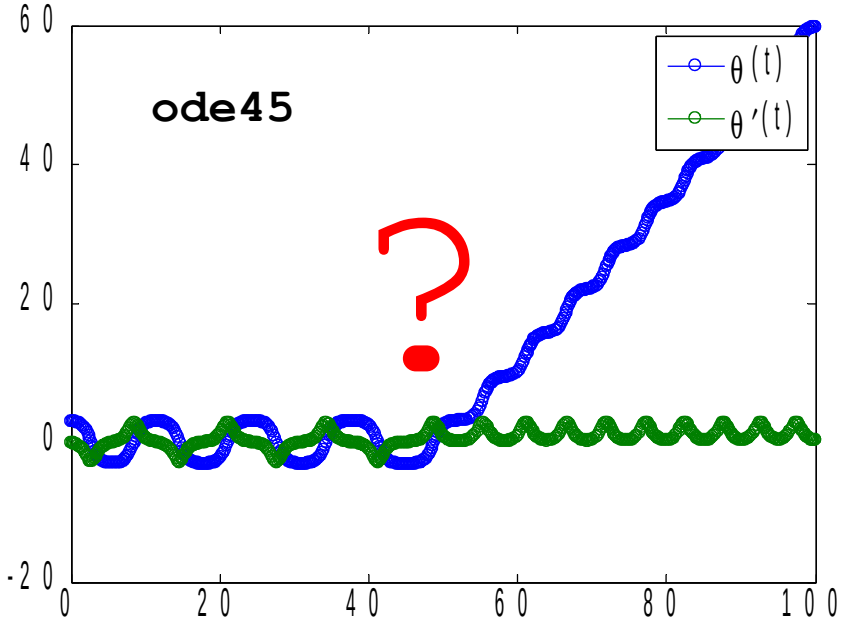
```
function yp = wah(t,y,p)
% WAH - pochodna rownania ruchu wahadla matematyczego
yp = [y(2);-p(1)/p(2)*sin(y(1))];

>> tspan=[0 10];        % czas: 0 - 10s
>> y0=[3; 0];           % wychylenie kątowe: 3rad, prędkość kątowa: 0rad/s

% parametry: przyspieszenie ziemskie 10m/s^2, długość wahadla: 5m
>> ode45(@wah,tspan,y0,[],[10 5]);
```

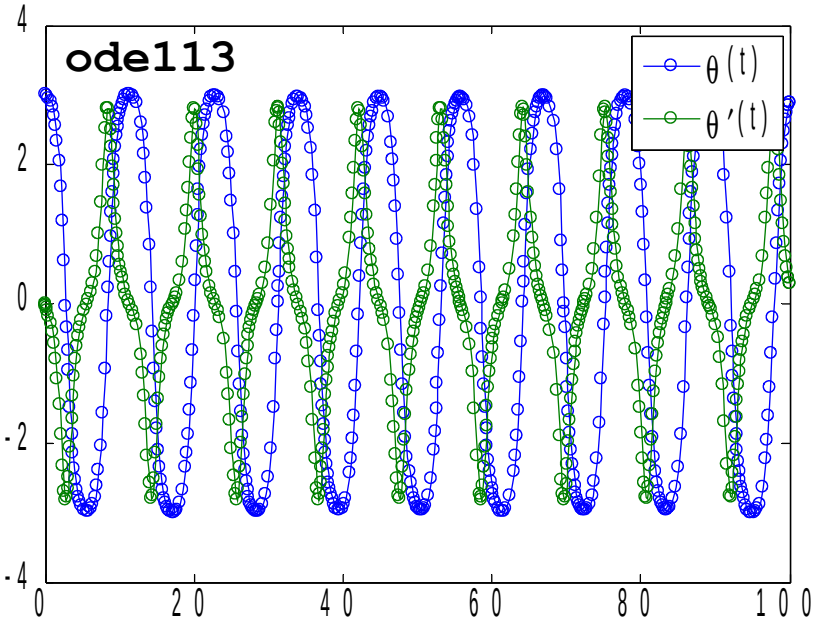




Stabilność solvera

```
function yp = wah(t,y,p)
% WAH - pochodna rownania ruchu wahadla matematyczego
yp = [y(2);-p(1)/p(2)*sin(y(1))];

>> tspan=[0 100];        % czas: 0 - 100s
>> y0=[3; 0];            % wychylenie kątowe: 3rad, prędkość kątowa: 0rad/s

% parametry: przyspieszenie ziemskie 10m/s^2, długość wahadla: 5m
>> ode45(@wah,tspan,y0,[],[10 5]);    % sypie się...
>> ode113(@wah,tspan,y0,[],[10 5]);   % wygląda ok
```





Dziś najważniejsze było...

- Ogromne możliwości obliczeń numerycznych
  - w MATLABie, ale podobne także w LabView, Octave'ie, R, Pythonie...
  - warto nauczyć się z nich korzystać

- Równania różniczkowe są bardzo szeroko stosowane w inżynierii biomedycznej
  - MATLAB dostarcza narzędzi do budowy i analizy modeli różniczkowych
  - warto nauczyć się z nich korzystać

Cele

- Rozwinięcie umiejętności poprawnego programowania:
  - programowanie proceduralne
  - elementy programowania obiektowego
  - tworzenie graficznego interfejsu użytkownika
- Nabycie umiejętności samodzielnego uczenia się
  - języka programowania
- Poznanie MATLABa jako narzędzia
  - rozwiązywania problemów numerycznych

Zakres problemów numerycznych

- Podstawowa analiza danych
  (wizualizacja, wczytywanie i zapisywanie plików)
- Podstawowe obliczenia statystyczne
- Metody interpolacji danych i dopasowywania krzywych
- Problemy algebry liniowej
- Całkowanie, różniczkowanie
- Rozwiązywanie równań różniczkowych zwyczajnych
- *Podstawy analizy sygnałów, operacje na sygnałach*
  (będzie na odrębnym przedmiocie)

A za 2 tygodnie...

- **Kolokwium**
  - PN 14.05 godz. 13.15-15.00, sala 314/A-1
- Kolokwium poprawkowe
  - CZ **31.05** godz. 17.05-18.45, sala 314/A-1