Wykład VIII

Rozwiązywanie równań i układów równań nieliniowych

- Równania nieliniowe w technice
- Zadanie wyznaczenia pierwiastków równania nieliniowego
- Metody iteracji z otaczaniem i podziałem (bisekcja i regula-falsi)
- Metody iteracji z wykorzystaniem siecznej i stycznej
- Zera wielomianu wartościami własnymi pewnej macierzy
- Problemy wielowymiarowe - układy równań

Równania i funkcje nieliniowe w technice

Opis funkcjami nieliniowymi względem wielkości wejściowej, wyjściowej, wewnętrznej (np. stanu) bądź parametrów jest powszechny w technice. Poniżej podano przykłady.

Nie-prostoliniowy ruch obiektów poddawanych zewnętrznym oddziaływaniom:

- tor planet w polu grawitacyjnym Słońca,
- ruch cząstek naładowanych w polu magnetycznym spektrometru masowego,
- Trajektoria pocisku przy prędkości początkowej, oporach powietrza i grawitacji.

Nieliniowości są powszechne w zagadnieniach technicznych z powodu ograniczeń fizycznych:

- tarcie suche opisane nieliniową funkcją względem prędkości w przeciwieństwie do tarcia wiskotycznego,
- nasycenia tranzystorów i wzmacniaczy (ograniczenie od napięcia zasilającego),
- nasycenie materiałów magnetycznych (np. blach transformatorowych).

Odpowiedzi liniowych układów dynamicznych są nieliniowymi funkcjami czasu i parametrów:

- odpowiedź inercyjna nieliniowa od czasu,
- wysokość piku rezonansowego odpowiedzi częstotliwościowej obiektu oscylacyjnego nieliniowa względem tłumienia.

Równania i układy równań nieliniowych powstają przy nakładaniu warunków (osiągnięcie określonej wartości, przecięcie trajektorii) na nieliniowe funkcje opisujące.

Przykład: Krzywa balistyczna

Tor pocisku wystrzelonego pod kątem α z prędkością początkową v_0 opisuje wzór:

$$y(x) = \left(tan(\alpha) + \frac{mg}{bv_0 cos(\alpha)}\right)x + g\frac{m^2}{b^2} ln\left(1 - \frac{xb}{mv_0 cos(\alpha)}\right),$$

gdzie b jest współczynnikiem oporu powietrza.

g = przyspieszenie ziemskie

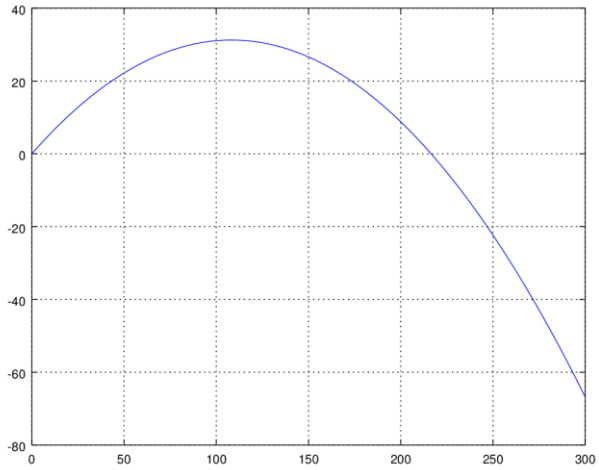
Bez oporu powietrza wzór redukuje się do:

$$y(x) = tan(\alpha)x - \frac{g}{2v_0^2 cos^2(\alpha)}x^2$$

m

Zadanie: Wyznacz zasięg kuli kamiennej o masie 5kg wystrzelonej z katapulty z prędkością 50m/s pod kątem 30º ze wzgórza o wysokości 50m nad płaskim terenem.

Inaczej: wyznacz x, przy którym $y(x)$ osiąga wartość -50.



Wyznaczanie pierwiastków równania nieliniowego (zer funkcji nieliniowej) – trudności i podstawowe idee

Trudności:

- możliwość braku zer (np. wielomian z zerami zespolonymi kiedy szukamy rzeczywistych),
- duże nagromadzenie zer w przedziale (np. dla funkcji sin(1/x) w pobliżu zera),
- możliwość braku zmiany znaku (np. dla zer wielomianu o parzystej krotności),
- nieciągłości, osobliwości.

Pomysły na rozwiązanie:

- zgrubnie narysować przebieg funkcji dla zorientowania się w problemie,
- znaleźć przedział, w którym funkcja zmienia znak i iteracyjnie go zawężać,
- wystartować z dowolnego punktu i podążać w kierunku malejących co do modułu wartości.

Informacja do wykorzystania:

- wartości funkcji,
- wartości lub przybliżenie pochodnych funkcji,
- poprzednio znalezione zera.

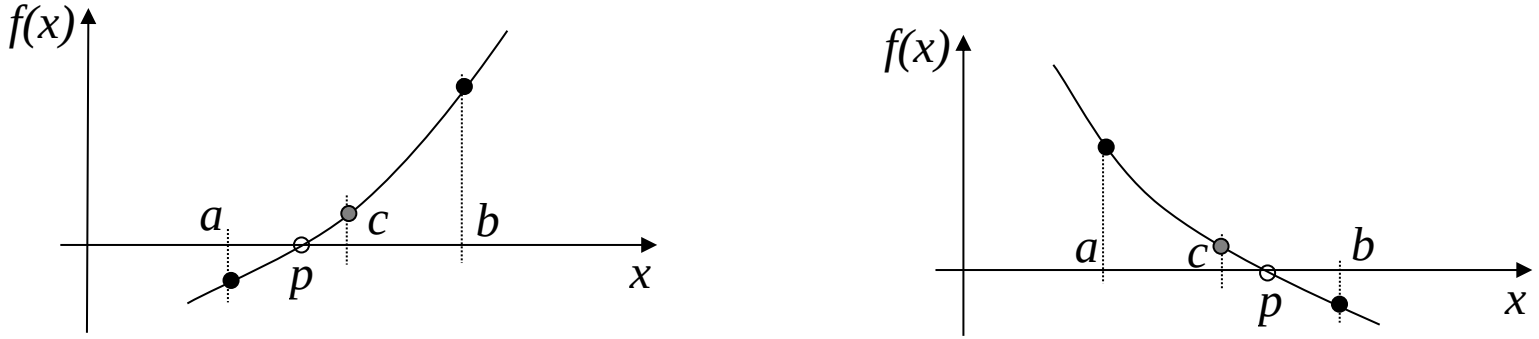
Sposób kontroli dokładności:

- różnica między wartością funkcji a zerem,
- różnica między bieżącym oszacowaniem zera a rozwiązaniem (ale to jest nieznane),
- różnica między dwoma kolejnymi oszacowaniami.

Metody iteracji z otaczaniem i podziałem (bisekcja i regula-falsi)

Idea otoczenia pierwiastka i iteracyjnego zbliżania się do niego tworzy podstawę metod obszaru zamkniętego (*closed domain*). Ponieważ dają one kontrolę przedziału zawierającego pierwiastek, dają też przewidywalną szybkość zbieżności i wielkość błędu.

Metoda bisekcji (połowienia przedziału)



W każdym kroku następuje wybór dwóch nowych punktów ograniczających przedział z zerem spośród dwóch starych punktów ograniczających i punktu środka przedziału.

Jeśli $f(a_k) \cdot f(c_k) < 0$: $a_{k+1} = a_k, \quad b_{k+1} = c_k$

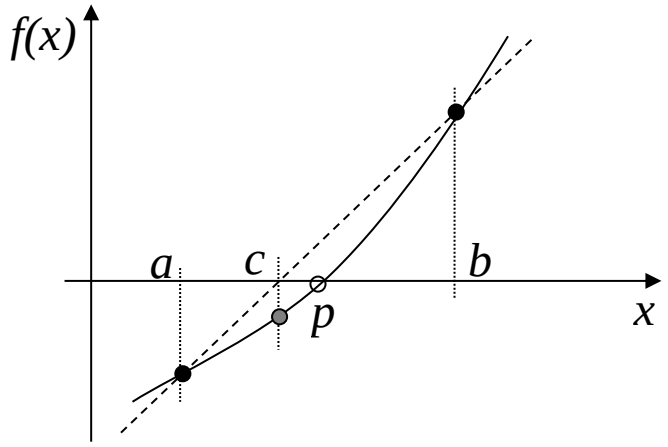
w przeciwnym przypadku: $a_{k+1} = c_k, \quad b_{k+1} = b_k$

Oszacowanie zera w *k*-tym kroku: $p_k = c_k = \frac{a_k + b_k}{2}$

Błąd oszacowania zera w *k*-tym kroku: $|p - p_k| \leq \varepsilon_k = \frac{b_0 - a_0}{2^{k+1}} = \frac{1}{2}\varepsilon_{k-1}$ (zbieżność liniowa)

Metoda *regula-falsi* (interpolacji liniowej)

Zbieżność metody połowienia jest niezależna od kształtu funkcji w otoczeniu zera. Tę zbieżność można przyśpieszyć stosując interpolację liniową funkcji na bazie punktów ograniczających. Zasada wyboru nowych punktów ograniczających pozostaje ta sama.



Przyrównując nachylenie prostej interpolującej przechodzącej przez punkty ograniczające a, b do nachylenia prostej przechodzącej przez b, i poszukiwane c (proporcje trójkąta):

$$\frac{f(b)-f(a)}{b-a}=\frac{f(c)-f(b)}{c-b}$$

$f(c)=0$

otrzymamy oszacowanie zera funkcji:

$$p_k = c_k = b_k - \frac{f(b_k)(b_k-a_k)}{f(b_k)-f(a_k)}$$

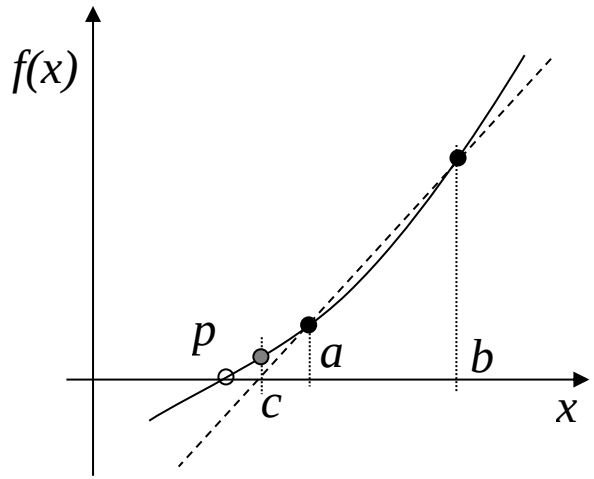
Metoda interpolacji liniowej będzie się zachowywała lepiej (będzie szybciej zbieżna) dla funkcji dobrze przybliżanych prostą w okolicach zera.

Błąd maleje w proporcjach liniowych (czyli zbieżność jest liniowa): $\varepsilon_k = K\varepsilon_{k-1}$, gdzie K zależy od pochodnych funkcji (krzywizny) wokół zera.

Metody iteracji w obszarze otwartym

Drugim pomysłem na poszukiwanie zera był start z dowolnego punktu i podążanie w kierunku malejących co do modułu wartości. Zacznijmy od przystosowania metody interpolacji liniowej do tego podejścia. Ponieważ rezygnujemy z zasady otaczania zera, to rozwiązanie interpolacyjne musimy zastąpić ekstrapolacyjnym.

Metoda siecznych (ekstrapolacji liniowej)



Następne oszacowanie zera identyczne jak w metodzie interpolacji liniowej (możliwa ekstrapolacja):

$$p_k = c_k = b_k - \frac{f(b_k)(b_k - a_k)}{f(b_k) - f(a_k)}$$

Wybór punktów do następnego oszacowania wg zasady „bliżej bieżącego oszacowania”:

dla $|c_k - a_k| < |c_k - b_k|$: $a_{k+1} = a_k, \quad b_{k+1} = c_k$

dla $|c_k - a_k| > |c_k - b_k|$: $a_{k+1} = c_k, \quad b_{k+1} = b_k$

Szybkość zbieżności metody siecznych dla zer jednokrotnych jest większa niż liniowa (zbieżność superliniowa), tzn.:

$$\varepsilon_k = K\varepsilon_{k-1}^{1.62},$$

gdzie K zależy od wartości pierwszej i drugiej pochodnej $f(x)$.

Metoda iteracji Newtona-Raphsona

Przy malejącej odległości punktów interpolacji wyrażenie na nachylenie prostej zbliża się do pochodnej (stycznej). Algorytm z użyciem pochodnej jest nazywany metodą Newtona-Raphsona.

$$p_{k+1} = p_k - \frac{f(p_k)}{f'(p_k)}$$

Warunek zbieżności (wyprowadzenie np. w J.H. Mathews, „Numerical methods for ...”):

dla $x \in [p-\delta, p+\delta]$: $\frac{|f(x)f''(x)|}{|f'(x)|} < 1$

Szybkość zbieżności (szybkość zmniejszania błędu):

Dla jednokrotnego zera *p* zbieżność kwadratowa: $\varepsilon_k = \frac{|f(p)|}{2|f'(p)|}\varepsilon_{k-1}^2$

Dla zera *p* o krotności *M* zbieżność liniowa: $\varepsilon_k = \frac{M-1}{M}\varepsilon_{k-1}$

Ta metoda ma zastosowanie przy generowaniu schematów iteracyjnych, kiedy znana jest analityczna pochodna funkcji i zastosowanie ogólne przy różnicowym przybliżeniu pochodnej.

Przykład Obliczanie pierwiastka kwadratowego liczby A jako zera funkcji $f(x) = x^2 - A$.

Zbudujemy zbieżny do pierwiastka kwadratowego schemat iteracyjny:

$f'(x) = 2x$, $x_{k+1} = x_k - (x_k^2 - A)/2x_k = 0.5(x_k - A/x_k) \rightarrow \sqrt{A}$

Metoda Mullera (ekstrapolacji/interpolacji kwadratowej)

Poprzednie metody siecznej i stycznej używały modelu liniowego funkcji w bieżącym punkcie dla predykcji (przez interpolację lub ekstrapolację) zera funkcji. Dla funkcji z dużą krzywizną w okolicy zera oznacza to wolną zbieżność algorytmu. Naturalne więc jest zwiększenie stopnia wielomianu interpolacyjnego do drugiego co powinno skutkować lepszą zbieżnością. To właśnie realizuje algorytm Mullera.

Równanie paraboli na trzech punktach $\{x_k, f_k\}, \{x_{k-1}, f_{k-1}\}, \{x_{k-2}, f_{k-2}\}$:

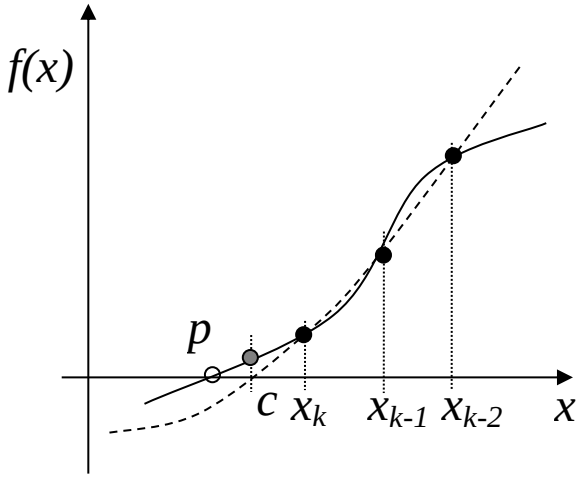
$$g(x) = a_2(x - x_k)^2 + a_1(x - x_k) + a_0$$

gdzie współczynniki:

$$a_2 = \frac{\delta_1 h_2 - \delta_2 h_1}{h_1 h_2 (h_1 - h_2)}, \quad a_1 = \frac{\delta_2 h_1^2 - \delta_1 h_2^2}{h_1 h_2 (h_1 - h_2)}, \quad a_0 = f_k$$

gdzie:

$$h_1 = x_{k-1} - x_k, \quad h_2 = x_{k-2} - x_k$$
$$\delta_1 = f_{k-1} - f_k, \quad \delta_2 = f_{k-2} - f_k$$



Zero paraboli interpolującej daje oszacowanie zera funkcji:

$$x_{k+1} = x_k - \frac{2a_0}{a_1 \pm \sqrt{a_1^2 - 4a_2 a_0}}$$

2c/(b+-sqrt(b^2-4ac))

o zbieżności superliniowej $\varepsilon_k = K\varepsilon_{k-1}^{1.84}$ (trochę lepsza od siecznej, trochę gorsza od stycznej).

Jak widać metoda wymaga kilku obliczeń. W dodatku pojawia się wyznaczanie pierwiastka równania kwadratowego, które dla pewnych wartości jest źle uwarunkowane. Spróbujmy inaczej.

Metoda odwrotnej interpolacji kwadratowej

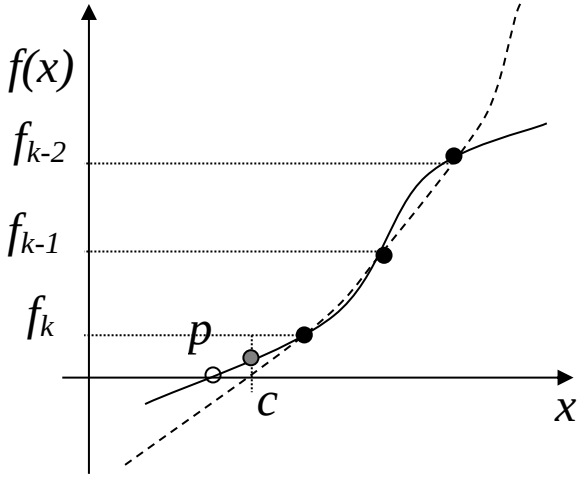
Zamiast wychodzić od naturalnej postaci wielomianu interpolującego jak w algorytmie Mullera, wykorzystajmy wiedzę z interpolacji i zastosujmy wielomian interpolujący w postaci Lagrange'a na relacji odwrotnej, tj. na punktach $\{f_k, x_k\}, \{f_{k-1}, x_{k-1}\}, \{f_{k-2}, x_{k-2}\}$. trzy punkty

Zero paraboli interpolującej daje oszacowanie zera funkcji:

$$x_{k+1} = \frac{f_{k-2}f_{k-1}}{(f_k - f_{k-2})(f_k - f_{k-1})} x_k + \frac{f_{k-2}f_k}{(f_{k-1} - f_{k-2})(f_{k-1} - f_k)} x_{k-1} + \frac{f_{k-1}f_k}{(f_{k-2} - f_{k-1})(f_{k-2} - f_k)} x_{k-2}$$

Algorytm Brenta zaimplementowany w Matlabie jako *fzero* jest kombinacją metody bisekcji, siecznych oraz odwrotnej interpolacji kwadratowej.

Implementacja w Octave zawiera również implementację odwrotnej interpolacji sześciennej.



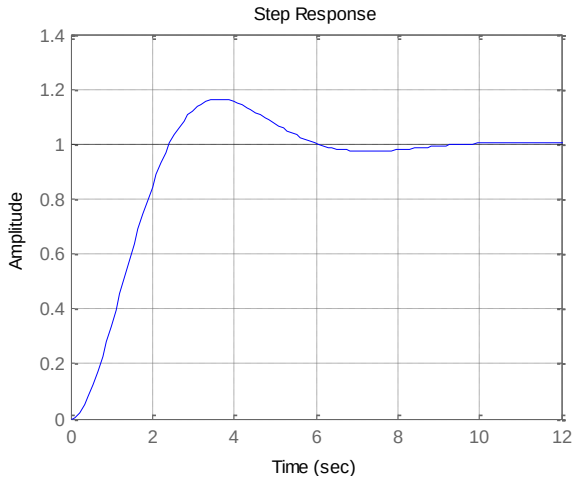
Implementacja funkcji *fzero* (zob. też „Numerical Recipes”)

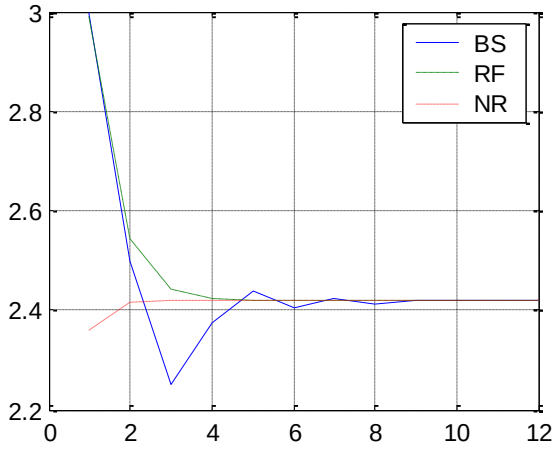
```
function [b,fval,exitflag,output] = fzero(FunFcnIn,x,varargin)
%FZERO  Scalar nonlinear zero finding.
%   Copyright 1984-2002 The MathWorks, Inc.
...
fc = fb;
% Main loop, exit from middle of the loop
while fb ~= 0
   % Insure that b is the best result so far, a is the previous
   % value of b, and c is on the opposite of the zero from b.
...
% Convergence test and possible exit
   m = 0.5*(c - b);
   toler = 2.0*tol*max(abs(b),1.0);
   if (abs(m) <= toler) | (fb == 0.0),
      break,
   end
% Choose bisection or interpolation
   if (abs(e) < toler) | (abs(fa) <= abs(fb))
      % Bisection
      d = m;  e = m;
      step='       bisection';
   else
      % Interpolation
      s = fb/fa;
      if (a == c)
         % Linear interpolation
         p = 2.0*m*s;
         q = 1.0 - s;
      else
         % Inverse quadratic interpolation
         q = fa/fc;
         r = fb/fc;
         p = s*(2.0*m*q*(q - r) - (b - a)*(r - 1.0));
         q = (q - 1.0)*(r - 1.0)*(s - 1.0);
      end;
      if p > 0, q = -q; else p = -p; end;
      % Is interpolated point acceptable
      if (2.0*p < 3.0*m*q - abs(toler*q)) & (p < abs(0.5*e*q))
         e = d;  d = p/q;
         step='       interpolation';
      else
         d = m;  e = m;
         step='       bisection';
      end;
   end % Interpolation
% Next point
...
end % Main loop
```

Przykład Eksperymentalne porównanie szybkości zbieżności poszczególnych metod

Problem: wyznaczenie momentu pierwszego przejścia przez wartość ustaloną odpowiedzi skokowej obiektu oscylacyjnego drugiego rzędu (np. wahadło, czujnik ciśnienia, amortyzator).

Zgrubnie odczytana wartość dla pierwszego przejścia to t=2.5. Przyjmiemy punkt startowy dla metod obszaru otwartego t_0=2, a dla metod przedziału zamkniętego t_0=2, t_1=4.






```
function C=nlsolvers(solver, its)

C=zeros(1,its);
a=2; b=4; c=a;

for i=1:its
switch(solver)
case 'bs' % bisection
    if f(a)*f(c)<0 b=c; else a=c;
    end
    c=(a+b)/2;
  case 'rf' % regula-falsi
    if f(a)*f(c)<0 b=c; else a=c;
    end
    c=b-f(b)*(b-a)/(f(b)-f(a));
  case 'nr' % newton-raphson
    c=c-f(c)/fp(c);
  otherwise,
    error('Unknown method');
  end
  C(i)=c;
end
```

```
function y=f(t)
% odpowiedz skokowa
K=1; w0=1; ksi=0.5;
p1=sqrt(1-ksi^2);
y=-1/p1*exp(-ksi*w0*t);
y=y*sin(w0*p1*t+asin(p1));

function y=fp(t)
% pochodna odpowiedzi skokowej
% czyli odpowiedz impulsowa
K=1; w0=1; ksi=0.5;
p1=sqrt(1-ksi^2);
y=w0/p1*exp(-ksi*w0*t);
y=y*sin(w0*p1*t);
```

MAIN

```
it=12;
cb=nlsolvers('bs',it);
cr=nlsolvers('rf',it);
cn=nlsolvers('nr',it);
plot(1:it,cb,1:it,cr,1:it,cn);
grid on,
legend('BS','RF','NR');
```

Przypadek szczególny - zera wielomianu poprzez wartości własne

Wartości własne macierzy były zerami wielomianu charakterystycznego macierzy (mało efektywna metoda rozwiązania zagadnienia własnego). Mając dany wielomian możemy postąpić odwrotnie, tzn. utworzyć macierz, dla której będzie on wielomianem charakterystycznym i wyznaczyć wszystkie zera wielomianu (rzeczywiste i zespolone) algorytmem rozwiązania zagadnienia własnego.

Jak można sprawdzić, wielomian (ogólniejszy wielomian możemy sprowadzić do tej postaci):

$$P(x) = x^N + a_{N-1}x^{N-1} + \ldots + a_1 x + a_0$$

podzielone przez a(N)

jest wielomianem charakterystycznym tzw. macierzy stowarzyszonej (*companion matrix*):

$$\mathbf{P} = \begin{bmatrix} -a_{N-1} & -a_{N-2} & \cdots & -a_1 & -a_0 \\ 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & \ddots & 1 & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$$

$$P(x) = \det[\mathbf{P} - x\mathbf{I}]$$

Stąd zera $P(x)$ to wartości własne **P**.

```
function r = roots(c)
%   Copyright 1984-2002 The MathWorks, Inc.
...
% Polynomial roots via a companion matrix
n = length(c);
if n > 1
    a = diag(ones(1,n-2),-1);
    a(1,:) = -c(2:n) ./ c(1);
    r = [r;eig(a)];
end
```

przesunięcie w lewo o "-1"

Problemy wielowymiarowe - układy równań

Uogólnienie metody Newtona-Raphsona (metoda stycznej w punkcie) na przypadek wielowymiarowy to metoda Newtona, którą przez analogię zapiszemy wektorowo jako:

$$\mathbf{p}_{k+1} = \mathbf{p}_k - [\mathbf{J}(\mathbf{p}_k)]^{-1}\mathbf{f}(\mathbf{p}_k)$$

gdzie (dla liczby równań równej liczbie zmiennych niezależnych):

wektor zmiennych: wektor funkcji: macierz Jakobianu:

$$\mathbf{p} = \begin{bmatrix} p_1 \\ \vdots \\ p_N \end{bmatrix} \qquad \mathbf{f}(\mathbf{p}) = \begin{bmatrix} f_1(\mathbf{p}) \\ \vdots \\ f_N(\mathbf{p}) \end{bmatrix} \qquad \mathbf{J}(\mathbf{p}) = \frac{\partial \mathbf{f}}{\partial \mathbf{p}} = \begin{bmatrix} \frac{\partial f_1}{\partial p_1} & \cdots & \frac{\partial f_1}{\partial p_N} \\ \vdots & & \vdots \\ \frac{\partial f_N}{\partial p_1} & \cdots & \frac{\partial f_N}{\partial p_N} \end{bmatrix}$$

W praktyce algorytm poszukiwania $\mathbf{f}(\mathbf{p}) = 0$ będzie wykonywany w następujących krokach:

1) Oblicz wartości funkcji i Jakobianu w bieżącym punkcie: $\mathbf{f}_k = \mathbf{f}(\mathbf{p}_k), \quad \mathbf{J}_k = \mathbf{J}(\mathbf{p}_k)$
2) Rozwiąż układ równań liniowych ze względu na **d**: $\mathbf{J}_k\mathbf{d} = -\mathbf{f}_k$
3) Wyznacz następne oszacowanie zera: $\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{d}$

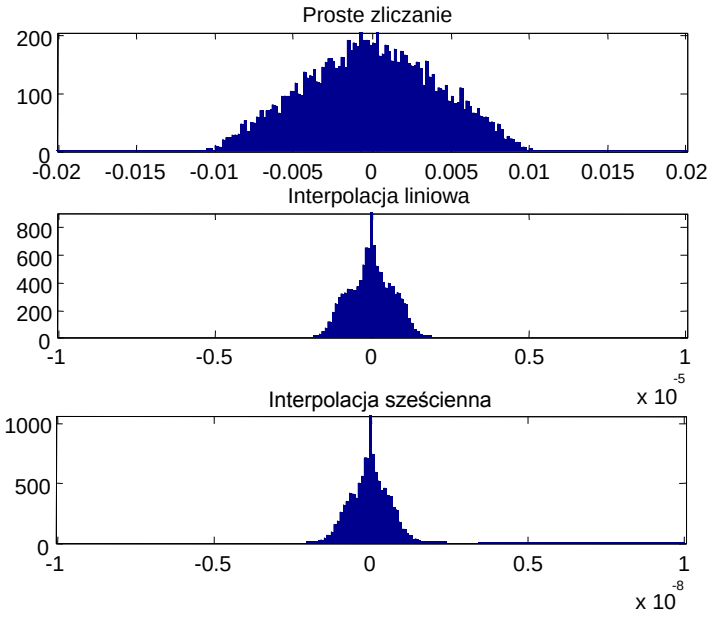
Ulepszona postać tego algorytmu jest zaimplementowana (Matlab) w *fsolve* jako algorytm *dogleg*.

Inny sposób rozwiązania to zmiana problemu układu równań nieliniowych na problem poszukiwania minimum sumy kwadratów składowych. Ponieważ suma kwadratów ma minimalną wartość zero dla wszystkich składowych zerowych, więc to problem równoważny. Funkcja *fsolve* może działać w ten sposób (algorytmy *lm* i *gn*). Algorytmy minimalizacji na następnym wykładzie.

Przykład Śledzenie okresu próbkowanego sygnału sinusoidalnego (np. napięcia/prądu sieciowego)
Pomysły na rozwiązanie:

- FFT i wybór maksymalnego prążka (duża rozdzielczość → dużo okresów → słabe śledzenie)
- Aproksymacja sinusem w jednym okresie (dopasowanie nieliniowe trzech parametrów)
- Wyznaczenie momentów zmiany znaku +/- i proste zliczanie próbek pomiędzy nimi
- Precyzyjniejsze wyznaczenie momentu zmiany – lokalna interpolacja lub aproksymacja.

Np. lokalna interpolacja pierwszego stopnia, lokalna interpolacja trzeciego stopnia, lokalna aproksymacja trzeciego stopnia, lokalna aproksymacja sinusem w okolicy zera
Porównajmy dokładność trzech wybranych metod: prostego zliczania i interpolacji pierwszego i trzeciego stopnia.




```
L=10000; dt=0.01; t=-
0.5:dt:1.5;
for i=1:L
  T=1+0.05*randn;  w=2*pi/T;
 y=sin(w*t+2*pi*0.05*randn);
  ys=sign(y);  yz=diff(ys);
  iz=find(yz>0);
  % proste zliczanie
  count=iz(2)-iz(1);
Tep=count*dt;
  % interpolacja pierwszego
stopnia
  y1=y(iz(1));
y2=y(iz(1)+1);
  dt1=dt*y2/(y2-y1);
  y1=y(iz(2));
y2=y(iz(2)+1);
  dt2=dt-dt*y2/(y2-y1);
  Tei=(count-1)*dt+dt1+dt2;
```

```
% Interpolacja trzeciego
stopnia
p=polyfit(-1:2,y(iz(1)-
1:iz(1)+2),3);
r=roots(p);
ir=find(imag(r)==0);
ic=find(r(ir)>=0 & r(ir)<=1);
dt1=dt*(1-r(ic));
p=polyfit(-1:2,y(iz(2)-
1:iz(2)+2),3);
r=roots(p);
ir=find(imag(r)==0);
ic=find(r(ir)>=0 & r(ir)<=1);
dt2=dt*r(ic);
Tea=(count-1)*dt+dt1+dt2;
Tref(i)=T; Tp(i)=Tep;
Ti(i)=Tei; Ta(i)=Tea;
end
dTp=(Tp-Tref)./Tref;
```

Podsumowanie, kluczowe elementy tego wykładu:

- Miejsce zerowe funkcji nieliniowej tworzącej równanie nieliniowe możemy wyznaczyć metodą:
  - Otoczenia zera i iteracyjnego zawężania przedziału:
    - bisekcja,
    - regula-falsi.
  - Iteracyjnego wyznaczania zera wielomianu interpolującego/ekstrapolującego funkcję nieliniową wokół bieżącego punktu:
    - metoda siecznych,
    - metoda Newtona-Raphsona (ekstrapolacja linią styczną, wersja wielowymiarowa),
    - ekstrapolacja kwadratowa (metoda Mullera) i ekstrapolacja sześcienna.
- W szczególnym przypadku wyznaczania miejsc zerowych wielomianu problem można opisać i rozwiązać używając wartości własnych macierzy stowarzyszonej
- Problem poszukiwania zer można wyrazić jako problem minimalizacji