ROZWIĄZYWANIE RÓWNAŃ
(znajdowanie pierwiastków)

$f(x)=0$

(w ogólności równań nieliniowych)

$f(x) = x^3 - x^2 + 2x - 1$
$f(x)=sinx-2cosx+1$
$f(x)= x-e^x$

Znajdowanie pierwiastków równań nieliniowych

Często istnieje potrzeba rozwiązania równania postaci f(x) = 0. Dla niektórych funkcji (np. $f(x) = x^2 - 4x + 1$) znane są sposoby analitycznego wyznaczenia pierwiastków tego równania.

Jednak w przypadku większości funkcji rozwiązanie analityczne jest trudne, czasochłonne lub nawet niemożliwe.

W wielu przypadkach zadowalające okazuje się użycie szybszych metod przybliżonych.

Pierwiastek równania odnaleziony przy ich pomocy obarczony jest jednak pewnym (z reguły możliwym do oszacowania) błędem.

Istnieje wiele metod przybliżonego rozwiązywania równań nieliniowych.

Przybliżone metody rozwiązywania równań

Przybliżone metody rozwiązywania równań polegają najczęściej na tworzeniu tzw. wzorów rekurencyjnych określających sposób wyznaczania kolejnych wyrazów ciągu liczbowego, którego granicą jest szukane rozwiązanie równania typu

$$F(x) = 0$$

Większość metod obliczeniowych należy do typowych metod iteracyjnych.

Znajdowanie pierwiastków równań nieliniowych

Podstawowe problemy tych metod to:

1. Lokalizacja pierwiastka (dobór punktu startowego).
2. Obliczanie przybliżeń pierwiastków.
3. Zbieżność procesu iteracyjnego.

Znajdowanie pierwiastków równań nieliniowych

Do najpopularniejszych i najprostszych należą metody iteracyjne:

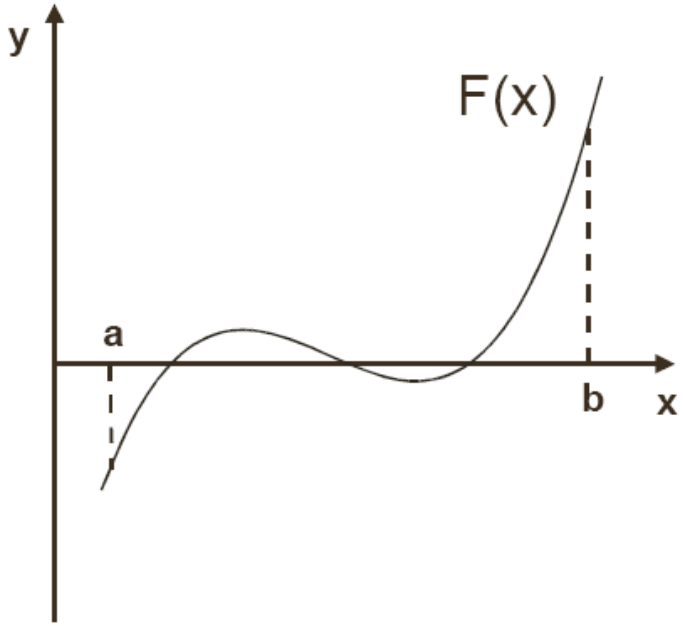
- bisekcji,
- siecznych
- stycznych (Newtona),

Lokalizacja pierwiastków

Twierdzenie 1 (Bolzano-Cauchy'ego):

Jeżeli funkcja $F(x)$ jest ciągła w przedziale $[a,b]$ domkniętym i na jego końcach przyjmuje wartości różnych znaków, tzn. $F(a)\cdot F(b)<0$, to między punktami a i b znajduje się co najmniej jeden pierwiastek równania $F(x)=0$.

Lokalizacja pierwiastków



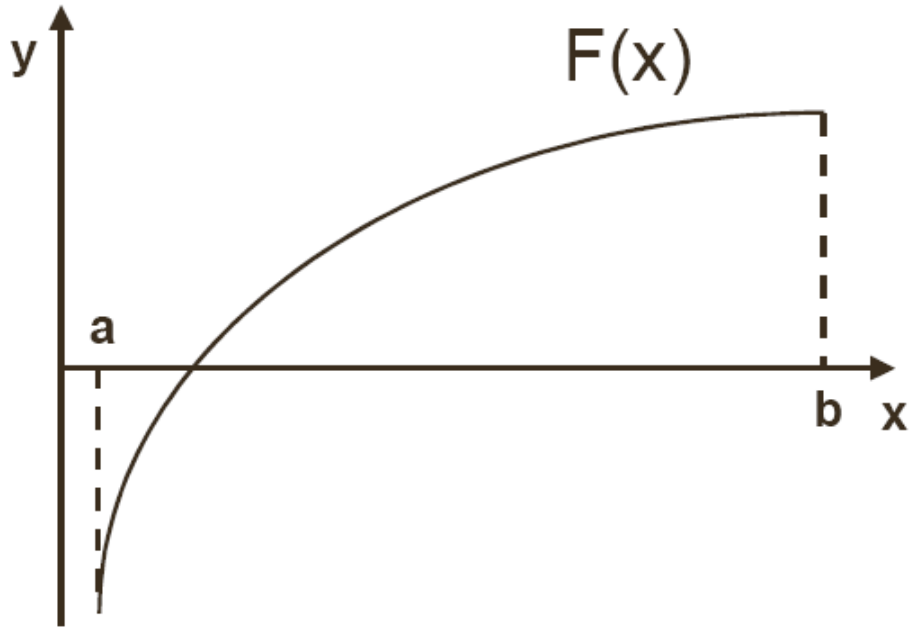
Przebieg funkcji między punktami a i b.

Lokalizacja pierwiastków

Twierdzenie 2:

Jeżeli w przedziale $[a,b]$ spełnione są założenia twierdzenia Bolzano - Cauchy'ego i dodatkowo $F'(x)$ jest stałego znaku w tym przedziale (co oznacza, że funkcja jest stale rosnąca lub stale malejąca), to przedział ten jest przedziałem izolacji pierwiastka równania $F(x)=0$ (w przedziale tym jest tylko jeden pierwiastek).

Lokalizacja pierwiastków



Przebieg funkcji między punktami a i b.

Twierdzenie 3:

Jeżeli

$$F(x) = a_n x^n + a_{n-1} x^{n-1} + \ \ldots \ + a_1 x + a_0 \quad , \quad (a_n \neq 0$$

to pierwiastki równania $F(x) = 0$ zawarte są w przedziale

$$|x| < 1 + \frac{\mathrm{M}}{|a_n|}$$

gdzie $$\mathrm{M} = \max\left\{ |a_0|, |a_1|, \ldots, |a_{n-1}| \right\}$$

Przykład:

Określić przedział zawierający pierwiastki równania:

$$x^4 - 3x^3 + 8x^2 - 5 = 0$$

$$|x| < 1 + \frac{8}{1} = 9$$

Pierwiastki znajdują się w przedziale: $(-9, 9)$

Metoda bisekcji

Niech funkcja $f(x)$ będzie funkcja ciągłą w przedziale $[a, b]$ oraz

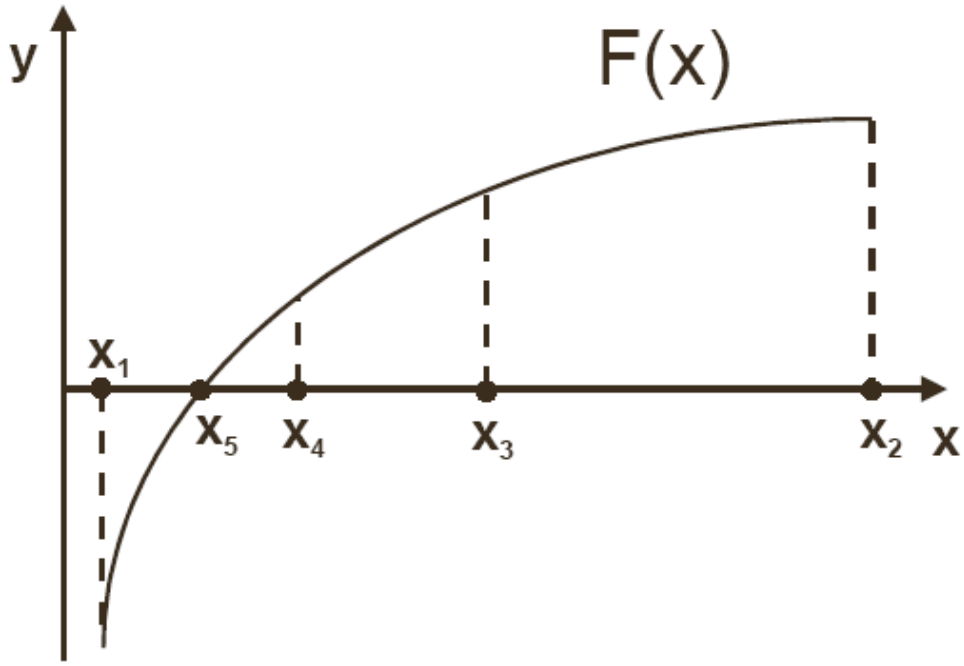
$$f(a)f(b) < 0.$$

Oznacza to, że wartość funkcji $f(x)$ zmienia znak w tym przedziale. Ponadto równanie $f(x) = 0$, ma w przedziale $[a, b]$ co najmniej jedno rozwiązanie.

Idea metody polega na połowieniu przedziału poszukiwań, za każdym razem biorąc tę część przedziału, na której wartość funkcji zmienia znak.

Połowienie takie jest kontynuowane tak długo, aż zostanie osiągnięta określona dokładność obliczeń

(oznaczana zwykle δ lub ε). Prowadzi to do następującego algorytmu, znanego jako *metoda bisekcji* lub *metoda połowienia*:



Kolejne przybliżenia poszukiwania pierwiastka w metodzie bisekcji.

Metoda bisekcji

1. Wybieramy przedział $<a, b>$, tak by $f(a)f(b)<0$
2. Dzielimy przedział na połowy:

$$x_0 = a + \frac{b-a}{2}$$

3. Mamy trzy przypadki:

$f(x_0) = 0, \quad znaleziono\ pierwiastek$

$f(x_0)\ ma\ ten\ sam\ znak\ co\ f(a)\ zatem\ pierwiastek$

$jest\ w\ przedziale\ (x_0, b)$

$f(x_0)\ ma\ ten\ sam\ znak\ co\ f(b)\ zatem\ pierwiastek$

$jest\ w\ przedziale\ (a, x_0)$

4. Wybieramy przedział zawierający pierwiastek jako nowy przedział $<a,b>$ i wracamy do kroku 2.

Metoda bisekcji

1) $c := (a+b)/2$

2) jeśli $b-c<=\varepsilon$ lub $f(c)=0$ to przyjmij , ze pierwiastkiem równania $f(x)=0$ jest wartość c, i zakończ program .

3) j e s l i $(f(b) * f(c))<=0$ to

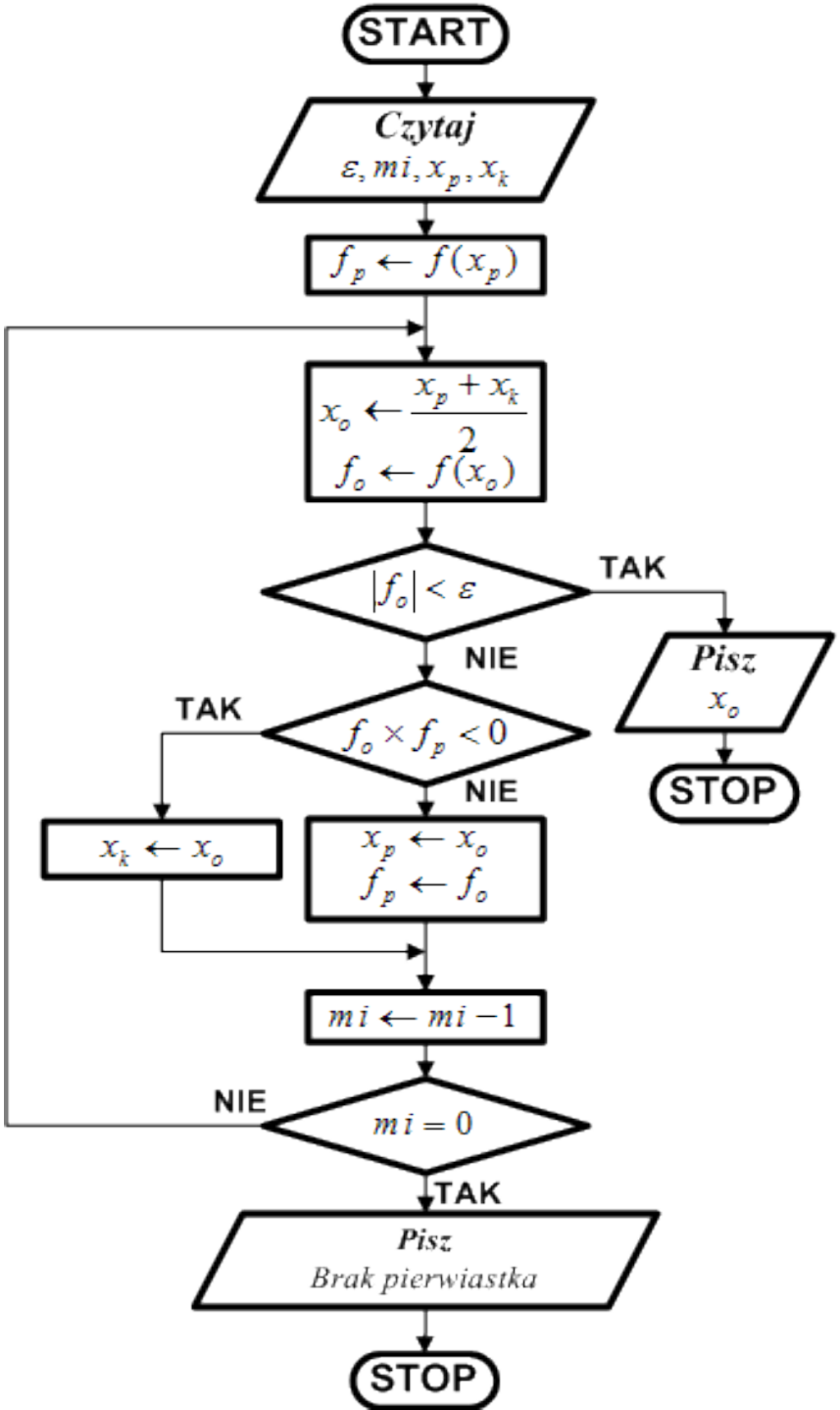
a:=c

w przeciwnym razie

b:=c

4) skocz z do punktu 1 .

Metoda bisekcji



```
#include<iostream>
#include"conio.h"
#include<math.h>
using namespace std;
double f(double);
int main()
{
	double a,b,x0,eps;
	int i;
	cout.precision(15);
	cout << "podaj a ";
	cin >> a;
	cout <<"podaj b ";
	cin >>b;
	cout <<"podaj eps ";
	cin >>eps;
	if (f(a)*f(b)>0)
	  {cout << "mozliwy brak pierwiastkow w przedziale";}
	else
```

```
        {
                i=0;
            while(fabs(a-b)>eps)
                 {
                                i++;
                                x0=(a+b)/2.;
                                if(fabs(x0)<eps) break;
                                if(f(a)*f(x0)<0)
                                          b=x0;
                                else
                                         a=x0;
                 }
       cout <<"x0="<<x0<<"  l. iteracji="<<i<<endl;
         }
       getch();
          return 0;
}
double f(double x)
{
       //return x*x*x*(x+sin(x*x-1)-1)-1;
       return (x*x)-1;
}
```

Metoda bisekcji

Z matematycznego punktu widzenia jeśli warunki początkowe zostaną spełnione, to metoda połowienia przedziału zawsze daje rozwiązanie.

Błąd tego rozumowania w odniesieniu do komputerów polega na tym, iż maszyny cyfrowe nie wykonują obliczeń z nieskończoną dokładnością

-to co jest zbieżne w matematyce, w obliczeniach numerycznych może być rozbieżne.

Niestety, jeśli założona dokładność będzie mniejsza od błędów zaokrągleń wyliczania wartości funkcji, to może się zdarzyć, iż algorytm nigdy nie da rozwiązania, zawieszając przy okazji program.

Metoda bisekcji

Przy zbliżaniu się do spodziewanego pierwiastka wartość funkcji może oscylować wokół źle przyjętego otoczenia zera, nigdy w nie nie wpadając (funkcje też są liczone z pewnym przybliżeniem). W takiej sytuacji krańce przedziału będą się do siebie coraz bardziej zbliżać, aż przekroczywszy dokładność zastosowanej arytmetyki liczb zmiennoprzecinkowych staną się sobie równe (w matematyce nigdy się to nie zdarzy).

Jeśli krańce przedziału zrównają się, to w kolejnych krokach wartość x0 będzie już zawsze taka sama i w każdym następnym obiegu algorytmu nic się więcej nie zmieni - dojdzie do zablokowania algorytmu

Metoda bisekcji

Zalety metody:

Prawie zawsze zbieżna

Wady

•Bardzo wolno zbieżna (3-4 iteracje dla 1 miejsca dziesiętnego)

•Nie uwzględnia postaci (kształtu) funkcji (po prostu dzieli przedział na 2)

Często stosowana jako wstępna metoda (tak aby się zbliżyć do pierwiastka), a następnie stosuje się metody szybciej zbieżne.

Błąd **metody bisekcji:**

Niech a_n, b_n, c_n oznaczają n-tą obliczoną wartość odpowiednio a, b i c. Ponadto niech α oznacza prawdziwą wartość pierwiastka równania $f(x) = 0$. Błąd popełniony w n-tym kroku w metodzie bisekcji $|\alpha - c_n|$ jest określony wzorem:

$$|\alpha - c_n| \leqslant \frac{1}{2^n}(b - a) \qquad (1)$$

zbieżność metody bisekcji - liniowa

$$\delta_{n+1} = \delta_n / 2$$

gdzie δ - przedział zawierający w kolejnym kroku miejsce zerowe funkcji;

jeśli ε jest żądaną tolerancją, to

$$\varepsilon = \delta_0 / 2^n \qquad \text{gdzie} \quad \delta_o = (b-a)$$

i z góry znamy liczbę iteracji

$$n = \log_2 \frac{\delta_0}{\varepsilon}$$

Wykładnik zbieżności

- Określa szybkość zbieżności metod iteracyjnych
- Metoda jest rzędu p, jeżeli istnieje stała c taka, że dla dwóch kolejnych przybliżeń x_k i x_{k+1} zachodzi

$$\lim_{k\to\infty} \frac{|\varepsilon_{k+1}|}{|\varepsilon_k|^p} = c$$

gdzie $\varepsilon_k = x_{k+1} - x_k$.

Przypadki specjalne

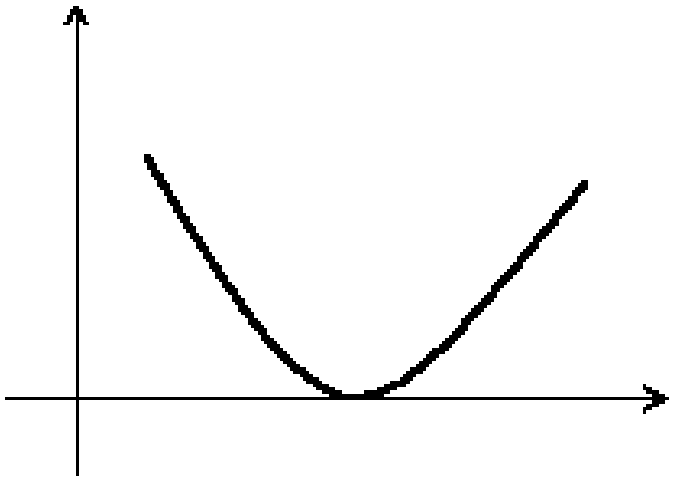
p=1 (metoda liniowa),
p>1 & p<2 (metoda superliniowa)
p=2 (metoda kwadratowa),
p=3 (metoda kubiczna)

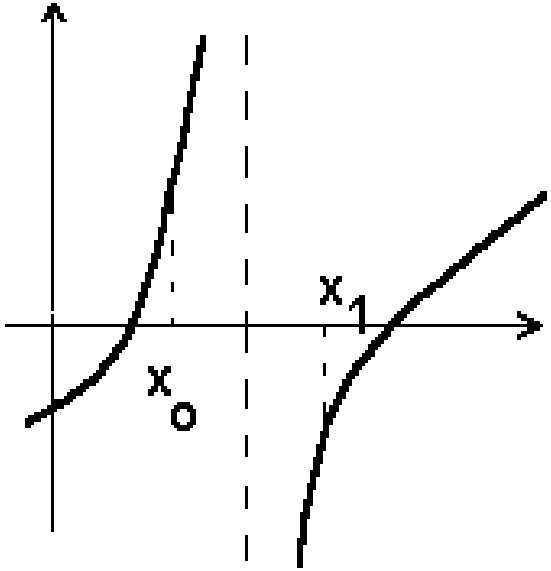
Zbieżność	Analizujemy kolejne ε	
1. $10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$...	linowa z C = 10^{-1}	p=1
2. $10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}$...	linowa z C = 10^{-2}	
3. $10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}$...	superliniowa	
4. $10^{-2}, 10^{-4}, 10^{-8}, 10^{-16}$...	kwadratowa	p=2, c=1 (w każdym kroku podwaja

nam się liczba cyfr znaczących)

Metoda bisekcji

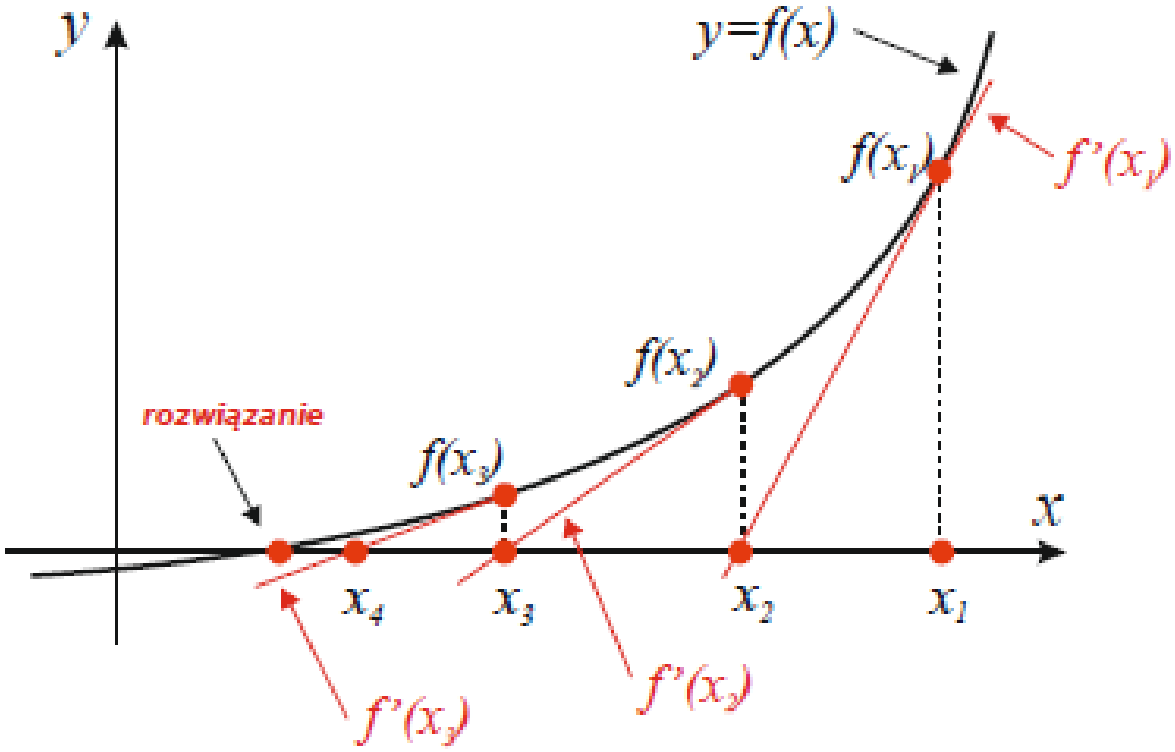
metoda nieskuteczna np. dla pierwiastków wielokrotnych, lub w przypadkach takich jak (warunki są OK):





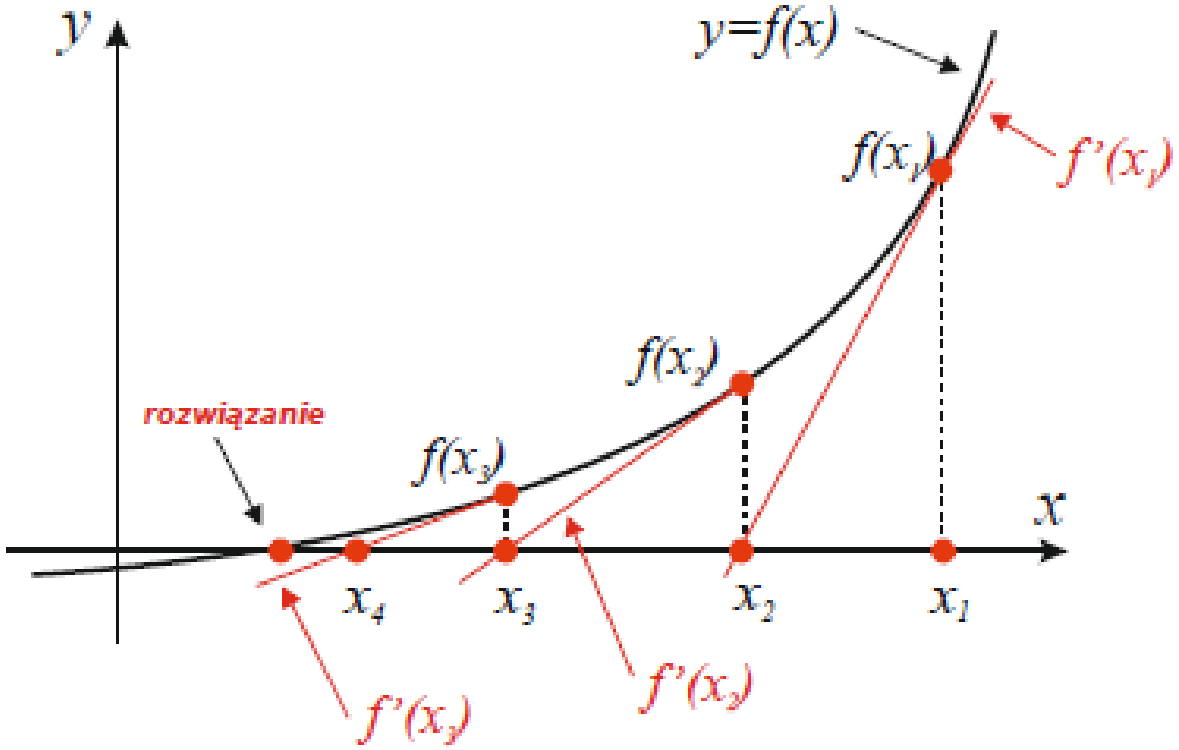
Metoda Newtona

Metoda Newtona



Metoda Newtona-Raphsona (Newtona, stycznej),

chcemy uwzględnić jakoś kształt funkcji, aby przyśpieszyć zbieżność



startujemy z punktu x_1

Metoda Newtona

startujemy z punktu x_o ;
Zapisując równanie stycznej w x_o

$$y(x) = f'(x_o)(x - x_o) + f(x_o)$$

Kolejne przybliżenie do rozwiązania x_r otrzymujemy w punkcie przecięcia stycznej z osia OX czyli w punkcie, w którym y=0

$$X_1 = X_o - \frac{f(X_o)}{f'(X_o)}$$

powtarzając tę konstrukcję w kolejnych punktach

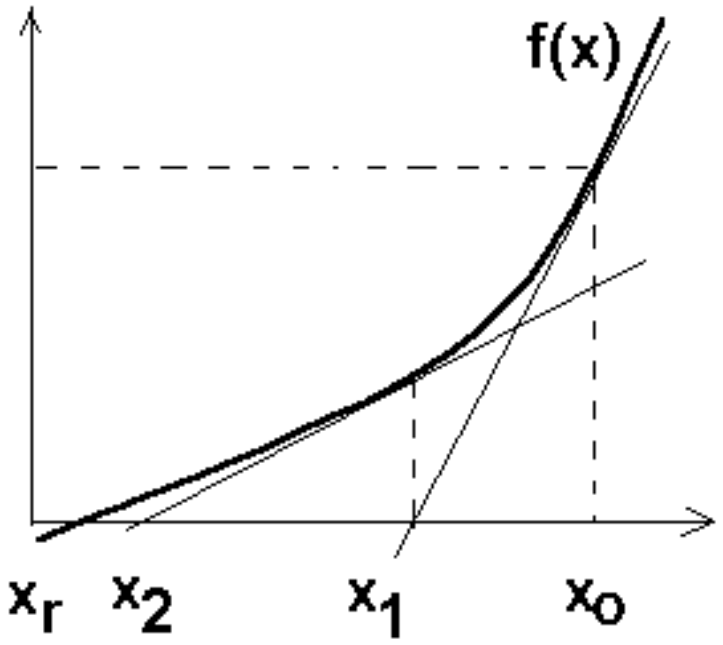
$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Aż spełniony będzie jeden z warunków kończących iteracje

$$|x_{k+1} - x_k| < \varepsilon$$

$$\frac{|x_{k+1} - x_k|}{|x_{k+1}|} < \varepsilon$$

$$|f(x_k)| < \varepsilon$$



Metoda Newtona

Metoda Newtona jest szybko zbieżna (**kwadratowo**)

Metoda jest zawsze zbieżna do x_r jeśli:

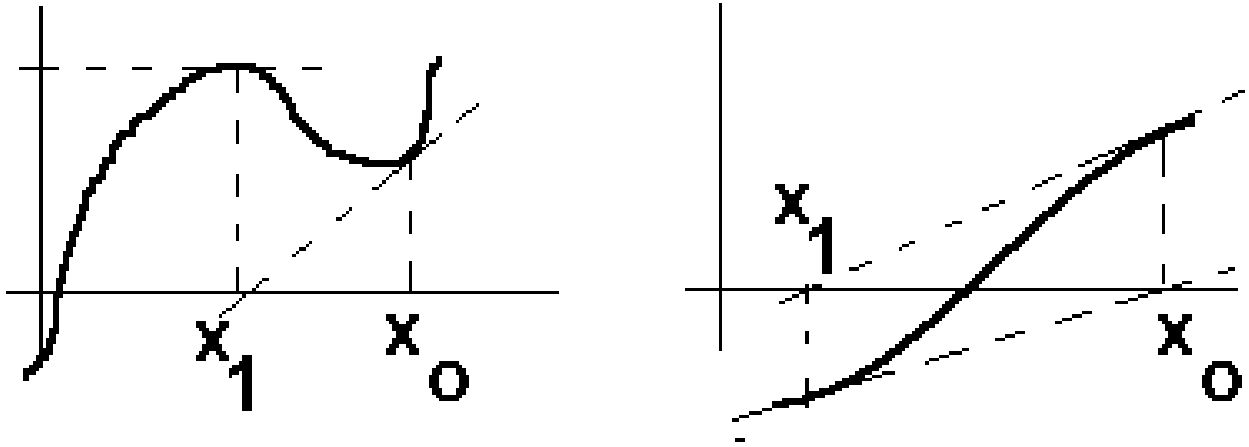
- w [x_o, x_r] nie ma innych pierwiastków,
- znak $f^{//}$ jest taki sam jak znak $f^{/}$

Zauważmy że w metodzie Newtona na starcie mamy tylko jeden punkt x_o

Metoda Newtona

- Metoda Newtona wymaga analitycznej znajomości pochodnej funkcji f(x) lub
- Gdy nie jest łatwo znaleźć $f'(x)$ lub gdy $f(x)$ dana jest w postaci stablicowanej, $f'(x)$ wyznaczamy numerycznie

Przykłady przypadkowego braku zbieżności w metodzie Newtona-Raphsona



Metoda Newtona

Metodę Newtona można łatwo uzyskać rozwijając funkcję

f(x) w szereg Taylora w pobliżu x_0

$$f(x) = f(x_o) + f'(x_o)(x - x_o) + f''(x_o)(x - x_o)^2/2! + \ldots$$

Wybierając tylko 2 pierwsze wyrazy rozwinięcia otrzymujemy

$$f(x) = f(x_o) + f'(x_o)(x - x_o) = 0$$

Metoda Newtona - przykład

Szukamy pierwiastków $x - x^{1/3} - 2 = 0$

Liczymy pochodną $f'(x) = 1 - \frac{1}{3}x^{-2/3}$

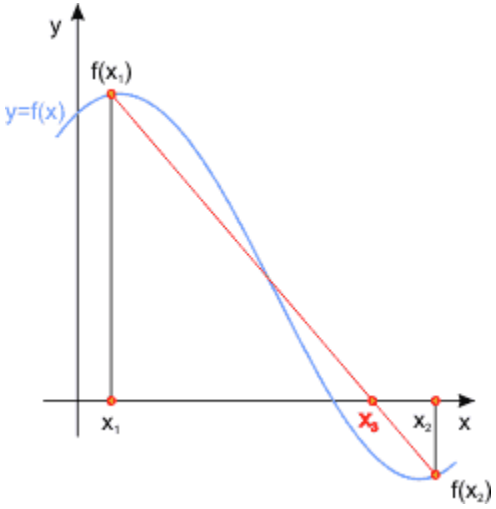
Równanie iteracyjne $x_{k+1} = x_k - \frac{x_k - x_k^{1/3} - 2}{1 - \frac{1}{3}x_k^{-2/3}}$

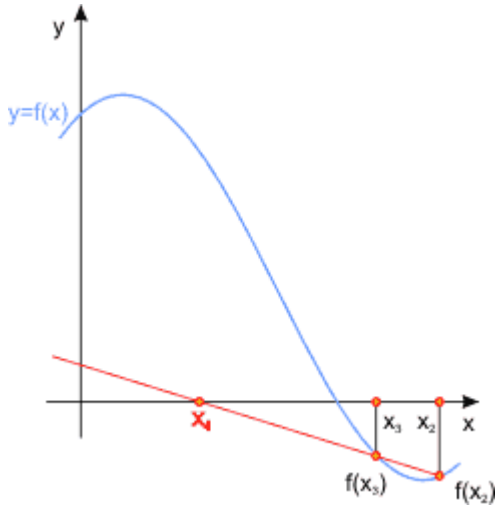
k	x_k	$f'(x_k)$	$f(x)$
0	3	0.83975005	-0.44224957
1	3.52664429	0.85612976	0.00450679
2	3.52138015	0.85598641	3.771×10^{-7}
3	3.52137971	0.85598640	2.664×10^{-15}
4	3.52137971	0.85598640	0.0

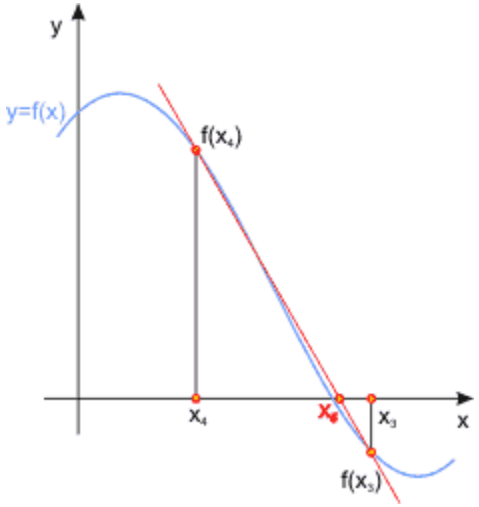
Metoda siecznych

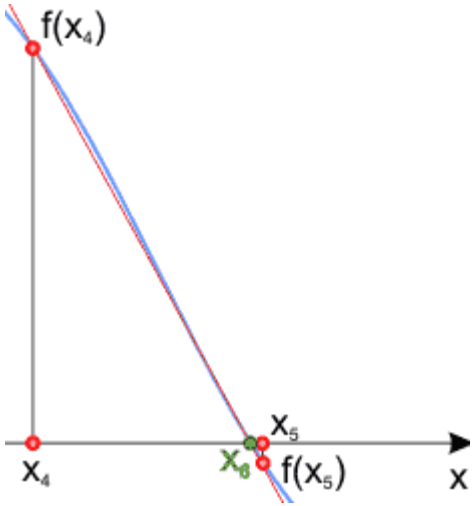
Metoda siecznych

Mamy daną funkcję f(x), dwa punkty startowe x_1 oraz x_2 i przedział <a,b> poszukiwań pierwiastka, do którego należą punkty x_1, x_2









Równanie prostej przechodzącej przez dwa punkty (x_A,y_A), (x_B,y_B)

$$(y-y_A)(x_B-x_A)-(y_B-y_A)(x-x_A)=0$$

Metoda siecznych

Szukamy kolejnych przybliżeń do rozwiązania prowadząc sieczne pomiędzy punktami x0,x1; x1,x2, itd..

Wzór ogólny na konkretne przybliżenia do miejsca zerowego

$$x_{n+1} = x_n - f(x_n)\left(\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}\right)$$

Iteracje tak jak poprzednio kończymy po uzyskaniu żądanej dokładności (uwaga na procesy rozbieżne! – ustalić maksymalna liczbę iteracji)

$$| x_{i-1} - x_{i-2} | < \varepsilon_x \quad \text{lub} \quad | f(x_i) | < \varepsilon_o$$

Metoda siecznych

Metoda siecznych można otrzymać jako przybliżenie metody Newtona

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Podstawiając za pochodną

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

$$x_{n+1} = x_n - f(x_n)\left(\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}\right)$$

Metoda Newtona

Metoda siecznych w ogólnym przypadku wymagać będzie większej liczby iteracji od metody Newtona.

Współczynnik zbieżności (informujący, jak błąd w kroku n + 1 zależy od błędu w kroku n) dla metody siecznych wynosi ok. **1.62**:

zaś w metodzie Newtona — dokładnie **2**:

Ale: metoda Newtona wymaga wyliczenia zarówno wartości funkcji, jak i wartości pochodnej tej funkcji.

Metoda siecznych

Program wyznacza zero funkcji f(x) metodą siecznych:

$$x_{i+1}=x_i - h*f(x_i)$$

$$h=(x_i-x_{i-1})/(f(x_i)-f(x_{i-1}))\quad i=2,3,...$$

Proces iteracyjny ulega przerwaniu, jeśli $|x_{i+1}-x_i|<eps$, $f(x_i)-f(x_{i-1})=0$, liczba iteracji przekroczy dozwoloną wartość, albo $|x_{i+1}-x\_i|$ w kolejnej iteracji będzie większe niż w poprzedniej.

Dane wejściowe: x1, x2, eps.

Metoda siecznych

Schemat algorytmu:

1. czytaj x1, x2 oraz eps
2. dane poprawne?
   if x1=x2 then
   wypisz komunikat o bledzie
   goto 1
3. poczatek iteracji
4. if f(x1)=f(x2) then
   wypisz komunikat o bledzie
   goto 1
5. h:=(x2-x1)/(f(x2)-f(x1))
6. x3:=x2-h*f(x2);
7. wypisz x oraz f(x)
8. if abs(x2-x1) < abs(x3-x2) then
   wypisz komunikat o bledzie
   goto 1
10. x2->x1; x3->x2
11. if abs(x2-x1)<eps then goto 12
    else goto 3
12. koniec programu

Metoda iteracji prostych

Metoda iteracji prostych

Równanie $f(x) = 0$

sprowadzamy do równoważnego

$$x = g(x)$$

Algorytm metody polega na wykorzystaniu schematu rekurencyjnego:

$$x_{k+1} = g(x_k),\ k = 0, 1, 2, \ldots$$

do generowania ciągu kolejnych przybliżeń poszukiwanego rozwiązania x.

Wygenerowany ciąg liczbowy $x_0, x_1, x_2, \ldots$ może okazać się zbieżny do rozwiązania x , ale tez może nie być zbieżny i wykazywać niestabilność.

Bardzo ważny jest wybór punktu startowego i odpowiedni wybór postaci $x = g(x)$

Rozważmy równanie $x^2 = 2$.
Przy założeniu, że $x \neq 0$ można napisać: $x = g(x) = 2/x$, a następnie skorzystać z reguły rekurencyjnej:

$$x_{k+1} = g(x) = \frac{2}{x_k}.$$

Przyjmując punkt startowy:

a) $x_0 = 0.5,\ x_1 = 2/0.5 = 4.0,\ x_2 = 2/4 = 0.5,\ x_3 = 2/0.5 = 4,\ \ldots$

b) $x_0 = 1.0,\ x_1 = 2/1.0 = 2.0,\ x_2 = 2/2 = 1.0,\ x_3 = 2/1.0 = 2,\ \ldots$

c) $x_0 = 2.0,\ x_1 = 2/2.0 = 1.0,\ x_2 = 2/1 = 2.0,\ x_3 = 2/2.0 = 1,\ \ldots$

Pokazane elementy wygenerowanych ciągów oscylują.

Takie zachowanie się algorytmu świadczy, że nie nadaje się on do rozwiązywania równania $x^2 = 2$.

Można jednak zaproponować inną postać algorytmu, dokonując przekształceń:

$$x^2 = 2 \quad \rightarrow x = \frac{2}{x} \quad \rightarrow x+x = x+\frac{2}{x} \quad \rightarrow 2x = x+\frac{2}{x} \quad \rightarrow x = \frac{1}{2}(x+\frac{2}{x})$$

Teraz formuła rekurencyjna przyjmie postać:

$$x_{k+1} = g(x_k) = \frac{1}{2}(x_k + \frac{2}{x_k})$$

Jeżeli teraz przyjmiemy $x_0 = 0.5$ to otrzymamy:

$$\begin{aligned} x_1 &= 2.25000 \\ x_3 &= 1.56944 \\ x_4 &= 1.42189 \\ x_5 &= 1.41423 \\ x_6 &= 1.41421 \end{aligned}$$

która dobrze przybliża liczbę $x^\star = \sqrt{2} \approx 1.41421356$.

$$x - x^{1/3} - 2 = 0$$

Trzy sposoby zapisu:

$$x_{k+1} = g_1(x_k) = x_k^{1/3} + 2$$

$$x_{k+1} = g_2(x_k) = (x_k - 2)^3$$

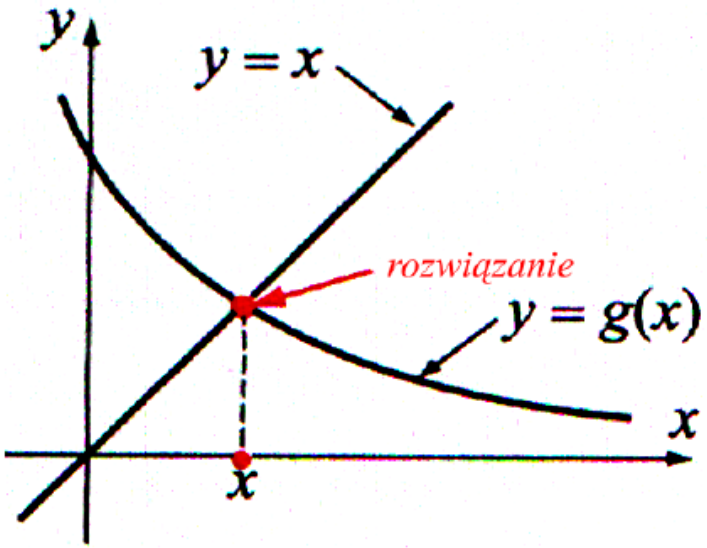
$$x_{k+1} = g_3(x_k) = \frac{6 + 2x_k^{1/3}}{3 - x_k^{2/3}}$$

k	$g_1(x_{k-1})$	$g_2(x_{k-1})$	$g_3(x_{k-1})$
0	3	3	3
1	3.4422495703	1	3.5266442931
2	3.5098974493	-1	3.5213801474
3	3.5197243050	-27	3.5213797068
4	3.5211412691	-24389	3.5213797068
5	3.5213453678	-1.451×10^{13}	3.5213797068
6	3.5213747615	-3.055×10^{39}	3.5213797068
7	3.5213789946	-2.852×10^{118}	3.5213797068
8	3.5213796042	∞	3.5213797068
9	3.5213796920	∞	3.5213797068

zbieżność **rozbieżność** **szybka zbieżność**

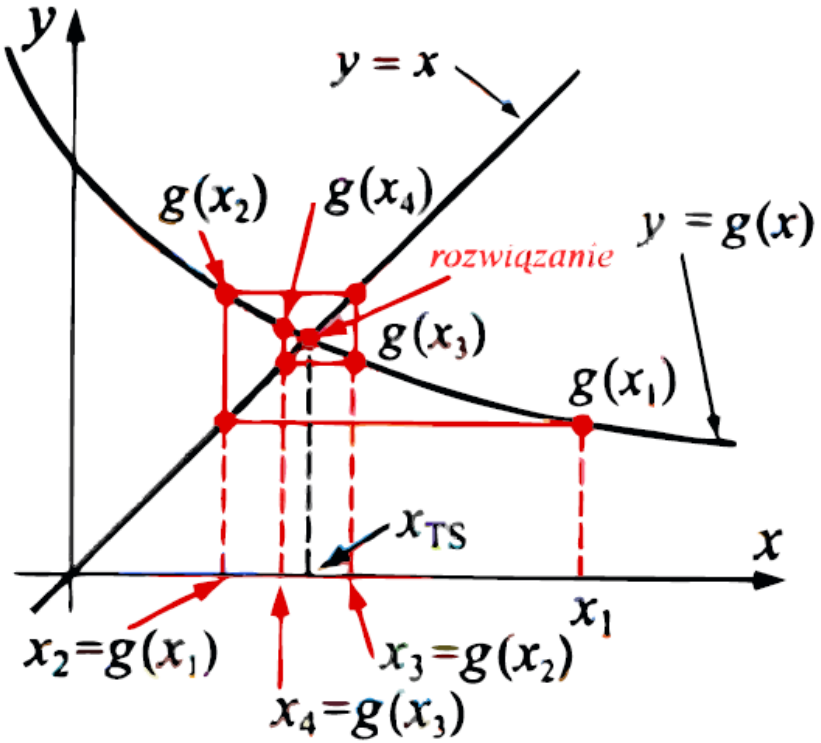
Metoda iteracji prostych (metoda punktu stałego)

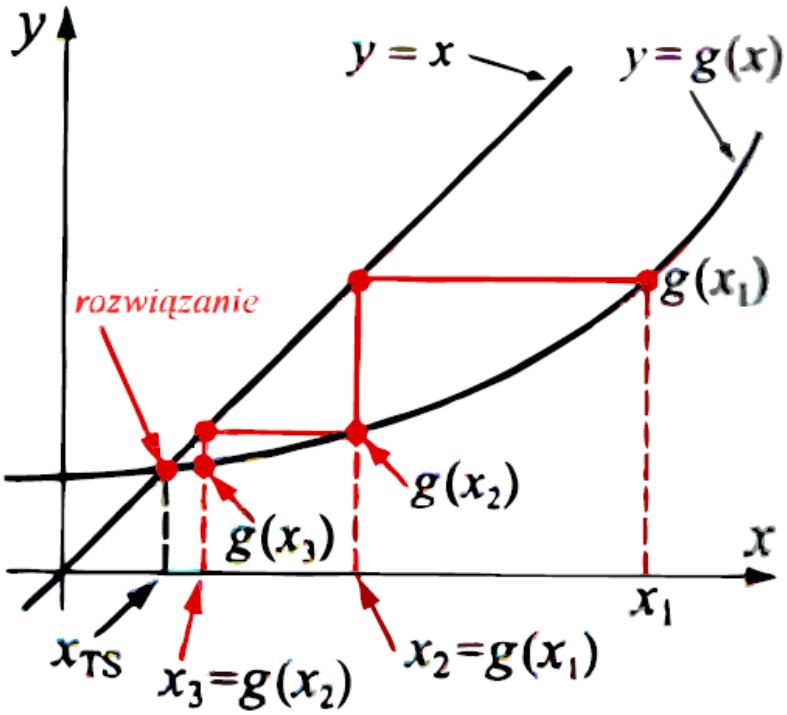
Równanie $f(x) = 0$ zastępujemy równaniem $x - g(x) = 0$, czyli $x = g(x)$



Rozwiązania szukamy iteracyjnie: $x_{k+1} = g(x_k)$

Metoda iteracji prostych (metoda punktu stałego)





Metoda interpolacji liniowej (regula falsi)

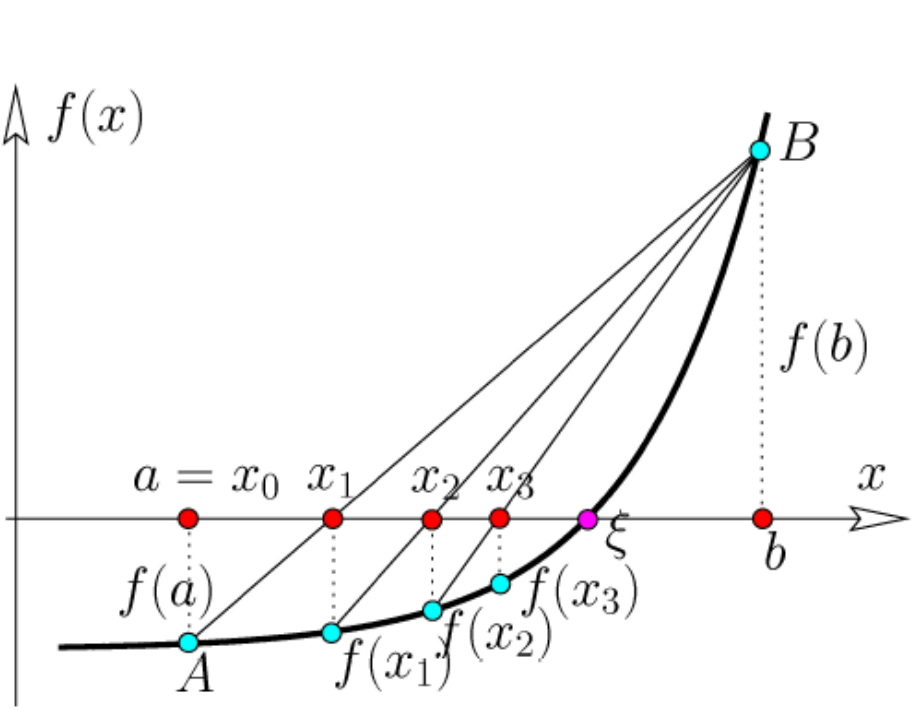
- Opisana w hinduskim tekście *Vaishali Ganit* (III w. p.n.e.)

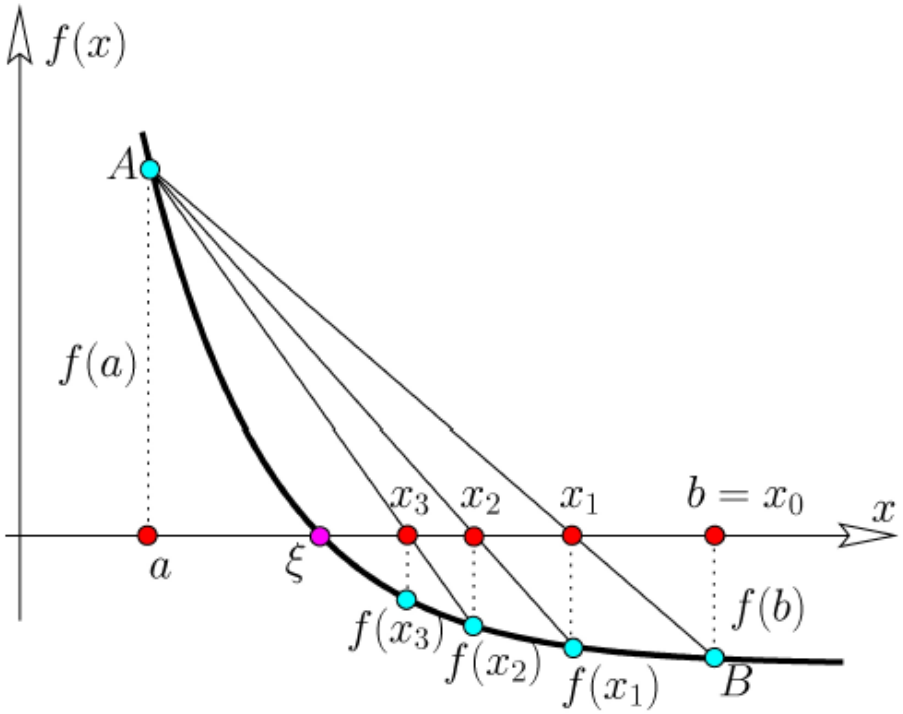
- *Dziewięć rozdziałów sztuki matematycznej* (九章算術) (200 p.n.e. – 100 n.e.) wykorzystuje algorytm do rozwiązywania równań liniowych

- *regula* – linia, *falsus* - fałszywy

Metoda interpolacji liniowej (regula falsi)

$$x_k = \frac{a\,f_b - b\,f_a}{f_b - f_a}$$





Metoda interpolacji liniowej (regula falsi)

W interpretacji geometrycznej metoda interpolacji liniowej oznacza zastąpienie krzywej $f(x)$ cięciwą łączącą punkty $A(a, f(a))$ i $B(b, f(b))$

$$\frac{x-a}{b-a} = \frac{y-f(a)}{f(b)-f(a)}.$$

Dla $y = 0$ mamy

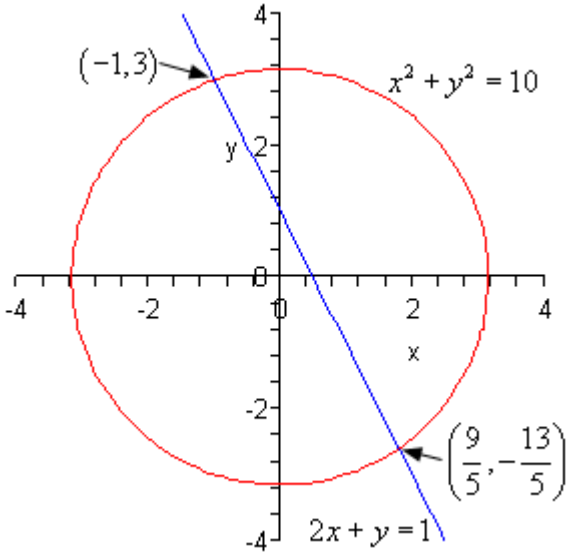
$$x_k = \frac{a\,f_b - b\,f_a}{f_b - f_a}$$

Układy równań nieliniowych

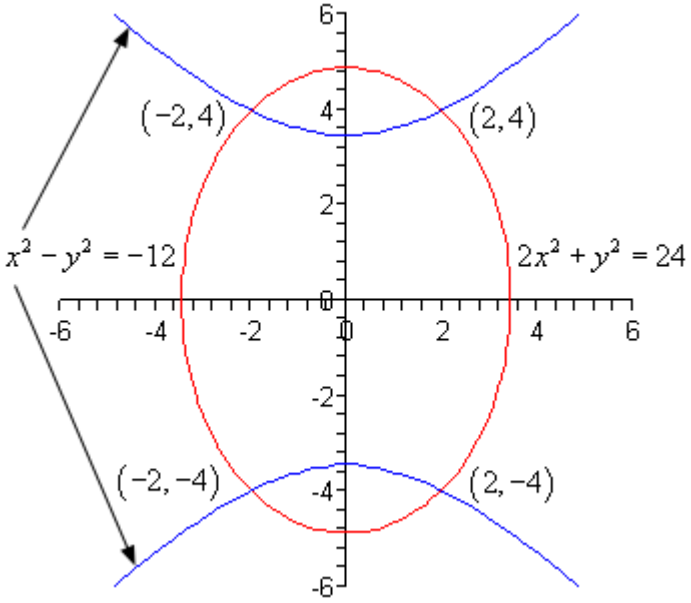
Układy równań nieliniowych

$$f(x,y) = 0$$
$$g(x,y) = 0$$

$$x^2 + y^2 = 10$$
$$2x + y = 1$$



$$2x^2 + y^2 = 24$$
$$x^2 - y^2 = -12$$



Rozwiązanie układu n równań nieliniowych sprowadza się do znalezienia $x_1, x_2, \ldots, x_n$ (czyli wektora $[x_1, x_2, \ldots, x_n]$)

Układy równań nieliniowych

Układ równań nieliniowych możemy zapisać następująco

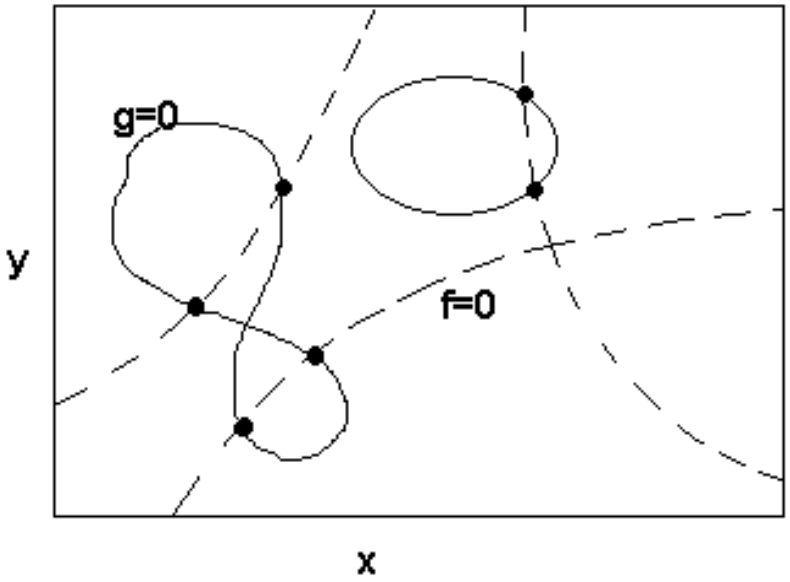
$$\begin{cases} f_1(x_1, x_2 \ldots x_n) = 0 \\ f_2(x_1, x_2 \ldots x_n) = 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ f_n(x_1, x_2 \ldots x_n) = 0 \end{cases}$$

$$\boldsymbol{f}(\mathbf{x}) = \begin{bmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \ldots \\ f_n(x_1, x_2, \ldots, x_n) \end{bmatrix} = 0$$

Rozwiązanie układu równań nieliniowych sprowadza się do znalezienia $x_1, x_2, \ldots, x_n$ (czyli wektora $[x_1, x_2, \ldots, x_n]$)

Dla n=2 mamy np.

$$f(x,y)=0$$
$$g(x,y)=0$$



Analogicznie jak dla metody Newtona, rozwijamy nasze funkcje w szeregi Taylora

$$0 \approx f(\mathbf{x}^0) + f'(\mathbf{x}^0)(\xi - \mathbf{x}^0)$$

Macierz określającą pochodne f'(x0) nazywamy macierzą Jacobiego i def. następująco

$$\mathbf{J}(f(\mathbf{x})) = f'(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

metodę Newtona uogólnia się na n-wymiarów

The Taylor series expansion is written for each $f_i(\mathbf{x})$

$$f_1(x+\Delta x) = f_1(x) + \frac{\partial f_1}{\partial x_1}(x)\Delta x_1 + \frac{\partial f_1}{\partial x_2}(x)\Delta x_2 + \ldots$$

$$\frac{\partial f_1}{\partial x_n}(x)\Delta x_n + \text{higher order terms}$$

$$\vdots$$

$$f_n(x+\Delta x) = f_n(x) + \frac{\partial f_n}{\partial x_1}(x)\Delta x_1 + \frac{\partial f_n}{\partial x_2}(x)\Delta x_2 + \ldots$$

$$\frac{\partial f_n}{\partial x_n}(x)\Delta x_n + \text{higher order terms}$$

This can be written more compactly in matrix form

$$\mathbf{f}(\mathbf{x}+\Delta\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{bmatrix} + \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \frac{\partial f_1}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{x}) & \frac{\partial f_2}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial f_2}{\partial x_n}(\mathbf{x}) \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}) & \frac{\partial f_n}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial f_n}{\partial x_n}(\mathbf{x}) \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{bmatrix}$$

\+ higher order terms

Analogicznie jak dla metody Newtona, metoda iteracyjna obliczania kolejnych pierwiastków daje się zapisać

$$x^{k+1} = x^k - J^{-1}(f(x^k))\, f(x^k)$$

k=0,1,2, …

Startujemy z $\mathbf{x}^0$, obliczamy $f(\mathbf{x}^0)$, $J^{-1}(f(\mathbf{x}^0))$, $\mathbf{x}^0$

problem – liczenie macierzy odwrotnej

$$* J(f(x^k)) \qquad x^{k+1} = x^k - J^{-1}(f(x^k))\, f(x^k)$$

Obliczanie macierzy odwrotnej Jacobianu zastępuje się rozwiązaniem układu równań liniowych, tożsamym z w powyższym równaniem

$$\mathbf{J}(f(\mathbf{x}^k))\mathbf{z}^k = -f(\mathbf{x}^k) \qquad (A\mathbf{z}=\mathbf{b})$$

$$\mathbf{z}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$$

Kolejne kroki w rozwiązywaniu problemu (metoda Newtona Raphsona): k=0, …, N -> max liczba kroków

- oblicz $f(\mathbf{x}^k)$,
- oblicz $\mathbf{J}\left(f(\mathbf{x}^k)\right) = f'(\mathbf{x}^k)$,
- rozwiąż układ równań liniowych $\mathbf{J}\left(f(\mathbf{x}^k)\right)\mathbf{z}^k = f(\mathbf{x}^k)$
- podstaw $\mathbf{x}^{k+1} = \mathbf{x}^k - \mathbf{z}^k$

Bardzo ważny wektor startowy $\mathbf{x}^0$

Metoda Newtona Raphsona, kwadratowo zbieżna, ale wymaga

obliczenia Jacobianu (pochodnych), rozwiązywania w każdym kroku układu równań liniowych (dodatkowy koszt numeryczny zarówno czasowy jak i pamięć RAM)

Iteracje kończy się gdy $\|\mathbf{x_{k+1}} - \mathbf{x_k}\| \leqslant \epsilon$

Uogólnienie metody Newtona-Raphsona (metoda stycznej w punkcie) na przypadek wielowymiarowy można zapisać wektorowo w następujący sposób:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\mathbf{f}(\mathbf{x}_k)}{\mathbf{J}(\mathbf{x}_k)}, \quad k = 0, 1, \ldots \tag{18}$$

wektor zmiennych	wektor funkcji	macierz Jakobianu
$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ x_n \end{bmatrix}$	$\mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ . \\ f_n \end{bmatrix}$	$\mathbf{J} = \frac{\partial f_i}{\partial x_j} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_2}{\partial x_n} \\ & \cdots\cdots & \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \end{bmatrix}$

lub $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k, \quad k = 0, 1, \ldots,$

gdzie $\Delta\mathbf{x}_k = -\Big(\mathbf{J}(\mathbf{x}_k)\Big)^{-1} \mathbf{f}(x_k).$

W celu uniknięcia odwracania Jakobianu, rozwiązywany jest układ równań $\mathbf{J}(\mathbf{x}_k)\,\Delta\mathbf{x}_k = -\mathbf{f}(x_k)$ do określenia $\Delta\mathbf{x}_k$.

Problemy metody NR dla układu r-ń nieliniowych

- Operacje na macierzach
- wektor startowy (trudny do oszacowania)
- odwracanie macierzy Jacobianu
- lub wielokrotne rozwiązywanie układu równań liniowych
- wielokrotne rozwiązania
- przynajmniej 2 macierze w pamięci operacyjnej komputera
- konieczność liczenia pochodnych (Jacobianu)
- alternatywa – metoda iteracji prostych

Metoda iteracji prostych Jacobiego (przykład CCD)

Układ równań nieliniowych (drugiego stopnia), równania metody sprzężonych klasterów (CCD). Niewiadome to amplitudy t_{ab}^{rs}

a,b =1 do occ, r,s 1 do virt

$$\langle rs||ab\rangle + (\varepsilon_r + \varepsilon_s - \varepsilon_a - \varepsilon_b)t_{ab}^{rs} + \sum_{t<u}^{\text{vir}} \langle rs||tu\rangle t_{ab}^{tu} + \sum_{c<d}^{\text{occ}} \langle cd||ab\rangle t_{cd}^{rs}$$

$$+ \sum_{d}^{\text{occ}} \sum_{u}^{\text{vir}} \langle ds||ub\rangle t_{ad}^{ru} - \sum_{c}^{\text{occ}} \sum_{u}^{\text{vir}} \langle cs||ub\rangle t_{ca}^{ru} - \sum_{d}^{\text{occ}} \sum_{t}^{\text{vir}} \langle ds||tb\rangle t_{ad}^{tr} + \sum_{c}^{\text{occ}} \sum_{t}^{\text{vir}} \langle cs||tb\rangle t_{ca}^{tr}$$

$$+ \sum_{c<d}^{\text{occ}} \sum_{t<u}^{\text{vir}} \langle cd||tu\rangle (t_{ab}^{rs} * t_{cd}^{tu} - t_{ab}^{rs} t_{cd}^{tu}) = 0$$

Rozwiązujemy iteracyjnie starując z t_{ab}^{rs} =0,

Pierwsza iteracja

$$\langle rs||ab\rangle + (\varepsilon_r + \varepsilon_s - \varepsilon_a - \varepsilon_b) t_{ab}^{rs,(1)} = 0,$$

$$t_{ab}^{rs,(1)} = -\frac{\langle rs||ab\rangle}{\varepsilon_r + \varepsilon_s - \varepsilon_a - \varepsilon_b}.$$