Rozwiązywanie równań nieliniowych

Metody Numeryczne

dr eng. Grzegorz Fotyga

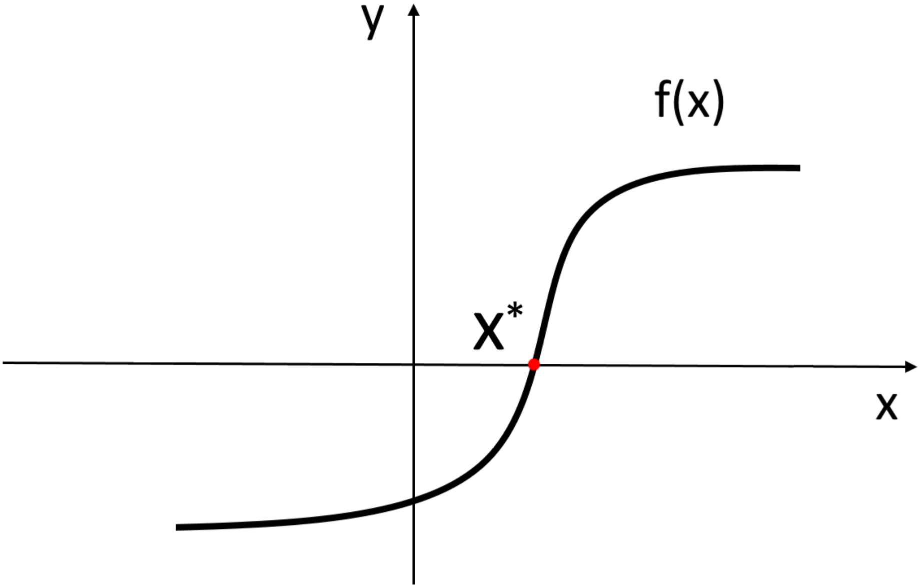
Gdańsk University of Technology
Faculty of Electronics, Telecommunications and Informatics
Department of Microwave and Antenna Engineering

6 kwietnia 2018

Lecture content

1. Basic assumptions
2. Klasyfikacja funkcji
3. Bisekcja
4. Metoda Newtona
5. Metoda siecznych

Basic assumptions (0)

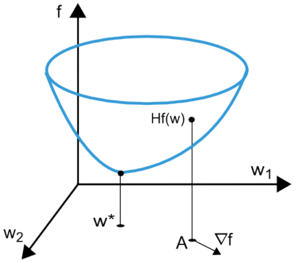


Podstawowe założenia (1)

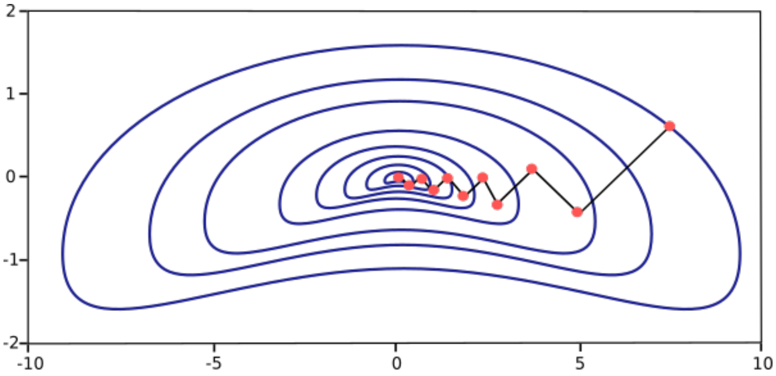
- Jedna z najczęściej spotykanych operacji w nauce i inzynierii to **rozwiązywanie równań nieliniowych**.
- funkcja $f(x)$ - szukamy punktu $x^* \in \mathbb{R}$, dla którego $f(x^*) = 0$.
- x^* jest nazywane miejscem zerowym (root) f.
- Punkt przecięcia dwóch krzywych: $f(x)$ i $g(x)$ odpowiada miejscu zerowemu funkcji: $h(x) = f(x) - g(x)$
- Wiele praktycznych problemów wymaga znalezienia miejsca zerowego funkcji **wielu zmiennych** (również zespolonych)
- Jest wiele *black-box* metod szukania miejsc zerowych. Jest konieczne, żeby mieć **wiedzę** o zasadach działania poszczególnych metod, żeby dobrać odpowiednią metodę i odpowiednie parametry w zależności od analizowanego problemu. Ew. żeby sprawnie znaleźć błędy.
- Algorytmy szukania miejsc zerowych dostarczają w kolejnych iteracjach kolejne przybliżenie położenia miejsca zerowego $(x_0, x_1, x_2, \dots)$. Ich działanie zaczyna się od **punktu startowego** lub od **przedziału**, który zawiera miejsce zerowe.

Podstawowe założenia (2) - neural networks

- *https : //www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network*
- The procedure used to carry out the learning process in a neural network is called the training algorithm.
- The learning problem in neural networks is formulated in terms of the minimization of a loss **multi-variable** function, $f(w)$.
- The learning problem for neural networks is formulated as searching of a parameter vector w* at which the loss function f takes a minimum value.

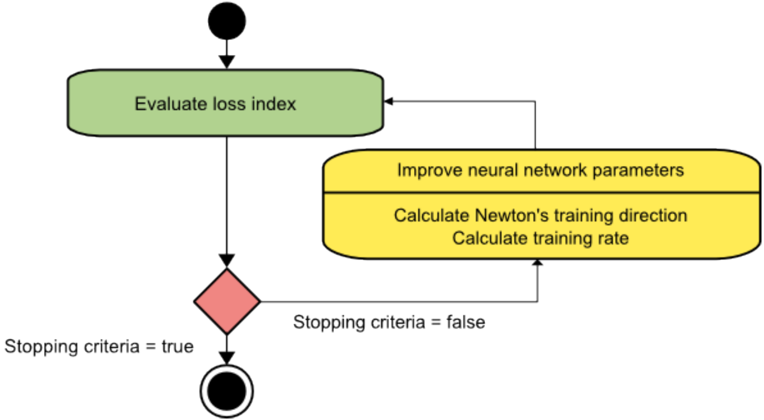


Podstawowe założenia (3) - neural networks

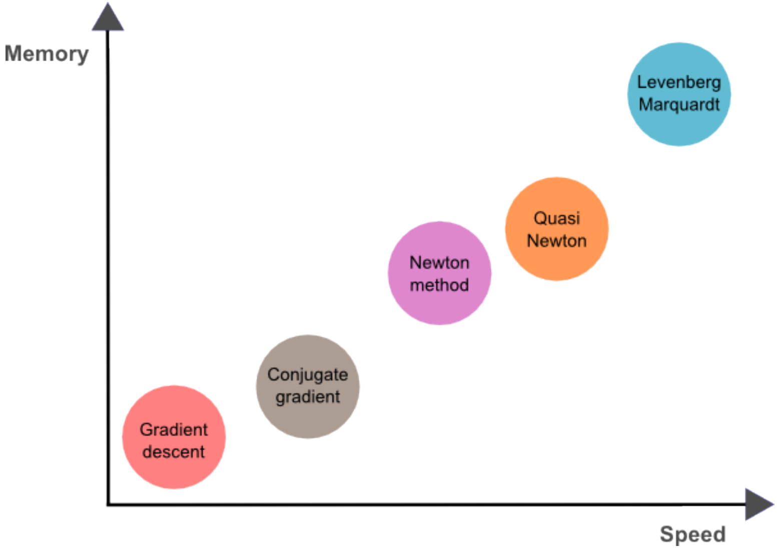


Podstawowe założenia (4) - neural networks

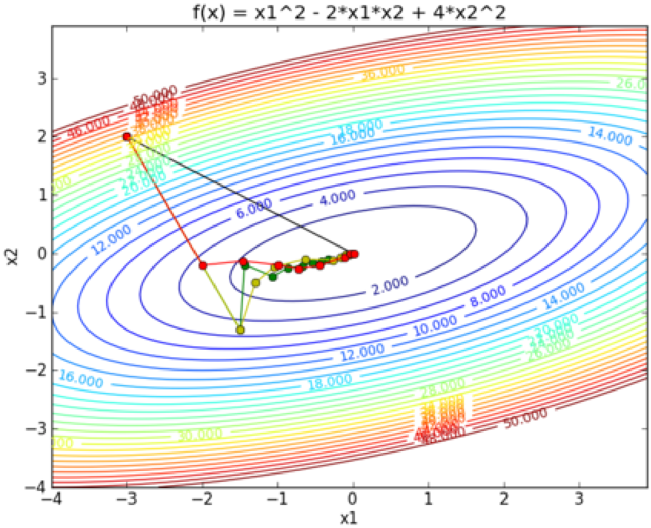
Newton's method



Podstawowe założenia (5) - neural networks



Podstawowe założenia (6) - optimization problems



Podstawowe założenia (7) - 3D graphics

Podstawowe założenia (8) - 3D graphics

The inverse square root of a floating point number is used in calculating a **normalized vector**. Programs can use normalized vectors to determine angles of incidence and reflection. 3D graphics programs must perform millions of these calculations every second to simulate lighting. Solution $\Rightarrow$ **Fast inverse square root** (Newton's method involved) *https://en.wikipedia.org/wiki/Fast_inverse_square_root*

Podstawowe założenia (9) - Quake

The following code is the fast inverse square root implementation from Quake III Arena, stripped of C preprocessor directives, but including the exact **original comment text**:

```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y  = number;
    i  = * ( long * ) &y;                       // evil floating point bit level hacking
    i  = 0x5f3759df - ( i >> 1 );               // what the fuck?
    y  = * ( float * ) &i;
    y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
//      y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration, this can be removed

    return y;
}
```

Newton

pol. skąd ta liczba?

Podstawowe założenia (10)

Wykład i laboratoria uwzględniają jedynie funkcje jednej zmiennej $f(x)$. Jednak zrozumienie zasad działania poszczególnych metod dla $f(x)$ jest **konieczne**, żeby zrozumieć i stosować metody *wielowymiarowe*.

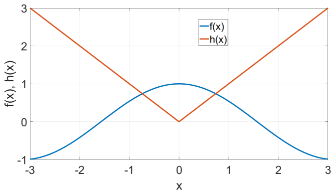
Klasyfikacja funkcji (1)

Klasyfikacja funkcji:

- *Ciągłość* - funkcja f jest *ciągła* jeżeli $(f(x_1) - f(x_2)) \to 0$, jeśli $x_1 \to x_2$.
- *Różniczkowalność* - funkcja jest *różniczkowalna*, jeżeli ma pochodną (f') w każdym punkcie swej dziedziny, której wartość w każdym punkcie jest skończona.

Przykład

- $f(x) = cos(x)$ jest ciągła i różniczkowalna
- $h(x) = |x|$ jest ciągła ale **nie** różniczkowalna dzięki **osobliwości** w $x = 0$.
- $g(x) = 1$ dla $x \in \mathbb{N}$ i $g(x) = -1$ w przeciwnym wypadku - jest **nieciągła** i **nie** różniczkowalna.

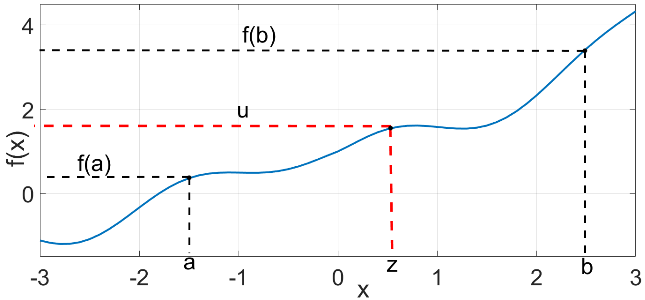


Bisekcja (1)

Bisekcja - **najprostsza** do implementacji metoda szukania miejsc zerowych (ale zdecydowanie **nie najszybsza**!)

Twierdzenie o wartości pośredniej (Twierdzenie Darboux)

- Załóżmy, że $f : [a, b] \to \mathbb{R}$ jest ciągła i $f(a) < u < f(b)$ or $f(b) < u < f(a)$. Wtedy istnieje $z \in (a, b)$, dla którego: $f(z) = u$.
- **Innymi słowy** - f osiąga **wszystkie wartości** z przedziału $f(a)$ i $f(b)$.



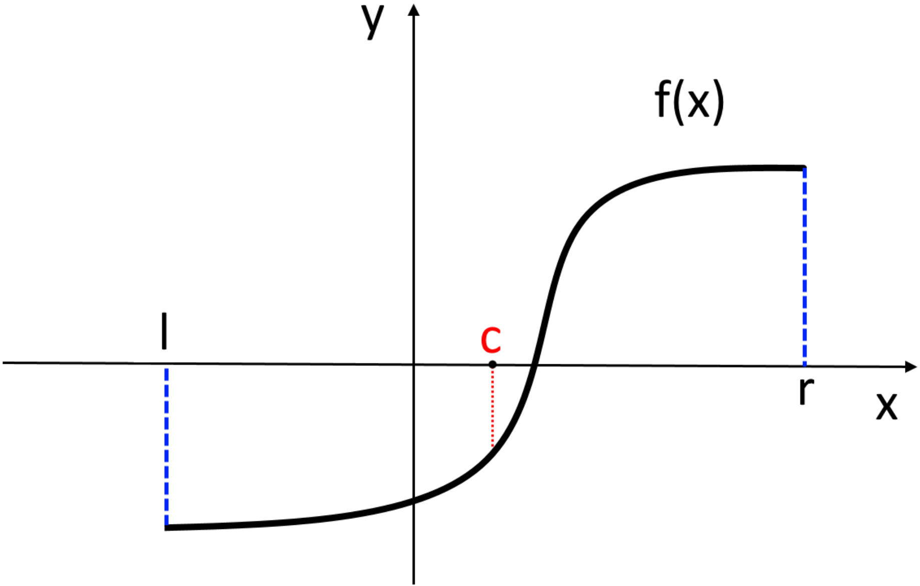
Bisekcja (2)

- Rozpatrzmy funkcję $f(x)$ i dwie wartości: l and r, takie, że: $f(l)$ i $f(r)$ mają **przeciwne znaki**: $f(l)f(r) < 0$
- Korzystając z twierdzenia o wartości pośredniej, między l i r **jest miejsce zerowe**.
- Fakt ten sugeruje, że można zastosować iteracyjną metodę podziału przedziału, w celu znalezienia x^*: dzielimy rekurencyjnie przedział $[l, r]$ na **równe odcinki**, wybierając tą połowę, gdzie występuje miejsce zerowe.

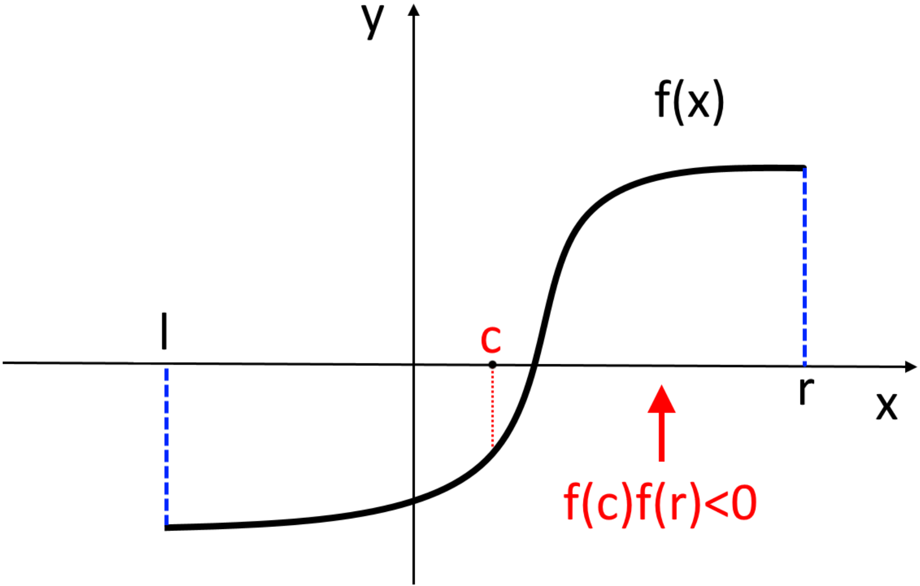
Algorithm 1: Bisekcja

Data: $f(x)$, l, r
for $k \leftarrow$ *1,2,3,...* **do**
    c $\leftarrow$ (l+r)/2
    if $|f(c)| < \epsilon_f$ *or* $|r - l| < \epsilon_x$ **then**
        return $x^* \approx c$
    else if $f(l)f(c) < 0$ **then**
        $r \leftarrow c$
    else
        $l \leftarrow c$
    end
end

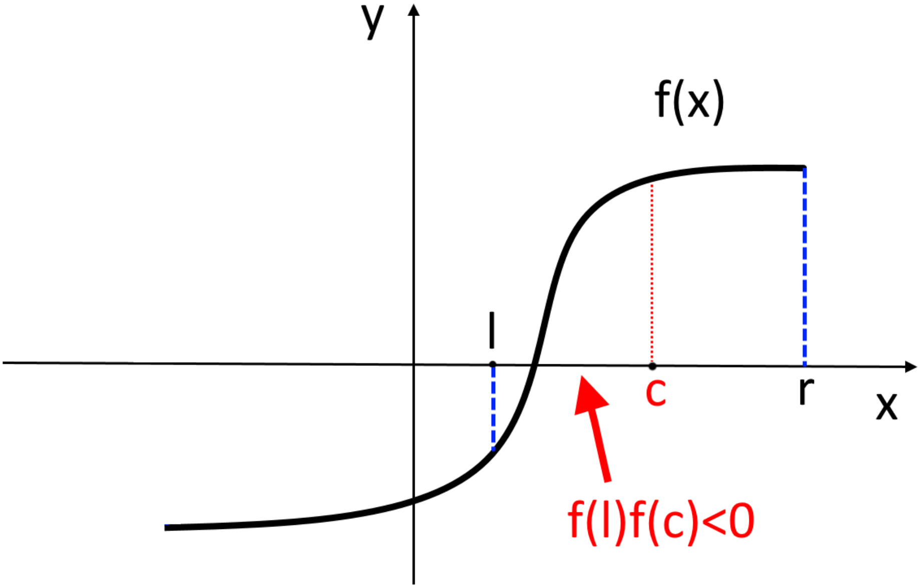
Bisekcja (3) - Przykład



Bisekcja (3) - Przykład



Bisekcja (3) - Przykład



Bisekcja (3) - Przykład

Przykład

- Oblicz $x = 1/\sqrt{11}$ używając bisekcji
- Ustaw: $\epsilon_f = \epsilon_x = 1e - 5$
- Następnie przekształć powyższą zależność do postaci: $f(x) = x^2 - 1/11$
- Przedział, który zawiera miejsce zerowe: $[0, 10] \Rightarrow x_0 = 5$
- Kolejne iteracje:
  1. k $= 1$: $c = 5$, $(f(0)f(5) < 0) \Rightarrow r = 5$
  2. k $= 2$: $c = 2.5$, $(f(0)f(2.5) < 0) \Rightarrow r = 2.5$
  3. k $= 3$: $c = 1.25$, $(f(0)f(1.25) < 0) \Rightarrow r = 1.25$
  4. ...
  5. k $= 20$: $c = 0.301504135131836$, $(f(x) = 0.000004347407495) < \epsilon_f \Rightarrow$ **BREAK**

Bisekcja (4) - Przykład

iteration	**c**	**f(c)**
1	5	-24.90909090909910
2	2.5	-6.15909090909090909
3	1.25	-1.47159090909090909
4	0.625	-0.29971590909090909
5	0.3125	-0.00674715909090909
...	...	...
18	0.301475524902344	0.000021598793947
19	0.301494598388672	0.000010098051544
20	**0.301504135131836**	**0.000004347407495**

Bisekcja (5) - błąd, zbieżność

- W każdej iteracji dzielimy przedział na **pół**, więc w każdej iteracji maksymalna wartość błędu jest zmniejszana o połowę: $E_{k+1} = E_k/2$, ponieważ $E_k = |r_k - l_k|/2$
- Więc bisekcja charakteryzuje się **liniowa zbieżnością**.
- Mimo, że bisekcja jest **wolna**, daje **gwarancję** zbieżności dla każdej funkcji ciągłej.
- **Jednak** jeżeli wiemy więcej o $f(x)$, możemy sformułować algorytm, który zbiega się **szybciej**.

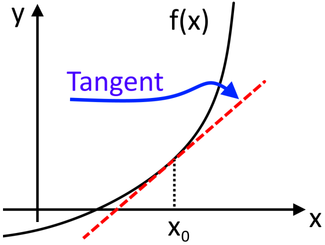
Metoda Newtona (1)

- rozpatrzmy funkcję $f(x)$ która jest ciągła i różniczkowalna.
- Metoda Newtona zaczyna się od zdefiniowania punktu początkowego: x_0 (możliwie blisko miejsca zerowego: x^*).
- Wartość f blisko x_0 (w x_1) może być **aproksymowana** za pomocą:

$$f(x_1) \approx f(x_0) + f'(x_0)(x_1 - x_0) \tag{1}$$

- Linia **styczna** do f w x_0 jest zdefiniowana następująco:

$$y = f(x_0) + f'(x_0)(x - x_0) \tag{2}$$



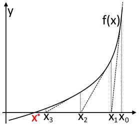
Metoda Newtona (2)

- Podstawienie $y = 0$ i przekształcenie (2) pozwala na zdefiniowanie sformułowania do obliczenia x_1:

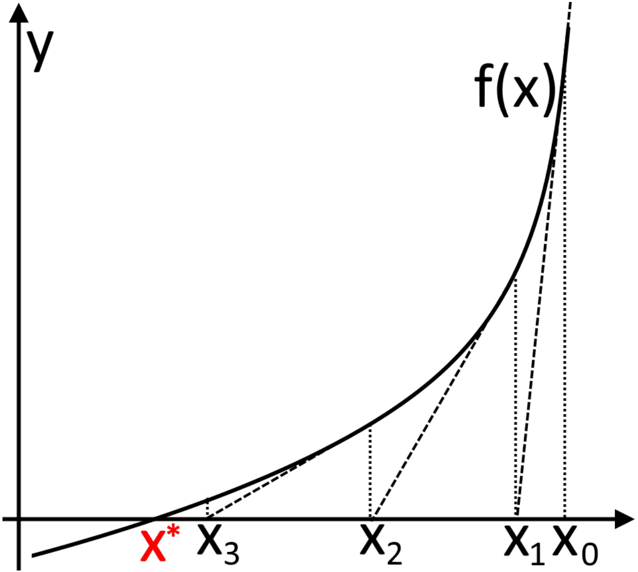
$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{3}$$

  zakładając $f'(x_0) \neq 0$.
- W rzeczywistości, x_1 może nie spełniać: $f(x_1) = 0$, ale **możemy mieć nadzieję**, że jest **bliżej** x^*, niż x_0.
- Ostatecznie otrzymujemy następujące **sformułowanie** na kolejne przybliżenia miejsca zerowego:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{4}$$



Metoda Newtona (2b)



Metoda Newtona (3)

Example

- Compute $x = 1/\sqrt{11}$ using Newton's method ($x^2 = 1/11, x > 0$)
- Set: $\epsilon_f = \epsilon_x = 1e - 10$
- Next, we transform the above formula to: $f(x) = x^2 - 1/11$
- The starting guess: $x_0 = 2$
- The derivative: $f'(x) = 2x$
- The subsequent steps:

iteration	x	$f(x_k)$
0	2.000000000000000	3.909090909090909
1	1.022727272727273	0.955061983471074
...	...	...
5	0.301547480958557	0.000021792363360
6	0.301511346742992	0.000000001305682
8	**0.301511344577764**	**0.000000000000000**

Metoda Newtona (4)

Cechy metoda Newtona

- Metoda Newtona **zbiega się**, jeżeli x_0 jest odpowiednio *blisko* x^*.
- Wymaga podania jednego **punktu startowego** (nie przedziału)t.
- Gwarantuje **kwadratową** zbieżność ($E_{k+1} < cE_k^2$, gdzie c jest stałą).
- Stop - jeżeli $|f(x_k)| < \epsilon_f$
- **Jednak** - wymaga wyznaczenia pochodnej: f',co może być **czasochłonne** (szczególnie dla problemów wielu zmiennych!)
- Jak ominąć tą niedogodność? $\Rightarrow$ **Metoda siecznych!**

Metoda siecznych (1) - własności

Metoda siecznych **własności**

- Nie wymaga jawnego obliczania pochodnej (f')
- f' jest aproksymowana za pomocą:

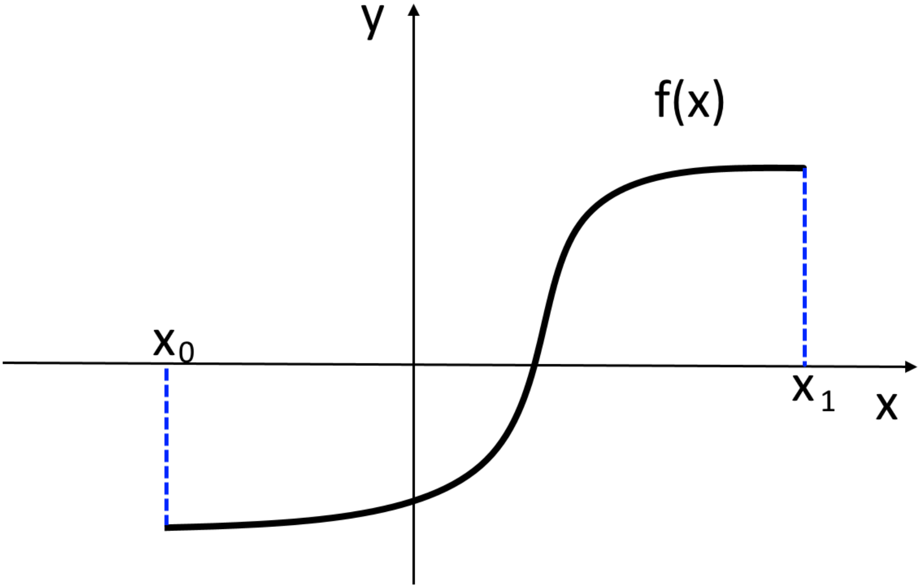
$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \tag{5}$$

- W efekcie (podstawiając równanie (5) to (4)), otrzymujemy:

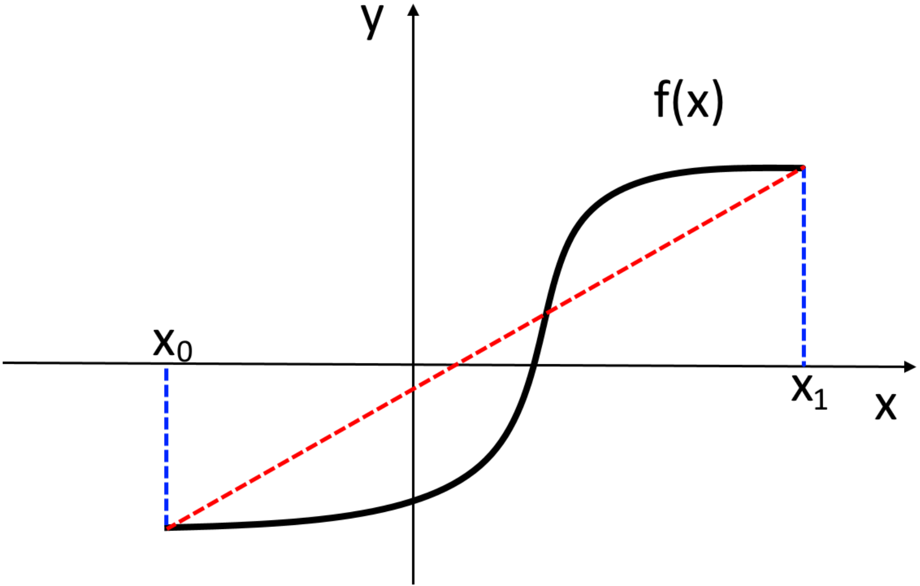
$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})} \tag{6}$$

- Zbieżność - między bisekcją (liniowa), a Newtona (kwadratowa): $E_{k+1} < cE_k^{(1+\sqrt{5})/2)}$, gdzie c jest stałą.
- Zaczyna się od zdefiniowania przedziału $[a, b]$ (podobnie jak w metodzie bisekcji), gdzie $x_0 = a$ i $x_1 = b$.

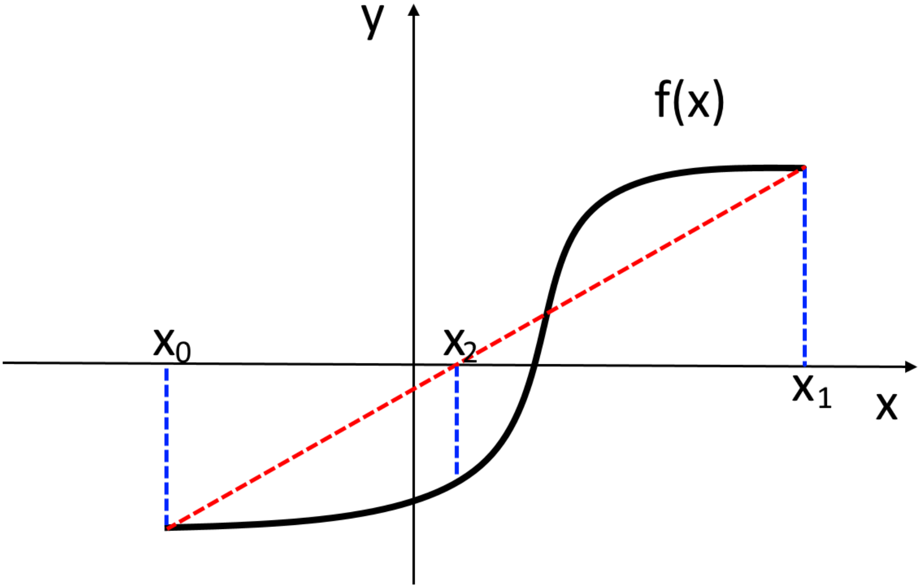
Metoda siecznych (2) - przykład



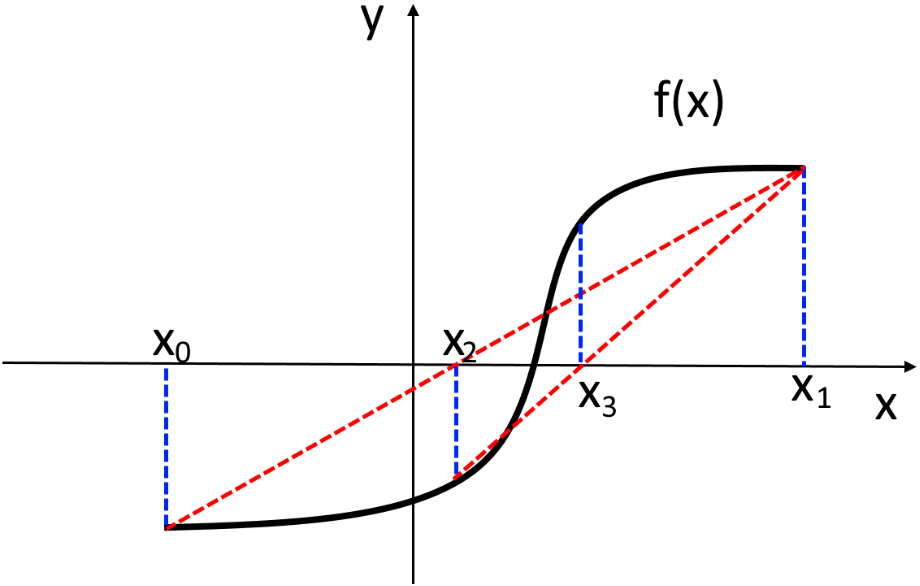
Metoda siecznych (3) - przykład



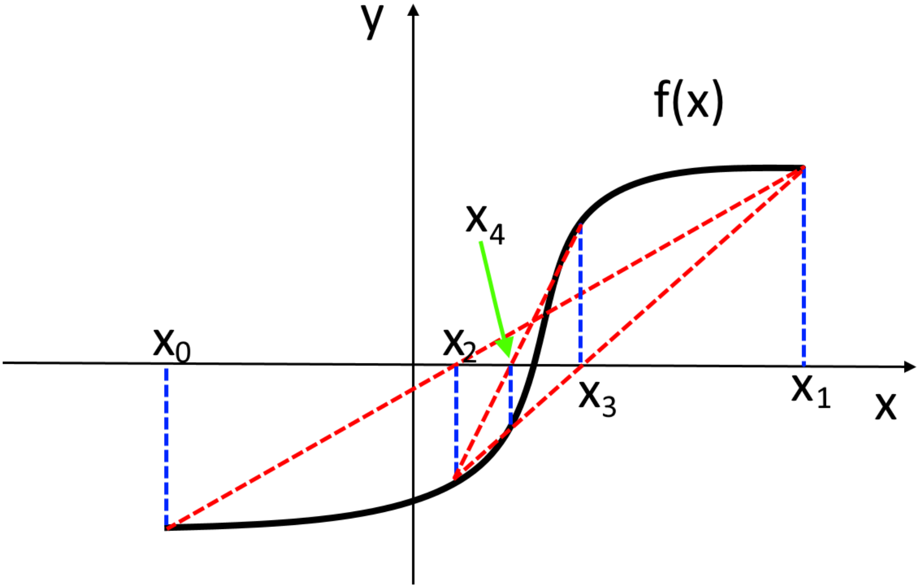
Metoda siecznych (4) - przykład



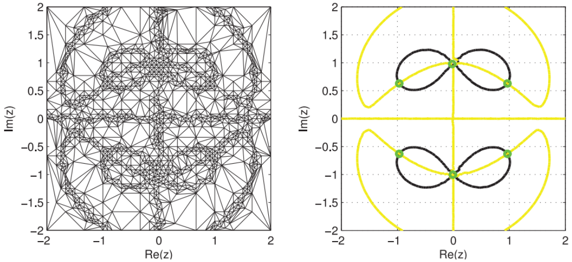
Metoda siecznych (5) - przykład



Metoda siecznych (6) - przykład



Szukanie miejsc zerowych funkcji zespolonych - przykład



Kowalczyk, Piotr. "Complex root finding algorithm based on delaunay triangulation." ACM Transactions on Mathematical Software (TOMS) 41.3 (2015): 19.

Thank you