CEL ĆWICZEŃ

Zaimplementować metodę bisekcji, metodę Newtona (metoda stycznych), oraz metodę siecznych do rozwiązywania *równań nieliniowych z jedną niewiadomą*. Przeprowadzić testy programów dla przykładowych równań.

WPROWADZENIE

Najprostszym równaniem z jedną niewiadomą jest równanie liniowe $ax=b,$ które ma jedno rozwiązanie $x=\frac{b}{a}$, pod warunkiem, że $a\neq 0.$ Nieco trudniejsze jest równanie kwadratowe $ax^2+bx+c=0,$ w którym $a\neq 0.$ Ma ono rozwiązania $x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$, które są liczbami rzeczywistymi $(x\in\mathbb{R}),$ gdy $b^2-4ac\geq 0.$ Ale dla równanie trzeciego stopnia, $ax^3+bx^2+cx+d=0,$ wzory są już dość skomplikowane i na ogół mało przydatne. Przykładowo, najprostsze nietrywialne równanie trzeciego stopnia, $x^3+x+1=0,$ ma dwa pierwiastki zespolone (pomijamy) i jeden rzeczywisty:

$$x=-\sqrt[3]{\frac{2}{3(-9+\sqrt{93})}}+\sqrt[3]{\frac{-9+\sqrt{93}}{18}}\approx -0{,}682328. \qquad (1)$$

Równanie wielomianowe trzeciego stopnia pojawia się gdy rozważamy równania stanu gazu van der Waalsa podającego dokładniej dla gazów rzeczywistych zależność pomiędzy ciśnieniem, objętością molową i temperaturą niż równanie gazy doskonałego $(P\upsilon=RT).$ Jeżeli dana jest temperatura (T) oraz ciśnienie (P), to obliczenie objętości molowej (v) wymaga rozwiązania równania:

$$(P+\alpha/\upsilon^2)(\upsilon-\beta)=RT.$$

W praktyce i tak nas interesuje to, co we wzorze (1) znajduje się na końcu – czyli przybliżenie dziesiętne. Ponadto w wielu sytuacjach występują równanie, które nie są nawet algebraiczne (wielomianowe), na przykład

$$x-\cos x=0,\quad (3-x)e^x-3=0,\quad M=x-e\sin x,\quad \upsilon\left(\frac{1}{R}+\mu\gamma\right)-\mu\upsilon^2+\alpha(e^{\upsilon/\beta}-1)+\frac{E}{R}=0,\quad \text{itp.} \qquad (2)$$

Drugie równanie z powyższego zestawu to równanie Keplera, które opisuje pewne parametry orbity eliptycznej, gdzie $x=E$ to tak zwana anomalia mimośrodowa, a M to anomalia średnia. Przy danych M i e należy obliczyć $E.$ Ostatnie równanie pojawia się przy analizie obwodu elektrycznego z diodą tunelową. W równaniu tym E jest napięciem z generatora napięcia, R to opór omowy (rezystancja), $\alpha,\ \beta>0$ to pewne parametry obwodu, a υ jest szukanym napięciem na diodzie (tzw. punkt pracy).

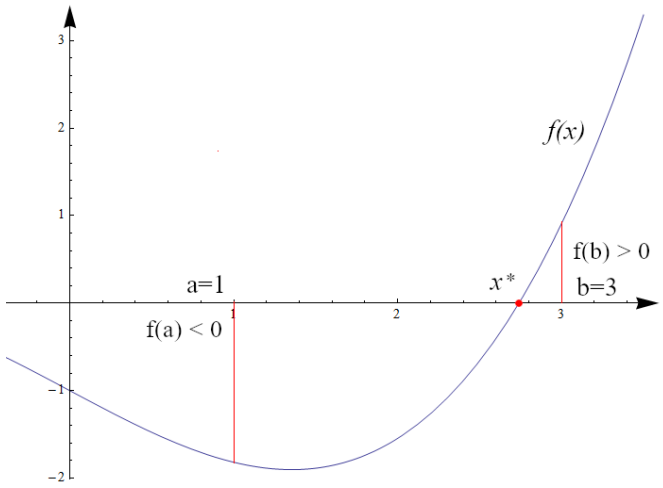
Widzimy, ze wszystkie te problemy możemy opisać jednym schematem: dane jest funkcja jednej zmiennej rzeczywistej $f:\mathbb{R}\supset D\to\mathbb{R},\quad y=f(x),$ z którym kojarzymy równanie $f(x)=0.$ Geometrycznie oznacza to punkt przecięcia wykresu funkcji $f(x)$ z osia Ox.

Metoda bisekcji (połowienia)

Metoda bisekcji opiera się na twierdzeniu zwanym **własnością Darboux**:

> *jeśli funkcja ciągła na przedziale* $[a,b]$ *przyjmuje na końcach tego przedziału wartości o przeciwnych znakach, to funkcja ta posiada miejsce zerowe w przedziale* $(a,b).$

Formalnie może to sformułować tak: jeżeli $f\in C([a,b])$ oraz $f(a)f(b)<0,$ to istnieje liczba $x^*\in(a,b)$ taka, że $f(x^*)=0.$



Rys. 1. Ilustracja własności Darboux: wartości funkcji dla a=1 I b=3 są różnego znaku (*f*(a)<0, *f*(b)>0). Zatem w przedziale [1, 3] funkcja musi mieć miejsce zerowe: *f*(x*)=0.

Opis algorytmu metody bisekcji:

1) *wybierz środek przedziału:* $x = (a+b)/2,$
2) *jeżeli* $f(x) = 0,$ *to* $x^* = x$ *jest szukanym pierwiastkiem i procedurę kończymy,*
3) *jeżeli* $f(x) \neq 0,$ *to wybieramy ten przedział* $[a,\ x]$ *albo* $[x,\ b]$*, w którym funkcja zmienia znak na krańcach,*
4) *jeżeli nie uzyskaliśmy żądanej dokładności, to wracamy do punktu* **1)**.

Poniżej jest przedstawiona przykładowa implementacja metody bisekcji w języku C++. Został on użyty do znajdowania pierwiastka równania $x = \cos x,$ tak więc $f(x) = x - \cos x.$ Wykresy funkcji $y = x$ oraz $y = \cos x$ przecinają się gdzieś na odcinku $[0,\ \pi/2],$ więc uruchamiając metodę bisekcji startujemy od przedziału $a = 0,0;\ b = 1,6.$

Warunek zakończenia iteracji w pętli **`do`** `{...}` **`while`**`()` jest zapisany następująco: $0,5\,|a-b| > eps$ (w kodzie użyto funkcji `fabs()`, która zwraca wartość bezwzględną z liczby). W metodzie bisekcji można łatwo kontrolować dokładność, z którą przybliżamy rozwiązanie: jeżeli pierwiastek znajduje się w przedziale $[a,\ b],$ to znamy go z dokładnością do $0,5\,|a-b|$ (wystarczy za przybliżenie wybrać środek przedziału $(a+b)/2$).

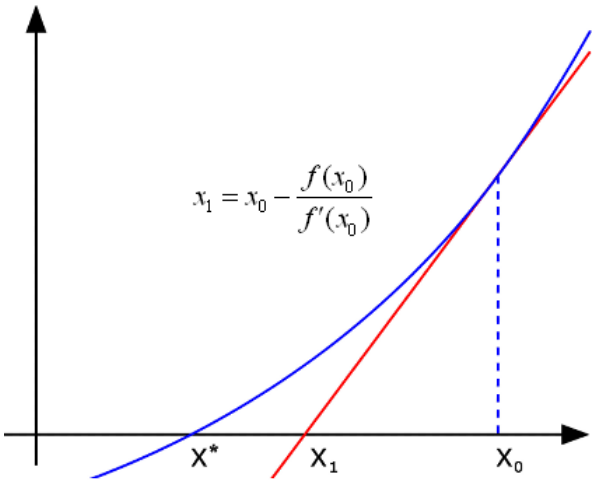
Uruchomienie przedstawionego programu daje wartość $x^* \simeq 0,739085$ jako przybliżoną wartość rozwiązania równania $x = \cos x$ na przedziale $[0,\ 1,6].$

bisekcja.cpp

```
#include <iostream.h>
#include <math.h>

double f(double x) { return x - cos(x);}
int main() {
    double a, b, x, y, ya, yb, eps;

    eps = 1E-8;
    a = 0.0; b = 1.6;
    ya = f(a); yb = f(b);
    do {
        x = (a + b)/2.0;
        cout << "x = " << x << "\n";
        y = f(x);
        if (y == 0) break;
        if (ya*y > 0) { a = x; ya = y;}
        else { b = x; yb = y;}
    } while (0.5*fabs(b-a)> eps);

    x = (a+b)/2.0;
    cout << "x = " << x << "\n";
    cout << "f(" << x << ")=" << f(x) << "\n";
    system("pause");
}
```

Metoda Newtona (metoda stycznych)

Daje ona szybką zbieżność choć pewną słabością jej jest to, że – na ogół – iteracje są zbieżne dopiero, gdy punkt startu będzie wybrany dostatecznie blisko szukanego rozwiązania. Idea metody jest następująca. Wybieramy punkt startu x_0 (możliwie najbliżej poszukiwanego rozwiązania), a następnie prowadzimy styczną do wykresu w tym punkcie. Styczna ta przecina oś OX w jakimś punkcie, który przyjmujemy za kolejne przybliżenie szukanego pierwiastka. Następnie powtarzamy ten krok. Idea ta jest zilustrowana na rysunku poniżej.



Rys. 1. Ilustracja do metody Newtona. Rysujemy styczną odpowiadająca punktowi x_0 i jako kolejne przybliżenie miejsca zerowego x* przyjmujemy punkt przecięcia tej stycznej z osią OX – na rys. jest to x_1.

Uzasadnienie wzoru na x_1 z powyższego rysunku jest następujące. Prosta która przechodzi przez punkt $(x_0, f(x_0))$ ma równanie $y = a(x - x_0) + f(x_0)$. Ale ma to być styczna, więc jej współczynnik kierunkowy a jest równy pochodnej funkcji $y = f(x)$ w punkcie $x = x_0$, czyli $a = f'(x_0)$ co daje $y = f'(x_0) \cdot (x - x_0) + f(x_0)$. Przyrównując do zera otrzymamy $f'(x_0) \cdot (x - x_0) + f(x_0) = 0$. Rozwiązując to równanie względem x otrzymamy podstawowy krok metody stycznych

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

co daje następujący ciąg przybliżeń

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad \text{dla } n = 0,1,2,\ldots \tag{3}$$

Gdy wywołujemy metodę Newtona dla równania $f(x) = 0$, to musimy podać punkt startu x_0. Ponadto należy dostarczyć do procedury także pochodną $f'(x)$.

Poniżej zaprezentowano kod realizujący metodę Newtona dla równań nieliniowych z jedną niewiadomą. W szczególności program został użyty do obliczenia pierwiastka równania trzeciego stopnia $x^3 + 2x^2 + 5x + 1 = 0$. Widać, że do uruchomienia procedury musimy określić lewą stronę, czyli $f(x) = x^3 + 2x^2 + 5x + 1$ oraz jej pochodną $Df(x) = f'(x) = 3x + 4x + 5$. Jako punkt startu wybrano $x_0 = 5,0$. Po dziesięciu iteracjach otrzymujemy przybliżenie rozwiązania $x^* \simeq -0,216757$. Jednak analiza wyników działania programu pokazuje, że wartość tą uzyskano już w iteracji numer 7.

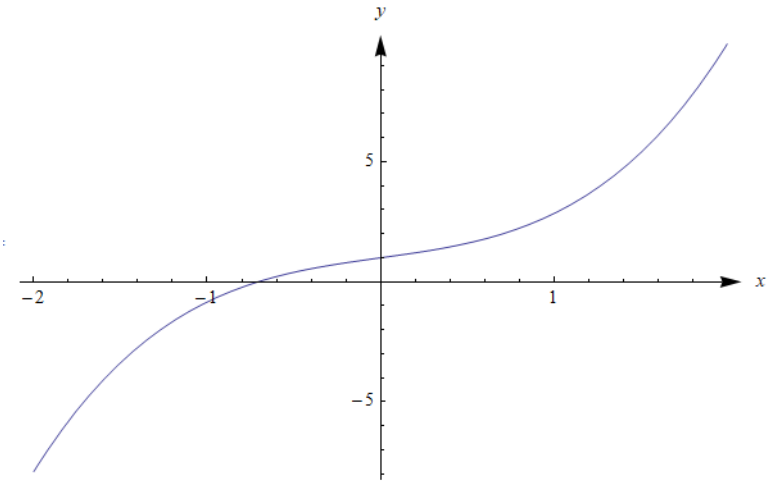
newton1D.cpp

```
#include <iostream.h>
#include <math.h>

double f(double x) { return x*x*x+2*x*x+5*x+1;}
double Df(double x) {return 3*x*x+4*x+5; }

int main() {
    double x;
    int n;

    x = 5.0;
    cout << "Punkt startu: " << x << "\n";
    for (n=1; n <= 10; n++) {
        x = x - f(x)/Df(x);
        cout << "iteracja: " << n << ", x = " << x << "\n";
    }

    system("pause");
}
```

Przykład 1. Szukamy miejsc zerowych równania:

$$\sin x + x^3 + 1 = 0.$$

Analiza wykresu funkcji $f(x) = \sin(x) + x^3 + 1$ (Rys. 2) sugeruje, że szukany pierwiastek znajduje się w przedziale $[-1, 0]$. Dlatego w metodzie bisekcji wybieramy $a = -1$, $b = 0$. Dla metody Newtona wybieramy jako punkt startu $x_0 = 0,0$. Jak widać z wykresu wybór $x_0 = -1,0$ jest też dobry. Iteracje również są zbieżne nawet dla tak odległego punktu startu jak $x_0 = -30$. Jeżeli dokładniej przyjrzymy się wykresowi funkcji, to możemy się przekonać, że w przypadku tego równania metoda Newtona jest zbieżna globalnie, tzn. dla każdego $x_0 \in \mathbb{R}$ ciąg metody Newtona $(x_n)_{n=0}^{\infty}$ jest zbieżny do rozwiązania.



Rys. 2. Wykres funkcji *f*(x)=sin(x)+x^3+1 na przedziale [-2, 2]. Wykres sugeruje, że miejsce zerowe znajduje się w przedziale [-1, 0].

Metoda *regula falsi*[1]

Geometryczna interpretacja jest podobna do metody siecznych. Przybliżamy funkcję ciągłą $f:[a,b]\to\mathbb{R}$ sieczną przechodząca przez punkty $(a, f(a))$ i $(b, f(b))$. Jeżeli funkcja przyjmuje różne znaki na końcach przedziału $[a, b]$, to oczywiście sieczna ta przetnie oś poziomą w jakimś punkcie $c \in (a, b)$ leżącym na odcinku. Punkt ten przyjmujemy jako przybliżenie miejsca zerowego. Dalsze postępowanie jest jednak inne niż w metodzie siecznych. Mianowicie, jeżeli $f(c) \neq 0$, to miejsce zerowe znajduje się albo w $[a, c]$, albo w $[c,b]$. Wybieramy ten przedział, w którym funkcja ma różne znaki na końcach i wykonujemy następny krok iteracji. W ten sposób na każdym etapie obliczeń przedział $[a, b]$ „obejmuje" szukane miejsce zerowe. Jak widać metoda ta jest bardzo zbliżona do metody bisekcji – jedyną różnicą jest inny wybór punktu w przedziale.

Jeżeli prosta ma przechodzić przez punkty $(a, f(a))$ i $(b, f(b))$, to jej równanie ma postać

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

co po przyrównaniu do zera daje c:

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

Opis algorytmu metody *regula falsi* jest następujący:

1) wybierz środek przedziału: $c = \big(af(a) - bf(b)\big) / \big(f(b) - f(a)\big)$,
2) jeżeli $f(c) = 0$, to $x^* = c$ jest szukanym pierwiastkiem i procedurę kończymy,
3) jeżeli $f(c) \neq 0$, to wybieramy ten przedział $[a, c]$ albo $[c, b]$, w którym funkcja zmienia znak na krańcach,
4) jeżeli nie uzyskaliśmy żądanej dokładności, to wracamy do punktu 1).

[1] Określenie *regula falsi* jest pochodzenia łacińskiego. Można ją przetłumaczyć jako fałszywa linia (lub fałszywa reguła).

ZADANIA DO WYKONANIA W RAMACH ĆWICZEŃ

1) Napisać jeden program w języku C++, który będzie implementował w formie funkcji trzy metody rozwiązywania równań nieliniowych z jedną niewiadomą: metodę bisekcji, metodę Newtona, i metodę (*regula falsi* lub siecznych). Dla każdego z podanych niżej przykładów należy zdefiniować prawą stronę w formie funkcji o prototypie `double f(double)` i nazwach wg schematu „*f_zad1_a*" (dla równania z Zadania 1 podpunkt a)). W funkcji głównej stworzyć sekcję dla każdego przykładu z zadań, które są poniżej, gdzie będzie komentarz np. /* Zadanie 1, podpunkt a … */, odpowiednie inicjalizacja zmiennych, kod wypisujący informację o uruchamianej funkcji i wyniki iteracji oraz wyraźnie wynik końcowy.

2) Raport (pdf) – maksymalnie 2 strony zawiera krótkie wprowadzenie do zagadnienia rozwiązywanie równań nieliniowych, opis dwóch wybranych metod, oraz przykład dwóch własnych równań, z wykresami obejmującymi miejsca zerowe. Równania te mają być rozwiązane dwoma wybranymi metodami. Dodać tabelkę, która będzie prezentowała dziesięć iteracji prowadzące do miejsca zerowego.

iteracja	Równanie1		Równanie2	
	Metoda 1	Metoda 2	Metoda 1	Metoda 2
1				
…				
10				

Zadanie 1. Stosując metodę bisekcji, znaleźć numerycznie rozwiązania poniższych równań w zbiorze liczb rzeczywistych $\mathbb{R}$.

a) $x^3 + 2x - 1 = 0.$
b) $x - 3\sin x - 1 = 0.$
c) $x \ln x + \sin x = 0.$
d) $e^{-3x} + x^5 - 3x^2 + 1 = 0.$

Zadanie 2. Stosując metodę Newtona, znaleźć numerycznie rozwiązania poniższych równań w zbiorze liczb rzeczywistych $\mathbb{R}$.

a) $x^5 + 3x^3 + 2 = 0.$
b) $e^x = x^2 + 2.$
c) (c1) $\operatorname{arctg} x = \dfrac{x}{x^2+1}$, (c2) $\operatorname{arctg} x = \dfrac{x}{x^2+1} - 1.$
d) $\sin(\sin(x)) = 0$ w przedziale $[1, 5]$.

Zadanie 3. Napisać program, który w oparciu o metodę Newtona będzie wyznaczał przybliżenie $n-$tego pierwiastka rzeczywistego $\sqrt[n]{c}$ z danej liczby $c \in \mathbb{R}$. Należy tak zaadaptować schemat Newtona, aby nie było już konieczne obliczanie odpowiedniej pochodnej. Argumentami wejściowymi do procedury mają być: $c =$ liczba rzeczywista, której pierwiastek będzie obliczany, $n =$ stopień pierwiastka, $iterNum =$ liczba iteracji.

Zadanie 4. W chemii ważny zagadnieniem jest ustalenie składu równowagowego w reakcji chemicznej. Na przykład w roztworach wodnych słabych kwasów, takich jak kwas octowy CH_3COOH, interesuje chemika jakie jest stężenie kationów wodorowych $H^+(aq)$, gdy rozpuścimy daną ilość kwasu w określonej objętości wody. Inaczej mówiąc dane jest analityczne stężenie słabego kwasu, c_0, a szukane

jest stężenie $x = [\mathrm{H}^+(aq)]$. Z teorii równowag chemicznych wynika, że równanie na x dla kwasów jednoprotonowych jest następujące:[2]

$$x^3 + K_a x^2 - (K_w + c_0 K_a)x - K_a K_w = 0, \tag{4}$$

gdzie $K_w = 10^{-14}$, K_a to tzw. stała kwasowa, która ilościowo opisuje jego „siłę", a $c_0 > 0$ jest stężeniem kwasu. Interesują nas tylko rozwiązania $x \in \mathbb{R}_+$ (rzeczywiste dodatnie) równania (4). W zadaniu chodzi o to, aby rozwiązać równanie dla kwasu octowego ($K_a = 1{,}754 \cdot 10^{-5}$), ale nie dla jednego stężenia, tylko dla zakresu $10^{-9} \le c_0 \le 1$. Ponieważ jest to zakres liczb obejmujący 9 rzędów wielkości, więc przyjmiemy, że na każdą dekadę będzie przykładowo sześć stężeń. Zatem w tym przypadku $c_0 \in \{q^k \cdot 10^{-9} : \ k = 0,1,2,\ldots,60\}$ gdzie $q = \sqrt[6]{10} \approx 1{,}468$. To oznacza ogólny schemat z pętlą zewnętrzną:

```
c0 = 1.0E-9; q = 1.468;
while (c0 <= 1) {
     szukaj x;
     zapisz od pliku (log(c0), -log(x));
     c0 = c0*q;
}
```

Dla każdego c_0 z tego zbioru znaleźć rozwiązanie $x(c_0)$, a następnie sporządzić wykres $-\log x(c_0) = f(\log c_0)$. To znaczy na osi poziomej zaznaczamy $\log c_0$, a na osi pionowej $-\log x(c_0)$, gdzie $x(c_0)$ jest rozwiązaniem numerycznym równania dla parametru c_0.

[2] W szkolnej chemii zagadnienie to jest rozważane w wersji uproszczonej – nie uwzględnia się w dysocjacji samej wody, co prowadzi do równanie kwadratowego $x^2 + K_a x - c_0 K_a = 0$. Dla stężeń c_0 bardzo małych to przybliżenie jest jednak nieporwane, i należy rozwiązywać dokładniejsze równanie (4).