Politechnika Świętokrzyska w Kielcach

Wydział Zarządzania i Modelowania Komputerowego

Algorytmy i struktury danych
Laboratorium
Algorytmy numeryczne

Politechnika Świętokrzyska
Kielce University of Technology

Przygotował: Radosław Kulig

Numer albumu: 093795

Kierunek: Inżynieria Danych

Studia: stacjonarne

Numer grupy: L02

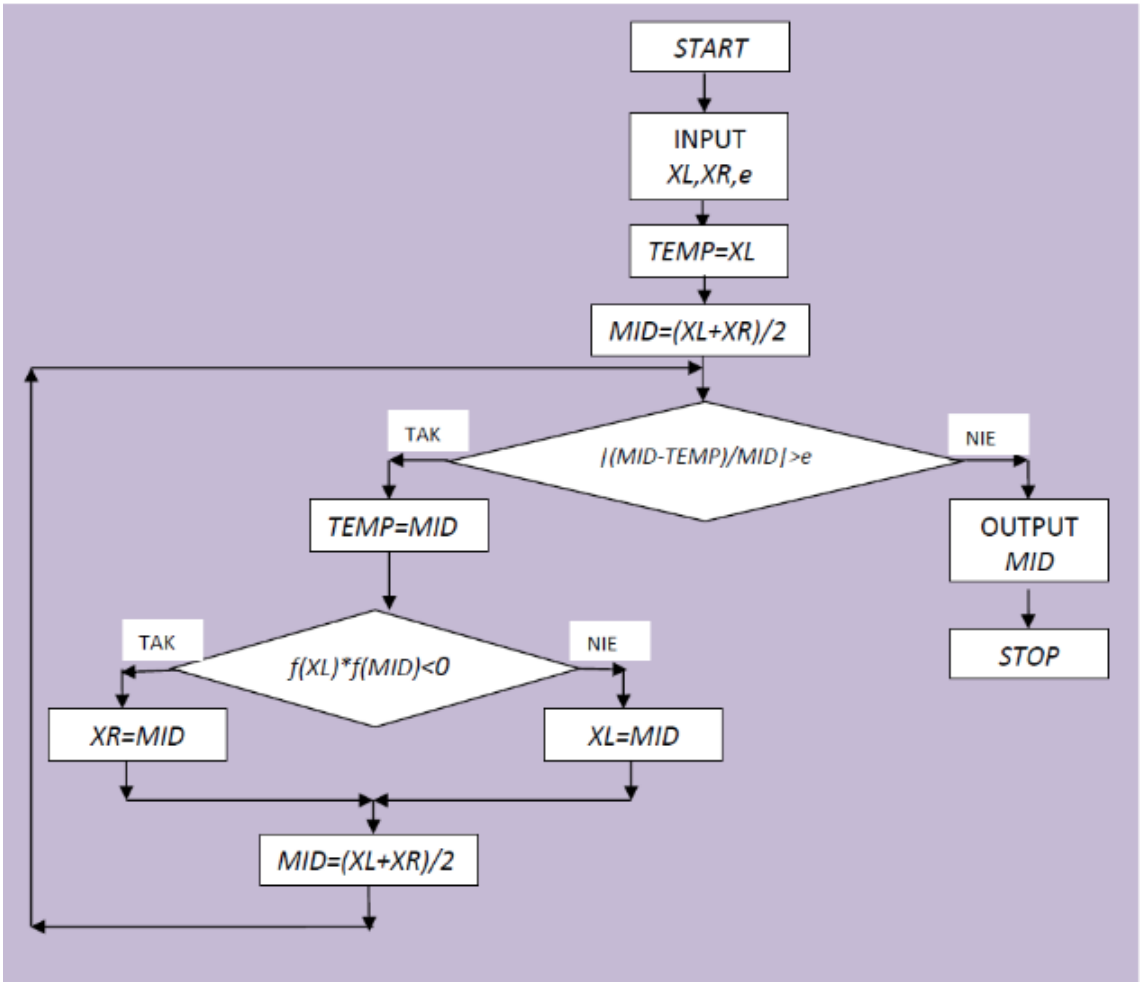
Wyznaczanie miejsca zerowego funkcji metodą bisekcji

Metoda bisekcji (ang. bisection method), zwana również metodą połowienia lub wyszukiwaniem binarnym pozwala stosunkowo szybko znaleźć pierwiastek dowolnej funkcji w zadanym przedziale poszukiwań `[a,b]`. Aby można było zastosować metodę bisekcji, funkcja musi spełniać kilka warunków początkowych:

- Funkcja musi być określona w przedziale `[a,b]`
- Funkcja musi być ciągła w przedziale `[a,b]`
- Na krańcach przedziału `[a,b]` funkcja musi mieć różne znaki

Rozwiązanie znajdowane jest za pomocą kolejnych przybliżeń. Z tego powodu należy określić dokładność, z którą chcemy otrzymać pierwiastek funkcji oraz dokładność wyznaczania samej funkcji. W każdym przybliżeniu algorytm wyznacza środek `MID` przedziału `[XL,XR]` jako średnią arytmetyczną krańców. Następnie sprawdzane jest, czy różnica pomiędzy środkami z kolejnych iteracji jest mniejsza od założonej dokładności wyliczania pierwiastka. Jeśli tak, to algorytm kończy pracę z wynikiem w MID.

W przeciwnym razie punkt `MID` dzieli przedział `[XL,XR]` na dwie równe połowy: `[XL,MID]` i `[MID,XR]`. Algorytm za nowy przedział `[XL,XR]` przyjmuję tę połówkę, w której funkcja zmienia znak na krańcach i kontynuuje wyznaczanie pierwiastka funkcji.



Implementacja algorytmu bisekcji

```
double vel(float c) {
    float g = 9.81, m = 68.1, t = 10;
    double result;
    result = g * m / c * (1 - exp(-(c / m) * t)) - 40;
    return result;
}
void bisekcja() {
    double XL = 12, XR = 16, e = 0.1;
    double TEMP = XL;
    double MID = (XL + XR) / 2;
    double epsilon = 1;
    cout << " | " << "XL" << " | " << "XR" << " | " << "MID" << " | " << "EPS" << " |
" << "\n";
    cout << " | " << XL << " | " << XR << " | " << MID << " | " << "---" << " | " <<
"\n";
    while (epsilon > e) {

        TEMP = MID;
```

```
        if (vel(XL) * vel(MID) < 0) XR = MID; else XL = MID;
        MID = (XL + XR) / 2;
        epsilon = abs((MID - TEMP) / MID) * 100;
        cout << " | " << XL << " | " << XR << " | " << MID << " | " << epsilon << " |
" << "\n";
    }
    cout << "Wynik algorytmy metody bisekcji: " << MID << "\n\n";
}
```

Wyznaczanie miejsca zerowego funkcji metodą Newtona-Raphsona

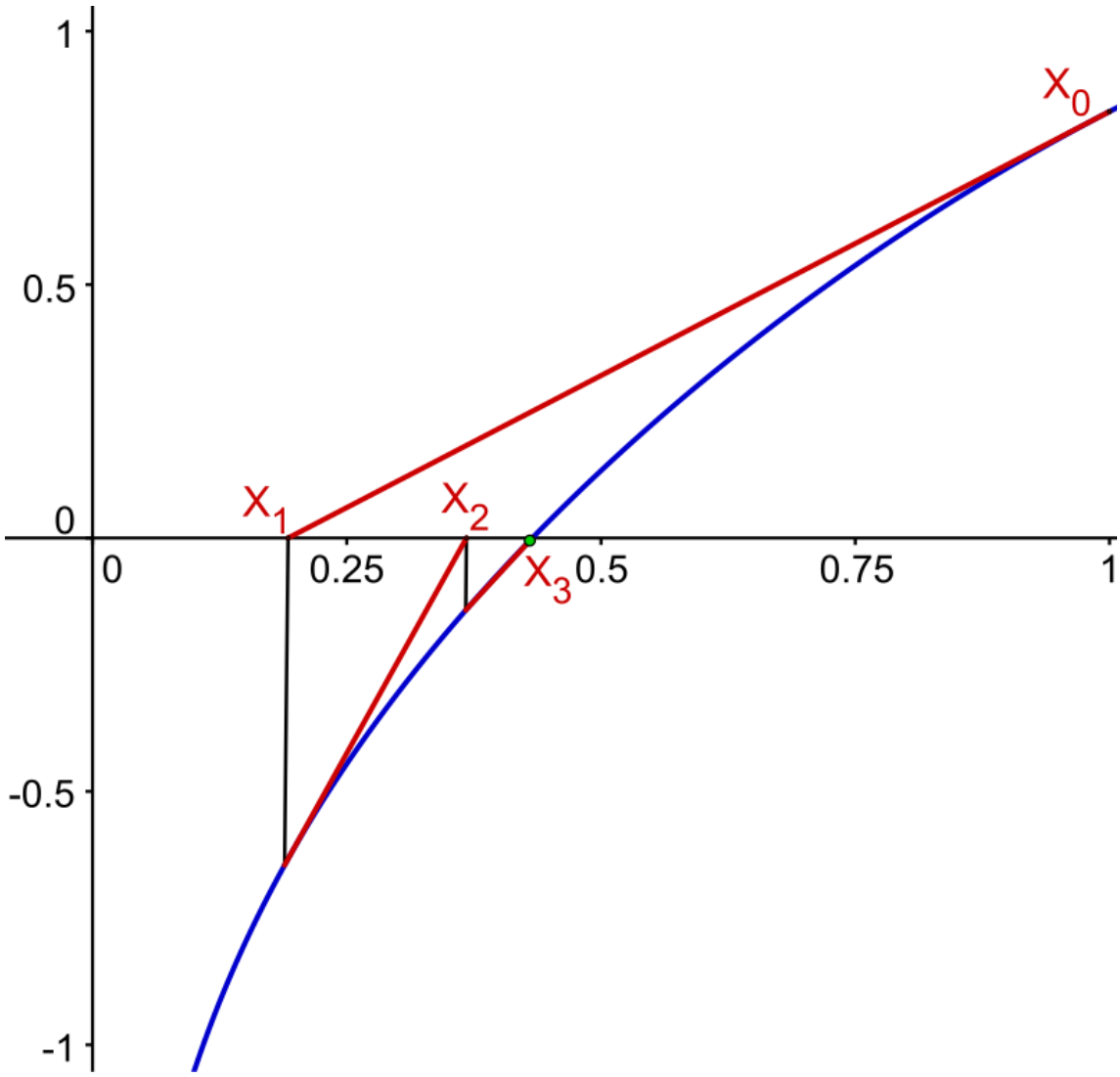
Metoda Newtona-Raphsona, nazywana również metodą stycznych, jest jedną z najczęściej stosowanych metod numerycznych do wyznaczania miejsc zerowych funkcji. Polega ona na iteracyjnym przybliżaniu pierwiastka poprzez wykorzystanie wartości funkcji oraz jej pochodnej w danym punkcie. W każdym kroku obliczane jest nowe przybliżenie zgodnie ze wzorem:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, k = 1, 2, \ldots.$$

Proces ten powtarza się aż do osiągnięcia zadanej dokładności rozwiązania. Metoda Newtona-Raphsona charakteryzuje się zbieżnością kwadratową w pobliżu pierwiastka, co oznacza, że liczba poprawnych cyfr przybliżenia rośnie w przybliżeniu dwukrotnie z każdą iteracją. Warunkiem jej skuteczności jest jednak istnienie i niezerowość pochodnej `f'(x)` w otoczeniu pierwiastka oraz odpowiedni wybór punktu startowego `x0`, ponieważ nieodpowiednie wartości początkowe mogą prowadzić do rozbieżności iteracji.

Założenia:

- W przedziale `[a,b]` znajduje się dokładnie jeden pierwiastek funkcji `f`.
- Funkcja ma różne znaki na krańcach przedziału, tj. `f(a)  f( b) < 0`.
- Pierwsza i druga pochodna funkcji mają stały znak w tym przedziale.



Implementacja algorytmu metody Newtona-Raphsona

```
double vel(double c) {
    double g = 9.81, m = 68.1, t = 10;
    double result;
    result = g * m / c * (1 - exp(-(c / m) * t)) - 40;
    return result;
}
double vel_der(double x) {
    double result = exp(-0.146843 * x) * (4.16764 * pow(10, -16) * x - 668.061 *
exp(0.146843 * x) + 2.83816 * pow(10, -15)) / pow(x, 2);
    return result;
}
void newtonRaphson() {
    double e = 0.01;
    double x = 10;
    double temp;
```

```
    double der = vel_der(x);
    cout << " | " << "x" << " | " << "|vel(x)|" << " | " << "\n";
    cout << " | " << x << " | " << abs(vel(x)) << " | " << "\n";
    do {
        temp = x - vel(x) / der;
        x = temp;
        der = vel_der(x);
        cout << " | " << x << " | " << abs(vel(x)) << " | " << "\n";
    } while (abs(vel(x)) > e);
    cout << "Wynik algorytmu metody Newtona-Raphsona: " << x << "\n";
}
```

Wyniki

Algorytm bisekcji

XL	XR	MID	EPS
12	16	14	---
14	16	15	6.66667
14	15	14.5	3.44828
14.5	15	14.75	1.69492
14.75	15	14.875	0.840336
14.75	14.875	14.8125	0.421941
14.75	14.8125	14.7812	0.211416
14.7812	14.8125	14.7969	0.105597
14.7969	14.8125	14.8047	0.0527704

Wynik algorytmy metody bisekcji: `14.8047`.

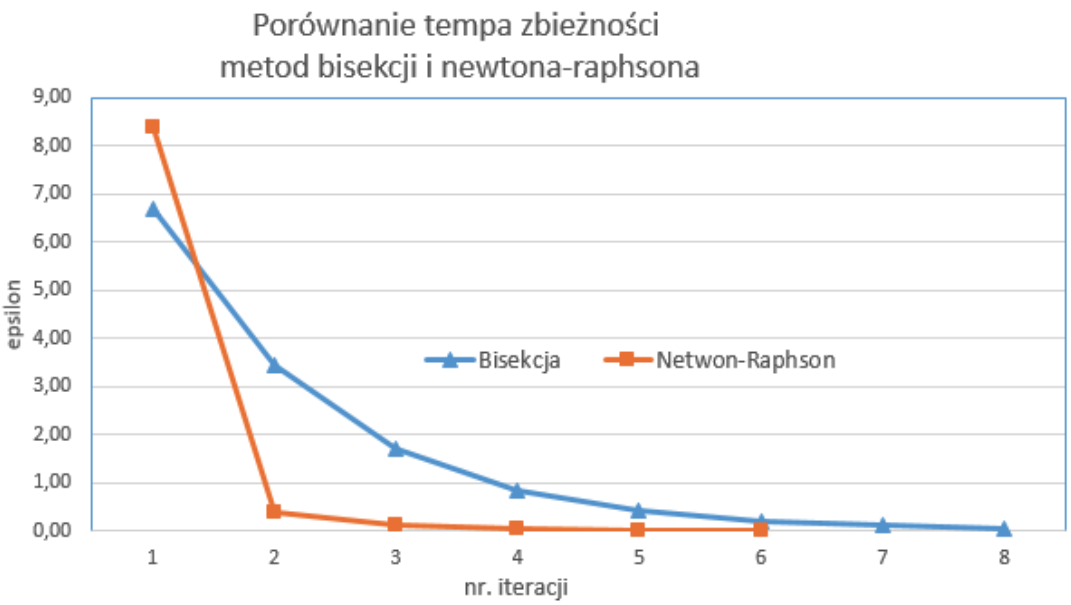
Algorytm Newtona-Raphsona

x	|vel(x)|
10	11.4215
11.7097	6.83073
13.1116	3.52177
14.0179	1.57401
14.4809	0.631897
14.6792	0.23869
14.7562	0.0877069
14.7848	0.0318761
14.7952	0.0115374

14.799	0.00416963

Wynik algorytmu metody Newtona-Raphsona: `14.799`.

Wykres



Wielkości wartości ε na wykresie powyżej są zależne od zadanych parametrów - przedziału `[XL,XL]` w metodzie bisekcji i wartości `x` w metodzie Newtona-Raphsona. Z punktu widzenia naszego ćwiczenia istotny jest jedynie kształt linii na wykresie, który odzwierciedla tempa zbieżności porównywanych metod.

Wnioski

W przeprowadzonych ćwiczeniach zaimplementowano i porównano dwie metody numeryczne służące do wyznaczania miejsc zerowych funkcji: metodę bisekcji oraz metodę Newtona-Raphsona. Obie metody pozwalają skutecznie znaleźć przybliżone rozwiązanie równania nieliniowego, jednak różnią się zasadą działania, szybkością zbieżności oraz wymaganiami wobec funkcji.

Metoda bisekcji charakteryzuje się dużą stabilnością i gwarancją zbieżności, pod warunkiem spełnienia założeń dotyczących ciągłości funkcji oraz zmiany znaku na krańcach przedziału. Jej główną wadą jest jednak wolne, liniowe tempo zbieżności, co powoduje konieczność wykonania większej liczby iteracji w celu osiągnięcia wymaganej dokładności.

Z kolei metoda Newtona-Raphsona, wykorzystująca wartości funkcji oraz jej pochodnej, cechuje się znacznie szybszym — kwadratowym — tempem zbieżności. Wymaga jednak odpowiedniego doboru punktu początkowego oraz znajomości pochodnej, gdyż w przeciwnym razie może nie zapewniać zbieżności rozwiązania.

Analiza wykresu potwierdza teoretyczne właściwości obu metod. Linie przedstawiające przebieg zbieżności ukazują, że w przypadku metody bisekcji tempo zbieżności jest liniowe, natomiast dla metody Newtona-Raphsona — kwadratowe, co przekłada się na znacznie szybsze osiąganie dokładnego rozwiązania. Podsumowując, metoda bisekcji

gwarantuje stabilność kosztem szybkości, natomiast metoda Newtona-Raphsona zapewnia większą efektywność obliczeniową przy spełnieniu bardziej restrykcyjnych warunków. Wybór odpowiedniej metody powinien zatem zależeć od charakteru analizowanej funkcji oraz oczekiwanej dokładności obliczeń.