Programowanie i metody numeryczne

Zadania – seria 7.

Równania nieliniowe.

Zadanie 1. `bisection` – Metoda bisekcji.

Napisz szablon funkcji

```
bool RootBisection(const auto &f, double &x, double a, double b, double eps)
```

która posługując się metodą bisekcji znajduje miejsce zerowe ciągłej funkcji $f : \mathbb{R} \supseteq X \to \mathbb{R}$ położone w przedziale $[a, b] \subset X$, przy czym $f(a)f(b) < 0$, z dokładnością ε. Funkcja `RootBisection` przyjmuje jako argumenty: referencję `x` zmiennej typu `double`, `f` – implementację funkcji f oraz trzy wartości `a`, `b` i `eps` typu `double`, odpowiadające kolejno liczbom a, b i ε. Gdy uda się znaleźć miejsce zerowe z żądaną dokładnością, funkcja `RootBisection` powinna umieścić je w zmiennej `x` oraz zwrócić wartość `true`, w przeciwnym razie ma zwrócić wartość `false`. Załóż, że w przedziale $[a, b]$ znajduje się co najwyżej jedno miejsce zerowe funkcji f.

Funkcja `RootBisection` powinna sprawdzać, czy wartości jej argumentów są poprawne oraz czy spełniają założenia (a więc np. czy odpowiadają $f(a)f(b) < 0$, $a < b$, $0 < \varepsilon \leqslant b - a$). Jeśli okaże się, że tak nie jest, funkcja powinna zgłosić odpowiedni wyjątek (pochodzący z biblioteki standardowej lub napisany specjalnie na jej potrzeby) opatrzony komunikatem wyjaśniającym przyczynę jego wystąpienia.

Napisz program testowy sprawdzający poprawność działania Twojego szablonu dla trzech przykładowych funkcji: $f(x) = x^2$, $g(x) = x^2 - 2$ oraz $h(x) = \mathrm{e}^x + x - 1$, poszukujący dla każdej z nich miejsca zerowego w przedziale $[-1, 6]$ z kilkoma przykładowymi dokładnościami: 0.1, 0.01, 0.001, 0.0001, 0.0000001.

Następnie, korzystając z tego szablonu, napisz program `bisection`, który przyjmuje jako argumenty wywołania trzy liczby zmiennoprzecinkowe określające wartości a, b i ε. Program powinien wczytywać ze standardowego wejścia liczby zmiennoprzecinkowe aż do napotkania znaku końca pliku, następnie konstruować wielomian z tymi liczbami jako współczynnikami i znajdować miejsce zerowe tego wielomianu w zadanym przedziale i z zadaną dokładnością. Możesz założyć, że w zadanym przedziale znajduje się co najwyżej jedno miejsce zerowe tego wielomianu.

Zadanie 2. `newton` – Metoda Newtona.

Napisz szablon funkcji

```
bool RootNewton(const auto &f, double &x, double x0, double eps, int N = 1000)
```

która posługując się metodą Newtona znajduje miejsce zerowe ciągłej funkcji $f : \mathbb{R} \supseteq X \to \mathbb{R}$ położone najbliżej punktu $x_0 \in X$ z dokładnością ε. Funkcja `RootNewton` przyjmuje jako argumenty referencję `x` zmiennej typu `double`, `f` – implementację funkcji f, dwie wartości `x0` i `eps` typu `double`, odpowiadające kolejno liczbom x_0 i ε oraz liczę całkowitą `N` określającą limit ilości iteracji z domyślną wartością 1000. Procedura znajdowania miejsca zerowego powinna trwać aż do osiągnięcia dokładności ε albo limitu liczby iteracji. Gdy limit ten zostanie osiągnięty przed znalezieniem miejsca zerowego z żądaną dokładnością, funkcja `RootNewton` powinna zwrócić wartość `false`, w przeciwnym razie – wartość `true`. W obu przypadkach znalezione najlepsze przybliżenie miejsca zerowego powinno zostać umieszczone w zmiennej `x`.

Napisz program testowy sprawdzający poprawność działania Twojego szablonu dla trzech przykładowych

funkcji: $f(x) = x^2$, $g(x) = x^2 - 2$ oraz $h(x) = \mathrm{e}^x + x - 1$, poszukujący dla każdej z nich miejsca zerowego w pobliżu punktu $x_0 = 1,4$ z różnymi przykładowymi dokładnościami: 0.1, 0.01, 0.001, 0.0001, 0.0000001.

Następnie, korzystając z tego szablonu, napisz program `newton`, który przyjmuje jako argumenty wywołania dwie liczby zmiennoprzecinkowe określające wartości x_0 i ε oraz liczbę całkowitą określającą limit ilości iteracji. Program powinien wczytywać ze standardowego wejścia liczby zmiennoprzecinkowe aż do napotkania znaku końca pliku, następnie konstruować wielomian z tymi liczbami jako współczynnikami i znajdować miejsce zerowe tego wielomianu w pobliżu zadanego punktu i z zadaną dokładnością. Program powinien również informować użytkownika, czy limit ilości iteracji został osiągnięty przed znalezieniem miejsca zerowego.

Zadanie 3. `steffensen` – Metoda Steffensena.

Napisz szablon funkcji

```
bool RootSteffensen(const auto &f, double &x, double x0, double eps,
                    int N = 1000)
```

która posługując się metodą Steffensena znajduje miejsce zerowe ciągłej funkcji $f : \mathbb{R} \supseteq X \to \mathbb{R}$ położone najbliżej punktu $x_0 \in X$ z dokładnością ε. Funkcja `RootSteffensen` przyjmuje jako argumenty referencję `x` zmiennej typu `double`, `f` – implementację funkcji f, dwie wartości `x0` i `eps` typu `double`, odpowiadające kolejno liczbom x_0 i ε oraz liczę całkowitą `N` określającą limit ilości iteracji z domyślną wartością 1000. Procedura znajdowania miejsca zerowego powinna trwać aż do osiągnięcia dokładności ε albo limitu liczby iteracji. Gdy limit ten zostanie osiągnięty przed znalezieniem miejsca zerowego z żądaną dokładnością, funkcja `RootSteffensen` powinna zwrócić wartość `false`, w przeciwnym razie – wartość `true`. W obu przypadkach znalezione najlepsze przybliżenie miejsca zerowego powinno zostać umieszczone w zmiennej `x`.

Napisz program testowy sprawdzający poprawność działania Twojego szablonu dla trzech przykładowych funkcji: $f(x) = x^2$, $g(x) = x^2 - 2$ oraz $h(x) = \mathrm{e}^x + x - 1$, poszukujący dla każdej z nich miejsca zerowego w pobliżu punktu $x_0 = 1,4$ z różnymi przykładowymi dokładnościami: 0.1, 0.01, 0.001, 0.0001, 0.0000001.

Następnie, korzystając z tego szablonu, napisz program `steffensen`, który przyjmuje jako argumenty wywołania dwie liczby zmiennoprzecinkowe określające wartości x_0 i ε oraz liczbę całkowitą określającą limit ilości iteracji. Program powinien wczytywać ze standardowego wejścia liczby zmiennoprzecinkowe aż do napotkania znaku końca pliku, następnie konstruować wielomian z tymi liczbami jako współczynnikami i znajdować miejsce zerowe tego wielomianu w pobliżu zadanego punktu i z zadaną dokładnością. Program powinien również informować użytkownika, czy limit ilości iteracji został osiągnięty przed znalezieniem miejsca zerowego.

Zadanie 4. `secant` – Metoda siecznych.

Napisz szablon funkcji

```
bool RootSecant(const auto &f, double &x, double x1, double x2, double eps,
                int N = 1000)
```

która posługując się metodą siecznych znajduje miejsce zerowe ciągłej funkcji $f : \mathbb{R} \supseteq X \to \mathbb{R}$ dla punktów początkowych $x_1, x_2 \in X$ z dokładnością ε. Funkcja `RootSecant` przyjmuje jako argumenty referencję `x` zmiennej typu `double`, `f` – implementację funkcji f, trzy wartości `x1`, `x2` i `eps` typu `double`, odpowiadające kolejno liczbom x_1, x_2 i ε oraz liczę całkowitą `N` określającą limit ilości iteracji z domyślną wartością 1000. Procedura znajdowania miejsca zerowego powinna trwać aż do osiągnięcia dokładności ε albo limitu liczby iteracji. Gdy limit ten zostanie osiągnięty przed znalezieniem miejsca zerowego z żądaną dokładnością, funkcja `RootSecant` powinna zwrócić wartość `false`, w przeciwnym razie – wartość `true`. W obu przypadkach znalezione najlepsze przybliżenie miejsca zerowego powinno zostać umieszczone w zmiennej `x`.

Napisz program testowy sprawdzający poprawność działania tego szablonu dla trzech przykładowych funkcji: $f(x) = x^2$, $g(x) = x^2 - 2$ oraz $h(x) = \mathrm{e}^x + x - 1$, poszukujący dla każdej z nich miejsca zerowego z punktami

początkowymi $x_1 = 1,4$ i $x_2 = -2$ z różnymi przykładowymi dokładnościami: 0.1, 0.01, 0.001, 0.0001, 0.0000001. Czy znalezione miejsce zerowe zawsze leży pomiędzy punktami x_1 i x_2?

Następnie, korzystając z tego szablonu, napisz program `secant`, który przyjmuje jako argumenty wywołania trzy liczby zmiennoprzecinkowe określające wartości x_1, x_2 i ε oraz liczbę całkowitą określającą limit ilości iteracji. Program powinien wczytywać ze standardowego wejścia liczby zmiennoprzecinkowe aż do napotkania znaku końca pliku, następnie konstruować wielomian z tymi liczbami jako współczynnikami i znajdować miejsce zerowe tego wielomianu z zadanymi punktami początkowymi i z zadaną dokładnością. Program powinien również informować użytkownika, czy limit ilości iteracji został osiągnięty przed znalezieniem miejsca zerowego.

Opracowanie: Bartłomiej Zglinicki.