Programowanie i metody numeryczne

Zadania – seria 8.

Różniczkowanie. Ekstrema.

Zadanie 1. `diff` – Różniczkowanie numeryczne.

Napisz zestaw szablonów funkcji:

- `double DiffForward(const auto &f, double x0, double h)`
- `double DiffBackward(const auto &f, double x0, double h)`
- `double DiffCentral(const auto &f, double x0, double h)`
- `double DiffRichardson(const auto &f, double x0, double h)`

których zadaniem jest obliczenie przybliżonej wartości pochodnej różniczkowalnej funkcji $y = f(x)$ w punkcie x_0. Funkcje te powinny posługiwać się odpowiednio następującymi wzorami:

$$
\begin{aligned}
f'_{\text{forward}}(x) &= \frac{f(x+h) - f(x)}{h}, \\
f'_{\text{backward}}(x) &= \frac{f(x) - f(x-h)}{h}, \\
f'_{\text{central}}(x) &= \frac{f(x+h) - f(x-h)}{2h}, \\
f'_{\text{richardson}}(x) &= \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h}.
\end{aligned}
$$

Każda z funkcji przyjmuje trzy argumenty: `f` – implementację funkcji f oraz dwie liczby zmiennoprzecinkowe `x0` i `h`, odpowiadające kolejno liczbom x_0 i h. Wartością zwracaną przez każdą z funkcji powinna być szukana przybliżona wartość pochodnej $f'(x_0)$.

Następnie napisz program `diff` testujący działanie Twoich szablonów. Program powinien przyjmować jako argumenty wywołania jedno ze słów: `forward`, `backward`, `central` lub `richardson`, określające metodę różniczkowania, oraz dwie liczby rzeczywiste określające kolejno wartości x_0 i h. Po uruchomieniu program ma wczytywać ze standardowego wejścia liczby rzeczywiste aż do napotkania znaku końca pliku, następnie konstruować wielomian z tymi liczbami jako współczynnikami i obliczać pochodną tego wielomianu posługując się wskazaną metodą.

Zadanie 2. `diferr` – Błąd w różniczkowaniu numerycznym.

Napisz program `diferr` analizujący błąd popełniany przy różniczkowaniu numerycznym metodą centralną w zależności od wartości h.

Program powinien obliczyć wartość pochodnej funkcji

$$f(x) = 0,5x^4 - 2x^3 - 0,25x^2 - 1,5x + 1,2$$

w punkcie $x_0 = 1,1$ dla trzydziestu różnych wartości h, danych wzorem

$$h_k = 0,1 \times 3^{-k},$$

a następnie dla każdej z tych wartości h wyznaczyć błąd przybliżenia, zdefiniowany jako moduł różnicy przybliżonej wartości pochodnej i wartości dokładnej $f'(1,1) = -6.648$. Następnie program powinien utworzyć plik tekstowy `diferr.txt` i zapisać w nim dane niezbędne do wykonania wykresu błędu przybliżenia w funkcji h.

Wykorzystując zewnętrzne narzędzie (np. skrypt napisany w języku Python) lub odpowiednią bibliotekę graficzną (np. ImPlot) sporządź wykres w skali logarytmiczno-logarytmicznej na podstawie danych z pliku `diferr.txt`.

Zadanie 3. `diffp` – Różniczkowanie numeryczne zbioru danych.

Napisz zestaw funkcji

- ```
  std::vector<double> DiffForwardP(const std::vector<double> &x, const
      std::vector<double> &y)
  ```

- ```
  std::vector<double> DiffCentralP(const std::vector<double> &x, const
      std::vector<double> &y)
  ```

których zadaniem jest obliczenie przybliżonej wartości pochodnej pewnej nieznanej funkcji $y = f(x)$, dla której znane są wartości $\big(y_i = f(x_i)\big)$ w zadanych punktach (x_i). Funkcje te powinny się posługiwać odpowiednio następującymi wzorami:

$$f'_{\text{forward}}(x_i) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i},$$
$$f'_{\text{central}}(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{x_{i+1} - x_{i-1}}.$$

Każda z funkcji przyjmuje jako argumenty stałe referencje wektorów `x` i `y` zawierających, odpowiednio, wartości (x_i) i (y_i) oraz zwraca wektor zawierający wartości $\big(y'_i = f'(x_i)\big)$ pochodnej funkcji $y = f(x)$ w punktach (x_i).

Następnie napisz program `diffp` testujący działanie Twoich funkcji. Program powinien przyjmować jako argument wywołania jedno ze słów: `forward` lub `central`, określające metodę różniczkowania. Po uruchomieniu program ma dwukrotnie wczytywać ze standardowego wejścia liczby rzeczywiste aż do napotkania znaku końca pliku, interpretując je, odpowiednio, jako wartości (x_i) i (y_i), a następnie obliczać pochodną zadaną metodą i wypisywać na standardowe wyjście w czytelnej formie wartości (x_i) i (y'_i).

Zadanie 4. `exnewton` – Znajdowanie ekstremów funkcji metodą Newtona.

Napisz szablon funkcji

```
bool ExNewton(double &x, const auto& f, double x0, double eps = 0.01, int N =
    1000)
```

która ma za zadanie znalezienie ekstremum ciągłej funkcji $f : \mathbb{R} \supseteq X \to \mathbb{R}$ położonego najbliżej punktu $x_0 \in X$ z dokładnością ε. Metoda znajdowania ekstremum składa się z dwóch kroków: w pierwszym należy numerycznie wyznaczyć pochodną funkcji, w drugim zaś zastosować metodę Newtona rozwiązywania równań do równania $f'(x) = 0$. Funkcja `ExNewton` 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. Procedura znajdowania ekstremum powinna trwać aż do

osiągnięcia dokładności ε albo limitu liczby iteracji. Gdy limit ten zostanie osiągnięty przed znalezieniem ekstremum z żądaną dokładnością, funkcja `ExNewton` powinna zwrócić wartość `false`, w przeciwnym razie – wartość `true`. W obu przypadkach znalezione najlepsze przybliżenie ekstremum powinno zostać umieszczone w zmiennej `x`.

Napisz program testowy sprawdzający poprawność działania tego szablonu dla dwóch przykładowych funkcji: $f(x) = (x-3)^2$ oraz $g(x) = \sin x$, poszukujący dla każdej z nich ekstremum w pobliżu punktu $x_0 = 1$ z różnymi przykładowymi dokładnościami: 0.1, 0.01, 0.001.

Następnie, korzystając z tego szablonu, napisz program `exnewton` szukający ekstremów dwóch przykładowych funkcji: $f(x) = (x-3)^2$ oraz $g(x) = \sin x$ w pobliżu punktu $x_0 = 1$ z różnymi przykładowymi dokładnościami: 0.1, 0.01, 0.001. Program powinien również informować użytkownika, czy limit ilości iteracji został osiągnięty przed znalezieniem ekstremum.

Opracowanie: Bartłomiej Zglinicki.