Rozwiązywanie równania nieliniowego metodą Newtona-Raphsona drugiego rzędu

1. *Zastosowanie*

Funkcja Newtona-Raphsona znajduję przybliżoną wartość pierwiastka równania nieliniowego z wykorzystaniem pierwszej i drugiej pochodnej w arytmetyce zmiennopozycyjnej i zmiennopozycyjnej arytmetyce przedziałowej.

2. *Opis metody*

Pierwiastek równania wyznaczany jest za pomocą iteracyjnego procesu Newtona-Raphsona drugiego rzędu w postaci:

$$x_{i+1} = x_i - \frac{f'(x_i) \pm \sqrt{[f'(x_i)]^2 - 2f(x_i)f''(x_i)}}{f''(x_i)} \qquad i = 0,1,2,\ldots,$$

W, którym wartość x_0 jest wprowadzona przez użytkownika. Proces zakończy się po osiągnięciu warunku

$$\frac{|x_{i+1} - x_i|}{\max(|x_{i+1}|, |x_i|)} < \varepsilon, \quad x_{i+1} \neq 0, \ \ x_i \neq 0,$$

W, którym wartość ε jest dokładnością wprowadzaną przez użytkownika lub, gdy $x_{i+1} = x_i = 0.$

3. *Wywołanie funkcji*

NewtonRaphson (*x, f, df, d2f, mit, eps, it*)

4. *Dane*

x – początkowy punkt, od którego funkcja zaczyna obliczanie kolejnych przybliżeń,
f – równanie w języku C++ dla której obliczane jest $f(x)$ dla danej wartości x,
df – równanie w języku C++ dla której obliczane jest $f'(x)$ dla danej wartości x,
d2f – równanie w języku C++ dla której obliczane jest $f''(x)$ dla danej wartości x,
mit – maksymalna liczba iteracji w procesie,
eps – błąd względny wyznaczenia pierwiastka.

5. *Wynik*

NewtonRaphson (*x, f, df, d2f, mit, eps, it*) – {przybliżona wartość pierwiastka,

liczba iteracji, wartość funkcji *f* dla obliczonej wartości pierwiastka, (dla zmiennopozycyjnej arytmetyki przedziałowej) szerokość przedziału wynikowego, *st* – kod z jakim zakończyła się funkcja}

6. *Inne parametry*

st – zmienna pomocnicza, która po wykonaniu funkcji *NewtonRaphson* przyjmuje jedną z następujących wartości:
1, jeżeli wartość $f''(x_i) = 0$,
2, jeżeli wartość $[f'(x_i)]^2 - 2f(x_i)f''(x_i) < 0$,
3, jeżeli osiągnięto maksymalną liczbę iteracji ε,
0, jeżeli nie wystąpi żadne z powyższych,
Uwaga: Jeżeli *st =1, 2*, to wartość funkcji *NewtonRaphson* nie jest obliczana, gdy *st* = 3 podawane jest ostatnie obliczone przybliżenie pierwiastka.

7. *Typy parametrów*

Integer: *mit, it, st,*
float128(dla arytmetyki zmiennopozycyjnej): *x, eps, fx,*
Interval<mpreal>(dla zmiennopozycyjnej arytmetyki przedziałowej): *x, fx,*
Mpreal(dla zmiennopozycyjnej arytmetyki przedziałowej): *eps,*
Function: *f, df, d2f.*

8. *Identyfikator nielokalny*

Function – float128 nazwa_funkcji(float128 x), w trybie interval przyjmuje sygnaturę Interval<mpreal> nazwa_funkcji(const Interval<mpreal>& x)

9. *Teksty procedur*

Dla arytmetyki zmiennopozycyjnej:

```cpp
struct NewtonResult {
    __float128 root;
    int iteration;
    __float128 last_f_value;
    int status;
};

NewtonResult newtonRaphsonSecondOrder(__float128 x0, __float128 tolerance, int maxIterations,
                                      EquationFunc equation, EquationFunc first_derivative, EquationFunc second_derivative) {
    __float128 num_x = x0;
    __float128 f_value;

    for (int i = 0; i < maxIterations; ++i) {
        __float128 f_value = equation(num_x);
        __float128 f_prime = first_derivative(num_x);
        __float128 f_double_prime = second_derivative(num_x);

        // Obsługa zerowej drugiej pochodnej
        if (f_double_prime == 0.0Q) {
            return {0, 0, 0, 1}
        }

        // Oblicz wyróżnik
        __float128 discriminant = (f_prime * f_prime) - (2 * f_value * f_double_prime);
        if (discriminant < 0.0Q) {
            return {0, 0, 0, 2}
        }

        // Oblicz pierwiastek z wyróżnika
        __float128 sqrt_discriminant = sqrtq(discriminant);

        // Próba obu pierwiastków wyróżnika
        __float128 numerator = f_prime + sqrt_discriminant;
        __float128 x_next = num_x - (numerator / f_double_prime);

        if (fabsq(x_next - num_x) >= tolerance) {
            numerator = f_prime - sqrt_discriminant;
            x_next = num_x - (numerator / f_double_prime);
        }

        // Warunek zbieżności
        if (fabsq(x_next - num_x) < tolerance) {
            return {x_next, i + 1, f_value}, 0;
        }

        // Aktualizacja przybliżenia
        num_x = x_next;
    }

    return {num_x, maxIterations, f_value, 3};
}
```

Dla zmiennopozycyjnej arytmetyki przedziałowej:

```cpp
struct NewtonResult {
    Interval<mpreal> root;
    int iteration;
    Interval<mpreal> last_f_value;
    int status;
};
NewtonResult newtonRaphsonSecondOrder(
    Interval<mpreal> x0, mpreal tolerance, int maxIterations,
    EquationFunc equation, EquationFunc first_derivative, EquationFunc second_derivative) {

    Interval<mpreal> num_x = x0;
    Interval<mpreal> f_value;

    for (int i = 0; i < maxIterations; ++i) {
        Interval<mpreal> f_value = equation(num_x);
        Interval<mpreal> f_prime = first_derivative(num_x);
        Interval<mpreal> f_double_prime = second_derivative(num_x);

        // Obsługa zerowej drugiej pochodnej
        if (f_double_prime.a <= 0 && f_double_prime.b >= 0) {
            return {0, 0, 0, 1};
        }

        // Oblicz wyróżnik
        Interval<mpreal> discriminant = f_prime * f_prime - (Interval<mpreal>(2, 2) * f_value * f_double_prime);
        if (discriminant < 0){
            return {0, 0, 0, 2};
        }

        // Oblicz pierwiastek z wyróżnika
        Interval<mpreal> sqrt_discriminant = ISqrt(discriminant);

        // Próba obu pierwiastków wyróżnika
        Interval<mpreal> x_next1 = num_x - ((f_prime + sqrt_discriminant) / f_double_prime);
        Interval<mpreal> x_next2 = num_x - ((f_prime - sqrt_discriminant) / f_double_prime);

        Interval<mpreal> x_next = (IntWidth(x_next1) < IntWidth(x_next2)) ? x_next1 : x_next2;

        // Warunek zbieżności
        if (abs(x_next.a - num_x.a) < tolerance && abs(x_next.b - num_x.b) < tolerance) {
            return {x_next, i+1, f_value, 0};
        }

        // Aktualizacja przybliżenia
        num_x = x_next;
    }

    return {num_x, maxIterations, f_value, 3};
}
```

10. Przykłady

a) **Równanie:** $\sin^2 x + \frac{1}{2}\sin x - \frac{1}{2} = 0$

Definicje funkcji (*f, df, d2f*):

Funkcja(*f*): $\sin^2 x + \frac{1}{2}\sin x - \frac{1}{2}$

Pierwsza pochodna(*df*): $\sin 2x + \frac{1}{2}\cos x$

Druga pochodna (*d2f*): $2\cos 2x - \frac{1}{2}\sin x$

Dane:

x = 0.6, *mit* = 20, *eps* = 1e-16 .

Wyniki:

NewtonRaphson (*x, f, df, d2f, mit, eps, it*) = {5.23598775598299e-01, 4, 0.0e+00, 0}

b) **Równanie:** $\sin^2 x + \frac{1}{2}\sin x - \frac{1}{2} = 0$

Definicje funkcji (*f, df, d2f*):

Funkcja(*f*): $\sin^2 x + \frac{1}{2}\sin x - \frac{1}{2}$

Pierwsza pochodna(*df*): $\sin 2x + \frac{1}{2}\cos x$

Druga pochodna (*d2f*): $2\cos 2x - \frac{1}{2}\sin x$

Dane:

x = [0.6, 0.6], *mit* = 20, *eps* = 1e-16 .

Wyniki:

NewtonRaphson (*x, f, df, d2f, mit, eps, it*) = {[5.2359877559829887e-01, 5.2359877559829888e-01], 4, [-1.2598322501371465e-32, -1.2587316935511890e-32], 0}

c) **Równanie:** $\sin^2 x + \frac{1}{2}\sin x - \frac{1}{2} = 0$

Definicje funkcji (*f, df, d2f*):

Funkcja(*f*): $\sin^2 x + \frac{1}{2}\sin x - \frac{1}{2}$

Pierwsza pochodna(*df*): $\sin 2x + \frac{1}{2}\cos x$

Druga pochodna (*d2f*): $2\cos 2x - \frac{1}{2}\sin x$

Dane:

x = [0.59, 0.61], *mit* = 20, *eps* = 1e-16 .

Wyniki:

st = 1(wartość drugiej pochodnej po podstawieniu x_i jest równa zeru)