Отчёт по выполненной работе

Введение

Метод хорд, также известный как метод секущих, применяется для численного решения нелинейных уравнений. В отличие от метода половинного деления, метод хорд использует прямую, соединяющую две точки на графике функции, для приближения корня. Мы будем искать корень уравнения $f(x)=0.5\exp(-\sqrt{x})-0.2\sqrt{(x^3)}+2$ с заданной точностью $\varepsilon = 10^{-6}$.

Локализация корня

Для начала локализуем корень, то есть определим интервал [a,b] , в котором находится корень уравнения. Для этого мы проверяем значения функции на концах интервала и увеличиваем b, пока значения функции f(a) и f(b) не станут разного знака.

Входные данные программы

- a, b: концы интервала для поиска корня.
- ε: точность решения.

Выходные данные программы

- Найденный корень уравнения с заданной точностью ε (epsilon).

Код программы на языке Си

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

// Определяем функцию, для которой ищем корень
double f(double x) {
   return 0.5 * exp(-sqrt(x)) - 0.2 * pow(x, 1.5) + 2;
}

// Метод хорд для нахождения корня
double find_root(double a, double b, double epsilon) {
   if (f(a) * f(b) >= 0) {
      fprintf(stderr, "Ошибка: функция должна иметь разные знаки на концах интервала.\n");
      return NAN; // Возвращаем NAN, если знаки не разные
   }

   double c;
   while ((b - a) / 2 > epsilon) {
      c = (f(b) * a - f(a) * b) / (f(b) - f(a)); // Метод хорд

      if (f(c) == 0) {
         return c; // Найден корень
      } else if (f(c) * f(a) < 0) {
         b = c; // Корень в [a, c]
      } else {
         a = c; // Корень в [c, b]
      }
   }
   return (a + b) / 2; // Возвращаем среднюю точку
```

```
}

int main() {
    double a, b; // Начальные значения
    double epsilon;

    // Запрос у пользователя на ввод интервалов и точности
    printf("Введите значение a: ");
    scanf("%lf", &a);
    printf("Введите значение b: ");
    scanf("%lf", &b);
    printf("Введите точность ε: ");
    scanf("%lf", &epsilon);

    // Локализация корня
    while (f(a) * f(b) >= 0) {
        b += 1; // Увеличиваем b, пока не найдем интервал с изменением знака
    }

    double root = find_root(a, b, epsilon);
    if (!isnan(root)) {
        printf("Найденный корень уравнения: %.6f\n", root);
    }

    return 0;
}
```

Объяснение кода

1. **Функция `f`:**

```
double f(double x) {
    return 0.5 * exp(-sqrt(x)) - 0.2 * pow(x, 1.5) + 2;
}
```

Эта функция вычисляет значение f(x) для заданного значения x. Здесь pow(x, 1.5) используется для возведения переменной x в степень 1.5 (что эквивалентно √x^3.

2. **Функция `find_root`**:

```
double find_root(double a, double b, double epsilon) {
    if (f(a) * f(b) >= 0) {
        fprintf(stderr, "Ошибка: функция должна иметь разные знаки на концах
интервала.\n");
        return NAN; // Возвращаем NAN, если знаки не разные
    }

    double c;
    while ((b - a) / 2 > epsilon) {
        c = (f(b) * a - f(a) * b) / (f(b) - f(a)); // Метод хорд

        if (f(c) == 0) {
```

```
            return c; // Найден корень
        } else if (f(c) * f(a) < 0) {
            b = c; // Корень в [a, c]
        } else {
            a = c; // Корень в [c, b]
        }
    }
    return (a + b) / 2; // Возвращаем среднюю точку
}
```

Эта функция реализует метод хорд для поиска корня уравнения. Она принимает начальные приближения a и b, а также точность ε.

3. **Функция `main`**:

```
int main() {
    double a, b; // Начальные значения
    double epsilon;

    // Запрос у пользователя на ввод интервалов и точности
    printf("Введите значение a: ");
    scanf("%lf", &a);
    printf("Введите значение b: ");
    scanf("%lf", &b);
    printf("Введите точность ε: ");
    scanf("%lf", &epsilon);

    // Локализация корня
    while (f(a) * f(b) >= 0) {
        b += 1; // Увеличиваем b, пока не найдем интервал с изменением знака
    }

    double root = find_root(a, b, epsilon);
    if (!isnan(root)) {
        printf("Найденный корень уравнения: %.6f\n", root);
    }

    return 0;
}
```

Эта функция задаёт начальные значения интервала, локализует корень, и затем ищет корень методом хорд, выводя результат на экран.

Заключение

Метод хорд (секущих) является эффективным численным методом для поиска корней нелинейных уравнений. В данном отчёте и коде представлено решение уравнения $f(x)=0.5\exp(-\sqrt{x})-0.2\sqrt{(x^3)}+2$ с заданной точностью $\varepsilon = 10^{-6}$. Метод использует итеративный процесс для последовательного приближения к корню уравнения до достижения заданной точности.