Санкт-Петербургский Государственный Университет
Прикладная математика и информатика

Отчёт по вычислительному практикуму 1
Приближённое решение нелинейных уравнений

Выполнил:
Яковлев Денис Михайлович
Группа 21.Б06-мм
st095998@student.spbu.ru

Под руководством Алцыбеева Глеба Олеговича

Преподавателя и ассистента по дисциплине "Вычислительный практикум"

12 Марта 2023 г.

Содержание

1 Преамбула

Обучаясь на программе "Прикладная математика и информатика каждый студент получает фундаментальные и прикладные знания в области математических дисциплин и информатики, а также формирует компетенции в результате освоения основной образовательной программы.
Дисциплина "Вычислительный практикум"("Computational Workshop") нацелена на формирование у студента следующих компетенций:

- **ПКП-4**: *Способен критически анализировать разрабатываемые решения, оценивать их эффективность и целесообразность*
- **ПКП-6**: *Способен обрабатывать, анализировать данные и делать выводы, используя соответствующий математический аппарат и современные прикладные программные средства*
- **ПКП-7**: *Способен преподавать математику и информатику в средней школе, специальных учебных заведениях на основе полученного фундаментального образования и научного мировоззрения*

2 Постановка задачи

Пусть дано алгебраическое или трансцендентное уравнение вида

$$f(x) = 0 \tag{1}$$

причем, известно, что все интересующие вычислителя корни находятся на отрезке $[A, B]$, на котором функция $f(x)$ определена и гладкая. Требуется найти все корни уравнения (1) на $[A, B]$ нечетной кратности (здесь $A, B, f(x)$ – параметры задачи). Решение задачи разбить на два этапа:

1. Процедура отделения корней уравнения (1) на отрезке $[A, B]$;
2. Уточнение корней уравнения (1) на отрезках перемены знака вида $[a_i, b_i]$;

    a. Методом половинного деления (методом бисекции);
    b. Методом Ньютона (методом касательных);
    c. Модифицированным методом Ньютона;
    d. Методом секущих

с заданной точностью $\epsilon > 0$ (ϵ – параметр задачи).

3 Ход работы

3.1 Примечание

1) Требования к оформлению задачи: вывести на печать название темы, исходные параметры задачи: A, B, вид функции $f(x), \epsilon$.

2) Отделение корней произвести способом табулирования $[A, B]$ с шагом $h > 0$ (где $h = \frac{(B-A)}{N}, N \geq 2-$параметр задачи). При реализации выбирать достаточно малые значения h. Результатом решения задачи отделения корней является последовательный

3) При уточнении корней на каждом из отрезков $[a_i, b_i]$ указанными методами, выводить на печать (для каждого метода)

   - название метода (для порядка));
   - начальное(ые) приближение(я) к корню;
   - количество шагов m (в каждом методе своё) для достижения точности ϵ, та-кой что $|x_m - x_{m-1}|< \epsilon$;
   - приближенное решение x_m уравнения (1), найденное каждым из упомянутых методов с точностью ϵ;
   - $|x_m - x_{m-1}|$ (в методе бисекции выводить длину последнего отрезка);
   - абсолютную величину невязки для прибл. решения $x_m : |f(x_m) - 0|$.

3.2 Пример: Входные данные

В качестве функции выбрана $f(x) = \sqrt{x} - 2cos(\frac{\pi x}{2})$, а её производная $f'(x) = \frac{1}{2\sqrt{x}} + \pi sin(\frac{\pi x}{2})$. Концы отрезка: $A = 0, B = 4.5$, $\epsilon = 10^{-8}$, $N = 10^4$

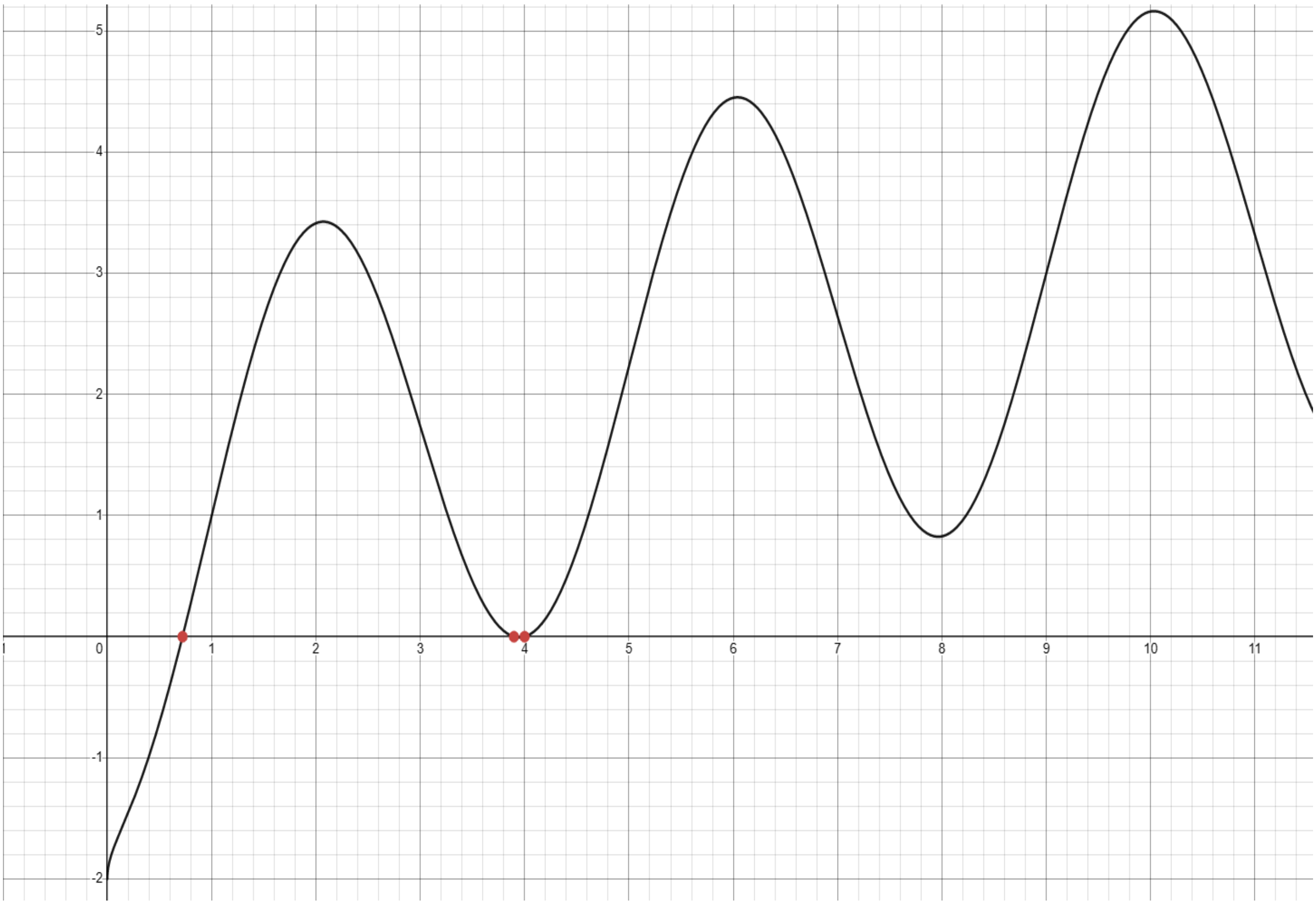

Рис. 1: График функции $f(x) = \sqrt{x} - 2cos(\frac{\pi x}{2})$. Корни выделены красным цветом

Ссылка на Git-репозиторий: `https://github.com/DeMiYak/NumericalMethods`

3.3 Вывод

```
    Тема: Приближённые решения нелинейных уравнений
f(x) = sqrt(x) - 2*cos(pi*x/2)
A = 0
B = 4.5
eps = 1e-8
N = 10000Метод



 бисекции:Между

 0.72045 и 0.7209:Начальное
 приближение: 0.720675Число
 итераций: 15Приближённое
 решение: 0.720882847595Длина
 последнего отрезка: 6.8664550712e-09Невязка
 |f(X)-0|: 1.67105285165e-08Между

 3.89745 и 3.8979:Начальное
 приближение: 3.897675Число
 итераций: 15Приближённое
 решение: 3.89780396576Длина
 последнего отрезка: 6.86645496017e-09Невязка
 |f(X)-0|: 2.72432121307e-09Между

 3.9996 и 4.00005:Начальное
 приближение: 3.999825Число
 итераций: 15Приближённое
 решение: 3.99999999847Длина
 последнего отрезка: 6.86645496017e-09Невязка
 |f(X)-0|: 3.81626286128e-10Число

 корней: 3Метод


 Ньютона:Между

 0.72045 и 0.7209:Начальное
 приближение: 0.720675Число
 итераций: 3Приближённое
 решение: 0.720882842728
|x{m}-x{m+1}|: 1.11022302463e-16Невязка
 |f(X)-0|: 2.22044604925e-16Между

 3.89745 и 3.8979:Начальное
```

```
 приближение: 3.897675Число
 итераций: 3Приближённое
 решение: 3.8978039767
|x{m}-x{m+1}|: 2.52686760405e-13Невязка
 |f(X)-0|: 2.22044604925e-16Между

 3.9996 и 4.00005:Начальное
 приближение: 3.999825Число
 итераций: 3Приближённое
 решение: 4
|x{m}-x{m+1}|: 8.91731133379e-13Невязка
 |f(X)-0|: 2.22044604925e-16Число

 корней: 3Модифицированный


 метод Ньютона:Между

 0.72045 и 0.7209:Начальное
 приближение: 0.720675Число
 итераций: 3Приближённое
 решение: 0.720882842728
|x{m} - x{m-1}|: 1.08446585045e-12Невязка
 |f(X)-0|: 3.33066907388e-16Между

 3.89745 и 3.8979:Начальное
 приближение: 3.897675Число
 итераций: 3Приближённое
 решение: 3.8978039767
|x{m} - x{m-1}|: 4.02158306656e-10Невязка
 |f(X)-0|: 2.5113244817e-13Между

 3.9996 и 4.00005:Начальное
 приближение: 3.999825Число
 итераций: 3Приближённое
 решение: 4
|x{m} - x{m-1}|: 1.04250741373e-09Невязка
 |f(X)-0|: 8.94395668638e-13Число

 корней: 3Метод


 секущих:Между

 0.72045 и 0.7209:Начальное
 приближение: 0.720882840903Число
 итераций: 1Приближённое
```

```
решение: 0.720882842728
|x{m}-x{m-1}|: 1.8250588818e-09Невязка
 |f(X)-0|: 2.63122856836e-14Между

3.89745 и 3.8979:Начальное
приближение: 3.89780430627Число
итераций: 2Приближённое
решение: 3.8978039767
|x{m}-x{m-1}|: 3.07913250452e-10Невязка
 |f(X)-0|: 2.22044604925e-16Между

3.9996 и 4.00005:Начальное
приближение: 3.99999980318Число
итераций: 2Приближённое
решение: 4
|x{m}-x{m-1}|: 9.6463725896e-11Невязка
 |f(X)-0|: 0Число

корней: 3
```

4 Приложение

4.1 Метод бисекции

```
cout << '\n' << '\n' << "Метод бисекции: " << endl;
    while(x < B)
    {
        double a = x, b = xNext;
        double c, delta, fappr=(a+b)/2;
        size_t m = 0;
        if(f(a)*f(b)<=0)
        {
            counter++;
            while(b - a >= 2*eps){
                m++;
                c = (a+b)/2;
                if(f(b)*f(c)<=0)
                {
                    a = c;
                } else b = c;
            }
            delta = (b-a)/2;

            cout << '\n' << "Между " << x << " и " << xNext << ":\n"<< "Начальное приближение: " << fappr << endl;
```

```
            cout << "Число итераций: " << m << "\Приближённоеn решение: " << c << "\Длинаn последнего отрезка: " << delta;
            cout << "\Невязкаn |f(X)-0|: " << fabs(f(c)) << endl;

        }

        x = xNext;
        xNext += h;
    }
    cout << '\n' << "Число корней: " << counter << endl;
```

4.2 Метод Ньютона

```
    cout << '\n' << '\n' << "Метод Ньютона: " << endl;
    while(x < B)
    {
        double a = x, b = xNext;
        double c = (a+b)/2, delta, fappr=(a+b)/2;
        size_t m = 0;
        if(f(a)*f(b) <= 0){
            counter++;
            while(fabs(c - a) >= eps){
                m++;
                a = c;
                c = fi(c);
                if(c > xNext || c < x){
                    cout << "Между "<< x << " и " << xNext << ": Ошибка: корень не может быть вычислен попробуйте( взять N побольше)" << endl;
                    m = 0;
                    break;
                }
            }
        }
        if(m){
            delta = fabs(c-a);

            cout << '\n' << "Между " << x << " и " << xNext << ":\n"<< "Начальное приближение: " << fappr << endl;
            cout << "Число итераций: " << m << "\Приближённоеn решение: " << c << "\n|x{m}-x{m+1}|: " << delta;
            cout << "\Невязкаn |f(X)-0|: " << fabs(f(c)) << endl;
        }
        x = xNext;
        xNext += h;
    }
    cout << '\n' << "Число корней: " << counter << endl;
```

4.3 Модифицированный метод Ньютона

```
cout << '\n' << '\n' << "Модифицированный метод Ньютона: " << endl;
while(x < B)
{
    double a = x, b = xNext;
    double c = (a+b)/2, delta, fappr=(a+b)/2;
    double cf = df(c);
    size_t m = 0;
    if(f(a)*f(b) <= 0){
        counter++;
        while(fabs(c - a) >= eps){
            m++;
            a = c;
            c = c - f(c)/cf;
            if(c > xNext || c < x){
                cout << "Между "<< x << " и " << xNext << ": Ошибка: корень не может быть вычислен попробуйте( взять N побольше)" << endl;
                m = 0;
                break;
            }
        }
    }
    if(m){
        delta = fabs(c-a);

        cout << '\n' << "Между " << x << " и " << xNext << ":\n"<< "Начальное приближение: " << fappr << endl;
        cout << "Число итераций: " << m << "\Приближённоеn решение: " << c << "\n|x{m} - x{m-1}|: " << delta;
        cout << "\Невязкаn |f(X)-0|: " << fabs(f(c)) << endl;
    }
    x = xNext;
    xNext += h;
}
cout << '\n' << "Число корней: " << counter << endl;
```

4.4 Метод секущих

```
cout << '\n' << '\n' << "Метод секущих: " << endl;
while(x < B){
    double a = x, b = xNext;
```

```
    double c = secf(b, a), delta, fappr=secf(b, a);
    size_t m = 0;
    if(f(a)*f(b)<=0){
        counter++;
        while(fabs(c - b) >= eps){
            m++;
            a = b;
            b = c;
            c = secf(c, a);
            if(c < x || c > xNext){
                cout << "Между "<< x << " и " << xNext << ": Ошибка: корень не может быть вычислен попробуйте( взять N побольше)" << endl;
                m = 0;
                break;
            }
        }
    }
    if(m){
        delta = fabs(c - b);
        cout << '\n' << "Между " << x << " и " << xNext << ":\n"<< "Начальное приближение: " << fappr << endl;
        cout << "Число итераций: " << m << "\Приближённоеn решение: " << c << "\n|x{m}-x{m-1}|: " << delta;
        cout << "\Невязкаn |f(X)-0|: " << fabs(f(c)) << endl;
    }
    x = xNext;
    xNext += h;
}
cout << '\n' << "Число корней: " << counter << endl;
```