Федеральное государственное автономное образовательное
учреждение высшего образования Национальный
исследовательский университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №2
Вычислительная математика

Вариант: 17

Группа	P3208
Студент	Щетинин С.В.
Преподаватель	Машина Е.А.

1 Цель работы

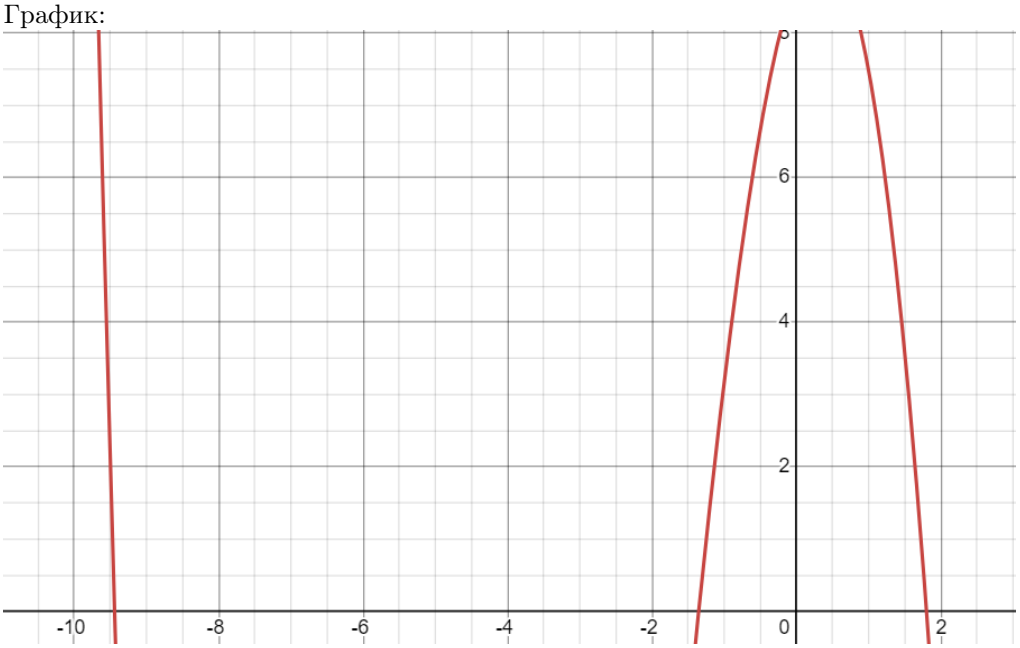
1. Найти приближенное значение определенного интеграла с требуемой точностью различными численными методами.

2 Порядок выполнения

1. Уравнение:

$$-0.38x^3 - 3.42x^2 + 2.51x + 8.75$$

График:



Точность: $\epsilon = 10^(-6)$ Первый корень: $[-10; -8]$
Метод секущих ($x_0 = -8$):

№	x_{k-1}	x_k	x_{k-1}	$f(x_{k+1})$	$\lvert x_{k+1} - x_k \rvert$
1	-8	-7.9	-10.351	37.7739	2.45099
2	-7.9	-10.351	-9.11556	-10.4811	1.23543
3	-10.351	-9.11556	-9.3839	-1.95752	0.268339
4	-9.11556	-9.3839	-9.44553	0.146391	0.0616264
5	-9.3839	-9.44553	-9.44124	-0.00180044	0.00428798
6	-9.44553	-9.44124	-9.44129	-1.62079e-06	5.20967e-05
7	-9.44124	-9.44129	-9.44129	1.79733e-11	4.69406e-08

Второй корень: $[-2; 0]$
Метод простых итераций ($x_0 = -2$):

№	x_k	x_{k+1}	$f(x_{k+1})$	$\lvert x_{k+1} - x_k \rvert$
1	2	1.81501	1.79934	0.184991
2	1.81501	1.79934	1.79798	0.0156694
3	1.79934	1.79798	1.79786	0.00135832
4	1.79798	1.79786	1.79785	0.000117981
5	1.79786	1.79785	1.79785	1.02494e-05
6	1.79785	1.79785	1.79785	8.90417e-07

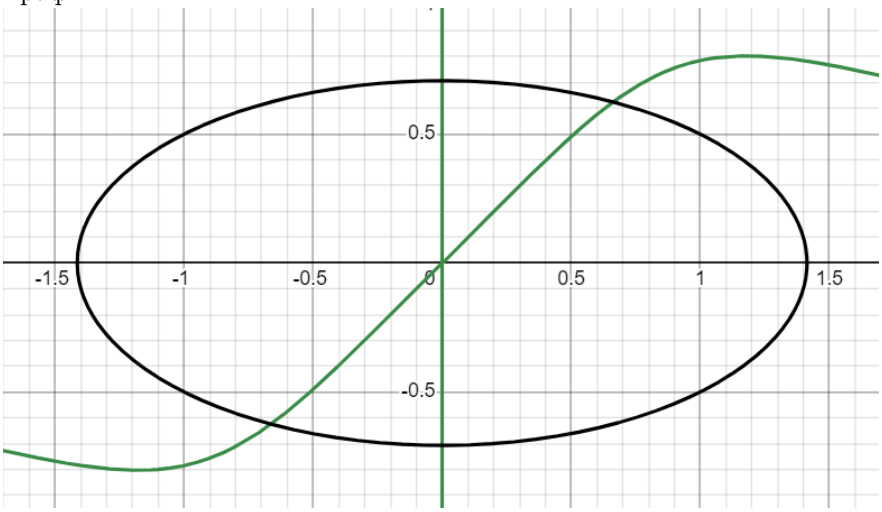
Третий корень: $[0; 2]$

№	a	b	x	$f(a)$	$f(b)$	$f(x)$	$\lvert a-b\rvert$
1	-2	0	-1	-6.91	8.75	3.2	2
2	-2	-1	-1.5	-6.91	3.2	-1.4275	1
3	-1.5	-1	-1.25	-1.4275	3.2	1.01094	0.5
4	-1.5	-1.25	-1.375	-1.4275	1.01094	-0.179336	0.25
5	-1.375	-1.25	-1.3125	-0.179336	1.01094	0.423315	0.125
6	-1.375	-1.3125	-1.34375	-0.179336	0.423315	0.123834	0.0625
7	-1.375	-1.34375	-1.35938	-0.179336	0.123834	-0.0272945	0.03125
8	-1.35938	-1.34375	-1.35156	-0.0272945	0.123834	0.0483842	0.015625
9	-1.35938	-1.35156	-1.35547	-0.0272945	0.0483842	0.0105735	0.0078125
10	-1.35938	-1.35547	-1.35742	-0.0272945	0.0105735	-0.0083534	0.00390625
11	-1.35742	-1.35547	-1.35645	-0.0083534	0.0105735	0.00111181	0.00195312
12	-1.35742	-1.35645	-1.35693	-0.0083534	0.00111181	-0.00362035	0.000976562
13	-1.35693	-1.35645	-1.35669	-0.00362035	0.00111181	-0.00125415	0.000488281
14	-1.35669	-1.35645	-1.35657	-0.00125415	0.00111181	-7.11417e-05	0.000244141
15	-1.35657	-1.35645	-1.35651	-7.11417e-05	0.00111181	0.000520343	0.00012207
16	-1.35657	-1.35651	-1.35654	-7.11417e-05	0.000520343	0.000224603	6.10352e-05
17	-1.35657	-1.35654	-1.35655	-7.11417e-05	0.000224603	7.67309e-05	3.05176e-05
18	-1.35657	-1.35655	-1.35656	-7.11417e-05	7.67309e-05	2.79469e-06	1.52588e-05
19	-1.35657	-1.35656	-1.35656	-7.11417e-05	2.79469e-06	-3.41735e-05	7.62939e-06
20	-1.35656	-1.35656	-1.35656	-3.41735e-05	2.79469e-06	-1.56894e-05	3.8147e-06
21	-1.35656	-1.35656	-1.35656	-1.56894e-05	2.79469e-06	-6.44735e-06	1.90735e-06

2. Система (метод Ньютона):

$$\begin{cases} \tan xy = x^2 \\ 0.5x^2 + 2y^2 = 1 \end{cases}$$

График:



$$\frac{\delta f}{\delta x} = \frac{y}{\cos^2(yx) - 2x}$$

$$\frac{\delta f}{\delta y} = \frac{x}{\cos^2(yx)}$$

$$\frac{\delta g}{\delta x} = x$$

$$\frac{\delta g}{\delta y} = 4y$$

$$\begin{pmatrix} \frac{y}{\cos^2(yx)-2x} & \frac{x}{\cos^2(yx)} \\ x & 4y \end{pmatrix} \cdot \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} \tan xy - x^2 \\ 0.5x^2 + 2y^2 + 1 \end{pmatrix}$$

$$\begin{cases} \dfrac{y\Delta x}{\cos^2(yx) - 2x} - 2x\Delta x + \dfrac{x\Delta y}{\cos^2(yx)} = \tan xy - x^2 \\ \Delta x x + yy\Delta y = 0.5x^2 + 2y^2 + 1 \end{cases}$$

При $x_0 = 0.6, y_0 = 0.6$ ответ $x = 0.66292, y = 0.62461$

3 Расчетные формулы метода

1. Метод половинного деления: $x_i = \frac{a_i + b_2}{2}$
2. Метод секущих: $x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i)$
3. Метод простой итерации: $x_{i+1} = \phi(x_i)$

4 Листинг программы

Программа написана на C++.

```
#include <iostream>
#include <string>
#include <utility>
#include <vector>
#include <map>
#include <set>
#include <cmath>
#include <sstream>

class UserInput {
public:
    std::pair<long double, long double> segment;
    long double x0_secant;
    long double x0_iter;
};


class Function {
private:
    const long double EPS = 1e-6;
    const long double STEP = 0.0001;
    const long double LEFT_CORN = -20;
    const long double RIGHT_CORN = 20;
    std::string _str_func;
    long double (*_func)(long double){};
    long double (*_x_func)(long double){};
    std::vector <std::pair<long double, long double>> _segments;
    long double _x1_secant;
    UserInput _user;
public:
    Function() = default;
    Function(std::string  _str_func,
             long double (*_func)(long double),
             long double (*_x_func)(long double));
    Function& operator=(const Function& f);

    [[nodiscard]] std::string get_str_func() const;
    std::vector <std::pair <long double, long double>> get_vector_segments_with_root();
    void setUserInput(UserInput user);
    [[nodiscard]] long double left_root() const;
    [[nodiscard]] long double right_root() const;
    [[nodiscard]] long double middle_root() const;

    friend bool is_correct_segment(const Function& f,
                                   std::pair <long double, long double> segment);
    friend bool is_correct_x0_iter(const Function &f,
                                   long double x0_iter);
    friend bool is_correct_x0_secant(const Function &f,
                                     long double x0_secant);

};

```

```cpp

Function::Function(std::string  _str_func,
                   long double (*const _func)(long double),
                   long double (*const _x_func)(long double))
        : _str_func(std::move(_str_func))
        , _func(_func)
        , _x_func(_x_func){

    std::vector <std::pair <long double, long double>> v_root =
            get_vector_segments_with_root();
    long double left = LEFT_CORN;

    for (size_t i = 0; i < v_root.size(); ++i) {
        if (i != v_root.size() - 1) {
            _segments.emplace_back(left, v_root[i + 1].first);
        } else {
            _segments.emplace_back(left, RIGHT_CORN);
        }
        left = v_root[i].second;
    }

}

Function& Function::operator=(const Function& f){
    if (this == &f) return *this;
    _str_func = f._str_func;
    _func = f._func;
    _x_func = f._x_func;
    _segments = f._segments;
    _x1_secant = f._x1_secant;
    _user = f._user;
    return *this;
}

std::string Function::get_str_func() const{
    return _str_func;
}

std::vector <std::pair <long double, long double>> Function::get_vector_segments_with_root() {
    std::vector <std::pair <long double, long double>> v_root;
    bool is_plus_pred = _func(LEFT_CORN) > 0;
    long double pred = LEFT_CORN;
    for (long double cur = LEFT_CORN; cur <= RIGHT_CORN; cur += STEP) {
        long double val = _func(cur);
        bool is_plus = val > 0;
        if (std::abs(val) < EPS) {
            //v_root.emplace_back(pred, cur + STEP);
        } else if (is_plus_pred != is_plus) {
            v_root.emplace_back(pred, cur);
        }
        is_plus_pred = is_plus;
        pred = cur;
    }
    return v_root;
}

void Function::setUserInput(UserInput user) {
    this->_user = user;
    _x1_secant = user.x0_secant + 0.1;
}

long double Function::left_root() const{
    long double cur_x = _x1_secant;
    long double pred_x = _user.x0_secant;
    long double pred_pred_x;
    long double cur_val = _func(cur_x);
    int cnt = 0;
    while (std::abs(cur_val) > EPS) {
        ++cnt;
        pred_pred_x = pred_x;
        pred_x = cur_x;
        long double value_pred_x = _func(pred_x);
        cur_x =
                pred_x - (pred_x - pred_pred_x) /
                         (value_pred_x - _func(pred_pred_x)) * value_pred_x;
        cur_val = _func(cur_x);
```

```
        /*std::cout << "\\hline " << cnt << " & " << pred_pred_x << " & "
        << pred_x << " & " << cur_x << " & " << cur_val << " & " << std::abs(cur_x - pred_x)
    << std::endl;*/
    }
    return cur_x;
}

long double Function::right_root() const{
    long double cur_x = _user.x0_iter;
    long double pred_x = RIGHT_CORN + 1;
    int cnt = 0;
    while (std::abs(cur_x - pred_x) > EPS) {
        ++cnt;
        pred_x = cur_x;
        cur_x = _x_func(cur_x);
        std::cout << "\\hline " << cnt << "&" << pred_x << "&" <<
                  cur_x << "&" <<
                  _x_func(cur_x) << "&" << std::abs(cur_x - pred_x) << "\\\\" << std::endl;
    }

    return cur_x;
}

long double Function::middle_root() const{

    long double l = _user.segment.first;
    long double r = _user.segment.second;
    bool is = false;
    if (_func(l) > _func(r)) {
        is = true;
    }
    int cnt = 0;
    while (r - l > EPS) {
        ++cnt;
        long double m = (r + l)/2;
        if (!is) {
            (_func(m) > 0) ? r = m : l = m;
        } else {
            (_func(m) > 0) ? l = m : r = m;
        }

    }

    return l;

}

bool is_correct_segment(const Function &f,
                        std::pair<long double, long double> segment) {
    return f._segments[1].first <= segment.first && f._segments[1].second >= segment.second;
}

bool is_correct_x0_iter(const Function &f,
                        long double x0_iter) {
    return f._segments[2].first <= x0_iter && f._segments[2].second >= x0_iter;
}

bool is_correct_x0_secant(const Function &f,
                          long double x0_secant) {
    return f._segments[0].first <= x0_secant && f._segments[0].second >= x0_secant;
}


std::map<int, Function>  init_functions() {
    std::map<int, Function> num2func;
    num2func[1] = Function(
            "2x^3 - 2x^2 - x + 1",
            [](long double x) {return 2*x*x*x - 2*x*x - x + 1;},
            [](long double x){return std::cbrt((2*x*x + x - 1)/2.);}
    );

    num2func[2] = Function(
            "-0.38x^3 - 3.42x^2 + 2.51x + 8.75",
            [](long double x) {return -0.38*x*x*x - 3.42*x*x + 2.51*x + 8.75;},
            [](long double x){
                return std::sqrt((-8.75 - 2.51 * x) / (-0.38 * x - 3.42));
```

```
            }
    );

    num2func[3] = Function(
            "(2^(x) - 1)* (3^(x) - 2) * (x - 2) + 1",
            [](long double x) {
                return static_cast<long double>(
                        (pow(2, x) - 1) * (pow(3, x) - 2) * (x - 2) + 1
                );
            },
            [](long double x){
                return static_cast<long double>(
                        -1 / ((pow(2, x) - 1) * (pow(3, x) - 2)) + 2
                );
            }
    );
    return num2func;
}

Function select_func(const std::map<int, Function>& num2func) {
    std::cout << "Please, input number of function:\n";
    for (auto &x : num2func) {
        std::cout << x.first << ". " << x.second.get_str_func() << '\n';
    }

    int num;
    std::cin >> num;
    return num2func.at(num);
}

void set_x0_secant(UserInput &userInput, const Function& func) {
    std::cout << "Input x0 for left root of equation (secant):\n";
    std::cin >> userInput.x0_secant;

    if (!is_correct_x0_secant(func, userInput.x0_secant)) {
        std::cout << "Invalid value for x0 secant\n";
        exit(1);
    }
}

void set_segment(UserInput &userInput, const Function& func) {
    std::cout << "\nInput segment for middle root of equation (binary search):\nLeft:\n";
    std::cin >> userInput.segment.first;
    std::cout << "\nRight:\n";
    std::cin >> userInput.segment.second;

    if (!is_correct_segment(func, userInput.segment)) {
        std::cout << "Invalid value for segment\n";
        exit(1);
    }
}

void set_x0_iter(UserInput &userInput, const Function& func) {
    std::cout << "\nInput x0 for right root of equation (iterations):\n";
    std::cin >> userInput.x0_iter;

    if (!is_correct_x0_iter(func, userInput.x0_iter)) {
        std::cout << "Invalid value for x0 iter\n";
        exit(1);
    }
}

void set_user_data_in_func(Function &func) {
    UserInput userInput;

    set_x0_secant(userInput, func);
    set_segment(userInput, func);
    set_x0_iter(userInput, func);

    func.setUserInput(userInput);
}

void determining_root(const Function& func) {
    std::cout << "Left root:\n";
    std::cout << func.left_root() << std::endl;
    std::cout << "Middle root:\n";
```

```
    std::cout << func.middle_root() << std::endl;
    std::cout << "Right root:\n";
    std::cout << func.right_root() << std::endl;
}

class System {
private:
    const long double EPS = 1e-4;
    std::string str_system;
    long double (*x_func)(long double){};
    long double (*y_func)(long double){};
    long double x0 = 0;
    long double y0 = 0;
public:
    System() = default;
    System(std::string  str_system,
           long double (*const x_func)(long double),
           long double (*const y_func)(long double))
           : str_system(std::move(str_system))
           , x_func(x_func)
           , y_func(y_func){

    }
    System& operator=(const System& f) {
        if (this == &f) return *this;
        str_system = f.str_system;
        x_func = f.x_func;
        y_func = f.y_func;
        x0 = f.x0;
        y0 = f.y0;
        return *this;
    };
    void set_xy(long double x0, long double y0) {
        this->x0 = x0;
        this->y0 = y0;
    }
    [[nodiscard]] std::string get_str_system() const {
        return str_system;
    }
    std::pair <long double, long double> count_root() {
        long double x0p = 0, y0p = 0;
        long double x0c = x0, y0c = y0;
        do {
            x0p = x0c, y0p = y0c;
            x0c = x_func(x0p);
            y0c = y_func(y0p);
        } while ( std::abs(x0c - x0p) >  EPS && std::abs(y0c - y0p) >  EPS);
        return {x0c, y0c};
    }
};

std::map <int, System>  init_systems() {
    std::map <int, System> num2system;
    num2system[1] = System(
            "(\n_|x^2 + y^2 - 4 = 0\n |1/x = y\n (\n",
            [](long double y){
                return std::sqrt(4 - y);
            },
            [](long double x){
                return 1/x;
            }
    );
    num2system[2] = System(
            "(\n_|x^2 + y^2 - 4 = 0\n |3x^2 = y\n (\n",
            [](long double y){
                return std::sqrt(y/3);
            },
            [](long double x){
                return std::sqrt(4 - x);
            }
    );
    num2system[3] = System(
            "(\n_|1/x = y\n |3x^2 = y\n (\n",
            [](long double y){
                return std::sqrt(y/3);
            },
```

```
            [](long double x){
                return 1/x;
            }
    );
    return num2system;
}

System select_system(const std::map<int, System>& num2system) {
    std::cout << "Input number of system\n";
    for (auto &x : num2system) {
        std::cout << x.first << ".\n " << x.second.get_str_system();
    }
    int num;
    std::cin >> num;
    return num2system.at(num);
}

void set_user_data_in_system(System &system) {
    long double x0, y0;
    std::cout << "Input x0:\n";
    std::cin >> x0;
    std::cout << "Input y0:\n";
    std::cin >> y0;
    system.set_xy(x0, y0);
}

int main() {

    const std::map<int, Function> num2func = init_functions();
    Function func = select_func(num2func);
    set_user_data_in_func(func);
    determining_root(func);

    std::map<int, System> num2system = init_systems();
    System system = select_system(num2system);
    set_user_data_in_system(system);
    std::pair <long double, long double> solve = system.count_root();
    std::cout << solve.first << " " << solve.second;

    return 0;
}
```

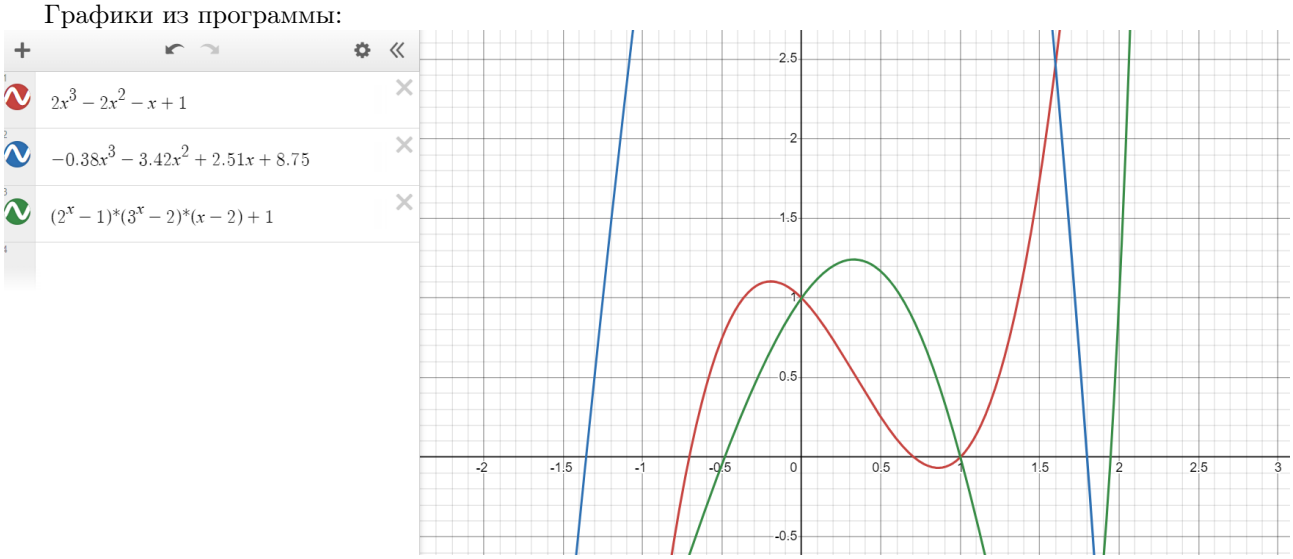
Листинг 1: Matrix.h

5 Примеры и результаты работы программы

```
Please, input number of function:
1. 2x^3 - 2x^2 - x + 1
2. -0.38x^3 - 3.42x^2 + 2.51x + 8.75
3. (2^(x) - 1)* (3^(x) - 2) * (x - 2) + 1

2

Input x0 for left root of equation (secant):

-8

Input segment for middle root of equation (binary search):
Left:

-2

Right:

0

Input x0 for right root of equation (iterations):

2
```

```
Left root:
-9.44129
Middle root:
-1.35656
Right root:
1.79785

Input number of system
1.
 (
_|x^2 + y^2 - 4 = 0
 |1/x = y
 (
2.
 (
_|x^2 + y^2 - 4 = 0
 |3x^2 = y
 (
3.
 (
_|1/x = y
 |3x^2 = y
 (

2

Input x0:

0.1

Input y0:

0.1

0.332942 1.56154
```

Графики из программы:



6 Вывод

В ходе выполнения лабораторной работы, я узнал некоторые методы решения нелинейных уравнений и систем нелинейных уравнений. А также написал программу для их реализации