



**Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**Лабораторная работа № 5
по курсу «Вычислительные алгоритмы»**

Тема Построение и программная реализация алгоритмов численного интегрирования

Студент Колосов Д.В.

Группа ИУ7-42Б

Оценка (баллы) _____

Преподаватель Градов В.М.

Москва.
2020 г.

Содержание

1	Цель работы	3
2	Задание	4
3	Результаты	5
4	Описание алгоритма	6
4.1	Квадратурная формула Гаусса	6
4.2	Формула Симпсона	7
5	Полученные результаты	8
5.1	Описать алгоритм вычисления n корней полинома Лежандра n -ой степени $P_n(x)$ при реализации формулы Гаусса . . .	8
5.2	Исследовать влияние количества выбираемых узлов сетки по каждому направлению на точность расчетов	8
5.3	3. Построить график зависимости $\epsilon(\tau)$ в диапазоне изменения $\tau=0.05-10$. Указать при каком количестве узлов получены результаты	9
6	Код программы	11
6.1	Основной модуль - формула Гаусса	11
6.2	Вспомогательный модуль для нахождения корней полинома Лежандра методом половинного деления	13
6.3	Вспомогательный модуль для решения СЛАУ методом Гаусса	14
6.4	Вспомогательный модуль для метода Симпсона	16
7	Контрольные вопросы	17
7.1	В каких ситуациях теоретический порядок квадратурных формул численного интегрирования не достигается	17
7.2	Построить формулу Гаусса численного интегрирования при одном узле	17
7.3	Построить формулу Гаусса численного интегрирования при двух узлах	17
7.4	Получить обобщенную кубатурную формулу, аналогичную (6.6) из лекции №6, для вычисления двойного интеграла методом последовательного интегрирования на основе формулы трапеций с тремя узлами по каждому направлению .	18

1 Цель работы

Получение навыков построения алгоритма вычисления двукратного интеграла с использованием квадратурных формул Гаусса и Симпсона.

2 Задание

Построить алгоритм и программу для вычисления двукратного интеграла при фиксированном значении параметра τ

$$\epsilon(\tau) = \frac{4}{\pi} \int_0^{\pi/2} d\phi \int_0^{\pi/2} [1 - \exp(-\tau \frac{l}{R})] \cos\theta \sin\theta d\theta$$

$$\text{где } \frac{l}{R} = \frac{2\cos\theta}{1-\sin^2\theta\cos^2\theta}$$

θ, ϕ - углы сферических координат

Применить метод последовательного интегрирования. По одному направлению использовать формулу Гаусса, а по другому - формулу Симпсона.

3 Результаты

1. Описать алгоритм вычисления n корней полинома Лежандра n -ой степени $P_n(x)$ при реализации формулы Гаусса.
2. Исследовать влияние количества выбираемых узлов сетки по каждому направлению на точность расчетов.
3. Построить график зависимости $\epsilon(\tau)$ в диапазоне изменения $\tau=0.05$ -10. Указать при каком количестве узлов получены результаты.

4 Описание алгоритма

4.1 Квадратурная формула Гаусса

Пусть интеграл вычисляется на стандартном интервале $[1; 1]$. Задача состоит в том, чтобы подобрать точки t_1, t_2, \dots, t_n и коэффициенты A_1, A_2, \dots, A_n так, чтобы квадратурная формула

$$\int_{-1}^1 f(t)dt = \sum_{i=1}^n A_i f(t_i)$$

Была точной для всех полиномов наивысшей возможной степени. Установлено, что эта наивысшая степень равна $N = 2n - 1$

Согласно вышенаписанной формуле:

$$\int_{-1}^1 t^k dt = \sum_{i=1}^n A_i t_i^k, k = 0, 1, 2, \dots, 2n - 1$$

Примняая во внимание, что:

$$\int_{-1}^1 t^k dt = \frac{1 - (-1)^{k+1}}{k+1} = \begin{cases} \frac{2}{k+1}, \text{mod}(k, 2) = 0 \\ 0, \text{mod}(k, 2) = 1 \end{cases}$$

Таким образом коэффициенты A_i и узлы t_i находятся из системы $2n$ уравнений.

$$\begin{cases} \sum_{i=1}^n A_i = 2, \\ \sum_{i=1}^n A_i t_i = 0, \\ \sum_{i=1}^n A_i t_i^2 = \frac{2}{3}, \\ \dots \\ \sum_{i=1}^n A_i t_i^{2n-1} = 0 \end{cases}$$

Системы нелинейна и найти решения довольно трудно. Для облегчения задачи воспользуемся полиномом Лежандра степени n и найдём его нули. Нули полинома Лежандра будут являться узлами формулы Гаусса. Проще всего найти нули полинома Лежандра учитывая рекуррентное соотношение

$$P_m(x) = \frac{1}{m}[(2m-1)xP_{m-1}(x) - (m-1)P_{m-2}(x)]$$

Далее системе решается методом Гаусса, откуда находятся коэффициенты A_i После чего подставляются в формулу:

$$\int_{-1}^1 f(t)dt = \sum_{i=1}^n A_i f(t_i)$$

При вычислении интеграла на произвольном интервале $[a; b]$, т.е. $\int_a^b f(x)dx$ для применения квадратурной формулы Гаусса необходимо выполнить преобразование переменной

$$x = \frac{b+a}{2} + \frac{b-a}{2}t$$

Таким образом получим:

$$\int_a^b f(x)dx = \frac{b-a}{2} \sum_{i=1}^n A_i f(x_i),$$

где

$$x_i = \frac{b+a}{2} + \frac{b-a}{2}t_i, i = 1, 2, \dots, n$$

4.2 Формула Симпсона

$$\int_a^b f(x)dx \approx \frac{h}{3} \sum_{i=0}^{\frac{N}{2}-1} (f_{2i} + 4f_{2i+1} + f_{2i+2})$$

5 Полученные результаты

5.1 Описать алгоритм вычисления n корней полинома Лежандра n -ой степени $P_n(x)$ при реализации формулы Гаусса

Для нахождения корней полинома Лежандра я применил метод половинного деления, опираясь на важное рекуррентное свойство полинома Лежандра:

$$P_m(x) = \frac{1}{m}[(2m-1)xP_{m-1}(x) - (m-1)P_{m-2}(x)]$$

Базой рекурсии в таком случае является:

$$P_0(x) = 1$$

$$P_1(x) = x$$

Очевидно, что для применения половинного деления необходимо сначала найти такой отрезок, чтобы на его концах знаки функции были различны (либо чтобы произведение обращалось в нуль), т.е.

$$P_n(x) * P_n(x + step) \leq 0$$

Далее с этими исходными данными применяется метод половинного деления, описанный мною выше.

5.2 Исследовать влияние количества выбираемых узлов сетки по каждому направлению на точность расчетов

Исследуем значения $\epsilon(\tau)$ при $\tau = 7$

Начальное количество узлов $N = 2, M = 2$:

Получим следующий результат $\epsilon(7) = 1.04718$ что, опираясь на физическое содержание задачи, не является точным результатом ($\epsilon < 1$).

Теперь увеличим количество узлов, а именно сделаем $N = 4, M = 4$

Получим следующий результат $\epsilon(7) = 1.00185$. Уже гораздо лучше, но ϵ всё ещё больше 1!

Ещё раз увеличим количество узлов, а именно сделаем $N = 16, M = 16$

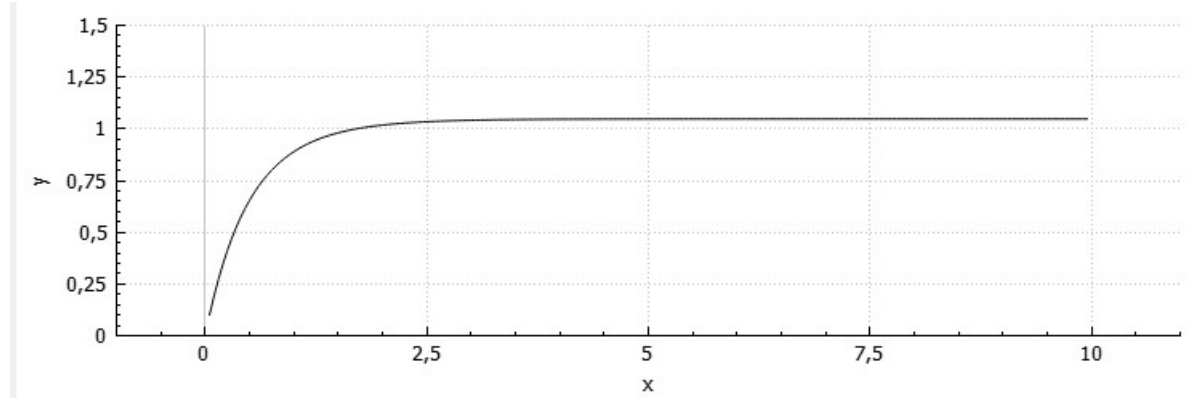
Получим следующий результат $\epsilon(7) = 0.99657$. Что гораздо ближе к правде, чем предыдущие результаты.

Таким образом можно сделать вывод, что количество узлов напрямую

влияет на точность результатов.

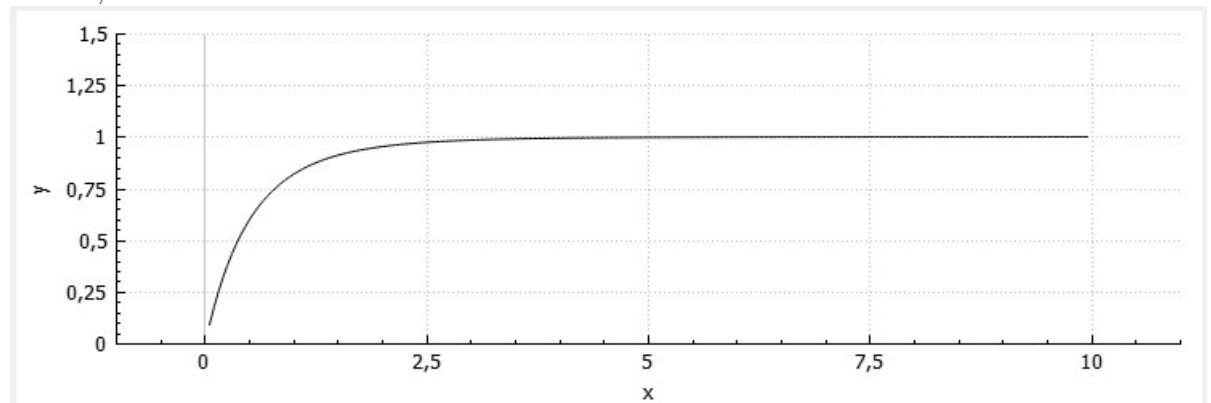
5.3 3. Построить график зависимости $\epsilon(\tau)$ в диапазоне изменения $\tau=0.05-10$. Указать при каком количестве узлов получены результаты

$N = 2, M = 2$:



Как видно, график при некоторых значениях лежит выше 1

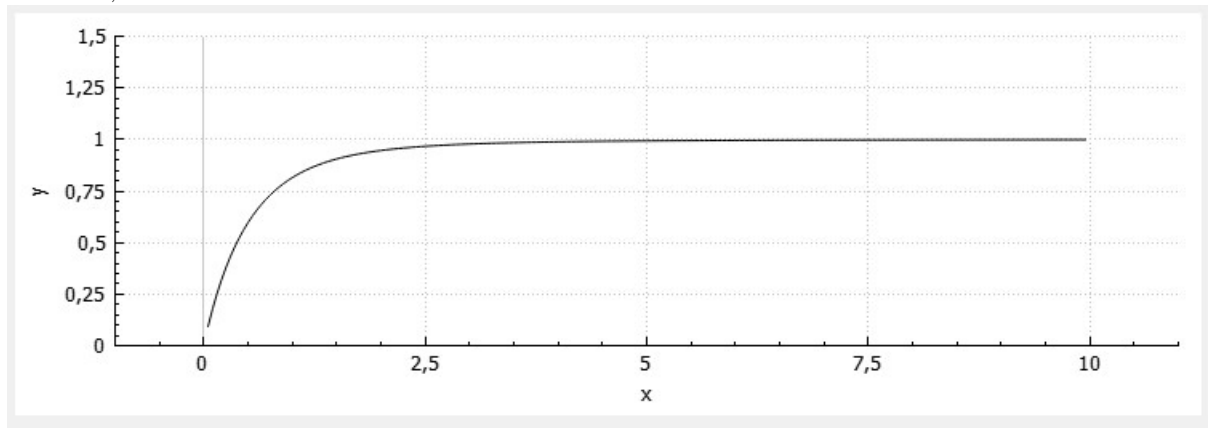
$N = 4, M = 4$:



Лучше, но в некоторых местах график все ещё выше 1. Это не очень заметно, но в этом можно убедиться в выведенных мною логах

```
1.00223
1.00223
1.00223
1.00224
1.00224
1.00224
1.00224
1.00224
```

$N = 16, M = 16$:



Приемлимая точность. В логах тоже всё хорошо:

```
0.998431
0.998452
0.998473
0.998494
0.998515
0.998535
0.998555
0.998574
```

6 Код программы

6.1 Основной модуль - формула Гаусса

```
QVector<double> x;
QVector<double> y;
for (double tau = 0.05; tau <= 10; tau += 0.05) {
    x.push_back(tau);

    N = ui->inputN->text().toInt();
    M = ui->inputM->text().toInt();

    h_y = (end - start) / M;

    int n = N;
    std::vector<std::vector<double>> P(n + 1);

    std::vector<double> t;
    for (double x = start_poly; x < end_poly - step; x += step) {
        if (P_i(x, n) * P_i(x + step, n) < 0) {
            t.push_back(half_search(x, x + step, n));
        }
    }

    std::vector<std::vector<double>> matrix(2 * n - 1);
    for (int i = 0; i < 2 * n - 1; i++) {
        matrix[i] = std::vector<double> (n + 1);
        for (int j = 0 ; j < n; j++) {
            matrix[i][j] = pow(t[j], i);
        }
        matrix[i][n] = (1 - pow(-1, i + 1)) / (i + 1);
    }

    for (int iter = 0; iter < n; iter++) {
        matrix_max_first(matrix, n, iter);
        matrix_normalize_rows(matrix, n, iter);
    }

    std::vector<double> A(n);
    matrix_get_solutions(matrix, n, A);

    double res = 0;
    for (int i = 0; i < N; i++) {
```

```

        res += A[i] * F((end + start) / 2 + (end - start) / 2 * t[i], tau, M, h_y);
    }
    res *= (end - start) / 2;

    res *= 4 / M_PI;
    qDebug() << res;
    y.push_back(res);
}

```

6.2 Вспомогательный модуль для нахождения корней полинома Лежандра методом половинного деления

```
double P_i(double x, int i) {
    if (i == 0)
        return 1;
    else if (i == 1)
        return x;

    return (double)1/i * ((2 * i - 1) * x * P_i(x, i - 1) - (i - 1) * P_i(x, i - 2))
}

double half_search(double start, double end, int n) {
    double half = (end + start) / 2;
    while (fabs(P_i(half, n) > EPS)) {
        if (P_i(half, n) * P_i(end, n) < 0) {
            start = half;
            half = (end + start) / 2;
        } else {
            end = half;
            half = (end + start) / 2;
        }
    }

    return half;
}
```

6.3 Вспомогательный модуль для решения СЛАУ методом Гаусса

```
void matrix_max_first(std::vector <std::vector <double>> &matrix,
                    int x_vars, int iter) {
    int mx = iter;
    for (int i = iter; i < x_vars; i++) {
        if (matrix[i][iter] > matrix[mx][iter])
            mx = i;
    }
    auto tmp = matrix[iter];
    matrix[iter] = matrix[mx];
    matrix[mx] = tmp;
}

void matrix_normalize_rows(std::vector <std::vector <double>> &matrix,
                        int x_vars, int iter) {
    for (int i = iter; i < x_vars; i++) {
        double normalize = matrix[i][iter];
        if (fabs(normalize) < 1E-06) {
            continue;
        }
        for (int j = iter; j < x_vars + 1; j++) {
            matrix[i][j] /= normalize;
        }
    }

    for (int i = iter + 1; i < x_vars; i++) {
        if (matrix[i][iter] < 1E-06)
            continue;
        for (int j = iter; j < x_vars + 1; j++) {
            matrix[i][j] -= matrix[iter][j];
        }
    }
}

void matrix_get_solutions(std::vector <std::vector <double>> &matrix,
                        int x_vars, std::vector <double> &x) {
    for (int i = x_vars - 1; i >= 0; i--) {
        double sigma = 0;
        for (int j = x_vars - 1; j > i; j--) {
            sigma += matrix[i][j] * x[j];
        }
    }
}
```

```
        if (fabs(matrix[i][i]) <= 1E-06) {  
            x[i] = 0;  
            continue;  
        }  
        x[i] = (matrix[i][x_vars] - sigma) / matrix[i][i];  
    }  
}
```

6.4 Вспомогательный модуль для метода Симпсона

```
double f(double phi, double tau, double t) {
    double l_r = (2 * cos(tau)) /
        (1 - sin(tau) * sin(tau) * cos(phi) * cos(phi));
    return (1 - exp(-t * l_r)) * cos(tau) * sin(tau);
}

double F(double phi, double tau, int M, double h_y) {
    double res = 0;
    for (int i = 0; i <= M / 2 - 1; i++) {
        res += f(phi, start + 2 * i * h_y, tau) +
            4 * f(phi, start + (2 * i + 1) * h_y, tau) +
            f(phi, start + (2 * i + 2) * h_y, tau);
    }

    res *= h_y / 3;
    return res;
}
```


7 Контрольные вопросы

7.1 В каких ситуациях теоретический порядок квадратурных формул численного интегрирования не достигается

Если подынтегральная функция не имеет соответствующих производных, то указанный теоретический порядок точности не достигается.

7.2 Построить формулу Гаусса численного интегрирования при одном узле

$$n = 1 \quad \left\{ \begin{array}{l} \sum_{i=1}^1 A_i = 2 \\ \sum_{i=1}^1 A_i t_i = 0 \end{array} \right.$$

Корень полинома Лежандра первой степени: $x = 0$

$$\left\{ \begin{array}{l} A_1 = 2 \\ A_1 * 0 = 0 \end{array} \right.$$

Решив систему, получим: $A_1 = 2$

В итоге:

$$\int_a^b f(x) dx = \frac{b-a}{2} (2 * f(\frac{b+a}{2}) + \frac{b-a}{2} * 0)$$

7.3 Построить формулу Гаусса численного интегрирования при двух узлах

$$n = 2 \quad \left\{ \begin{array}{l} \sum_{i=1}^2 A_i = 2 \\ \sum_{i=1}^2 A_i t_i = 0 \\ \sum_{i=1}^2 A_i t_i^2 = \frac{2}{3} \end{array} \right.$$

Корни полинома Лежандра второй степени: $x = \pm \sqrt{\frac{1}{3}}$

$$\begin{cases} A_1 + A_2 = 2 \\ A_1 * \sqrt{\frac{1}{3}} + A_2 * -\sqrt{\frac{1}{3}} = 0 \\ A_1 * \sqrt{\frac{1}{3}}^2 + A_2 * (-\sqrt{\frac{1}{3}})^2 = \frac{2}{3} \end{cases}$$

Решив систему, получим: $A_1 = A_2 = 1$

В итоге:

$$\int_a^b f(x)dx = \frac{b-a}{2} (1 * f(\frac{b+a}{2} + \frac{b-a}{2}\sqrt{\frac{1}{3}}) + 1 * f(\frac{b+a}{2} + \frac{b-a}{2}(-\sqrt{\frac{1}{3}})))$$

7.4 Получить обобщенную кубатурную формулу, аналогичную (6.6) из лекции №6, для вычисления двойного интеграла методом последовательного интегрирования на основе формулы трапеций с тремя узлами по каждому направлению

$$N = M = 3$$

$$\int_c^d \int_a^b f(x, y) dx dy == h_x (\frac{1}{2}F_0 + F_1 + \frac{1}{2}F_2 + F_3)$$

$$F_0 = \int_c^d f(x_0, y) dy = h_y (\frac{1}{2}f(x_0, y_0) + f(x_0, y_1) + \frac{1}{2}f(x_0, y_2) + f(x_0, y_3))$$

$$F_1 = \int_c^d f(x_1, y) dy = h_y (\frac{1}{2}f(x_1, y_0) + f(x_1, y_1) + \frac{1}{2}f(x_1, y_2) + f(x_1, y_3))$$

$$F_2 = \int_c^d f(x_2, y) dy = h_y (\frac{1}{2}f(x_2, y_0) + f(x_2, y_1) + \frac{1}{2}f(x_2, y_2) + f(x_2, y_3))$$

$$F_3 = \int_c^d f(x_3, y) dy = h_y (\frac{1}{2}f(x_3, y_0) + f(x_3, y_1) + \frac{1}{2}f(x_3, y_2) + f(x_3, y_3))$$

Таким образом, получаем обобщённую формулу:

$$\int_c^d \int_a^b f(x, y) dx dy == h_x * h_y (\frac{1}{4}(f(x_0, y_0) + f(x_0, y_2) + f(x_2, y_0) + f(x_2, y_2)) + \frac{1}{2}(f(x_0, y_1) + f(x_0, y_3) + f(x_2, y_1) + f(x_2, y_3) + f(x_1, y_0) + f(x_1, y_2) + f(x_3, y_0) + f(x_3, y_2)) + (f(x_1, y_1) + f(x_1, y_3) + f(x_3, y_1) + f(x_3, y_3)))$$