Федеральное государственное автономное
образовательное учреждение высшего образования
«Научно-образовательная корпорация ИТМО»

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия

Отчёт по лабораторной работе №3

По дисциплине «Вычислительная математика» (4 семестр)

Студент:
Дениченко Александр P3212
Практик:
Наумова Надежда Александровна

Санкт-Петербург
2024 г.

1 Цель работы

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

Вариант – 8

2 Вычислительная часть

1. Вычислить интеграл, приведенный в таблице 1, точно.
2. Вычислить интеграл по формуле Ньютона – Котеса при n = 6.
3. Вычислить интеграл по формулам средних прямоугольников, трапеций и Симпсона при n = 10 .
4. Сравнить результаты с точным значением интеграла.
5. Определить относительную погрешность вычислений для каждого метода.
6. В отчете отразить последовательные вычисления.

Интеграл по варианту:

$$\int_2^3 3x^3 - 2x^2 - 7x - 8$$

Точное вычисление интеграла:

$$\int_2^3 3x^3 - 2x^2 - 7x - 8 = \left(\frac{3x^4}{4} - \frac{2x^3}{3} - \frac{7x^2}{2} - 8x\right)\Big|_2^3 = \frac{243}{4} - \frac{301}{6} = \frac{127}{12} \approx 10.583$$

Вычисление по формуле Ньютона – Котеса:

Берём n=6, тогда коэффициенты Котеса для равноотстоящих узлов:

$$c_6^0 = c_6^6 = \frac{42(b-a)}{840}$$

$$c_6^1 = c_6^5 = \frac{216(b-a)}{840}$$

$$c_6^2 = c_6^4 = \frac{27(b-a)}{840}$$

$$c_6^3 = \frac{272(b-a)}{840}$$

Границы известны a=2; b=3:

$$c_6^0 = c_6^6 = \frac{42}{840}$$

$$c_6^1 = c_6^5 = \frac{216}{840}$$

$$c_6^2 = c_6^4 = \frac{27}{840}$$

$$c_6^3 = \frac{272}{840}$$

Найдём шаг разбиения:

$$h = \frac{3-2}{6} = \frac{1}{6}$$

Запишем определенный интеграл в виде:

$$\int_2^3 3x^3 - 2x^2 - 7x - 8 = c_6^0 \cdot f(a) + c_6^1 \cdot f(a + \frac{1}{6}) + c_6^2 \cdot f(a + \frac{2}{6}) + c_6^3 \cdot f(a + \frac{3}{6}) + c_6^4 \cdot f(a + \frac{4}{6}) + c_6^5 \cdot f(a + \frac{5}{6}) + c_6^6 \cdot f(b) =$$

$$= \frac{42}{840} \cdot f(2) + \frac{216}{840} \cdot f(2 + \frac{1}{6}) + \frac{27}{840} \cdot f(2 + \frac{2}{6}) + \frac{272}{840} \cdot f(2 + \frac{3}{6}) + \frac{27}{840} \cdot f(2 + \frac{4}{6}) + \frac{216}{840} \cdot f(2 + \frac{5}{6}) + \frac{42}{840} \cdot f(3) \approx 10.617$$

Относительная погрешность:

$$\epsilon = \frac{|10.617 - 10.583|}{10.583} \cdot 100 = 0.32\%$$

Вычисление по формуле прямоугольников со средними высотами:

По условия дано n = 10, тогда делим отрезок интегрирования на 10 равных частей по формуле:

$$h = \frac{b-a}{n} = \frac{3-2}{10} = 0.1$$

По формуле средних прямоугольников:

$$I = h \sum_{i=1}^{n} y_{i-\frac{1}{2}}$$

i	0	1	2	3	4	5	6	7	8	9	10
x_i	2	2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9	3
y_i	-6	-3.737	-1.136	1.821	5.152	8.875	13.008	17.569	22.576	28.047	34
$x_{i-1/2}$		2.05	2.15	2.25	2.35	2.45	2.55	2.65	2.75	2.85	2.95
$y_{i-1/2}$		-4.9096	-2.4799	0.2969	3.4386	6.9634	10.8891	15.2339	20.0156	25.2524	30.9621

Таблица 1: Приближенное вычисление интеграла методом средних прямоугольников

Подсчёт:

$$I = 0.1 \cdot (-4.9096 - 2.4799 + 0.2969 + 3.4386 + 6.9634 + 10.8891 + 15.2339 + 20.0156 + 25.2524 + 30.9621) = 10.8737$$

Относительная погрешность:

$$\epsilon = \frac{|10.8737 - 10.583|}{10.583} \cdot 100 = 2.75\%$$

Вычисление методом трапеций:

По условия дано n = 10, тогда делим отрезок интегрирования на 10 равных частей по формуле:

$$h = \frac{b-a}{n} = \frac{3-2}{10} = 0.1$$

По формуле трапеций:

$$I = h \cdot \left(\frac{y_0 + y_{10}}{2} + \sum_{i=1}^{n-1} y_i \right)$$

i	0	1	2	3	4	5	6	7	8	9	10
x_i	2	2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9	3
y_i	-6	-3.737	-1.136	1.821	5.152	8.875	13.008	17.569	22.576	28.047	34

Таблица 2: Приближенное вычисление интеграла методом трапеций

Подсчёт:

$$I = 0.1 \cdot \left(\frac{-6 + 43}{2} + (-6 - 3.737 - 1.136 + 1.821 + 5.152 + 8.875 + 13.008 + 17.569 + 22.576 + 28.047) \right) = 10.0175$$

Относительная погрешность:

$$\epsilon = \frac{|10.0175 - 10.583|}{10.583} \cdot 100 = 5.34\%$$

Вычисление методом Симпсона:

По условия дано n = 10, тогда делим отрезок интегрирования на 10 равных частей по формуле:

$$h = \frac{b-a}{n} = \frac{3-2}{10} = 0.1$$

По формуле Симпсона:

$$I = \frac{h}{3}(y_0 + 4 \cdot (y_1 + y_3 + y_5 + y_7 + y_9) + 2 \cdot (y_2 + y_4 + y_6 + y_8) + y_{10}) =$$

i	0	1	2	3	4	5	6	7	8	9	10
x_i	2	2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9	3
y_i	-6	-3.737	-1.136	1.821	5.152	8.875	13.008	17.569	22.576	28.047	34

Таблица 3: Приближенное вычисление интеграла методом Симпсона

$$= \frac{0.1}{3}(-6 + 4 \cdot (-3.737 + 1.821 + 8.875 + 17.569 + 28.047) + 2(-1.136 + 5.152 + 13.008 + 22.576) + 34) = 10.5833$$

Относительная погрешность:

$$\epsilon = \frac{|10.5833 - 10.583|}{10.583} \cdot 100 = 0.0028\%$$

3 Машинная реализация

Листинг 1: Первый узел валидации

```
1  private CalculateError isValidInterval(RequestFuncUser requestFuncUser) {
2      ArrayList<Double> kritPoints = new Functions().getErrPoints((int) requestFuncUser.
           getTypeFunc());
3      if(requestFuncUser.getA()>requestFuncUser.getB()){
4          double tmp = requestFuncUser.getA();
5          requestFuncUser.setA(requestFuncUser.getB());
6          requestFuncUser.setB(tmp);
7      }
8      Double a = requestFuncUser.getA();
9      Double b = requestFuncUser.getB();
10     Functions functions = new Functions();
11
12     if(a.equals(b)) {
13         return CalculateError.INCORRECT_BOUNDS_NULL_S;
14     }
15     for (Double point : kritPoints) {
16         logger.info("point > a: "+(point>a)+"; point<b: "+(point<b));
17         if (point.equals(a) && !Double.isNaN(functions.f_dx(a, (int) requestFuncUser.
               getTypeFunc()))
18                 && Double.isFinite(functions.f_dx(a, (int) requestFuncUser.getTypeFunc()))
                       {
19             requestFuncUser.setA(a + 0.0000001);
20         } else if (point.equals(b) && !Double.isNaN(functions.f_dx(b, (int) requestFuncUser.
               getTypeFunc()))
21                 && Double.isFinite(functions.f_dx(b, (int) requestFuncUser.getTypeFunc()))
                       {
```

```
22              requestFuncUser.setB(b - 0.0000001);
23          }else if (Double.isNaN(functions.f_dx(a, (int) requestFuncUser.getTypeFunc())) ||
24                  Double.isNaN(functions.f_dx(b, (int) requestFuncUser.getTypeFunc()))
25                  || Double.isInfinite(functions.f_dx(a, (int) requestFuncUser.getTypeFunc()))
26                  || Double.isInfinite(functions.f_dx(b, (int) requestFuncUser.getTypeFunc()))
                        ) {
27              return CalculateError.INCORRECT_BOUNDS_INT_ERR;
28          }
29          if(point>a && point<b && (Double.isNaN(functions.f_dx(point, (int) requestFuncUser.
                getTypeFunc()))
30                  || Double.isInfinite(functions.f_dx(point, (int) requestFuncUser.getTypeFunc
                        ())) )){
31              logger.warn("somnitelno␣no␣okey");
32          }
33      }
34      logger.info("f_dx(a):␣"+functions.f_dx(a, (int) requestFuncUser.getTypeFunc())+";␣f_dx(b
            ):␣"+functions.f_dx(b, (int) requestFuncUser.getTypeFunc()));
35      return null;
36  }
```

Листинг 2: Второй узел валидации

```
1   if(critInterval.isEmpty()){
2       mathMethod = getMethod((int) pointRequest.getMethod(), pointRequest);
3       mathMethod.calculate();
4       tmp = mathMethod.getAnswer().outAnswer();
5   }else {
6       double a = pointRequest.getA();
7       double b = pointRequest.getB();
8       double crit1 = critInterval.get(0);
9       double crit2 = critInterval.get(1);
10      if (a < crit1 && b > crit2) {
11          RequestFuncUser interval1 = pointRequest.clone();
12          interval1.setB(crit1);
13          logger.info(interval1.toString());
14
15          RequestFuncUser interval2 = pointRequest.clone();
16          interval2.setA(crit2);
17          logger.info(interval2.toString());
18
19          MathMethod mathMethod1 = getMethod((int) interval1.getMethod(), interval1);
20          MathMethod mathMethod2 = getMethod((int) interval2.getMethod(), interval2);
21
22
23          mathMethod1.calculate();
24          mathMethod2.calculate();
25          logger.info(mathMethod1.answerInfo.toString());
26          logger.info(mathMethod2.answerInfo.toString());
27
28          double e = mathMethod1.answerInfo.e;
29          double ans = mathMethod1.answerInfo.answer + mathMethod2.answerInfo.answer;
30          double r = (mathMethod1.answerInfo.r + mathMethod2.answerInfo.r)/2;
31          long n = mathMethod1.answerInfo.n + mathMethod2.answerInfo.n;
32          AnswerInfo answerInfo = new AnswerInfo(e, ans, ans, r, n);
33          if(Math.abs(interval1.getA()) == Math.abs(interval2.getB())){
34              answerInfo = new AnswerInfo(0, 0, 0, 0, 0);
35          }
```

```
36          tmp = answerInfo.outAnswer();
37      } else if (a >= crit1 && b <= crit2) {
38          AnswerInfo answerInfo = new AnswerInfo(0, 0, 0, 0, 0);
39          tmp = answerInfo.outAnswer();
40      } else if (a >= crit1 && a <= crit2 && b > crit2) {
41          RequestFuncUser interval1 = pointRequest.clone();
42          interval1.setA(crit2);
43          MathMethod mathMethod1 = getMethod((int) interval1.getMethod(), interval1);
44          mathMethod1.calculate();
45          logger.info(mathMethod1.answerInfo.toString());
46          tmp = mathMethod1.answerInfo.outAnswer();
47      } else if (a < crit1 && b <= crit2 && b >= crit1) {
48          RequestFuncUser interval1 = pointRequest.clone();
49          interval1.setB(crit1);
50          MathMethod mathMethod1 = getMethod((int) interval1.getMethod(), interval1);
51          mathMethod1.calculate();
52          logger.info(mathMethod1.answerInfo.toString());
53          tmp = mathMethod1.answerInfo.outAnswer();
54      } else if ((b < crit1)||(a > crit2)) {
55          mathMethod = getMethod((int) pointRequest.getMethod(), pointRequest);
56          mathMethod.calculate();
57          tmp = mathMethod.getAnswer().outAnswer();
58      } else {
59          AnswerInfo answerInfo = new AnswerInfo(0, 0, 0, 0, 0);
60          tmp = answerInfo.outAnswer();
61      }
62  }
```

Листинг 3: Метод средних прямоугольников

```
1   public void calculate() {
2       if (a > b) {
3           double tmp = a;
4           a = b;
5           b = tmp;
6       }
7
8       double step, sum, r = e + 1;
9       long n = 4;
10      double currAns = 0, prevAns = 0;
11
12      while (r > e){
13          step = (b - a) / n;
14          sum = 0;
15          for (int i = 0; i < n; i++) {
16              double aNew = a + step / 2 + step * i;
17              sum += functions.f(aNew, (int) number);
18          }
19          prevAns = currAns;
20          currAns = sum * step;
21          if (n > 4) {
22              r = Math.abs(currAns - prevAns) / 3.0;
23          }
24          n *= 2;
25          if(n > 1000000000){
26              break;
27          }
```

```
28      }
29      answerInfo = new AnswerInfo(e, currAns, prevAns, r, n / 2);
30      }
```

Листинг 4: Метод левых прямоугольников

```
1  public void calculate() {
2      if (a > b) {
3          double tmp = a;
4          a = b;
5          b = tmp;
6      }
7
8      double step, sum, r = e + 1;
9      long n = 4;
10     double currAns = 0, prevAns = 0;
11
12     while (r > e){
13         step = (b - a) / n;
14         sum = 0;
15         for (int i = 0; i < n; i++) {
16             double aNew = a + step * i;
17             sum += functions.f(aNew, (int) number);
18         }
19         prevAns = currAns;
20         currAns = sum * step;
21         if (n > 4) {
22             r = Math.abs(currAns - prevAns) / 3.0;
23         }
24         n *= 2;
25         if(n > 1000000000){
26             break;
27         }
28     }
29     answerInfo = new AnswerInfo(e, currAns, prevAns, r, n / 2);
30 }
```

Листинг 5: Метод правых прямоугольников

```
1  public void calculate() {
2      if (a > b) {
3          double tmp = a;
4          a = b;
5          b = tmp;
6      }
7
8      double step, sum, r = e + 1;
9      long n = 4;
10     double currAns = 0, prevAns = 0;
11
12     while (r > e){
13         step = (b - a) / n;
14         sum = 0;
15         for (int i = 1; i <= n; i++) {
16             double aNew = a + step * i;
17             sum += functions.f(aNew, (int) number);
18         }
```

```
19          prevAns = currAns;
20          currAns = sum * step;
21          if (n > 4) {
22              r = Math.abs(currAns - prevAns) / 3.0;
23          }
24          n *= 2;
25          if(n > 1000000000){
26              break;
27          }
28      }
29      answerInfo = new AnswerInfo(e, currAns, prevAns, r, n / 2);
30  }
```

Листинг 6: Метод Симпсона

```
1   public void calculate() {
2       if (a > b) {
3           double tmp = a;
4           a = b;
5           b = tmp;
6       }
7
8       double aNew = a, step, sum = 0, r = e + 1;
9       long n = 4;
10      double y0 = functions.f(a, (int) number);
11      double yn = functions.f(b, (int) number);
12      double currAns = 0, prevAns = 0;
13
14      while (r >= e) {
15          step = (b - a) / n;
16          sum = 0;
17          aNew = a + step;
18          for (int i = 1; i < n; i++) {
19              if (i % 2 == 0) {
20                  sum += 2 * functions.f(aNew, (int) number);
21              } else {
22                  sum += 4 * functions.f(aNew, (int) number);
23              }
24              aNew += step;
25          }
26          prevAns = currAns;
27          currAns = step / 3 * (y0 + sum + yn);
28          if (n > 4) {
29              r = Math.abs(currAns - prevAns) / 15.0;
30          }
31          n *= 2;
32          if(n > 1000000000){
33              break;
34          }
35      }
36
37      answerInfo = new AnswerInfo(e, currAns, prevAns, r, n / 2);
38  }
```

Листинг 7: Метод Трапеций

```
1   public void calculate() {
```

```
2       if (a > b) {
3           double tmp = a;
4           a = b;
5           b = tmp;
6       }
7
8       double aNew = a, step, sum = 0, r = e + 1;
9       long n = 4;
10      double y0 = functions.f(a, (int) number);
11      double yn = functions.f(b, (int) number);
12      double currAns = 0, prevAns = 0;
13
14      while (r > e) {
15          step = (b - a) / n;
16          sum = 0;
17          aNew = a + step;
18          for (int i = 1; i < n; i++) {
19              sum += functions.f(aNew, (int) number);
20              aNew += step;
21          }
22          prevAns = currAns;
23          currAns = step * ((y0 + yn) / 2 + sum);
24          if (n > 4) {
25              r = Math.abs(currAns - prevAns) / 3.0;
26          }
27          n *= 2;
28          if(n > 1000000000){
29              break;
30          }
31      }
32      answerInfo = new AnswerInfo(e, currAns, prevAns, r, n / 2);
33  }
```

4 Пример работы программы

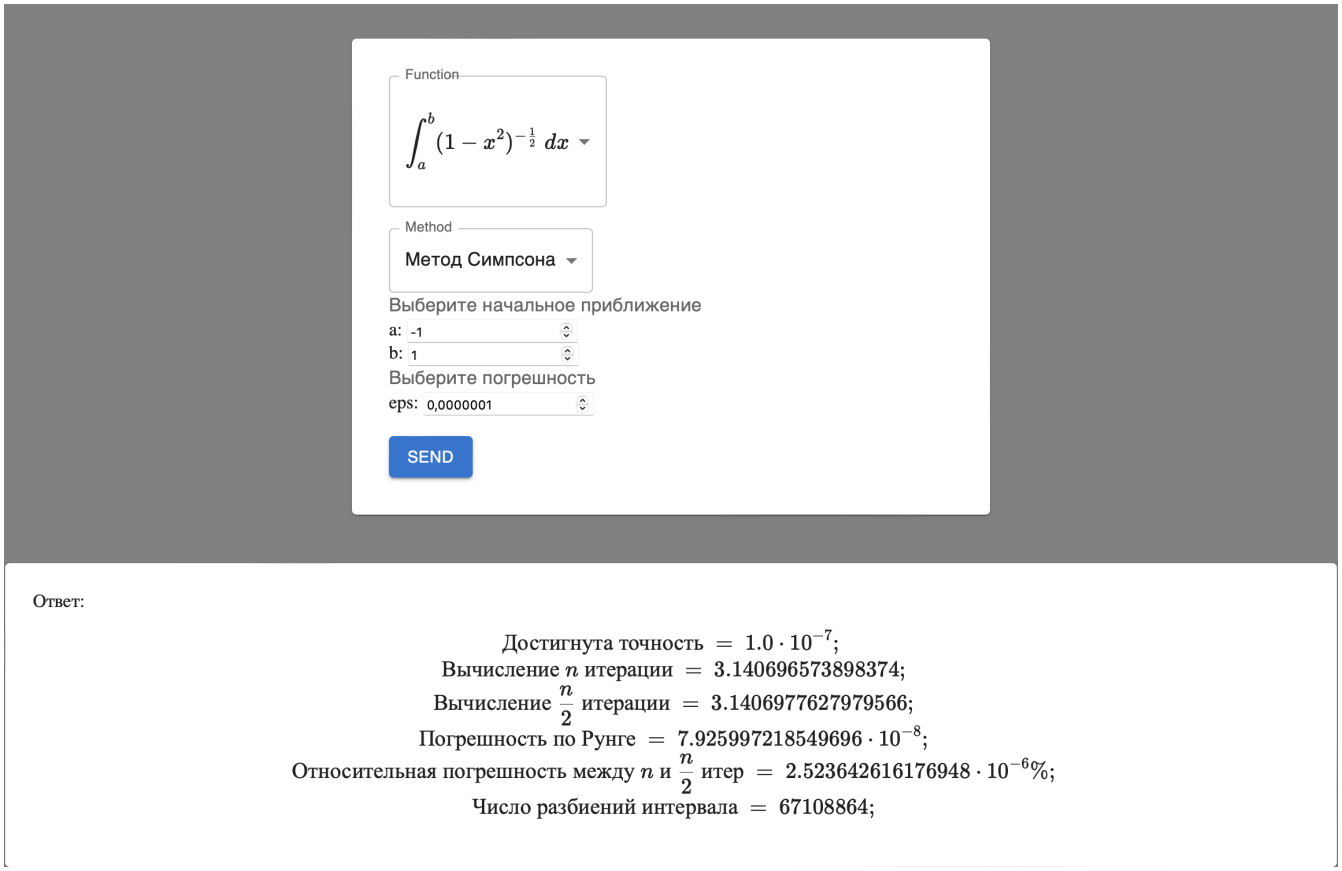

Рис. 1: Front: React

```
2024-03-20 00:14:31,143 INFO  [main] o.s.b.StartupInfoLogger: Starting Web4Application using Java 20.0.1 with PID 36664 (/Users/alexalex/Desktop/cm_math/lab3/cm3/tar
2024-03-20 00:14:31,145 INFO  [main] o.s.b.SpringApplication: No active profile set, falling back to 1 default profile: "default"
2024-03-20 00:14:31,514 INFO  [main] o.s.b.w.e.t.TomcatWebServer: Tomcat initialized with port(s): 8080 (http)
2024-03-20 00:14:31,524 INFO  [main] o.a.j.l.DirectJDKLog: Initializing ProtocolHandler ["http-nio-8080"]
2024-03-20 00:14:31,526 INFO  [main] o.a.j.l.DirectJDKLog: Starting service [Tomcat]
2024-03-20 00:14:31,526 INFO  [main] o.a.j.l.DirectJDKLog: Starting Servlet engine: [Apache Tomcat/10.1.15]
2024-03-20 00:14:31,565 INFO  [main] o.a.j.l.DirectJDKLog: Initializing Spring embedded WebApplicationContext
2024-03-20 00:14:31,566 INFO  [main] o.s.b.w.s.c.ServletWebServerApplicationContext: Root WebApplicationContext: initialization completed in 402 ms
2024-03-20 00:14:31,692 INFO  [main] o.a.j.l.DirectJDKLog: Starting ProtocolHandler ["http-nio-8080"]
2024-03-20 00:14:31,700 INFO  [main] o.s.b.w.e.t.TomcatWebServer: Tomcat started on port(s): 8080 (http) with context path ''
2024-03-20 00:14:31,704 INFO  [main] o.s.b.StartupInfoLogger: Started Web4Application in 0.783 seconds (process running for 1.482)
2024-03-20 00:15:24,759 INFO  [http-nio-8080-exec-1] o.a.j.l.DirectJDKLog: Initializing Spring DispatcherServlet 'dispatcherServlet'
2024-03-20 00:15:24,761 INFO  [http-nio-8080-exec-1] o.s.w.s.FrameworkServlet: Initializing Servlet 'dispatcherServlet'
2024-03-20 00:15:24,765 INFO  [http-nio-8080-exec-1] o.s.w.s.FrameworkServlet: Completed initialization in 4 ms
2024-03-20 00:15:24,842 INFO  [http-nio-8080-exec-2] c.e.w.c.AttemptController: Функция: 3.0; Метод: 5.0; Граница левая: -1.0; Граница правая: 1.0; Точность: 1.0E-7;
2024-03-20 00:15:24,843 INFO  [http-nio-8080-exec-2] c.e.w.v.DataValidation: point > a: false; point<b: true
2024-03-20 00:15:24,843 INFO  [http-nio-8080-exec-2] c.e.w.v.DataValidation: point > a: true; point<b: false
2024-03-20 00:15:24,843 INFO  [http-nio-8080-exec-2] c.e.w.v.DataValidation: f_dx(a): -1.5707963267948966; f_dx(b): 1.5707963267948966
2024-03-20 00:15:24,843 INFO  [http-nio-8080-exec-2] c.e.w.m.m.SimpsonMethod: Метод Симпсона
2024-03-20 00:15:32,545 INFO  [http-nio-8080-exec-2] c.e.w.m.AnswerInfo: т: 1.0E-7; ans: 3.140696573898374; abs: 7.925997218549696E-8; count: 67108864
2024-03-20 00:15:32,546 INFO  [http-nio-8080-exec-2] c.e.w.c.AttemptController: Успешно
```

Рис. 2: Backend: String logs

5 Диаграммы

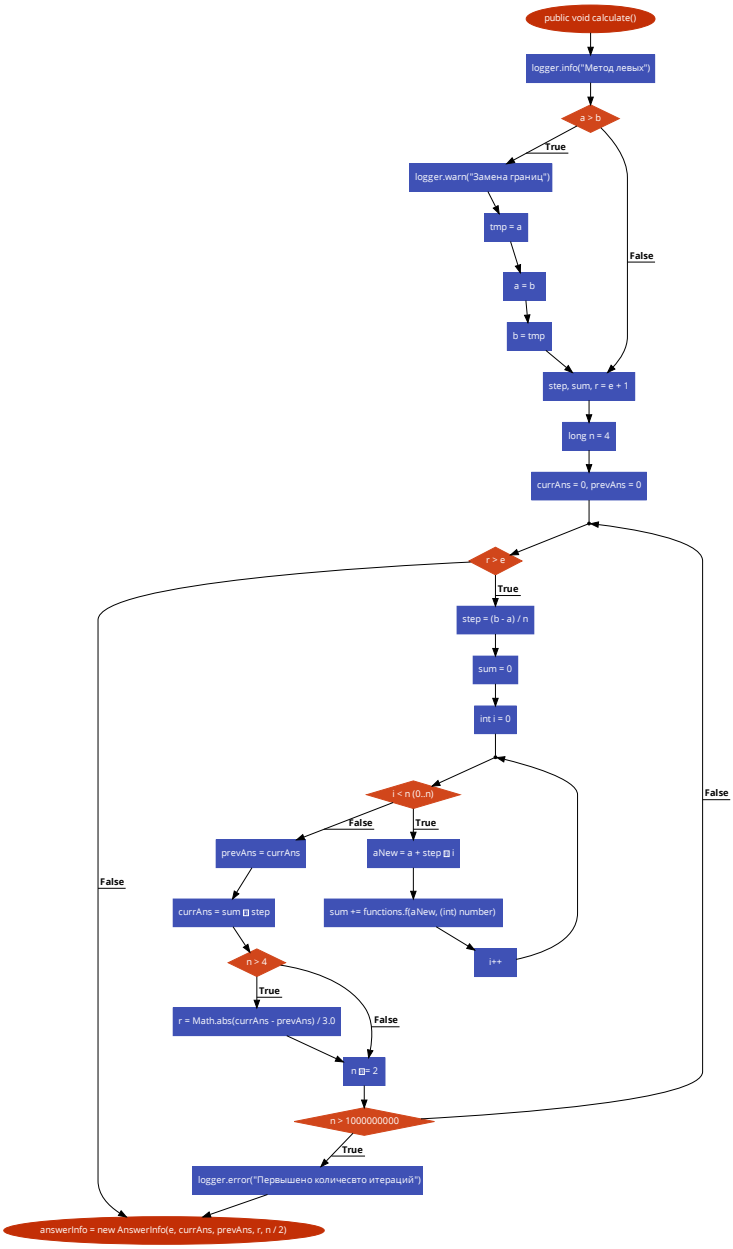

Рис. 3: Левые прямоугольники

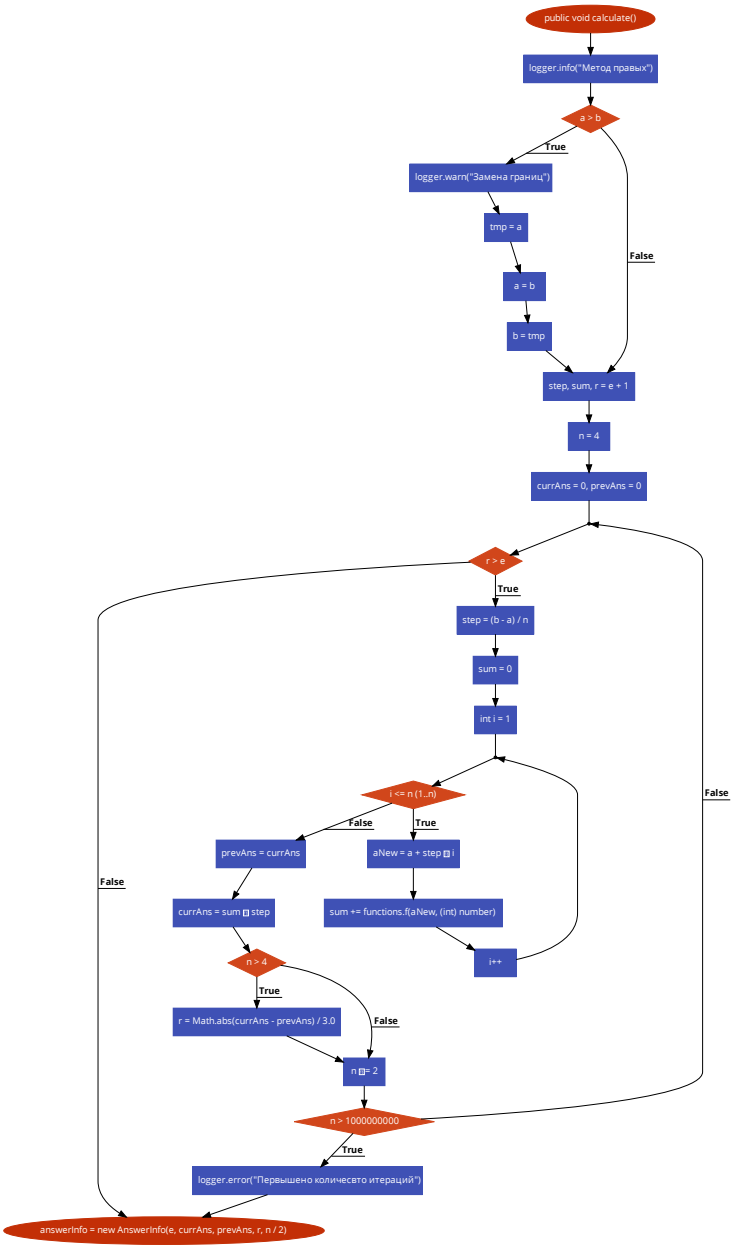

Рис. 4: Правые прямоугольники

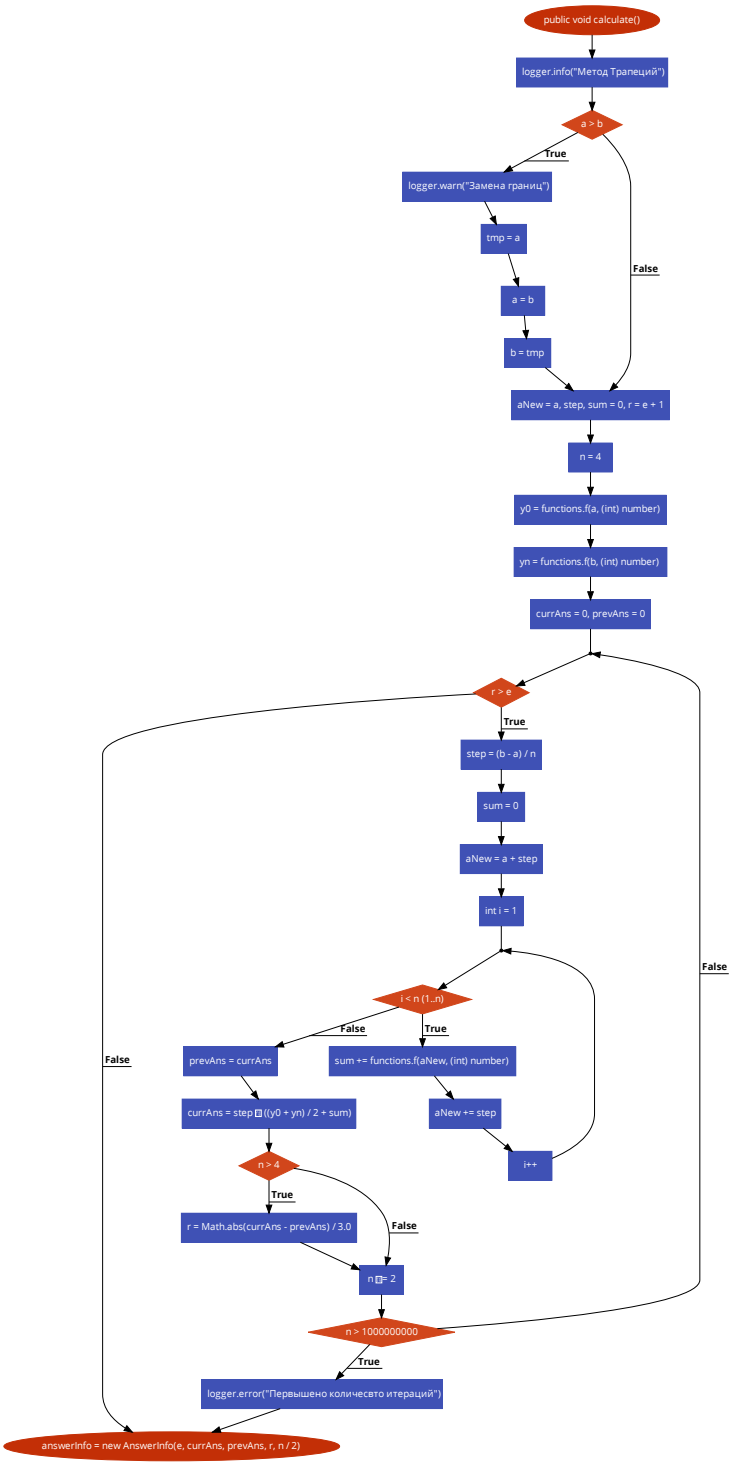

Рис. 5: Трапеции

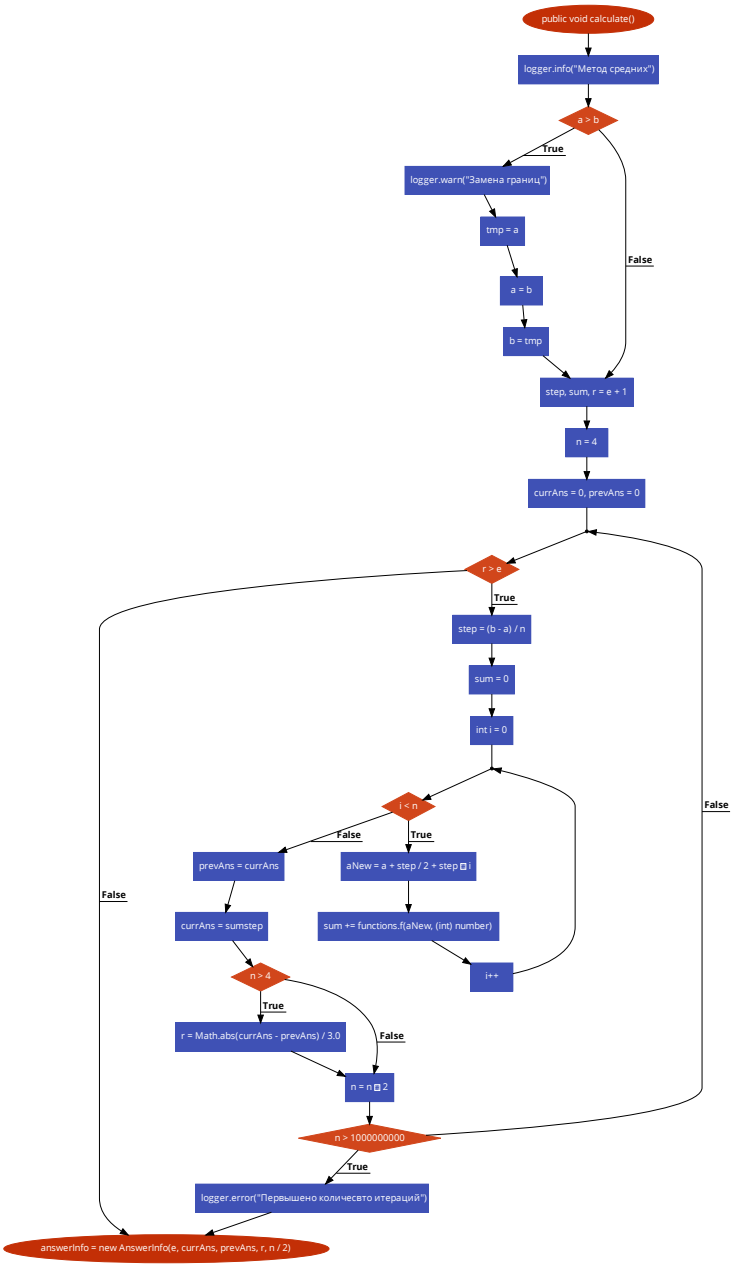

Рис. 6: Центральные прямоугольники

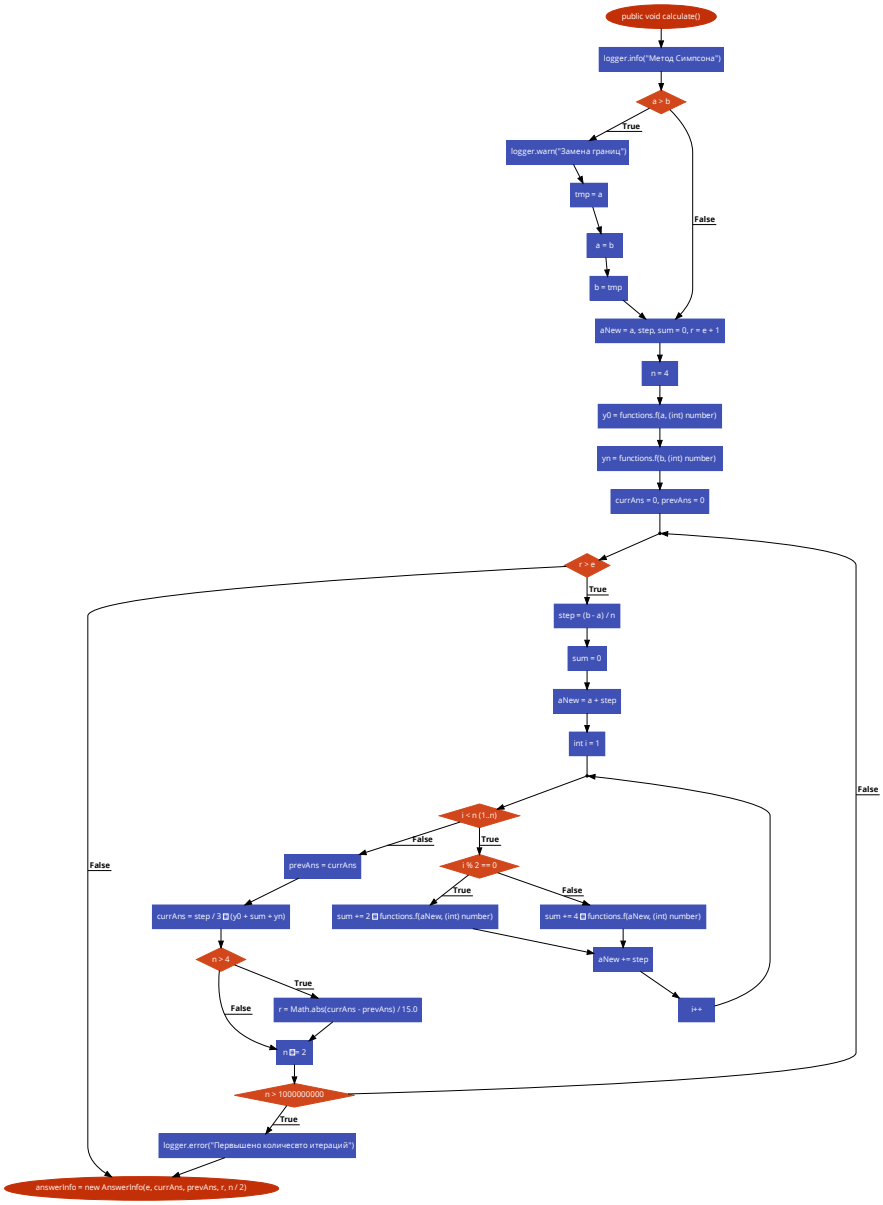

Рис. 7: Симпсон

6 GitHub

Ссылка на мой репозиторий на GitHub: `https://github.com/Alex-de-bug/cm_math/tree/main/lab3`.

7 Вывод

При работе были изучены несколько численных методов вычисления интегралов, написаны несколько алгоритмов для реализации. Подсчёт руками помог усвоить все тонкости работы численных методов. Придумана обработка точек разрыва в том числе неустранимых 2го рода.