“КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ”

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по семестровой работе по дисциплине “Численные методы” на тему

“Интерполирование трансцендентных функций. Вычисление интеграла с помощью квадратурной формы.”

Вариант 29

Работу выполнил:

Студент группы 09-163

Голиков Д. Я.

Работу проверила:

Гнеденкова В. Л.

Казань, 2023

СОДЕРЖАНИЕ

Постановка задачи

Одна из специальный функций математической физики - интеграл Френеля, определяется следующим образом:

$$C(x) = \int_0^x \cos\left(\frac{\pi t^2}{2}\right) dt$$

Цель задания - изучить и сравнить различные способы приближенного вычисления этой функции.

Для этого:

1. Протабулировать $C(x)$ на отрезке $[a, b]$ с шагом h с точностью ε, основываясь на ряде Тейлора, предварительно вычислив его:

$$C(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \left(\frac{\pi}{2}\right)^2}{(2n)!\,(4n+1)} x^{4n+1}$$

Где $a = 0$, $b = 1.5$, $h = 0.15$, $\varepsilon = 10^{-6}$, и получить таким образом таблицу:

x_0	x_1	x_2	...	x_n
f_0	f_1	f_2	...	f_n

$$f_i = C(x_i),\ x_i = a + i \cdot h,\ i = 0,\ \dots,\ n$$

2. По полученной таблице значений построить интерполяционный полином Лагранжа:

$$L_n(x) = \sum_{i=0}^{n} f(x_i) \prod_{i \neq j,\ j=0}^{n} \frac{x - x_j}{x_i - x_j}$$

И вычислить погрешности интерполирования

$$\varepsilon(x) = |C(x) - L_n(x)|,\ \varepsilon = \max_{x \in [a,b]} \varepsilon(x)$$

В качестве узлов интерполяции взять:

1) Равномерно распределенные узлы (как в задании 1)

2) Чебышевские узлы интерполяции, вычислить по формуле:

$$x_i = \cos\left(\frac{2i+1}{2n+2}\pi\right) + \frac{a+b}{2},\ i = 0,\ \ldots,n$$

3. Выяснить зависимость максимальной погрешности интерполирования от числа узлов интерполяции.

4. На сетке узлов $\{x_i\}$, где $x_i = a + i \cdot h,\ i = 0,\ \ldots,\ n,\ h = \frac{b-a}{n}$, построить таблицу приближенных значений $C(x)$, используя составные квадратурные формулы

1) Прямоугольников (правых, левых, центральных);

2) Трапеций;

3) Симпсона;

4) Гаусса;

ЧАСТЬ 1

Первой частью решения задания является функция табуляции ряда. Для правильной работы данной функции мы используем следующий алгоритм:

Нахождение некоторой величины q_n, которая вычисляется следующим образом:

$$q_n(x) = \frac{a_{n+1}(x)}{a_n(x)}$$

Данная величина позволяет нам избежать вычисление факториала в ряде Тейлора.

В моем варианте

$$q_n = -\frac{(4i+1)\cdot\left(\frac{\pi}{2}\right)^2\cdot x^4}{(2i+1)\cdot(2i+2)\cdot(4i+5)}$$

$$a_{n+1}(x) = a_n(x)\cdot q_n(x)$$

Значение рядя вычисляется с точностью ε , т. е. до тех пор, пока разница между двумя

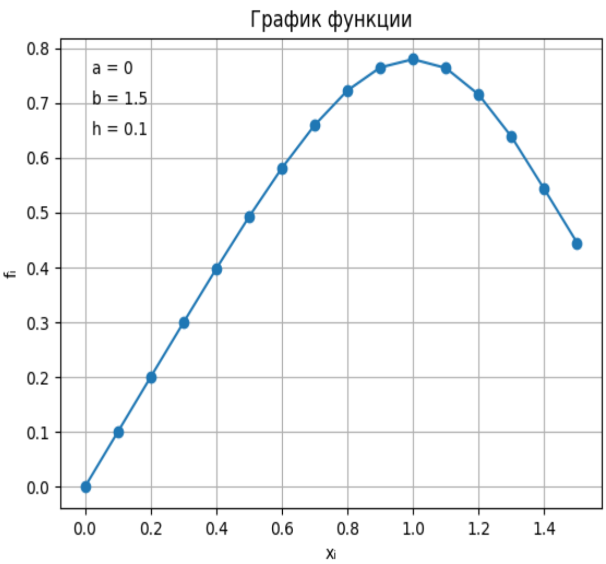
членами ряда $|a_{n+1} - a_n| \geq \varepsilon$

Рассмотрим результат табулирования функции на отрезке $[0{,}1.5]$ при различных условиях.

1) Равномерно распределенные узлы.

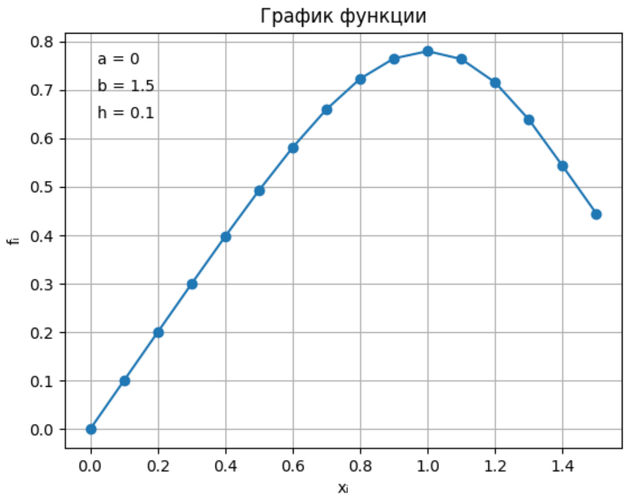
Рассмотрим случай, где $\varepsilon = 10^{-6}$. Количество узлов - 16

xi	fi
0.0	0.0
0.1	0.09999753263
0.2	0.1999210576
0.3	0.2994009763
0.4	0.3974807592
0.5	0.4923442255
0.6	0.5810954470
0.7	0.6596523525
0.8	0.7228441717
0.9	0.7648230198
1.0	0.7798934005
1.1	0.7638066694
1.2	0.7154377219
1.3	0.6385504432
1.4	0.5430957866
1.5	0.4452611748



Попробуем уменьшить погрешность до 10^{-2}

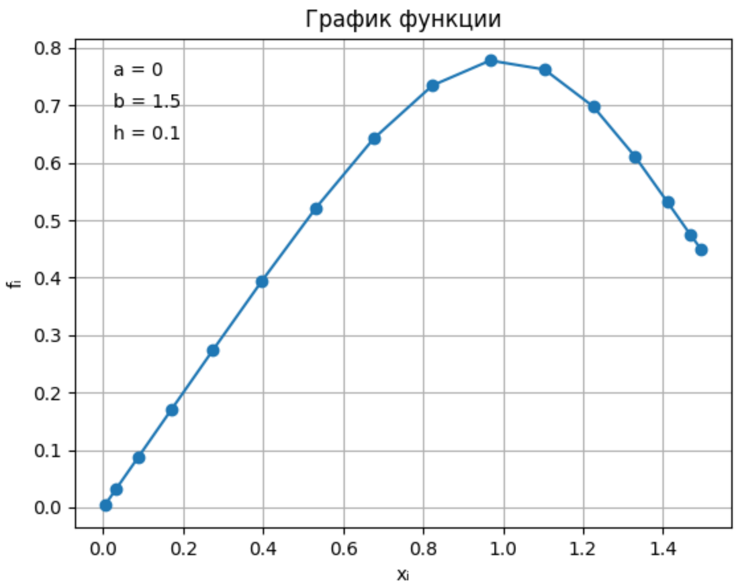
xi	fi
0.0	0.0
0.1	0.09999753260
0.2	0.1999210432
0.3	0.2994004215
0.4	0.3974733813
0.5	0.4922893716
0.6	0.5810975339
0.7	0.6596677763
0.8	0.7229311946
0.9	0.7648141331
1.0	0.7798405077
1.1	0.7635420301
1.2	0.7154911785
1.3	0.6388340918
1.4	0.5430146680
1.5	0.4448122663



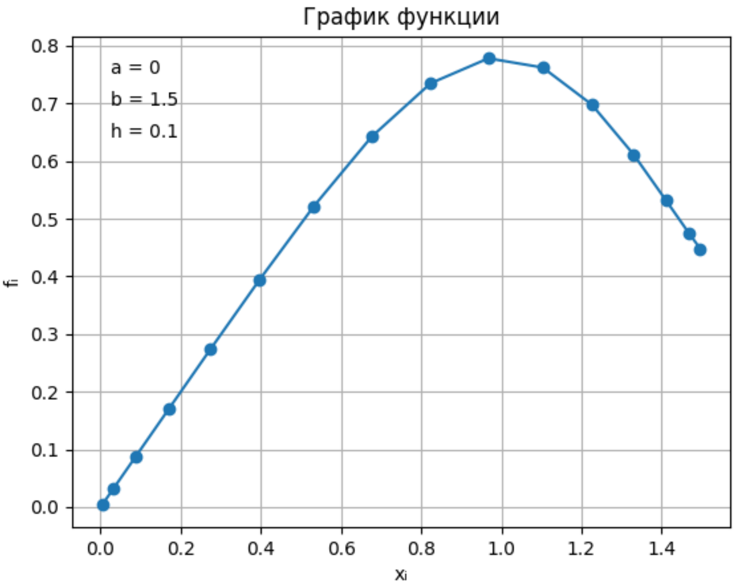
Видно, что для данного примера существенное уменьшение точности вычисления пр отражается на графике (различие в среднем начинается с 5-го знака после запятой)

2) Чебышевские узлы интерполяции (с аналогичной точностью)

xi	fi
0.0036114550	0.003611455000
0.0322947483	0.03229473963
0.0885590519	0.08855770789
0.1702421598	0.1702068794
0.2742050367	0.2738227965
0.3964524473	0.3940427100
0.5322864921	0.5218396625
0.6764871449	0.6423562728
0.8235128555	0.7348433395
0.9677135076	0.7782749385
1.103547552	0.7626369898
1.225794963	0.6980491841
1.329757840	0.6114657471
1.411440948	0.5316641438
1.467705252	0.4759163510
1.496388545	0.4486093136



```
xi         fi
0.0036114550 0.0036114550
0.0322947483 0.03229473963
0.0885590519 0.08855770788
0.1702421598 0.1702068760
0.2742050367 0.2738225495
0.3964524473 0.3940359002
0.5322864921 0.5218401044
0.6764871449 0.6423661752
0.8235128555 0.7349699430
0.9677135076 0.7782445891
1.103547552 0.7623575770
1.225794963 0.6981325017
1.329757840 0.6119201028
1.411440948 0.5315648643
1.467705252 0.4756545570
1.496388545 0.4481864264
```



Видно, что в обоих случаях значения на краях отрезка почти не изменилась, но ближе к середине отрезка различия начинаются уже с 3-го значащего знака

ЧАСТЬ 2.

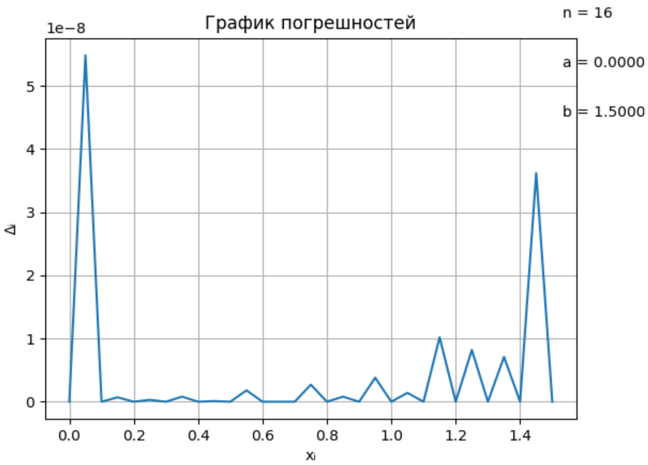
Второй частью решения задания является построение интерполяционного полинома Лагранжа.

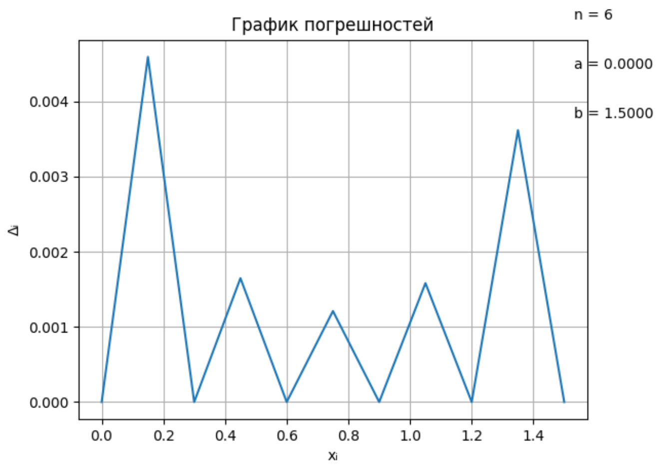
В первую очередь, построим таблицу с равномерно распределенными узлами. Возьмем количество узлов, равное 6, а размер полинома Лагранжа - 16, 6 из которых совпадают с узлами, а остальные получаются путем их равномерного распределения между узлами.

Построим таблицы. Слева- таблица, посчитанная по ряду Тейлора из Части 1, справа - значения интерполяционного полинома Лагранжа.

xi	fi	fi
0.0	0.0	0.0
0.1	0.09999753263	0.1051787158
0.2	0.1999210576	0.2030716994
0.3	0.2994009763	0.2994009763
0.4	0.3974807592	0.3958800549
0.5	0.4923442255	0.4910466573
0.6	0.5810954470	0.5810954470
0.7	0.6596523525	0.6607107590
0.8	0.7228441717	0.7238993283
0.9	0.7648230198	0.7648230198
1.0	0.7798934005	0.7786315563
1.1	0.7638066694	0.7622952474
1.2	0.7154377219	0.7154377219
1.3	0.6385504432	0.6411686511
1.4	0.5430957866	0.5469164855
1.5	0.4452611748	0.4452611748

Видно, что значения в узлах совпадают, но при сравнении узлов в других точках есть некоторая погрешность. Посмотрим на графики:





Сейчас можно увидеть закономерность:

1) Погрешность в узлах равно 0

2) Погрешность на краях отрезка при равномерном распределении узлов резко возрастает, и чем больше узлов, тем больше это заметно, хотя при 6 или 16 узлах (как на графиках) эта погрешность довольно мала.

3) При отдалении от краев отрезка погрешность уменьшается

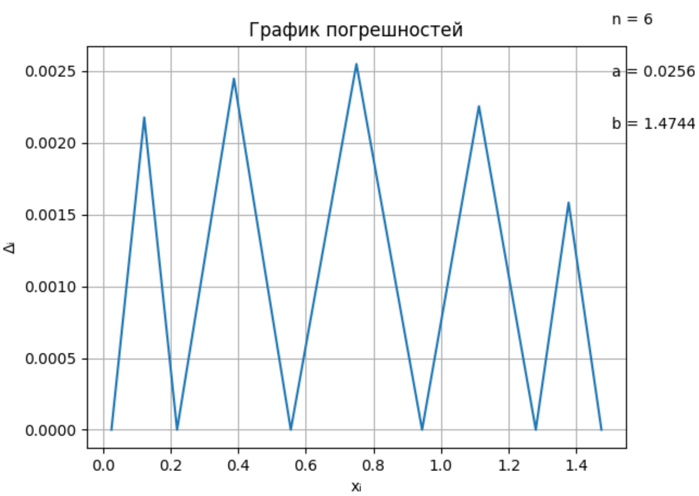
Для Чебышевских узлов интерполяции таблица будет выглядеть следующим образом:

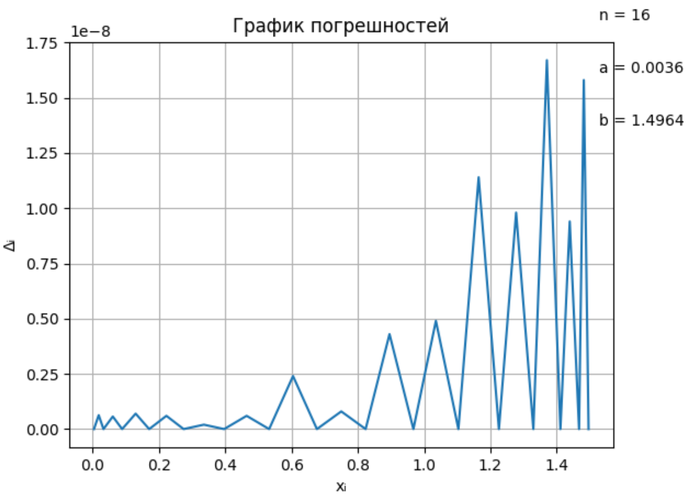
6 узлов являются Чебышевскими (посчитаны по формуле)

Между каждым таким узлом равномерно распределено ещё два узла

Сначала показаны значения функции, посчитанные по ряду Тейлора, потом значения, полученные по Интерполяционному полиному Лагранжа

x	fi	fi
0.0255556302	0.02555562751	0.02555562751
0.09026039153	0.09025891337	0.09251315451
0.1549651529	0.1549431043	0.1565899002
0.2196699142	0.2195437381	0.2195437381
0.3317418481	0.3307518367	0.3285168764
0.4438137820	0.4395839908	0.4375545121
0.5558857159	0.5429309299	0.5429309299
0.6852952385	0.6489300021	0.6511478819
0.8147047611	0.7304907566	0.7326928050
0.9441142838	0.7750906194	0.7750906194
1.056186218	0.7748554444	0.7729409513
1.168258152	0.7341747081	0.7321765678
1.280330086	0.6555479221	0.6555479221
1.345034847	0.5970254366	0.5982778356
1.409739608	0.5333652598	0.5349230149
1.474444370	0.4694000556	0.4694000556





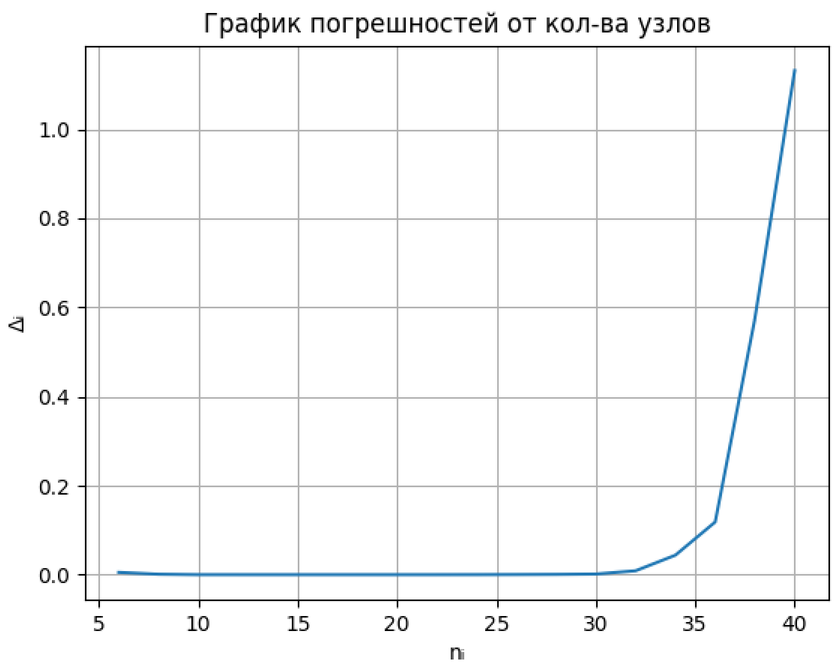
Сейчас мы уже не можем увидеть четких закономерностей, а из следующих частей более наглядно будет представлена зависимость погрешности от количества узлов в обоих случаях.

ЧАСТЬ 3

Выявим зависимость максимальной погрешности интерполирования от количества узлов интерполяции. Рассмотрим таблицы, в которые внесем значения максимальной погрешности для каждого n.

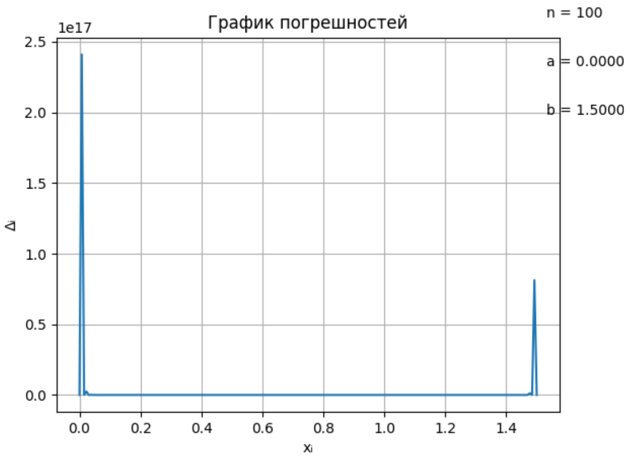
Сначала рассмотрим равномерное распределение узлов:

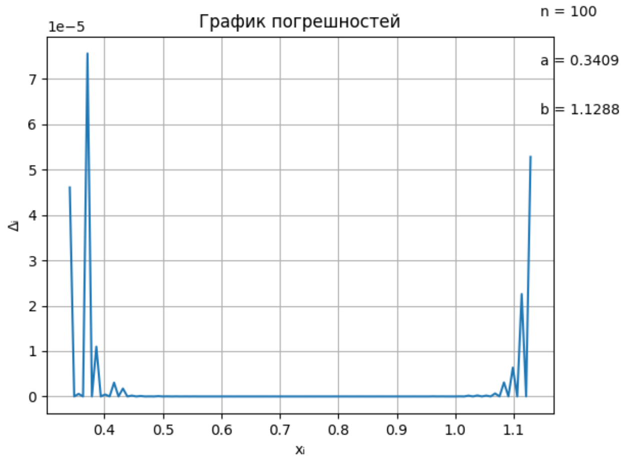
n	max_er
6,	0.00518118317
8,	0.0010311313
10,	0.00002931489
12,	0.0000076155
14,	3.031E-7
16,	1.403E-7
18,	0.0000010823
20,	0.0000069558
22,	0.0000141850
24,	0.0000963660
26,	0.0003118123
28,	0.00065342984
30,	0.00142047032
32,	0.0087612725
42,	3.728765881
44,	5.392088254
46,	75.88861563
48,	409.1318889
50,	702.0808770



При последующем увеличении n максимальная погрешность будет стремительно расти. Можно заметить, что примерно до 30 узлов максимальная погрешность практически не меняется, но потом начинает резко возрастать.

Теперь посмотрим на график погрешности для 100 узлов. Очевидно, что максимальна погрешность будет очень большой, в моем случае около $2.4 \cdot 10^{17}$

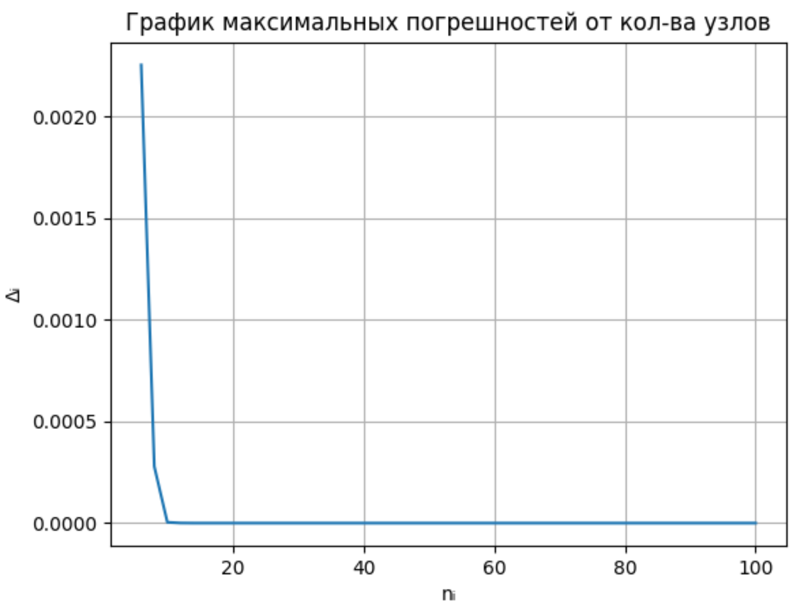




Однако, если взять участок $[a, b]$, $a \approx 0.34$, $b \approx 1.129$, то максимальная погрешность не превысит 10^{-4}.

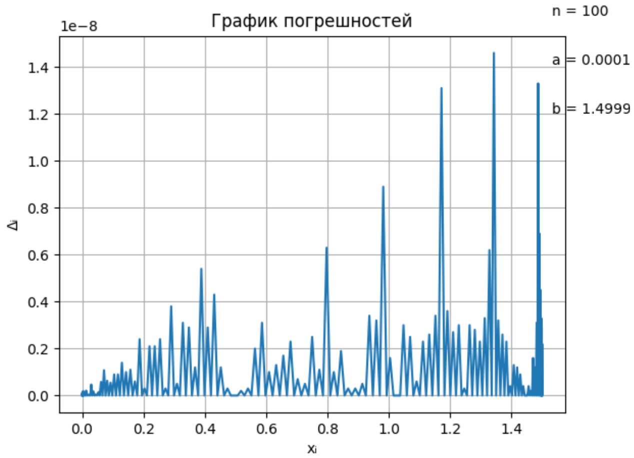
Теперь посмотрим на ту же зависимость при распределении узлов, используя полином Чебышева:

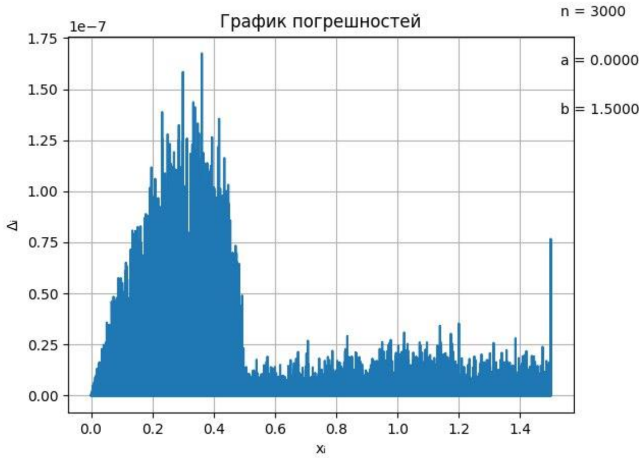
```
n    max_er
 6, 0.00225424114
16, 2.44E-8
26, 2.03E-8
36, 1.81E-8
46, 1.99E-8
56, 2.90E-8
66, 1.99E-8
76, 2.17E-8
86, 2.26E-8
96, 2.09E-8
```



Во втором случае погрешность быстро уменьшается и даже при довольно большом количестве узлов очень мала (около $2 \cdot 10^{-8}$)

Так как сложность вычисления интерполяционного полинома Лагранжа как минимум квадратична, слишком большое количество узлов потребует много времени для вычисления, поэтому будет сложно найти такое n, при котором погрешность во втором случае начнет резко увеличиваться.





Из графиков видно, что даже при $n = 3000$ погрешность не превышает $2 \cdot 10^{-7}$. Вычисление последнего графика заняло более двух часов.

ЧАСТЬ 4

Четвертой частью решения является вычисление различных квадратурных формул.

Для вычисления приближенного значения интегралов будем делить отрезок на n частей и при каждой итерации увеличивать n в 2 раза. Для каждого отрезка будем считать значение интеграла по формуле, после чего складывать. Расчеты будем вести до тех пор, пока $|S_{2n} - S_n| > \varepsilon$, $\varepsilon = 10^{-4}$

Квадратурная формула прямоугольников

Формула левых прямоугольников:

$$\int_a^b f(x)dx \approx f(a)(b-a)$$

Формула правых прямоугольников:

$$\int_a^b f(x)dx \approx f(b)(b-a)$$

Формула центральных прямоугольников

$$\int_a^b f(x)dx \approx f\left(\frac{b-a}{2}\right)(b-a)$$

Построим таблицу для левых прямоугольников:

C(x)	erf(x)	err	n
0.000000000000	0.000000000000	0.0000000000	0
0.099997532630	0.099998819310	0.0000012867	8
0.199921057600	0.199962221300	0.0000411637	8
0.299400976000	0.299490615100	0.0000896391	32
0.397480759200	0.397577919600	0.0000971604	128
0.492344225900	0.492418370400	0.0000741445	512
0.581095447000	0.581186544600	0.0000910976	1024
0.659652351800	0.659748657000	0.0000963052	2048
0.722844171800	0.722934803400	0.0000906316	4096
0.764823021200	0.764900681800	0.0000776606	8192
0.779893400300	0.779954359500	0.0000609592	16384
0.763806665900	0.763895365100	0.0000886992	16384
0.715437722700	0.715497968900	0.0000602462	32768
0.638550454500	0.638624627900	0.0000741734	32768
0.543095783000	0.543181598800	0.0000858158	32768
0.445261176200	0.445348017000	0.0000868408	32768

Из таблицы видно, что для этих формулы, чтобы добиться требуемой точности, нужно сделать довольно много разделений этого отрезка.

Построим таблицу для правых прямоугольников:

C(x)	erf(x)	err	n
0.000000000000	0.000000000000	0.0000000000	0
0.099997532630	0.099995735120	0.0000017975	8
0.199921057600	0.199863557700	0.0000574999	8
0.299400976000	0.299353239700	0.0000477363	64
0.397480759200	0.397431416500	0.0000493427	256
0.492344225900	0.492269697600	0.0000745283	512
0.581095447000	0.581004116200	0.0000913308	1024
0.659652351800	0.659555969600	0.0000963822	2048
0.722844171800	0.722753485500	0.0000906863	4096
0.764823021200	0.764745563700	0.0000774575	8192
0.779893400300	0.779832289100	0.0000611112	16384
0.763806665900	0.763717593100	0.0000890728	16384
0.715437722700	0.715378040500	0.0000596822	32768
0.638550454500	0.638475158500	0.0000752960	32768
0.543095783000	0.543010869600	0.0000849134	32768
0.445261176200	0.445171880300	0.0000892959	32768

Ситуация практически не изменилась. Для достаточной точности требуется довольно много разделений.

Теперь построим таблицу для центральных прямоугольников:

C(x)	erf(x)	err	n
0.000000000000	0.000000000000	0.0000000000	0
0.099997532630	0.099997659730	0.0000001271	8
0.199921057600	0.199925122400	0.0000040648	8
0.299400976000	0.299431766300	0.0000307903	8
0.397480759200	0.397513229100	0.0000324699	16
0.492344225900	0.492368671300	0.0000244454	32
0.581095447000	0.581110240600	0.0000147936	64
0.659652351800	0.659682862800	0.0000305110	64
0.722844171800	0.722857987300	0.0000138155	128
0.764823021200	0.764845289200	0.0000222680	128
0.779893400300	0.779925360400	0.0000319601	128
0.763806665900	0.763816727000	0.0000100611	256
0.715437722700	0.715448360800	0.0000106381	256
0.638550454500	0.638583322400	0.0000328679	128
0.543095783000	0.543118017300	0.0000222343	64
0.445261176200	0.445250857300	0.0000103189	256

Можно заметить, что в последнем случае погрешность немного уменьшилась, а для достижения требуемой точности потребовалось намного меньше делений отрезка.

Квадратурная формула трапеций

$$\int_a^b f(x)dx \approx \frac{(b-a)}{2}\big(f(a)+f(b)\big).$$

Составим таблицу и посмотрим на различия:

C(x)	erf(x)	err	n
0.000000000000	0.000000000000	0.0000000000	0
0.099997532630	0.099997277240	0.0000002554	8
0.199921057600	0.199912889500	0.0000081681	8
0.299400976000	0.299385437900	0.0000155381	16
0.397480759200	0.397464488300	0.0000162709	32
0.492344225900	0.492331997000	0.0000122289	64
0.581095447000	0.581065859000	0.0000295880	64
0.659652351800	0.659637095600	0.0000152562	128
0.722844171800	0.722816541800	0.0000276300	128
0.764823021200	0.764811887600	0.0000111336	256
0.779893400300	0.779877420800	0.0000159795	256
0.763806665900	0.763786544200	0.0000201217	256
0.715437722700	0.715416447300	0.0000212754	256
0.638550454500	0.638534026600	0.0000164279	256
0.543095783000	0.543084755900	0.0000110271	128
0.445261176200	0.445281813900	0.0000206377	256

Мы получили похожую ситуацию с формулой центральных прямоугольников. Погрешность ниже, чем в первых формулах прямоугольника и нам не нужно слишком много разделений отрезка для достижения нужной точности.

Квадратурная формула Симпсона

$\int_a^b f(x)dx \approx \frac{(b-a)}{6}\big(f(a) + 4f(c) + f(b)\big)$, где $c = \frac{a+b}{2}$

Построим таблицу:

C(x)	erf(x)	err	n
0.000000000000	0.000000000000	0.0000000000	0
0.099997532630	0.099997532220	0.0000000004	8
0.199921057600	0.199921044800	0.0000000128	8
0.299400976000	0.299400880700	0.0000000953	8
0.397480759200	0.397480377000	0.0000003822	8
0.492344225900	0.492343184200	0.0000010417	8
0.581095447000	0.581093404100	0.0000020429	8
0.659652351800	0.659649786200	0.0000025656	8
0.722844171800	0.722844230100	0.0000000583	8
0.764823021200	0.764823777900	0.0000007567	16
0.779893400300	0.779896021800	0.0000026215	16
0.763806665900	0.763812848600	0.0000061827	16
0.715437722700	0.715438427600	0.0000007049	32
0.638550454500	0.638551458200	0.0000010037	32
0.543095783000	0.543096747100	0.0000009641	32
0.445261176200	0.445262747000	0.0000015708	16

Снова можно заметить уменьшение погрешности при том, что количество разбиений отрезка тоже существенно сократилось

Квадратурная формула Гаусса

$\int_c^d f(x)dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)dx \approx \sum_{i=1}^{n} S_i(f)$, где

$S_i(f) = \frac{h_n}{2}\left[f\left(z_{i-1} + \frac{h_n}{2}\left(1 - \frac{1}{\sqrt{3}}\right)\right) + f\left(z_{i-1} + \frac{h_n}{2}\left(1 + \frac{1}{\sqrt{3}}\right)\right)\right],$

z_i - точки разбиения отрезка интегрирования на n частей $z_i = c + i \cdot h_n$, $h_n = \frac{d-c}{n}$

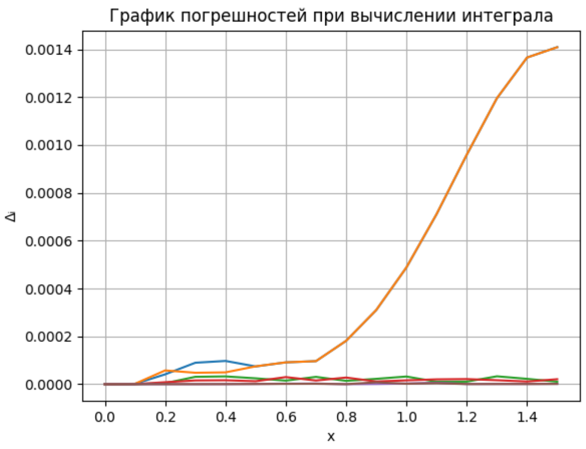
Снова построим таблицу:

C(x)	erf(x)	err	n
0.000000000000	0.000000000000	0.0000000000	0
0.099997532630	0.099997532900	0.0000000003	8
0.199921057600	0.199921066200	0.0000000086	8
0.299400976000	0.299401039700	0.0000000637	8
0.397480759200	0.397481014200	0.0000002550	8
0.492344225900	0.492344920500	0.0000006946	8
0.581095447000	0.581096810400	0.0000013634	8
0.659652351800	0.659654068100	0.0000017163	8
0.722844171800	0.722844148100	0.0000000237	8
0.764823021200	0.764815296900	0.0000077243	8
0.779893400300	0.779891652900	0.0000017474	16
0.763806665900	0.763802543400	0.0000041225	16
0.715437722700	0.715437253000	0.0000004697	32
0.638550454500	0.638549785500	0.0000006690	32
0.543095783000	0.543095140500	0.0000006425	32
0.445261176200	0.445260077200	0.0000010990	16

Ситуация снова похоже на предыдущую (с использованием формулы Симпсона), количество разбиений отрезка намного ниже, чем у других формул, при это достигнута требуемая точность

Построим графики погрешности, но для наглядности ограничим n до 1024, таким образом при вычислении по формулам правых и левых прямоугольников погрешность сильно увеличится, но количество вычислений не будет превышать последующие

(на втором графике уберем формулы левых и правых прямоугольников):



Из графиков можно сделать вывод:

Вычисления по формулам правых и левых прямоугольников дают большую погрешность, чтобы добиться требуемой погрешности, нужно сильно увеличить количество разбиений отрезка, следовательно количество вычислений

Остальные формулы дают большую точность, однако формулы Симпсона и Гаусса заметно выигрывают.

ВЫВОД

В работе были рассмотрены способы вычисления интерполяционного полинома Лагранжа по равномерно распределенным узлам и по корням полинома Чебышева.

При вычислении интерполяционного полинома Лагранжа по равномерно распределенным узлам, максимальная погрешность полинома при увеличении количества узлов сначала постепенно убывает, затем начинает быстро возрастать и достигать больших значений уже на 40 узлах.

В случае вычисления полинома Лагранжа по корням полинома Чебышева погрешность стабильно маленькая и мы не наблюдаем резкого увеличения погрешности даже при очень большом количестве узлов (в моем случае, даже при 3000 узлах погрешность мала)

Таким образом, мы выяснили, что более точным способом вычисления интегралов с заданной точностью является интерполяционный полином, в котором за узлы интерполяции взяты корни полинома Чебышева

Так же в ходе выполнения работы мы получили результаты вычисления интегралов с заданной точностью с помощью различных составных квадратурных формул и узнали, что наиболее эффективными методами являются квадратурные формулы Симпсона и Гаусса.

ПРИЛОЖЕНИЕ

Основной код моей программы можно просмотреть и скачать по ссылке: https://github.com/D4eNst/project-numerical-methods.

Код написан на языке программирования Python.

Код хорошо структурирован и документирован, здесь я приложу примеры использования:

```
from math import pi, cos
from decimal import Decimal
from solution import Solution


def main():
    a = Decimal('0')
    b = Decimal('1.5')
    n = 16
    epsilon = Decimal('1e-6')

    solution1 = Solution(
        a=a, b=b, n=n, epsilon=epsilon,
        qn_func=lambda x, i: -1 * (4 * i + 1) * (Decimal(pi) / 2) ** 2 * (x ** 4) /
(
                (2 * i + 1) * (2 * i + 2) * (4 * i + 5)),
        a0_func=lambda x: x,
        fi_from_t=lambda t: Decimal(cos((Decimal.from_float(pi) * t * t) / 2)),
        num_points=2,
        type_nodes=1,
        type_interpolation=1)
if __name__ == "__main__":
    main()
```

```
ⓒ solution.Solution

def __init__(self,
             a: Decimal,
             b: Decimal,
             n: int,
             epsilon: Decimal,
             a0_func: (Decimal) -> Decimal,
             qn_func: (Decimal, int) -> Decimal,
             fi_from_t: (Decimal) -> Decimal,
             **kwargs: Any) -> Any
```

Params: a – Начальное значение интервала.
b – Конечное значение интервала.
n – Количество узлов.
epsilon – Точность вычислений.
a0_func – Функция для вычисления a0. (1 задание)
qn_func – Функция для вычисления qn. (1 задание)
fi_from_t – Функция ϕ(t). (3 задание)
kwargs – Дополнительные аргументы.

Keyword args: num_points – количество точек между узлами для интерполяции (чтобы использовать по умолчанию для функций в классе)
type_nodes – int: Тип распределения узлов (1 - равномерно, 2 - полином Чебышева).
type_interpolation – int: Параметр для выбора функции интерполяции (1 - Логранжа, 2 - Ньютона)

Инициализация класса Solution.

Все последующие примеры я буду дописывать в фунцию main.

Вывести значения узлов и посчитанные по первому заданию значения в этих узлах:

```
solution1.show_console_tabulation()
```

Вот так можно вывести в столбик все значения интерполяционного полинома:

```
print("Полином")
print(*map(str, solution1.calculate_interpolation()), sep="\n")
```

При этом также можно вывести иксы этого полинома:

```
print("x")
print(*solution1.interpolate_nodes(), sep="\n")
```

Список погрешностей можно вывести так:

```
print(*solution1.inaccuracy())
```

А для того, чтобы вывести график, как в отчете, можно использовать метод:

```
solution1.show_graph_inaccuracy()
```

Или

```
solution1.show_graph_inaccuracy(only_normal_er=True, epsilon=Decimal('1e-4'))
```

Таблицы из части 4 я составлял с помощью такого кода:

```
erf_xes = [Decimal('0.0')] + solution1.tabulate_erf_x(formula='gauss')
print("      C(x)            erf(x)          err")
for erf_x, f in zip(erf_xes, solution1.f_from_x_values):
    print(f"{f:2.12f} {erf_x:2.12f} {abs(f - erf_x):2.10f}")
```

А графики можно вывести так:

```
solution1.show_graph_erf_x_error(methods=["center_rect", "trapezoid", "simpson",
"gauss"])
```

Или

```
solution1.show_graph_erf_x_error(methods= "all")
```

Выведем список максимальных погрешностей до тех пор, пока она не превысит единицу, с шагом 2:

```
print(f"\n----------------------------------\nЗависимость погрешности от количества
узлов")
dict_for_max_inaccuracy = {
    'num_points': 2,
    'stop_on_large_er': True,
    'max_n': 107,
    'start_n': 6,
    'stop_n': 100,
    'step': 2
}
max_inaccuracy_list = solution1.error_vs_nodes_dependency(**dict_for_max_inaccuracy)
print("n    max_er")
for i in range(len(max_inaccuracy_list)):
    print(f'{int(dict_for_max_inaccuracy.get("start_n", n + 1)) + i*10:2}, '
          f'{str(max_inaccuracy_list[i])}')
```

Теперь график:

```
solution1.show_graph_error_vs_nodes_dependency(**dict_for_max_inaccuracy)
```

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. **Даутов Р.З. Тимербаев М.Р.** Численные методы. Решение задач линейной алгебры и дифференциальных уравнений: учебное пособие.(2021)

2. **Даутов Р.З. Тимербаев М.Р.** Численные методы. Приближение функций: учебное пособие.(2021)

3. **Глазырина Л.Л., Карчевский М.М**. Введение в численные методы - Учебные пособия (2012)

4. **Даутов Р.З.** Практикум по курсу численные методы. Решение задачи Коши для системы ОДУ (2014)

5. **Даутов Р.З., Карчевский М.М.** Основы численных методов линейной алгебры: учеб. пособие (2018)