Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Вычисление математических функций»**

**Выполнил**:

студент группы 3823Б1ПМ1-1

Алемаев М.В.

**Проверил**:

преподаватель каф. ВВСП,

Волокитин В.Д.

Нижний Новгород

2024

**Содержание**

[Постановка задачи 3](#_Toc152527359)

[Метод решения 4](#_Toc152527360)

[Руководство пользователя 6](#_Toc152527361)

[Описание программной реализации 7](#_Toc152527362)

[Подтверждение корректности 8](#_Toc152527363)

[Результаты экспериментов 12](#_Toc152527364)

[Заключение 17](#_Toc152527365)

[Приложение 18](#_Toc152527366)

[Источники 20](#_Toc152527367)

# Постановка задачи

1. Написать ряды Маклорена для следующих функций: синус, косинус, натуральный логарифм и экспонента.
2. Написать несколько методов суммирования.
3. Проверить на корректность.
4. Создать пользовательский интерфейс для вычисления рядов суммирования.
5. Провести ряд экспериментов, в ходе которых узнать, какой способ суммирования лучше всего подходит в данном случае.

**Метод решения**

Для начала приведем несколько ряды разложения соответствующих функций:









Наиболее эффективно члены последовательности вычислять рекуррентно через коэффициент *next = ai+1/ai*.

Найдем next для соответствующих функций:

* exp. next = *x/i*
* log. next = *-(x\*i)/(i+1)*
* sin, cos. next = *-(x \* x) / (i\*(i-1))*

где x – аргумент функции, i – степень.

Для решения поставленной задачи необходимо реализовать 4 вида суммирования:

1. **Прямое суммирование.** Суммирование членов в порядке их следования – от младшей степени к старшей.

2. **Обратное суммирование.** Суммирование элементов последовательности от последнего к первому – от старшей степени к младшей.

3. **Попарное суммирование.** Прямое суммирование элементов последовательности, но от обычного прямого суммирования он отличается тем, что складываются по два числа вида *ai + aj+1*, где 2 | i.Если число элементов нечетно, то отдельно прибавляется последний элемент

4. **Обратное попарное суммирование.** Обратное суммирование элементов последовательности, но от обычного обратного суммирования он отличается тем, что складываются по два числа вида *ai + aj-1*. Если число элементов нечетно, то отдельно прибавляется первый элемент.

**Руководство пользователя**

Пользователь сначала вводит название одной из четырех функций, перечисленных ранее. Затем вводит число элементов последовательности и аргумент функции.

Программа в свою очередь выводит проводит суммирование четырьмя различными способами, выводит результат и погрешность.

**Описание программной реализации**

Рекуррентно считаем все элементы последовательности и вносим их в массив *values*, состоящий из *n* элементов,в порядке их следования. Для этого введем функцию *next*, при помощи которой будем вычислять последующий элемент. В неё будем вносить аргумент функции и нужную нам степень.

Суммируем массив values 4-мя различными способами в функции *infelicity*, которая одинакова для всех рядов. При этом не стоит забывать, что начиная с какого-то номера элементы будут обращаться в ноль. Затем для полученных значений ищем погрешность. Погрешность будем определять как разницу между значением функции встроенной в библиотеку *cmath,* и той, что получили путем суммирования.

Приведем пример для экспоненты:

Exp\_t(n,x,next)

1. e = exp(x);
2. values = {};
3. values[0] = 1;
4. for i = 1 to n do I = I + 1
5. values[i] = values[i-1]\*next(x,i);
6. INFELISITY(e, values, n);

Аналогично делается для остальных функций.

INFELISITY(reference, values, n)

1. s = 0;
2. for i = 0 to n
3. s += values[i];
4. //Выводим значение и ищем погрешность
5. For I = n-1 to -1 do I = I - 1
6. s += values[i];
7. //Выводим значение и ищем погрешность
8. for i = 0 to n-1 do I = I + 2
9. s += values[i] + values[i+1];
10. if (n%2)
11. s+=values[n-1];
12. //Выводим значение и ищем погрешность
13. for i = n-1 to 0 do I = I - 2
14. s += values[i] + values[i+1];
15. if (n%2)
16. s += values[0];

Не стоит забывать что в случае с попарными способами суммирования число членов последовательности может быть нечетным. Тогда мы добавляем последний (первый) элемент в частном порядке.

**Подтверждение корректности**

В целях подтверждения корректности проверим сходимость ряда при увеличении числа членов последовательности. Если программа написана правильно, то при увеличении числа членов погрешность будет постепенно уменьшаться.

Продемонстрируем графики иллюстрирующие сходимость ряда при следующих значениях аргумента:

* 1.9 для натурального логарифма
* 15 для остальных

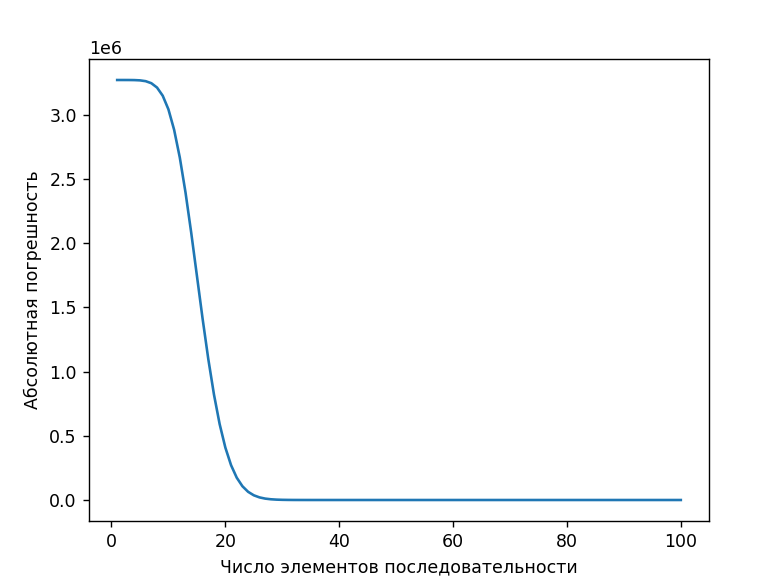


Рис. 1 Погрешность вычисления экспоненты

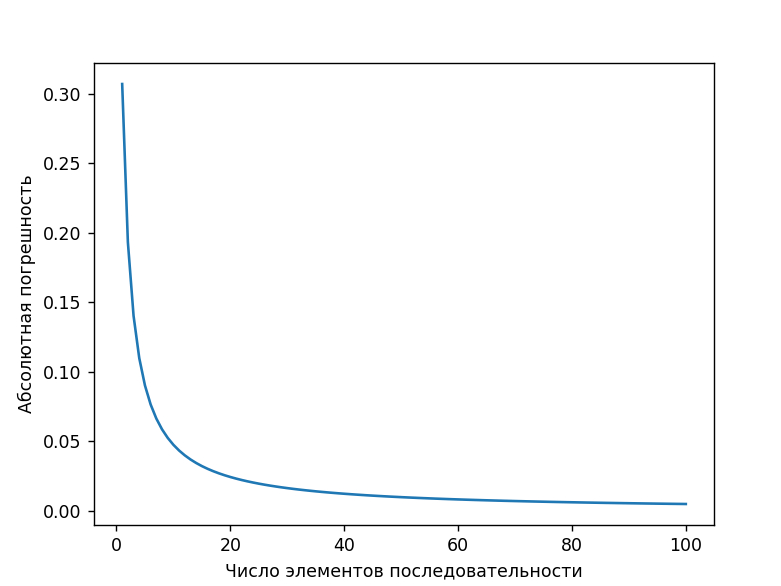


Рис. 2 погрешность вычисления натурального логарифма

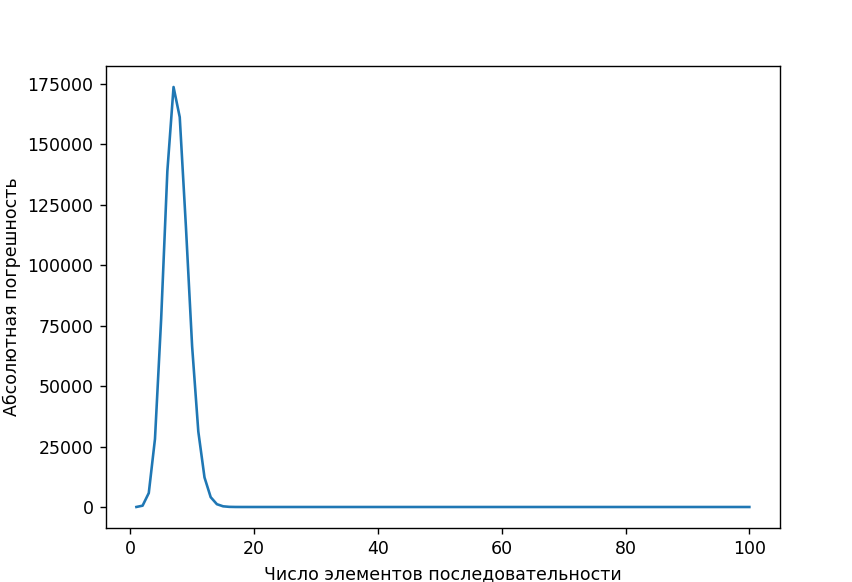


Рис. 3 погрешность вычисления синуса

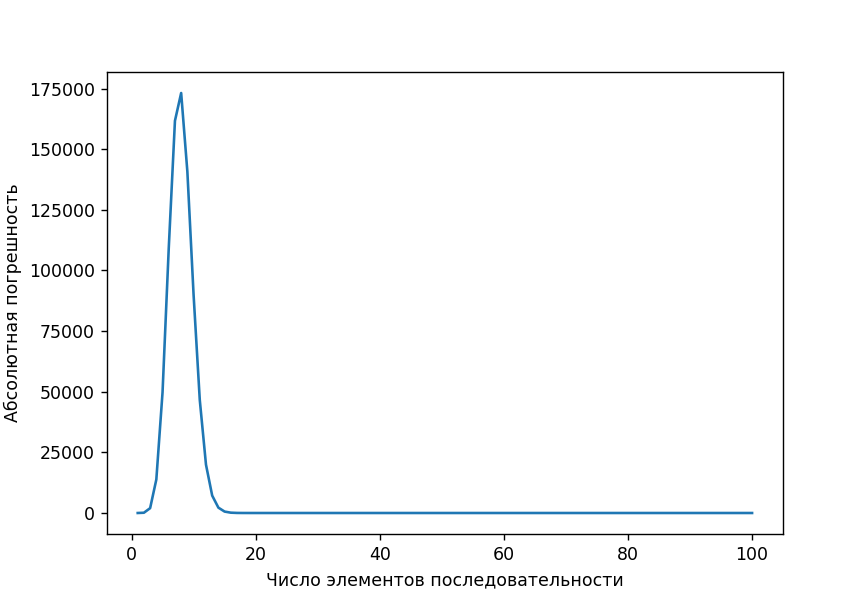


Рис. 4 погрешность вычисления косинуса

**Результаты экспериментов**

Для эксперимента были использованы значения аргумента от 0 до 30.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Прямое суммирование | | Обратное суммирование | | Попарное суммирование | | Обратное попарное суммирование | |
| Аргу-мент | Погрешность | Аргу-мент | Погрешность | Аргу-мент | Погрешность | Аргу-мент | Погрешность |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 2.38419e-07 | 1 | 0 | 1 | 2.38419e-07 | 1 | 0 |
| 2 | 4.76837e-07 | 2 | 0 | 2 | 0 | 2 | 0 |
| 3 | 1.90735e-06 | 3 | 0 | 3 | 0 | 3 | 0 |
| 4 | 3.8147e-06 | 4 | 7.62939e-06 | 4 | 0 | 4 | 3.8147e-06 |
| 5 | 0 | 5 | 1.52588e-05 | 5 | 1.52588e-05 | 5 | 1.52588e-05 |
| 6 | 3.05176e-05 | 6 | 3.05176e-05 | 6 | 3.05176e-05 | 6 | 3.05176e-05 |
| 7 | 0.000244141 | 7 | 0 | 7 | 0.00012207 | 7 | 0.00012207 |
| 8 | 0.000732422 | 8 | 0.000488281 | 8 | 0.000244141 | 8 | 0.000244141 |
| 9 | 0.000976562 | 9 | 0 | 9 | 0.000488281 | 9 | 0 |
| 10 | 0 | 10 | 0 | 10 | 0 | 10 | 0 |
| 11 | 0.0117188 | 11 | 0.0078125 | 11 | 0.015625 | 11 | 0.0078125 |
| 12 | 0.015625 | 12 | 0.015625 | 12 | 0.015625 | 12 | 0.015625 |
| 13 | 0 | 13 | 0.0625 | 13 | 0 | 13 | 0.03125 |
| 14 | 0 | 14 | 0 | 14 | 0 | 14 | 0.125 |
| 15 | 0.25 | 15 | 0.25 | 15 | 0.25 | 15 | 0.5 |
| 16 | 2 | 16 | 3 | 16 | 2 | 16 | 3 |
| 17 | 2 | 17 | 4 | 17 | 4 | 17 | 0 |
| 18 | 20 | 18 | 8 | 18 | 4 | 18 | 12 |
| 19 | 0 | 19 | 16 | 19 | 16 | 19 | 16 |
| 20 | 32 | 20 | 64 | 20 | 96 | 20 | 96 |
| 21 | 384 | 21 | 640 | 21 | 128 | 21 | 384 |
| 22 | 1280 | 22 | 1280 | 22 | 512 | 22 | 1280 |
| 23 | 3072 | 23 | 5120 | 23 | 2048 | 23 | 2048 |
| 24 | 8192 | 24 | 6144 | 24 | 6144 | 24 | 8192 |
| 25 | 32768 | 25 | 32768 | 25 | 40960 | 25 | 40960 |
| 26 | 32768 | 26 | 49152 | 26 | 16384 | 26 | 32768 |
| 27 | 65536 | 27 | 98304 | 27 | 0 | 27 | 98304 |
| 28 | 262144 | 28 | 524288 | 28 | 262144 | 28 | 262144 |
| 29 | 262144 | 29 | 1.04858e+06 | 29 | 262144 | 29 | 262144 |
| 30 | 3.14573e+06 | 30 | 1.04858e+06 | 30 | 3.14573e+06 | 30 | 2.09715e+06 |

Таб. 1 Измерение погрешности экспоненты

Из таблицы видно, что наиболее предпочтительным способом суммирования для экспоненты является попарное суммирование. А худшим способом является обратное суммирование.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Прямое суммирование | | Обратное суммирование | | Попарное суммирование | | Обратное попарное суммирование | |
| Аргу-мент | Погрешность | Аргу-мент | Погрешность | Аргу-мент | Погрешность | Аргу-мент | Погрешность |
| 0.1 | 1.66893e-06 | 0.1 | 2.38419e-07 | 0.1 | 0 | 0.1 | 2.38419e-07 |
| 0.2 | 0 | 0.2 | 0 | 0.2 | 0 | 0.2 | 0 |
| 0.3 | 3.57628e-07 | 0.3 | 0 | 0.3 | 1.19209e-07 | 0.3 | 0 |
| 0.4 | 5.96046e-08 | 0.4 | 5.96046e-08 | 0.4 | 0 | 0.4 | 5.96046e-08 |
| 0.5 | 1.19209e-07 | 0.5 | 0 | 0.5 | 0 | 0.5 | 0 |
| 0.6 | 5.96046e-08 | 0.6 | 0 | 0.6 | 0 | 0.6 | 0 |
| 0.7 | 5.96046e-08 | 0.7 | 0 | 0.7 | 0 | 0.7 | 0 |
| 0.8 | 1.49012e-08 | 0.8 | 0 | 0.8 | 0 | 0.8 | 0 |
| 0.9 | 1.49012e-08 | 0.9 | 0 | 0.9 | 0 | 0.9 | 7.45058e-09 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1,1 | 0 | 1,1 | 0 | 1,1 | 0 | 1,1 | 0 |
| 1,2 | 1.49012e-08 | 1,2 | 0 | 1,2 | 1.49012e-08 | 1,2 | 1.49012e-08 |
| 1,3 | 2.98023e-08 | 1,3 | 0 | 1,3 | 2.98023e-08 | 1,3 | 0 |
| 1,4 | 5.96046e-08 | 1,4 | 0 | 1,4 | 0 | 1,4 | 2.98023e-08 |
| 1,5 | 5.96046e-08 | 1,5 | 0 | 1,5 | 2.98023e-08 | 1,5 | 0 |
| 1,6 | 5.96046e-08 | 1,6 | 0 | 1,6 | 2.98023e-08 | 1,6 | 0 |
| 1,7 | 1.19209e-07 | 1,7 | 0 | 1,7 | 1.19209e-07 | 1,7 | 0 |
| 1,8 | 1.19209e-07 | 1,8 | 5.96046e-08 | 1,8 | 1.19209e-07 | 1,8 | 5.96046e-08 |
| 1,9 | 5.96046e-08 | 1,9 | 0 | 1,9 | 1.19209e-07 | 1,9 | 0 |

Таб. 2 погрешность вычисления натурального логарифма

Из второй таблицы видно, что лучшим способом суммирования для натурального логарифма являются обратная сумма и обратно-попарное суммирование, в то время, как прямое суммирование значительно уступает. Это связанно с тем, что ряд натурального логарифма довольно долго сходится, поэтому большое количество маленьких по модулю элементов находится в конце. Однако аргументы в данном случае довольно малы, поэтому полную оценку аппроксимации натурального логарифма дать нельзя, из-за того, что область его сходимости – (0;2].

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Прямое суммирование | | Обратное суммирование | | Попарное суммирование | | Обратное-попарное суммирование | |
| Аргу-мент | Погрешность | Аргу-мент | Погрешность | Аргу-мент | Погрешность | Аргу-мент | Погрешность |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 2 | 0 | 2 | 5.96046e-08 | 2 | 0 | 2 | 0 |
| 3 | 1.3411e-07 | 3 | 4.47035e-08 | 3 | 4.47035e-08 | 3 | 4.47035e-08 |
| 4 | 1.78814e-07 | 4 | 5.96046e-08 | 4 | 1.78814e-07 | 4 | 5.96046e-08 |
| 5 | 4.76837e-07 | 5 | 9.53674e-07 | 5 | 5.96046e-07 | 5 | 9.53674e-07 |
| 6 | 2.17557e-06 | 6 | 3.45707e-06 | 6 | 1.2219e-06 | 6 | 1.54972e-06 |
| 7 | 7.7486e-07 | 7 | 3.57628e-07 | 7 | 4.76837e-07 | 7 | 7.27177e-06 |
| 8 | 1.65701e-05 | 8 | 1.2517e-06 | 8 | 1.67489e-05 | 8 | 1.2517e-06 |
| 9 | 6.82473e-06 | 9 | 9.11951e-06 | 9 | 7.7486e-06 | 9 | 9.11951e-06 |
| 10 | 1.98483e-05 | 10 | 7.67708e-05 | 10 | 1.27554e-05 | 10 | 7.67708e-05 |
| 11 | 0.000101328 | 11 | 0.000208139 | 11 | 9.76324e-05 | 11 | 0.00033021 |
| 12 | 0.000526607 | 12 | 0.000684261 | 12 | 0.000558972 | 12 | 0.000440121 |
| 13 | 0.00252315 | 13 | 0.00179952 | 13 | 0.00241733 | 13 | 0.00204366 |
| 14 | 0.00383741 | 14 | 0.00607979 | 14 | 0.00389445 | 14 | 0.00559151 |
| 15 | 0.000776708 | 15 | 0.0137746 | 15 | 0.000775576 | 15 | 0.0137746 |
| 16 | 0.024894 | 16 | 0.0340571 | 16 | 0.0249098 | 16 | 0.0350337 |
| 17 | 0.0599696 | 17 | 0.0418984 | 17 | 0.0580167 | 17 | 0.0126016 |
| 18 | 0.26586 | 18 | 0.270497 | 18 | 0.523641 | 18 | 0.33495 |
| 19 | 0.244205 | 19 | 0.0915779 | 19 | 0.240512 | 19 | 0.0896247 |
| 20 | 1.30048 | 20 | 1.30849 | 20 | 1.27107 | 20 | 1.13647 |
| 21 | 1.59929 | 21 | 2.22728 | 21 | 1.45678 | 21 | 1.96166 |
| 22 | 9.09167 | 22 | 2.50543 | 22 | 9.09068 | 22 | 2.00152 |
| 23 | 76.9535 | 23 | 49.1419 | 23 | 77.0122 | 23 | 49.638 |
| 24 | 35.0648 | 24 | 42.9759 | 24 | 34.8451 | 24 | 41.9681 |
| 25 | 81.3903 | 25 | 122.831 | 25 | 79.5148 | 25 | 137.784 |
| 26 | 214.204 | 26 | 294.365 | 26 | 205.266 | 26 | 262.404 |
| 27 | 1400.94 | 27 | 1571.23 | 27 | 1393.01 | 27 | 1571.21 |
| 28 | 3323.49 | 28 | 4967.45 | 28 | 3287.26 | 28 | 4899.44 |
| 29 | 3521.33 | 29 | 1720.33 | 29 | 3505.34 | 29 | 1592.33 |
| 30 | 17497.9 | 30 | 6067.01 | 30 | 17609.8 | 30 | 5799.01 |

Таб. 3 погрешность вычисления синуса

Для синуса лучшими при маленьких значениях лучше подходит прямое и попарное суммирование, в то время, как для больших аргументов больше подходят обратное и обратное попарное.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Прямое суммирование | | Обратное суммирование | | Попарное суммирование | | Обратное-попарное суммирование | |
| Аргу-мент | Погрешность | Аргу-мент | Погрешность | Аргу-мент | Погрешность | Аргу-мент | Погрешность |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 2 | 2.98023e-08 | 2 | 8.9407e-08 | 2 | 2.98023e-08 | 2 | 2.98023e-08 |
| 3 | 0 | 3 | 1.19209e-07 | 3 | 5.96046e-08 | 3 | 1.19209e-07 |
| 4 | 5.96046e-08 | 4 | 0 | 4 | 1.19209e-07 | 4 | 0 |
| 5 | 1.19209e-07 | 5 | 3.57628e-07 | 5 | 1.49012e-07 | 5 | 5.96046e-07 |
| 6 | 3.33786e-06 | 6 | 3.33786e-06 | 6 | 3.15905e-06 | 6 | 3.33786e-06 |
| 7 | 2.68221e-06 | 7 | 3.99351e-06 | 7 | 2.32458e-06 | 7 | 3.99351e-06 |
| 8 | 8.9407e-06 | 8 | 7.77841e-06 | 8 | 5.37932e-06 | 8 | 7.77841e-06 |
| 9 | 3.21865e-06 | 9 | 2.563e-06 | 9 | 2.98023e-06 | 9 | 2.563e-06 |
| 10 | 0.000197589 | 10 | 0.000131369 | 10 | 0.000198483 | 10 | 0.000161886 |
| 11 | 0.000320393 | 11 | 0.000335044 | 11 | 0.000328268 | 11 | 0.000335044 |
| 12 | 6.07371e-05 | 12 | 0.000103951 | 12 | 5.8651e-05 | 12 | 0.000103951 |
| 13 | 0.00243342 | 13 | 0.00107473 | 13 | 0.00270808 | 13 | 0.00314993 |
| 14 | 0.00856712 | 14 | 0.00319229 | 14 | 0.00857452 | 14 | 0.0068544 |
| 15 | 0.00940192 | 15 | 0.0135942 | 15 | 0.0103098 | 15 | 0.0096879 |
| 16 | 0.0342572 | 16 | 0.0189388 | 16 | 0.033768 | 16 | 0.0260189 |
| 17 | 0.0254652 | 17 | 0.0239426 | 17 | 0.00788733 | 17 | 0.0388352 |
| 18 | 0.185561 | 18 | 0.109535 | 18 | 0.199168 | 18 | 0.109535 |
| 19 | 0.0276235 | 19 | 0.170475 | 19 | 0.0901216 | 19 | 0.171452 |
| 20 | 3.71659 | 20 | 3.31189 | 20 | 3.77936 | 20 | 3.37683 |
| 21 | 2.43105 | 21 | 1.75085 | 21 | 2.39968 | 21 | 1.81335 |
| 22 | 2.96481 | 22 | 2.83785 | 22 | 3.85546 | 22 | 2.59371 |
| 23 | 12.00462 | 23 | 18.33062 | 23 | 11.71273 | 23 | 6.31092 |
| 24 | 71.5025 | 24 | 17.7367 | 24 | 71.4401 | 24 | 18.2367 |
| 25 | 183.754 | 25 | 241.112 | 25 | 183.721 | 25 | 241.353 |
| 26 | 138.632 | 26 | 184.637 | 26 | 120.758 | 26 | 298.647 |
| 27 | 736.896 | 27 | 882.833 | 27 | 672.888 | 27 | 851.833 |
| 28 | 2548 | 28 | 149.494 | 28 | 2484 | 28 | 424.525 |
| 29 | 811.857 | 29 | 7364.79 | 29 | 811.841 | 29 | 8515.81 |
| 30 | 30392.9 | 30 | 27057.8 | 30 | 30360.8 | 30 | 24880.8 |

Таб. 4 погрешность вычисления косинуса

Ряд косинуса очень похож на ряд синуса, поэтому результаты для него аналогичны.

В качестве вывода можно выделить:

* Чем дальше аргумент оказывался от нуля, тем ниже оказывалась точность.
* Для тригонометрических функций при маленьких аргументах наиболее эффективными оказались прямое и попарное суммирование, а для больших – обратное и обратное попарное.
* Аппроксимацию натурального логарифма сложно оценить в силу малости аргументов.

**Заключение**

* В работе были прописаны ряды разложения ряда элементарных функций.
* Была проведена проверка на корректность работы.
* Были описаны программная реализация и принцип работы.
* Был проведен ряд экспериментов и озвучены выводы.

**Приложение**

float nextsin(float x, size\_t i) {

return -(x \* x) / (i\*(i-1.0f));

}

float nextcos(float x, size\_t i) {

return -(x \* x) / (i \* (i - 1.0f));

}

float nextexp(float x, size\_t i) {

return x / i;

}

float nextlog(float x, size\_t i) {

return -(x \* i) / (i + 1.0f);

}

void infelicity(float reference, float\* values, int n) {

float s = 0;

for (int i = 0; i < n; i++)

s += values[i];

printf("Прямое суммирование\n");

cout << "Значение - " << s << endl;

cout << "Погрешность - " << abs(s - reference) << endl << endl;

s = 0;

for (int i = n - 1; i >= 0; i--)

s += values[i];

printf("Обратное суммирование\n");

cout << "Значение - " << s << endl;

cout << "Погрешность - " << abs(s - reference) << endl << endl;

s = 0;

for (int i = 0; i + 1 < n; i+=2)

s += (values[i] + values[i + 1]);

if (n % 2)

s += values[n - 1];

printf("Прямое попарное суммирование\n");

cout << "Значение - " << s << endl;

cout << "Погрешность - " << abs(s - reference) << endl << endl;

s = 0;

for (int i = n-1; i-1 >= 0; i -= 2) {

s += (values[i] + values[i - 1]);

}

if (n % 2)

s += values[0];

printf("Обратное попарное суммирование\n");

cout << "Значение - " << s << endl;

cout << "Погрешность - " << abs(s - reference) << endl << endl;

}

void exp\_t(float x, int n, float (\*next)(float, size\_t)) {

float e = exp(x);

float\* values = (float\*)malloc(n \* sizeof(float));

values[0] = 1;

for (int i = 1; i < n; i++) {

values[i] = values[i-1] \* next(x, i);

}

infelicity(e, values, n);

free(values);

}

void log\_t(float x, int n, float (\*next)(float, size\_t)) {

x--;

float lg = log1p(x+1);

float\* values = (float\*)malloc(n \* sizeof(float));

values[0] = x;

for (int i = 1; i < n; i++)

values[i] = values[i-1] \* next(x,i);

infelicity(lg, values, n);

free(values);

}

void sin\_t(float x, int n, float (\*next)(float, size\_t)) {

float sinx = sin(x);

float\* values = (float\*)malloc(n \* sizeof(float));

for (int i = 0; i < n; i++)

values[i] = 0;

values[0] = x;

for (int i = 3; i < 2\*n; i += 2)

values[i/2] = values[i/2-1] \* next(x, i);

infelicity(sinx, values, n);

free(values);

}

void cos\_t(float x, int n, float (\*next)(float, size\_t)) {

float cosx = cos(x);

float\* values = (float\*)malloc(n \* sizeof(float));

for (int i = 0; i < n; i++)

values[i] = 0;

values[0] = 1;

int c = 1;

for (int i = 2; i < 2\*n; i += 2) {

values[i/2] = values[i / 2 - 1] \* next(x, i);

}

infelicity(cosx, values, n);

free(values);

}

**Литература**

1. https://ru.wikipedia.org/wiki/Ряд\_Тейлора