Лабораторная работа №2
Ильин Денис, z3243

Задание 1

1. Посмотрим на сумму трех чисел: a1,a2,a3. Путсь они даны с некоторой погрешностью $|\sigma a1|, |\sigma a2|, |\sigma a3| < \varepsilon$. Оценим погрешность суммы этих трех чисел: $\sigma(a1 + a2 + a3) = \sigma((a1 + a2) + a3) = \frac{\Delta a3+\Delta(a1+a2)}{|a1+a2+a3|} + \varepsilon_{\text{окр}} = \frac{a3*\sigma a3}{|a1+a2+a3|} + \frac{\Delta(a1+a2)}{|a1+a2+a3|} + \varepsilon_{\text{окр}}$. Следовательно, после такого представления видим, что погрешность суммы трех чисел зависит от $\Delta(a1 + a2)$. Поэтому будем складывать числа так, чтобы погрешность была минимальной.Это будет достигаться при сложении чисел в порядке возрастания

2. В алгоритме Кэхэна уменьшение погрешности достигается введением дополнительной переменной для хранения нарастающей суммы погрешности. Данный алгоритм складывает младшие бдиты суммы, которые исчезают из-за округления при наивном суммировании

```python
import numpy as np
import math as m
def exact(N):
    return 1/2*(m.sin(N)-(1/m.tan(1/2))*m.cos(N)+(1/m.tan(1/2)))
def sumsin(N):
    parts=[np.full((1), m.sin(k)) for k in range(1,N+1)]
    samples=np.concatenate(parts)
    return np.random.permutation(samples)
def direct_sum(x):
    """Последовательная сумма всех элементов вектора x"""
    s=0.
    for e in x:
        s+=e
    return s
def relative_error(x0, x):
    return np.abs(x0-x)/np.abs(x)
for M in [10, 100, 1000, 10000, 100000, 1000000]:
    print(exact(M),direct_sum(sumsin(M)),relative_error(exact(M),direct_sum(sumsin(M))))
print("SORTED MAX")
for M in [10, 100, 1000, 10000, 100000, 1000000]:
    x=sumsin(M)
    sorted_x=x[np.argsort(x)]
    print(exact(M),direct_sum(sorted_x),relative_error(exact(M),direct_sum(sorted_x)))
print("SORTED MIN")
for M in [10, 100, 1000, 10000, 100000, 1000000]:
    x=sumsin(M)
    sorted1_x=x[np.argsort(x)[::-1]]
    print(exact(M),direct_sum(sorted1_x),relative_error(exact(M),direct_sum(sorted1_x)))
print("СОРТИРОВКА ПО МОДУЛЮ")
for M in [10, 100, 1000, 10000, 100000, 1000000]:
    x=sumsin(M)
    y=sorted(x, key=np.abs)
    print(exact(M),direct_sum(y),relative_error(exact(M),direct_sum(y)))
```

(t-sum) восстанавливает старшие разряды y; вычитание y восстанавливает младшие разряды. Таким образом, сложение происходит в двух переменных: sum хранит сумму, c хранит часть суммы, которая не попала в sum, чтобы быть учтенной в sum на следующей итерации.

Задание 2

X	$\langle\Delta\rangle$	$\langle\sigma\rangle$	$\langle x\rangle$
x_n	εm	ε	m
$\frac{1}{N}\sum x_n$	εm	ε	m
$\left(\frac{1}{N}\sum x_n\right)^2$	$2\varepsilon m^2$	2ε	m^2
$\lvert x_n - \frac{1}{N}\sum x_n\rvert$	$2\varepsilon m$	$\frac{2\varepsilon m}{\sqrt{d}}$	$\sqrt{d}$
$\lvert x_n - \frac{1}{N}\sum x_n\rvert^2$	$4\varepsilon m\sqrt{d}$	$\frac{4\varepsilon m}{\sqrt{d}}$	d
x_n^2	$2\varepsilon m^2$	2ε	m^2
$x_n^2 - \left(\frac{1}{N}\sum x_n\right)^2$	$4\varepsilon m^2$	$\frac{4\varepsilon m^2}{d}$	d

$$N D_N^2 - (N-1) D_{N-1}^2 = \sum_{i=1}^{N} (x_i - \bar{x}_N)^2 - \sum_{i=1}^{N-1} (x_i - \bar{x}_{N-1})^2 = (x_N - \bar{x}_N)^2 +$$

$$+ \sum_{i=1}^{N-1} \left((x_i - \bar{x}_N)^2 - (x_i - \bar{x}_{N-1})^2 \right) = (x_N - \bar{x}_N)^2 + \sum_{i=1}^{N-1} (x_i - \bar{x}_N + x_i - \bar{x}_{N-1}) \cdot$$

$$\cdot (\bar{x}_{N-1} - \bar{x}_N) = (x_N - \bar{x}_N)^2 + (\bar{x}_N - x_N)(\bar{x}_{N-1} - \bar{x}_N) = (x_N - \bar{x}_N)(x_N - \bar{x}_{N-1})$$

$$\Rightarrow \boxed{D_N = D_{N-1} + \frac{1}{N}\left[(x_N - \bar{x}_N)(x_N - \bar{x}_{N-1}) - D_{N-1} \right]}$$

Задание 3

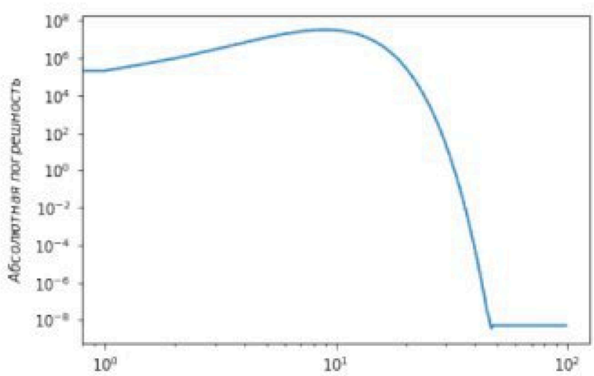
1) При отрицательных аргументах ряд становится знакочередующимся. Так как $lim_{k\to\infty} \frac{x^k}{k!} = 0$, то при больших значениях k будут вычитаться маленькие по модулю значения, что приведет к еще более маленьким значениям, на запись которых может не хватить количества разрядов, поэтому для x<0 следует считать следующим образом: $e^{-x} = \frac{1}{e^x} = \frac{1}{\sum\limits_{k=1}^{N} \frac{x^K}{k!}}$

2) $\sigma e^x = |x|\varepsilon$, а $\Delta e^x = e^x \cdot \sigma e^x$. То есть $\Delta e^x \le e^x|x|\varepsilon \le 2|x|\varepsilon$. Тогда при x близких к 0 абсолютная погрешность будет мала.

3) $\frac{exp(\theta)x^{N+1}}{(N+1)!} \le \frac{exp(2)(2)^{N+1}}{(N+1)!} = 3 * 10^{-10}$ при N=17 для x=2

4)

```
def plot_error(x0,err):
    #mask=np.logical_and(err>0,err<np.inf)
    #plt.loglog(x0[mask],err[mask],".k")
    plt.loglog(x0,err)
    #plt.loglog(x0,[eps]*len(err),"--r") # машинная точность для сравнения
    plt.xlabel("$Количество\;слагаемых\;частичной\;суммы$")
    plt.ylabel("$Абсолютная\;погрешность$")
    plt.show()
def cum_exp_taylor(x, N=None):
    """Вычисляет все частичные суммы ряда Тейлора для экспоненты по N-ую включительно."""
    acc = np.empty(N+1, dtype=float)
    acc[0] = 1 # k-ая частичная сумму. Начинаем с k=0.
    xk = 1 # Степени x^k.
    inv_fact = 1 # 1/k!.
    for k in range(1, N+1):
        xk = xk*x
        inv_fact /= k
        acc[k] = acc[k-1]+xk*inv_fact
    return acc[-1]
```



5) Используя запись полинома n-ой степени в виде мономов требует максимум n операций сложения и $\frac{n^2+n}{2}$ операций умножения, если степени считаются путем последовательного перемножения и каждый моном оценивается индивидуально. Это может быть сведено к n операциям сложения и 2n-1 операции умножения, если считать степени x итеративно. Однако схема Горнера требует только n сложений и n умножений

6)

```
x=-10
N=100
def relative(x,x0):
    return np.abs(x-x0)/np.abs(x0)
def taylor1(x,n):
    return x**n/f(n)
X=[]
for i in range(0,N):
    X.append(taylor1(x,i))
X=np.array(X)
K=Kahan_sum(X)
print(relative(K,np.exp(x)))
```

```
7.2440008560647795e-09
```