Numerical Methods
Lab Report 2. Roundoff errors

Maryna Borovyk

October 11, 2024

1 Backward summation for the Basel problem

Code

```
def backward_summation(nmax):
    sum = 0
    for k in range(nmax, 0, -1):  # in range from nmax to 1, with step -1
        sum += 1.0 / (k * k)  # add 1/k^2 to sum
    analytical_value = math.pi * math.pi / 6
    relative_error = (sum - analytical_value) / analytical_value * 100
    return sum, relative_error

nmax = main()

for i in [1, 2, 4, 8]:
    sum, relative_error = backward_summation(i * nmax)
    print(f"For {i} nmax:")
    print(f"Sum: {sum}")
    print(f"Relative error: {relative_error}%")


main()
```

```
For 1 nmax:
Sum: 1.6449340563115145
Relative error: -6.405552757250759e-07%
For 2 nmax:
Sum: 1.6449340615798704
Relative error: -3.2027763786253795e-07%
For 4 nmax:
Sum: 1.6449340642140484
Relative error: -1.6013881893126897e-07%
For 8 nmax:
Sum: 1.6449340655311373
Relative error: -8.006941621498115e-08%
```

2 Calculation of the exponential function

Code

```
def exponential(x, n_terms):
    sum = 1
    term = 1
    for i in range(1, n_terms):  # in range from 1 to n_terms - 1
        term *= x / i
        sum += term
    return sum

2 usages
def relative_error(sum, x):
    return (sum - math.exp(x)) / math.exp(x) * 100
```

```python
for x in [0.1, 20, -20]:
    print(f"For x = {x}:")
    print(f"Maclaurin series: {exponential(x,  n_terms: 100)}")
    print(f"Correct Maclaurin series: {math.exp(x)}")
    print(f"relative error: {relative_error(exponential(x,  n_terms: 100), x)}\n")

    x = -20
# How can this function be used to get the correct results for x = -20?
print(f"Correct Maclaurin series for x = -20: {1/exponential(-x,  n_terms: 100)}")
print(f"Correct relative error for x = -20: {relative_error(1/exponential(-x,  n_terms: 100), x)}")
```

Results

```
For x = 0.1:
Maclaurin series: 1.1051709180756473
Correct Maclaurin series: 1.1051709180756477
relative error: -4.0182853401836004e-14

For x = 20:
Maclaurin series: 485165195.40979046
Correct Maclaurin series: 485165195.4097903
relative error: 3.685629884788795e-14

For x = -20:
Maclaurin series: 6.147561828914626e-09
Correct Maclaurin series: 2.061153622438558e-09
relative error: 198.25830360191324
```

To avoid catastrophic cancelation in alternating series I used a formula:

$$e^{-x} = \frac{1}{e^x}$$

```
Correct Maclaurin series for x = -20: 2.061153622438557e-09
Correct relative error for x = -20: -4.013192435284797e-14
```

3 Kahan summation for Basel sum

Code

```
1 usage
def kahan_summation(input_list):
    sum = 0.0
    c = 0.0
    for number in input_list:
        y = number - c
        t = sum + y
        c = (t - sum) - y # (t - sum) recovers the high-order part of y;
        sum = t
    return sum
```

```
for i in [1, 2, 4, 8]:
    sum, relative_error = backward_summation(i * nmax)
    print(f"For {i} nmax:")
    print(f"Sum: {sum}")
    print(f"Relative error: {relative_error}%")

terms = [1.0 / (k * k) for k in range(1, nmax + 1)]
kahan_sum = kahan_summation(terms)
analytical_value = math.pi * math.pi / 6
relative_difference = (kahan_sum - analytical_value) / analytical_value * 100

print("Kahan Summation Basel Sum: ", kahan_sum)
print("Relative percentage difference using Kahan Summation: ", relative_difference, "%")
```

Results

```
Kahan Summation Basel Sum:  1.6449340563115145
Relative percentage difference using Kahan Summation:  -6.405552757250759e-07 %
```