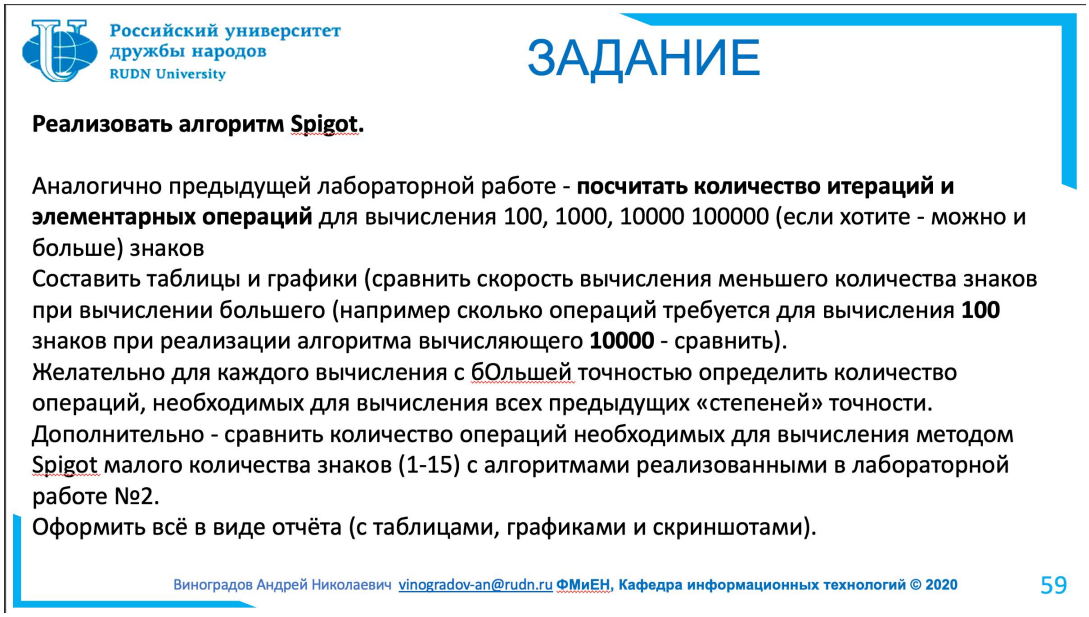
Лабораторная работа №3

Задание:



Выполнение:

1. Подготовка

In [106…
```
!pwd
```

```
/Users/artyem.petrov/dev/university/4-1/it-computer-practice/lab03
```

In [107…
```
!../.venv/bin/pip install matplotlib
```

```
Requirement already satisfied: matplotlib in /Users/artyem.petrov/de
v/university/4-1/it-computer-practice/.venv/lib/python3.13/site-pack
ages (3.10.8)
Requirement already satisfied: contourpy>=1.0.1 in /Users/artyem.pet
rov/dev/university/4-1/it-computer-practice/.venv/lib/python3.13/sit
e-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /Users/artyem.petrov/
dev/university/4-1/it-computer-practice/.venv/lib/python3.13/site-pa
ckages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /Users/artyem.pe
trov/dev/university/4-1/it-computer-practice/.venv/lib/python3.13/si
te-packages (from matplotlib) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /Users/artyem.pe
trov/dev/university/4-1/it-computer-practice/.venv/lib/python3.13/si
te-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in /Users/artyem.petrov/d
ev/university/4-1/it-computer-practice/.venv/lib/python3.13/site-pac
kages (from matplotlib) (2.3.5)
Requirement already satisfied: packaging>=20.0 in /Users/artyem.petr
ov/dev/university/4-1/it-computer-practice/.venv/lib/python3.13/site
-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /Users/artyem.petrov/de
v/university/4-1/it-computer-practice/.venv/lib/python3.13/site-pack
ages (from matplotlib) (12.0.0)
Requirement already satisfied: pyparsing>=3 in /Users/artyem.petrov/
dev/university/4-1/it-computer-practice/.venv/lib/python3.13/site-pa
ckages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in /Users/artye
m.petrov/dev/university/4-1/it-computer-practice/.venv/lib/python3.1
3/site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /Users/artyem.petrov/dev/
university/4-1/it-computer-practice/.venv/lib/python3.13/site-packag
es (from python-dateutil>=2.7->matplotlib) (1.17.0)

[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: /Users/artyem.petrov/dev/university/4-1/it-
computer-practice/.venv/bin/python -m pip install --upgrade pip
```

In [108…

```python
import time
import matplotlib.pyplot as plt
import json
```

2. Реализуем класс для подсчета операций:

In [109…

```python
class SpigotCounter:
    def __init__(self):
        self.ops = 0
        self.iterations = 0

    def add(self, count=1):
        self.ops += count
```

3. Теперь реализуем саму функцию:

In [110…

```python
def spigot_pi(n_digits, counter=None):
    if counter is None:
        counter = SpigotCounter()

    # Длина массива
    L = (10 * n_digits) // 3 + 1
    a = [2] * L
    digits = []

    # Буфер для недействительных цифр
    invalid_count = 0

    for i in range(n_digits):
        counter.iterations += 1

        carry = 0
        # Проходим по массиву с конца к началу
        for j in range(L-1, -1, -1):
            counter.add(2)  # умножение и сложение
            x = a[j] * 10 + carry

            if j > 0:
                denominator = 2*j + 1
                counter.add(2)  # умножение и сложение для denomina
                a[j] = x % denominator
                carry = (x // denominator) * j
                counter.add(3)  # деление, умножение, взятие остатк
            else:
                # Последний элемент (j = 0)
                a[0] = x % 10
                digit = x // 10
                counter.add(2)  # деление и взятие остатка

                # Коррекция цифры
                if digit == 9:
                    invalid_count += 1
                elif digit == 10:
                    digit = 0
                    for k in range(len(digits)-invalid_count, len(d
                        digits[k] = (digits[k] + 1) % 10
                        if digits[k] != 0:
                            break
                    invalid_count = 1
                else:
                    # Сбрасываем недействительные цифры
                    invalid_count = 0

                digits.append(digit)

    # Преобразуем цифры в строку
    result = str(digits[0]) + "."
    for d in digits[1:]:
        result += str(d)

    return result, counter
```

4. Реализуем функцию для сравнения скорости, точности и кол-ва операция для вычисления числа Pi:

In [111…

```
def compute_and_compare():
    precisions = [100, 1000, 10000]
    results = []

    for n in precisions:
        counter = SpigotCounter()
        start_time = time.time()

        pi_str, counter = spigot_pi(n, counter)
        elapsed_time = time.time() - start_time

        results.append({
            'digits': n,
            'operations': counter.ops,
            'time': elapsed_time,
            'iterations': counter.iterations
        })

        # Показываем первые и последние 20 цифр
        preview = pi_str[:30] + "..." + pi_str[-30:] if n > 60 else

        print(f"\n{n} цифр π:")
        print(f"  Время: {elapsed_time:.3f} сек")
        print(f"  Операций: {counter.ops:,}")
        print(f"  Итераций: {counter.iterations}")
        print(f"  Операций на цифру: {counter.ops/n:.1f}")
        print(f"  Первые/последние цифры: {preview}")

    return results
```

5. Реализуем функцию для сравнения с результатами Лабораторной работы №2

In [112…

```
def compare_with_lab2():
    # Данные из ЛР2
    lab2_data = {
        'Мэчин': {
            'ops': [19, 33, 51, 51, 73, 73, 99, 129, 163, 201, 243,
            'time': [1.63338114465804e-05, 1.137499938370714e-05, 1
                     1.6379808272807180e-05, 2.37580862212348e-05,
                     4.095898475497961e-05, 5.5334086901830435e-05,
                     3.75209806238667e-05, 4.241609891913507e-05, 4
                     5.03750852380738e-05, 6.25420707189739e-05, 6.
            'iterations': [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 10,
        },
        'Нилаканта': {
            'ops': [14, 28, 56, 119, 259, 553, 1197, 2576, 5558, 11
                    119644, 257959, 535857],
            'time': [3.91699722968419e-06, 4.91708484417379e-06, 1.
                     4.4791988329051896e-05, 0.0081039340886897644,
```

```
                    0.00479249999525683, 0.0211459080666082, 0.036
                    0.176286009017099, 0.846224249876097, 3.945642
                    18.23386294197726, 86.5178642991612, 388.16923
            'iterations': [2, 4, 8, 17, 37, 79, 171, 368, 794, 1710
                           17092, 36850, 76551]
        }
    }

    # Вычисляем Spigot для 1-15 цифр
    spigot_results = []
    spigot_times = []
    spigot_iterations = []

    for prec in range(1, 16):
        counter = SpigotCounter()
        start_time = time.time()
        pi_str, counter = spigot_pi(prec, counter)
        elapsed_time = time.time() - start_time

        spigot_results.append(counter.ops)
        spigot_times.append(elapsed_time)
        spigot_iterations.append(counter.iterations)

    print("ТОЧНОСТЬ | ОПЕРАЦИИ SPIGOT | ОПЕРАЦИИ МЭЧИН | ОПЕРАЦИИ Н
    print("-" * 130)

    for i in range(15):
        print(f"{i+1:^9} | {spigot_results[i]:^15} | {lab2_data['Мэ
              f"{spigot_times[i]:^15.6f} | {lab2_data['Мэчин']['tim

    return {
        'Spigot': {
            'ops': spigot_results,
            'time': spigot_times,
            'iterations': spigot_iterations
        },
        'Мэчин': lab2_data['Мэчин'],
        'Нилаканта': lab2_data['Нилаканта']
    }
```

6. Реализуем функция для вывода графиков

In [113…

```
def plot_results(spigot_results_big, comparison_data):
    plt.figure(figsize=(15, 10))

    # Извлекаем данные из comparison_data
    spigot_ops_small = comparison_data['Spigot']['ops']
    spigot_times_small = comparison_data['Spigot']['time']

    machin_ops = comparison_data['Мэчин']['ops']
    machin_times = comparison_data['Мэчин']['time']

    nilakantha_ops = comparison_data['Нилаканта']['ops']
    nilakantha_times = comparison_data['Нилаканта']['time']
```

```
    # График 1: Операции для Spigot при разной точности (большие зн
    plt.subplot(2, 2, 1)
    precisions_big = [r['digits'] for r in spigot_results_big]
    ops_big = [r['operations'] for r in spigot_results_big]
    plt.plot(precisions_big, ops_big, 'bo-', linewidth=2, markersiz
    plt.xlabel('Количество цифр')
    plt.ylabel('Арифметических операций')
    plt.title('Spigot: Операции vs Точность (большие значения)')
    plt.grid(True, alpha=0.3)

    # График 2: Время для Spigot (большие значения)
    plt.subplot(2, 2, 2)
    times_big = [r['time'] for r in spigot_results_big]
    plt.plot(precisions_big, times_big, 'ro-', linewidth=2, markers
    plt.xlabel('Количество цифр')
    plt.ylabel('Время (сек)')
    plt.title('Spigot: Время vs Точность (большие значения)')
    plt.grid(True, alpha=0.3)

    # График 3: Сравнение операций всех алгоритмов для малой точнос
    plt.subplot(2, 2, 3)
    x = list(range(1, 16))

    plt.plot(x, spigot_ops_small, 'go-', label='Spigot', linewidth=
    plt.plot(x, machin_ops, 'bo-', label='Мэчин', linewidth=2, mark
    plt.plot(x, nilakantha_ops, 'ro-', label='Нилаканта', linewidth

    plt.xlabel('Точность (количество цифр)')
    plt.ylabel('Арифметических операций')
    plt.title('Сравнение операций (1-15 цифр)')
    plt.legend()
    plt.grid(True, alpha=0.3)

    # График 4: Сравнение времени всех алгоритмов (логарифмическая
    plt.subplot(2, 2, 4)

    plt.semilogy(x, spigot_times_small, 'go-', label='Spigot', line
    plt.semilogy(x, machin_times, 'bo-', label='Мэчин', linewidth=2
    plt.semilogy(x, nilakantha_times, 'ro-', label='Нилаканта', lin

    plt.xlabel('Точность (количество цифр)')
    plt.ylabel('Время выполнения (сек, log scale)')
    plt.title('Сравнение времени (логарифмическая шкала)')
    plt.legend()
    plt.grid(True, alpha=0.3, which='both')

    plt.tight_layout()

    # Сохраняем графики
    timestamp = time.strftime("%Y%m%d_%H%M%S")
    plt.savefig(f'spigot_comparison_{timestamp}.png', dpi=150, bbox
    plt.show()

    return timestamp
```

7. Анализ операций для вычисления меньшей точности при расчете большей

In [114…

```
def analyze_operations_for_subprecision():
    target_digits = [100, 1000, 10000]

    for main_digits in target_digits:
        print(f"\nПри вычислении {main_digits} цифр:")

        for sub_digits in [10, 50, 100, 500, 1000]:
            if sub_digits <= main_digits:
                # Оценка: операции ~ O(n^2) для Spigot
                total_ops_estimate = 10 * main_digits * main_digits
                ops_for_sub = 10 * sub_digits * sub_digits
                percentage = (ops_for_sub / total_ops_estimate) * 1

                print(f"  - Для {sub_digits} цифр: ~{ops_for_sub:,}
```

8. Собираем результаты:

8.1. Вычисление Spigot для разной точности (большие значения)

In [115…

```
spigot_results_big = compute_and_compare()

json.dumps(spigot_results_big)
```

```
100 цифр π:
  Время: 0.005 сек
  Операций: 233,500
  Итераций: 100
  Операций на цифру: 2335.0
  Первые/последние цифры: 3.14159265358979323846264338 32...406286208
998627034825342117067

1000 цифр π:
  Время: 0.557 сек
  Операций: 23,335,000
  Итераций: 1000
  Операций на цифру: 23335.0
  Первые/последние цифры: 3.14159265358979323846264338 32...612001927
8766111959092164101 98

10000 цифр π:
  Время: 59.526 сек
  Операций: 2,333,350,000
  Итераций: 10000
  Операций на цифру: 233335.0
  Первые/последние цифры: 3.14159265358979323846264338 32...695968815
920550000165525637567
```

Out[115… '[{"digits": 100, "operations": 233500, "time": 0.0047233104705810
55, "iterations": 100}, {"digits": 1000, "operations": 23335000, "
time": 0.5565197467803955, "iterations": 1000}, {"digits": 10000,
"operations": 2333350000, "time": 59.526089906692505, "iteration
s": 10000}]'

```
100 цифр π:
  Время: 0.005 сек
  Операций: 233,500
  Итераций: 100
  Операций на цифру: 2335.0
  Первые/последние цифры: 3.14159265358979323846264338 32...40628620899862703482534211 7067

1000 цифр π:
  Время: 0.557 сек
  Операций: 23,335,000
  Итераций: 1000
  Операций на цифру: 23335.0
  Первые/последние цифры: 3.14159265358979323846264338 32...61200192787661119590921641 0198

10000 цифр π:
  Время: 59.526 сек
  Операций: 2,333,350,000
  Итераций: 10000
  Операций на цифру: 233335.0
  Первые/последние цифры: 3.14159265358979323846264338 32...69596881592055000016552563 7567
'[{"digits": 100, "operations": 233500, "time": 0.004723310470581055, "iterations": 100}, {"digits": 1000, "operations": 23335000, "time": 0.556519746780395
5, "iterations": 1000}, {"digits": 10000, "operations": 2333350000, "time": 59.526089906692505, "iterations": 10000}]'
```

8.2. Сравнение с ЛР2 (малые значения 1-15)

In [116…

```
comparison_data = compare_with_lab2()

json.dumps(comparison_data)
```

ТОЧНОСТЬ | ОПЕРАЦИИ SPIGOT | ОПЕРАЦИИ МЭЧИН | ОПЕРАЦИИ НИЛАКАНТА | ВРЕМЯ SPIGOT (с) | ВРЕМЯ МЭЧИН (с) | ВРЕМЯ НИЛАКАНТА (с)

ТОЧНОСТЬ	ОПЕРАЦИИ SPIGOT	ОПЕРАЦИИ МЭЧИН	ОПЕРАЦИИ НИЛАКАНТА	ВРЕМЯ SPIGOT (с)	ВРЕМЯ МЭЧИН (с)	ВРЕМЯ НИЛАКАНТА (с)
1	25	19	14	0.000004	0.000016	0.000004
2	92	33	28	0.000004	0.000011	0.000005
3	222	51	56	0.000005	0.000017	0.000013
4	380	51	119	0.000008	0.000016	0.000045
5	580	73	259	0.000011	0.000024	0.008104
6	864	73	553	0.000017	0.000026	0.000953
7	1155	99	1197	0.000023	0.000041	0.004792
8	1488	129	2576	0.000029	0.000055	0.021146
9	1926	163	5558	0.000035	0.000025	0.036717
10	2350	201	11970	0.000042	0.000038	0.176286
11	2816	243	25758	0.000051	0.000042	0.846224
12	3408	243	55559	0.000063	0.000042	3.945642
13	3965	289	119644	0.000072	0.000050	18.233863
14	4564	339	257959	0.000083	0.000063	86.517864
15	5310	339	535857	0.000096	0.000063	388.169231

Out[116… '{"Spigot": {"ops": [25, 92, 222, 380, 580, 864, 1155, 1488, 1926, 2350, 2816, 3408, 3965, 4564, 5310], "time": [4.0531158447265625e-06, 3.814697265625e-06, 5.0067901611328125e-06, 7.867813110351562e-06, 1.0967254638671875e-05, 1.6927719116210938e-05, 2.288818359375e-05, 2.9325485229492188e-05, 3.4809112548828125e-05, 4.1961669921875e-05, 5.1021575927734375e-05, 6.318092346191406e-05, 7.200241088867188e-05, 8.296966552734375e-05, 9.608268737792969e-05], "iterations": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]}, "\\u041c\\u044d\\u0447\\u0438\\u043d": {"ops": [19, 33, 51, 51, 73, 73, 99, 129, 163, 201, 243, 243, 289, 339, 339], "time": [1.6333811 4465804e-05, 1.137499938370714e-05, 1.65298424446927e-05, 1.637980827280718e-05, 2.37580862212348e-05, 2.56249139457941e-05, 4.09589 8475497961e-05, 5.5334086901830434e-05, 2.493798880311988e-05, 3.7 5209806238667e-05, 4.241609891913507e-05, 4.21250158178493e-05, 5. 03750852380738e-05, 6.25420707189739e-05, 6.25420707189739e-05], " iterations": [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 10, 11, 12, 1 2]}, "\\u041d\\u0438\\u043b\\u0430\\u043a\\u0430\\u043d\\u0442\\u0430": {"ops": [14, 28, 56, 119, 259, 553, 1197, 2576, 5558, 11970, 25758, 55559, 119644, 257959, 535857], "time": [3.91699722968419e-06, 4.91708484417379e-06, 1.2500807869675756e-05, 4.47919883290519e-05, 0.008103934088689764, 0.0009527998065576611, 0.0047924999952 5683, 0.0211459080666082, 0.0367169170853324, 0.176286009017099, 0.846224249876097, 3.945642041013343, 18.23386294197726, 86.517864 2991612, 388.1692307920026], "iterations": [2, 4, 8, 17, 37, 79, 1 71, 368, 794, 1710, 3694, 7937, 17092, 36850, 76551]}}'

ТОЧНОСТЬ | ОПЕРАЦИИ SPIGOT | ОПЕРАЦИИ МЭЧИН | ОПЕРАЦИИ НИЛАКАНТА | ВРЕМЯ SPIGOT (с) | ВРЕМЯ МЭЧИН (с) | ВРЕМЯ НИЛАКАНТА (с)

1	25	19	14	0.000004	0.000016	0.000004
2	92	33	28	0.000004	0.000011	0.000005
3	222	51	56	0.000005	0.000017	0.000013
4	380	51	119	0.000008	0.000016	0.000045
5	580	73	259	0.000011	0.000024	0.008104
6	864	73	553	0.000017	0.000026	0.000953
7	1155	99	1197	0.000023	0.000041	0.004792
8	1488	129	2576	0.000029	0.000055	0.021146
9	1926	163	5558	0.000035	0.000025	0.036717
10	2350	201	11970	0.000042	0.000038	0.176286
11	2816	243	25758	0.000051	0.000042	0.846224
12	3408	243	55559	0.000063	0.000042	3.945642
13	3965	289	119644	0.000072	0.000050	18.233863
14	4564	339	257959	0.000083	0.000063	86.517864
15	5310	339	535857	0.000096	0.000063	388.169231

'{"Spigot": {"ops": [25, 92, 222, 380, 580, 864, 1155, 1488, 1926, 2350, 2816, 3408, 3965, 4564, 5310], "time": [4.0531158447265625e-06, 3.814697265625e-06, 5.0067901611328125e-06, 7.867813110351562e-06, 1.0967254638671875e-05, 1.6927719116210938e-05, 2.288818359375e-05, 2.9325485229492188e-05, 3.4809112548828125e-05, 4.1961669921875e-05, 5.1021575927734375e-05, 6.318092346191406e-05, 7.200241088867188e-05, 8.296966552734375e-05, 9.608268737792969e-05], "iterations": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]}, "\\u041c\\u044d\\u0447\\u0438\\u043d": {"ops": [19, 33, 51, 51, 73, 73, 99, 129, 163, 201, 243, 243, 289, 339, 339], "time": [1.6333811446580 4e-05, 1.137499938370714e-05, 1.65298424446927e-05, 1.637980827280718e-05, 2.37580862212348e-05, 2.56249139457941e-05, 4.0958984754979 61e-05, 5.5334086901830434e-05, 2.493798880311988e-05, 3.752098062386 67e-05, 4.241609891913507e-05, 4.212501581784 93e-05, 5.03750852380738e-05, 6.25420707189739e-05, 6.25420707189739e-05], "iterations": [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 10, 11, 12, 12]}, "\\u041d\\u0438\\u043b\\u0430\\u043a\\u0430\\u043d\\u0442\\u0430": {"ops": [14, 28, 56, 119, 259, 553, 1197, 2576, 5558, 11970, 25758, 55559, 119644, 257959, 535857], "time": [3.91699722968419e-06, 4.91708484417379e-06, 1.2500807869675756e-05, 4.47919883290519e-05, 0.008103934088689764, 0.0009527998065576611, 0.00479249999525683, 0.02114590806 66082, 0.0367169170853324, 0.176286009017099, 0.846224249876097, 3.945642041013343, 18.23386294197726, 86.5178642991612, 388.1692307920026], "iterations": [2, 4, 8, 17, 37, 79, 171, 368, 794, 1710, 3694, 7937, 17092, 36850, 76551]}}'

8.3. Анализ операций

In [117…
```
analyze_operations_for_subprecision()
```

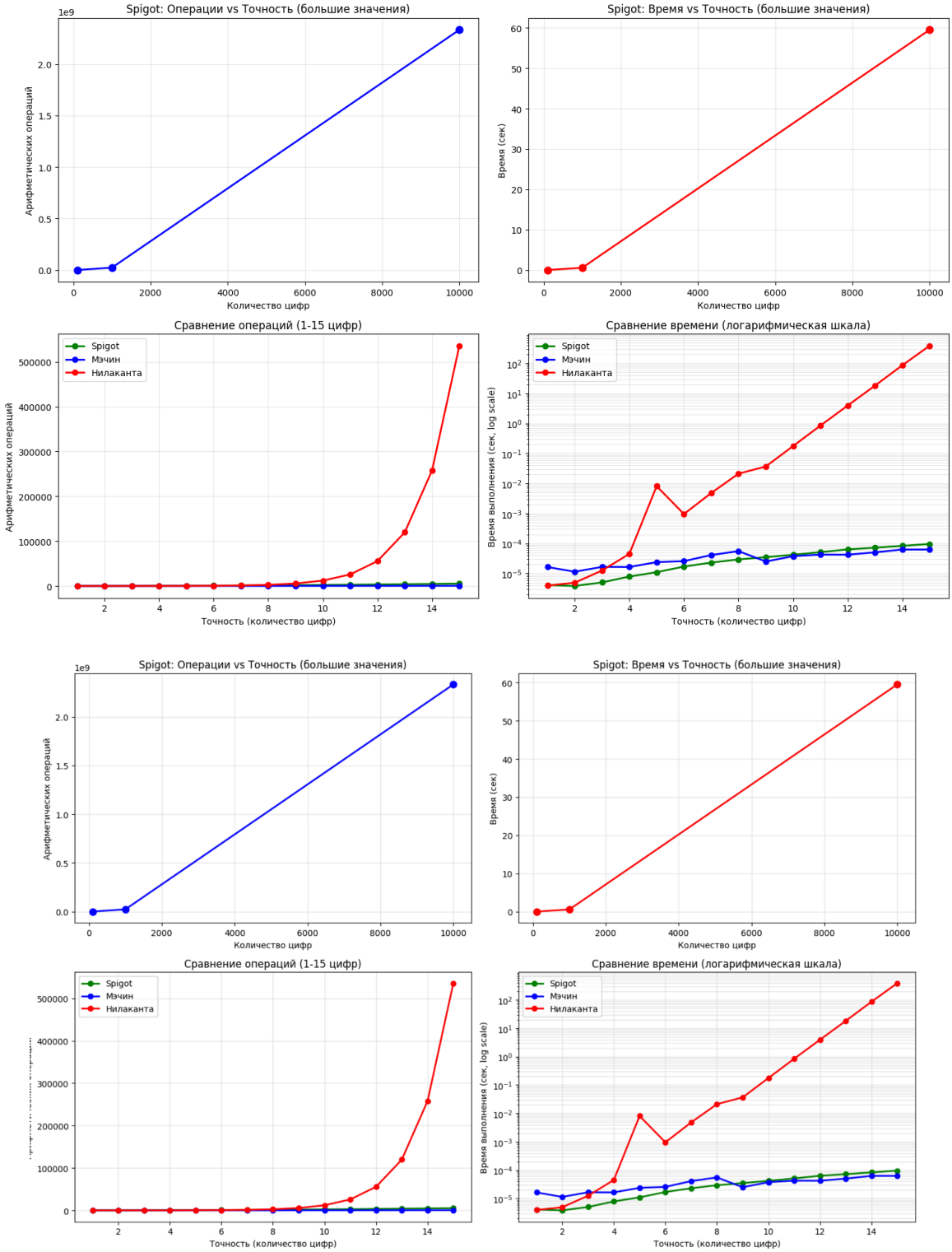
```
При вычислении 100 цифр:
  - Для 10 цифр: ~1,000 операций (1.0% от полного расчета)
  - Для 50 цифр: ~25,000 операций (25.0% от полного расчета)
  - Для 100 цифр: ~100,000 операций (100.0% от полного расчета)

При вычислении 1000 цифр:
  - Для 10 цифр: ~1,000 операций (0.0% от полного расчета)
  - Для 50 цифр: ~25,000 операций (0.2% от полного расчета)
  - Для 100 цифр: ~100,000 операций (1.0% от полного расчета)
  - Для 500 цифр: ~2,500,000 операций (25.0% от полного расчета)
  - Для 1000 цифр: ~10,000,000 операций (100.0% от полного расчета)

При вычислении 10000 цифр:
  - Для 10 цифр: ~1,000 операций (0.0% от полного расчета)
  - Для 50 цифр: ~25,000 операций (0.0% от полного расчета)
  - Для 100 цифр: ~100,000 операций (0.0% от полного расчета)
  - Для 500 цифр: ~2,500,000 операций (0.2% от полного расчета)
  - Для 1000 цифр: ~10,000,000 операций (1.0% от полного расчета)
```


```
При вычислении 100 цифр:
  - Для 10 цифр: ~1,000 операций (1.0% от полного расчета)
  - Для 50 цифр: ~25,000 операций (25.0% от полного расчета)
  - Для 100 цифр: ~100,000 операций (100.0% от полного расчета)

При вычислении 1000 цифр:
  - Для 10 цифр: ~1,000 операций (0.0% от полного расчета)
  - Для 50 цифр: ~25,000 операций (0.2% от полного расчета)
  - Для 100 цифр: ~100,000 операций (1.0% от полного расчета)
  - Для 500 цифр: ~2,500,000 операций (25.0% от полного расчета)
  - Для 1000 цифр: ~10,000,000 операций (100.0% от полного расчета)

При вычислении 10000 цифр:
  - Для 10 цифр: ~1,000 операций (0.0% от полного расчета)
  - Для 50 цифр: ~25,000 операций (0.0% от полного расчета)
  - Для 100 цифр: ~100,000 операций (0.0% от полного расчета)
  - Для 500 цифр: ~2,500,000 операций (0.2% от полного расчета)
  - Для 1000 цифр: ~10,000,000 операций (1.0% от полного расчета)
```


8.4. Графики

In [118…

```python
timestamp = plot_results(spigot_results_big, comparison_data)
```



8.5. Итоговая таблица результатов

In [119…

```
print("Spigot алгоритм (большая точность):")
print("Цифр | Операции      | Время (с)  | Итераций | Операций на ц
print("-" * 75)
for r in spigot_results_big:
    ops_per_digit = r['operations'] / r['digits']
    print(f"{r['digits']:^5} | {r['operations']:^13,} | {r['time']:

print(f"\nГрафик сохранен как: spigot_comparison_{timestamp}.png")
```