Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО».

Факультет программной инженерии и компьютерной техники

Лабораторная работа №6
Вариант №12
Численное решение обыкновенных дифференциальных уравнений

Выполнил
Путинцев Данил Денисович
Группа P3207
Проверил(а)
Преподаватель: Рыбаков Степан Дмитриевич

Санкт-Петербург 2025 год

Цели лабораторной работы

Решить задачу Коши для обыкновенных дифференциальных уравнеий численными методами

Листинг программы

```
#Необходимо реализовать методы 1, 3, 5
#1. Метод Эйлера
#3. Метод Рунге-Кутта 4-го порядка
#5. Мношаговый метод Милна

import math

equations_funcs = [
    lambda x, y: 2 * x - y,
    lambda x, y: y / x if x != 0 else float('inf'),
    lambda x, y: math.exp(-x) + y ** 2,
    lambda x, y: y * (1 - y),
    lambda x, y: math.sin(x) * y
]

def euler_method(start_points, interval, h, epsilon, equation, flag=True):
    x0, y0 = start_points
    xn = interval[1]
    f = equations_funcs[equation]

    def calculate(f, x0, y0, xn, step):
        x = [x0]
        y = [y0]
        while x[-1] < xn:
            current_x = x[-1]
            current_y = y[-1]
            delta = min(step, xn - current_x)
            y_next = current_y + delta * f(current_x, current_y)
            x.append(current_x + delta)
            y.append(y_next)
        return x, y

    x_h, y_h = calculate(f, x0, y0, xn, h)
    x_h2, y_h2 = calculate(f, x0, y0, xn, h / 2)
    max_error = 0.0
    for i in range(len(y_h)):
        idx = 2 * i
        if idx < len(y_h2):
            max_error = max(max_error, abs(y_h[i] - y_h2[idx]))
```

```
  if flag:
    print("\n" + "=" * 40)
    print(f"{'Метод Эйлера':^40}")
    print("=" * 40)
    print(f"{'x':<10}{'y (h)':<15}{'y (h/2)':<15}")
    for i in range(len(y_h)):
      h2_val = y_h2[2 * i] if 2 * i < len(y_h2) else "N/A"
      print(f"{x_h[i]:<10.3f}{y_h[i]:<15.6f}{str(h2_val):<15}")

    print(f"\nМаксимальная погрешность: {max_error:.6f}")
    print(f"Целевая точность: {epsilon:.6f}")

    if max_error > epsilon:
      print("\nТребуемая точность не достигнута!")
      print(f"Рекомендуемый шаг: {h / 2:.4f}")
    else:
      print("\nТочность в пределах допустимого!")

  return {
    "x_values": x_h,
    "y_values": y_h,
    "x_half_step": x_h2,
    "y_half_step": y_h2,
    "max_error": max_error,
    "is_precision_achieved": max_error <= epsilon
  }


def runge_kutta_method(start_points, interval, h, epsilon, equation, flag=True):
  x0, y0 = start_points
  xn = interval[1]
  f = equations_funcs[equation]

  def calculate(f, x0, y0, xn, step):
    x = [x0]
    y = [y0]
    while x[-1] < xn:
      current_x = x[-1]
      current_y = y[-1]

      k1 = step * f(current_x, current_y)
      k2 = step * f(current_x + step / 2, current_y + k1 / 2)
      k3 = step * f(current_x + step / 2, current_y + k2 / 2)
```

```
        k4 = step * f(current_x + step, current_y + k3)

        y_next = current_y + (k1 + 2 * k2 + 2 * k3 + k4) / 6

        delta = min(step, xn - current_x)
        x.append(current_x + delta)
        y.append(y_next)
    return x, y

  x_h, y_h = calculate(f, x0, y0, xn, h)
  x_h2, y_h2 = calculate(f, x0, y0, xn, h / 2)

  max_error = 0.0
  for i in range(len(y_h)):
    idx = 2 * i
    if idx < len(y_h2):
        max_error = max(max_error, abs(y_h[i] - y_h2[idx]))

  if flag:
    print("\n" + "=" * 45)
    print(f"{'Метод Рунге-Кутта 4-го порядка':^45}")
    print("=" * 45)
    print(f"{'x':<10}{'y (h)':<18}{'y (h/2)':<18}")
    for i in range(len(y_h)):
        h2_val = y_h2[2 * i] if 2 * i < len(y_h2) else "N/A"
        print(f"{x_h[i]:<10.3f}{y_h[i]:<18.6f}{str(h2_val):<18}")

  return {
    "x_values": x_h,
    "y_values": y_h,
    "x_half_step": x_h2,
    "y_half_step": y_h2,
    "max_error": max_error,
    "is_precision_achieved": max_error <= epsilon
  }


exact_solutions = [
  lambda x, x0, y0: 2 * (x - 1) + (y0 - 2 * (x0 - 1)) * math.exp(-(x - x0)),
  lambda x, x0, y0: y0 * (x / x0),
  None,
  lambda x, x0, y0: y0 / (y0 + (1 - y0) * math.exp(-(x - x0))),
  lambda x, x0, y0: y0 * math.exp(math.cos(x0) - math.cos(x))
]
```

```
def milne_method(start_points, interval, h, epsilon, equation, flag=True):
    x0, y0 = start_points
    xn = interval[1]
    f = equations_funcs[equation]
    exact_sol = exact_solutions[equation]

    if exact_sol is None:
        print("Ошибка: Для выбранного уравнения нет точного решения!")
        return None

    rk_result = runge_kutta_method(start_points, interval, h, epsilon, equation,
flag=False)
    x_rk = rk_result["x_values"]
    y_rk = rk_result["y_values"]

    if len(x_rk) < 4:
        print("Слишком мало точек! Уменьшите шаг h.")
        return None

    x = x_rk[:4]
    y = y_rk[:4]
    f_vals = [f(x[i], y[i]) for i in range(4)]

    n = 4
    while x[-1] < xn:
        x_p = x0 + n * h
        y_p = y[n - 4] + (4 * h / 3) * (2 * f_vals[n - 3] - f_vals[n - 2] + 2 * f_vals[n - 1])
        f_p = f(x_p, y_p)

        y_c = y[n - 2] + (h / 3) * (f_vals[n - 2] + 4 * f_vals[n - 1] + f_p)
        f_c = f(x_p, y_c)

        x.append(x_p)
        y.append(y_c)
        f_vals.append(f_c)
        n += 1

    exact_y = [exact_sol(xi, x0, y0) for xi in x]
    errors = [abs(y[i] - exact_y[i]) for i in range(len(y))]
    max_error = max(errors)
    if flag:
        print("\n" + "=" * 60)
```

```
        print(f"{'Метод Милна':^60}")
        print("=" * 60)
        print(f"{'x':<10}{'y (числ)':<15}{'y (точн)':<15}{'Погрешность':<15}")
        for i in range(len(x)):
            print(f"{x[i]:<10.3f}{y[i]:<15.6f}{exact_y[i]:<15.6f}{errors[i]:<15.6f}")

        print(f"\nМаксимальная погрешность: {max_error:.6f}")
        print(f"Целевая точность: {epsilon:.6f}")

    return {
        'x': x,
        'y': y,
        'exact': exact_y,
        'errors': errors,
        'max_error': max_error,
        'is_precision_achieved': max_error <= epsilon
    }


def compare_methods(start_points, interval, h, epsilon, equation):
    euler_result = euler_method(start_points, interval, h, epsilon, equation)
    rk_result = runge_kutta_method(start_points, interval, h, epsilon, equation)
    milne_result = None

    if exact_solutions[equation] is not None:
        milne_result = milne_method(start_points, interval, h, epsilon, equation)

    headers = ["x", "Эйлер", "Рунге-Кутта", "Милн", "Точное"]
    data = []
    x_values = euler_result["x_values"]

    for i in range(len(x_values)):
        x = x_values[i]

        euler_val = euler_result["y_values"][i] if i < len(euler_result["y_values"]) else "N/A"
        rk_val = rk_result["y_values"][i] if i < len(rk_result["y_values"]) else "N/A"
        milne_val = milne_result["y"][i] if milne_result and i < len(milne_result["y"]) else
"N/A"
        exact_val = exact_solutions[equation](x, *start_points) if
exact_solutions[equation] else "N/A"

        row = [
            f"{x:.3f}",
            f"{euler_val:.6f}" if isinstance(euler_val, (int, float)) else str(euler_val),
```

```
        f"{rk_val:.6f}" if isinstance(rk_val, (int, float)) else str(rk_val),
        f"{milne_val:.6f}" if isinstance(milne_val, (int, float)) else str(milne_val),
        f"{exact_val:.6f}" if isinstance(exact_val, (int, float)) else str(exact_val)
    ]
    data.append(row)

print("\n" + "=" * 85)
print(f"{'Сводная таблица результатов':^85}")
print("=" * 85)
print("{:<10} {:<15} {:<15} {:<15} {:<15}".format(*headers))
for row in data:
    print("{:<10} {:<15} {:<15} {:<15} {:<15}".format(*row))

print("\nМаксимальные погрешности:")
print(f"Эйлер: {euler_result['max_error']:.6f}")
print(f"Рунге-Кутта: {rk_result['max_error']:.6f}")
if milne_result:
    print(f"Милн: {milne_result['max_error']:.6f}")
```

Скриншоты результатов выполнения программы при различных исходных данных

0. y' = 2x - y

1. y' = y/x

2. y' = e^(-x) + y^2

3. y' = y(1-y)

4. y' = sin(x)*y

Введите номер функции: 1

Введите начальные условия в формате 'x0 y0' (например, '0 1'): -3.21451 2.21412

Введите нижнюю границу интервала: -5

Введите верхнюю границу интервала: 5

Введите шаг: 1

Введите точность: 0.1

1. Метод Эйлера

2. Метод Рунге-Кутта 4-го порядка

3. Метод Милна

4. Все методы

Выберете метод для решение ОДУ: 4

==

Метод Эйлера

==

x	y (h)	y (h/2)
-3.215	2.214120	2.21412
-2.215	1.525331	1.5253307288513644
-1.215	0.836541	0.8365414577027291
-0.215	0.147752	0.1477521865540939
0.785	-0.541037	-0.5410370845945414
1.785	-1.229826	-1.2298263557431766
2.785	-1.918616	-1.918615626891812
3.785	-2.607405	-2.607404898040447
4.785	-3.296194	-3.296194169189082
5.000	-3.443946	N/A

Максимальная погрешность: 0.000000

Целевая точность: 0.100000

Точность в пределах допустимого!

===

Метод Рунге-Кутта 4-го порядка

===

x	y (h)	y (h/2)
-3.215	2.214120	2.21412
-2.215	1.525331	1.5253307288513644
-1.215	0.836541	0.836541457702729
-0.215	0.147752	0.1477521865540939
0.785	-0.541037	-0.5410370845945429
1.785	-1.229826	-1.22982635574318
2.785	-1.918616	-1.9186156268918173

3.785	-2.607405	-2.6074048980404543
4.785	-3.296194	-3.296194169189092
5.000	-3.984983	N/A

==

Метод Милна

==

x	y (числ)	y (точн)	Погрешность
-3.215	2.214120	2.214120	0.000000
-2.215	1.525331	1.525331	0.000000
-1.215	0.836541	0.836541	0.000000
-0.215	0.147752	0.147752	0.000000
0.785	-0.541037	-0.541037	0.000000
1.785	-1.229826	-1.229826	0.000000
2.785	-1.918616	-1.918616	0.000000
3.785	-2.607405	-2.607405	0.000000
4.785	-3.296194	-3.296194	0.000000
5.785	-3.984983	-3.984983	0.000000

Максимальная погрешность: 0.000000

Целевая точность: 0.100000

==

Сводная таблица результатов

==

x	Эйлер	Рунге-Кутта	Милн	Точное
-3.215	2.214120	2.214120	2.214120	2.214120
-2.215	1.525331	1.525331	1.525331	1.525331
-1.215	0.836541	0.836541	0.836541	0.836541
-0.215	0.147752	0.147752	0.147752	0.147752

0.785	-0.541037	-0.541037	-0.541037	-0.541037
1.785	-1.229826	-1.229826	-1.229826	-1.229826
2.785	-1.918616	-1.918616	-1.918616	-1.918616
3.785	-2.607405	-2.607405	-2.607405	-2.607405
4.785	-3.296194	-3.296194	-3.296194	-3.296194
5.000	-3.443946	-3.984983	-3.984983	-3.443946

Максимальные погрешности:

Эйлер: 0.000000

Рунге-Кутта: 0.000000

Милн: 0.000000

!("/exit" to quit) Введите команду: => 4

Отправка: 4

0. y' = 2x - y

1. y' = y/x

2. y' = e^(-x) + y^2

3. y' = y(1-y)

4. y' = sin(x)*y

Введите номер функции: 3

Введите начальные условия в формате 'x0 y0' (например, '0 1'): 0.214 1.214

Введите нижнюю границу интервала: -1

Введите верхнюю границу интервала: 1

Введите шаг: 0.1

Введите точность: 0.01

1. Метод Эйлера

2. Метод Рунге-Кутта 4-го порядка

3. Метод Милна

4. Все методы

Выберете метод для решение ОДУ: 4

==

Метод Эйлера

==

x	y (h)	y (h/2)
0.214	1.214000	1.214
0.314	1.188020	1.188939434974798
0.414	1.165683	1.1672431810319777
0.514	1.146370	1.148368345490551
0.614	1.129590	1.1318789155080593
0.714	1.114952	1.1174206397679287
0.814	1.102135	1.1047027165735048
0.914	1.090879	1.0934842225541672
1.000	1.082353	1.084910464129447

Максимальная погрешность: 0.002606

Целевая точность: 0.010000

Точность в пределах допустимого!

===

Метод Рунге-Кутта 4-го порядка

===

x	y (h)	y (h/2)
0.214	1.214000	1.214
0.314	1.189771	1.1897705722806828
0.414	1.168666	1.1686655757856466
0.514	1.150204	1.150204053175046
0.614	1.133995	1.1339949515104002
0.714	1.119717	1.1197170980036066
0.814	1.107104	1.1071043156147324
0.914	1.095934	1.0959342009572277
1.000	1.086020	1.0860195532033152

==

Метод Милна

==

x	y (числ)	y (точн)	Погрешность
0.214	1.214000	1.214000	0.000000
0.314	1.189771	1.189771	0.000000
0.414	1.168666	1.168666	0.000000
0.514	1.150204	1.150204	0.000000
0.614	1.133995	1.133995	0.000000
0.714	1.119717	1.119717	0.000000
0.814	1.107104	1.107104	0.000001
0.914	1.095934	1.095934	0.000000
1.014	1.086019	1.086020	0.000001

Максимальная погрешность: 0.000001

Целевая точность: 0.010000

==

Сводная таблица результатов

==

x	Эйлер	Рунге-Кутта	Милн	Точное
0.214	1.214000	1.214000	1.214000	1.214000
0.314	1.188020	1.189771	1.189771	1.189771
0.414	1.165683	1.168666	1.168666	1.168666
0.514	1.146370	1.150204	1.150204	1.150204
0.614	1.129590	1.133995	1.133995	1.133995
0.714	1.114952	1.119717	1.119717	1.119717
0.814	1.102135	1.107104	1.107104	1.107104
0.914	1.090879	1.095934	1.095934	1.095934
1.000	1.082353	1.086020	1.086019	1.087338

Максимальные погрешности:

Эйлер: 0.002606

Рунге-Кутта: 0.000000

Милн: 0.000001

0. y' = 2x - y

1. y' = y/x

2. y' = e^(-x) + y^2

3. y' = y(1-y)

4. y' = sin(x)*y

Введите номер функции: 4

Введите начальные условия в формате 'x0 y0' (например, '0 1'): -0.94224 1.214

Введите нижнюю границу интервала: -1

Введите верхнюю границу интервала: 2

Введите шаг: 0.25

Введите точность: 0.1

1. Метод Эйлера

2. Метод Рунге-Кутта 4-го порядка

3. Метод Милна

4. Все методы

Выберете метод для решение ОДУ: 4

==

Метод Эйлера

==

x	y (h)	y (h/2)
-0.942	1.214000	1.214
-0.692	0.968506	0.9917769843228489
-0.442	0.813965	0.8513535538348989
-0.192	0.726878	0.7743887667577023
0.058	0.692159	0.749546102257266

0.308	0.702148	0.7721059080224096
0.558	0.755323	0.8433503006323486
0.808	0.855269	0.9700602118215895
1.058	1.009803	1.1638997069811818
1.308	1.229753	1.439994540151295
1.558	1.526617	1.8136093269376896
1.808	1.908238	2.2937307517090275
2.000	2.264827	2.7341974725943516

Максимальная погрешность: 0.469371

Целевая точность: 0.100000

Требуемая точность не достигнута!

Рекомендуемый шаг: 0.1250

==

Метод Рунге-Кутта 4-го порядка

==

x	y (h)	y (h/2)
-0.942	1.214000	1.214
-0.692	1.012154	1.012153398124939
-0.442	0.885240	0.8852396142394795
-0.192	0.818992	0.8189924849815365
0.058	0.805385	0.8053856884861752
0.308	0.842723	0.8427236803268394
0.558	0.935621	0.9356212821201747
0.808	1.095025	1.0950271731936876
1.058	1.337856	1.337862488564536
1.308	1.685184	1.6851998306449008
1.558	2.157275	2.157308521046587
1.808	2.763856	2.7639177135276873
2.000	3.489694	3.489788845210249

==

Метод Милна

==

x	y (числ)	y (точн)	Погрешность
-0.942	1.214000	1.214000	0.000000
-0.692	1.012154	1.012153	0.000001
-0.442	0.885240	0.885240	0.000000
-0.192	0.818992	0.818992	0.000000
0.058	0.805384	0.805386	0.000001
0.308	0.842724	0.842724	0.000001
0.558	0.935620	0.935621	0.000001
0.808	1.095020	1.095027	0.000007
1.058	1.337846	1.337863	0.000016
1.308	1.685186	1.685201	0.000014
1.558	2.157332	2.157311	0.000022
1.808	2.764041	2.763922	0.000120
2.058	3.490076	3.489795	0.000281

Максимальная погрешность: 0.000281

Целевая точность: 0.100000

==
==============

Сводная таблица результатов

==
==============

x	Эйлер	Рунге-Кутта	Милн	Точное
-0.942	1.214000	1.214000	1.214000	1.214000
-0.692	0.968506	1.012154	1.012154	1.012153
-0.442	0.813965	0.885240	0.885240	0.885240
-0.192	0.726878	0.818992	0.818992	0.818992

0.058	0.692159	0.805385	0.805384	0.805386
0.308	0.702148	0.842723	0.842724	0.842724
0.558	0.755323	0.935621	0.935620	0.935621
0.808	0.855269	1.095025	1.095020	1.095027
1.058	1.009803	1.337856	1.337846	1.337863
1.308	1.229753	1.685184	1.685186	1.685201
1.558	1.526617	2.157275	2.157332	2.157311
1.808	1.908238	2.763856	2.764041	2.763922
2.000	2.264827	3.489694	3.490076	3.313633

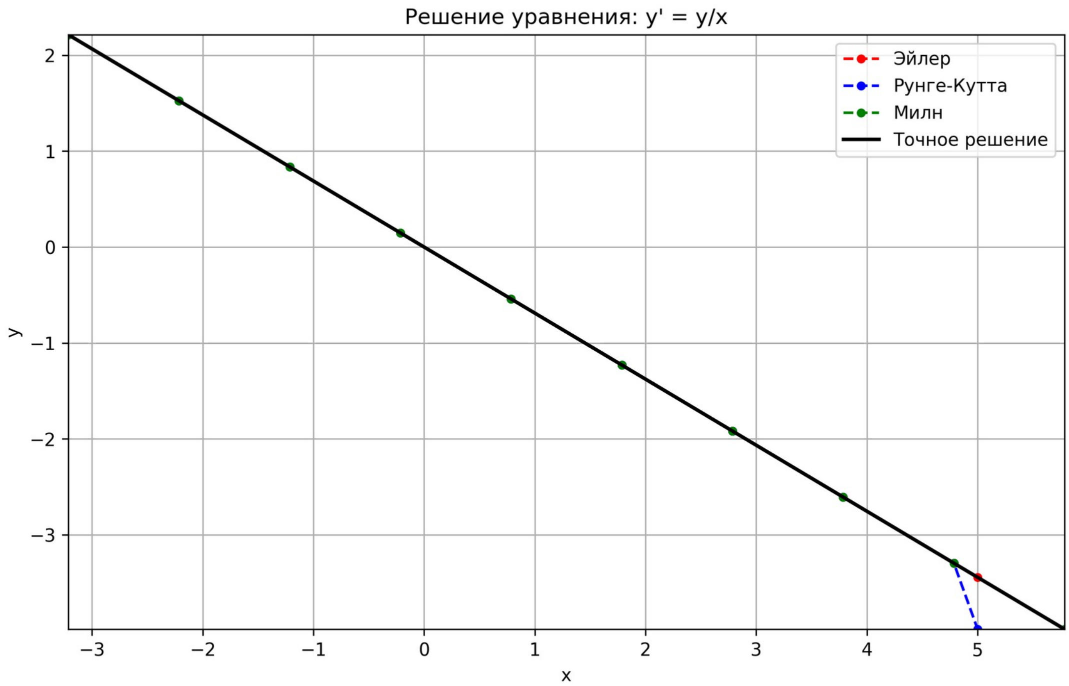
Максимальные погрешности:

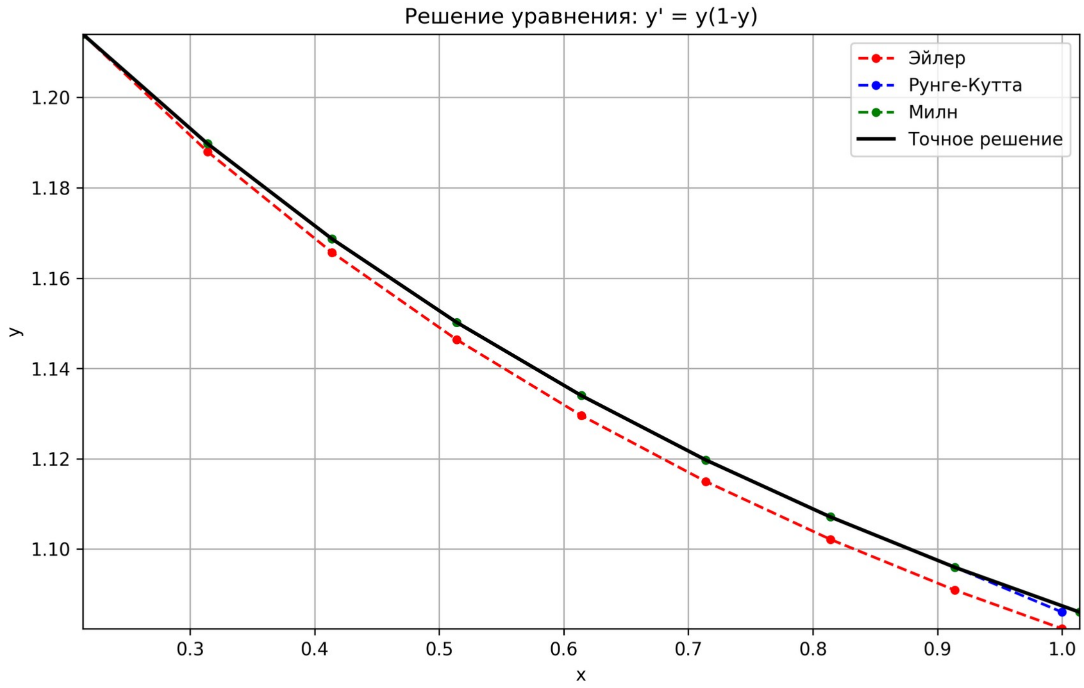
Эйлер: 0.469371

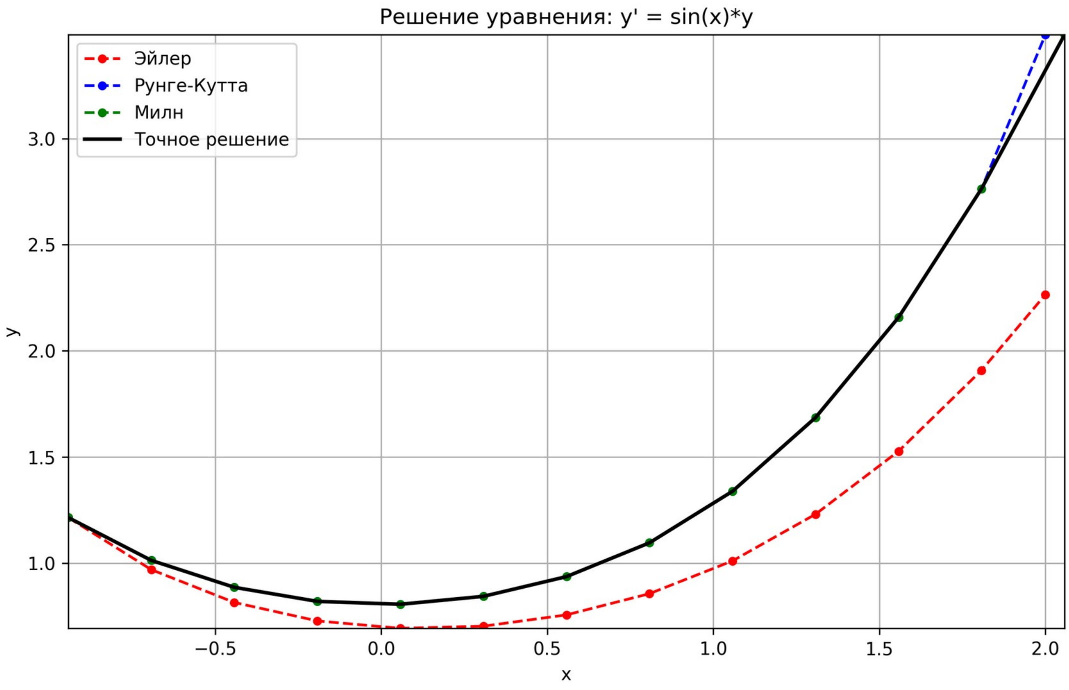
Рунге-Кутта: 0.000095

Милн: 0.000281

Графики точного решения и полученного приближенного решения







Выводы

В ходе работы были реализованы три метода решения ОДУ: Эйлера, Рунге-Кутты 4-го порядка и Милна. Сравнение показало, что метод Рунге-Кутты точнее Эйлера, а Милна эффективен для гладких решений. Визуализация подтвердила теоретические выводы. Работа позволила освоить практическое применение численных методов и оценку их точности.