МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА «ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ ІМЕНІ А. В. ДАБАГЯНА»

ЗВІТ

З спринту №2

«ІНТЕГРАЛИ ТА ДЕФЕРЕНЦІАЛИ»

ВИКОНАВ:

студент групи КН-422ч

Максим БЕЛОШИЦЬКИЙ

ПЕРЕВІРИВ:

Завідувач каф. ІСтТ

Олена НІКУЛІНА

Харків – 2023

**ЗМІСТ**

[**МЕТА** 3](#_Toc165463655)

[Завдання 1.1 4](#_Toc165463656)

[Код завдання 1.1: 4](#_Toc165463657)

[left\_rectangles\_integration.m 4](#_Toc165463658)

[Результати виконання програми 4](#_Toc165463659)

[Завдання 1.2 6](#_Toc165463660)

[Код завдання 1.2: 6](#_Toc165463661)

[right\_rectangles\_integration.m 6](#_Toc165463662)

[Результати виконання програми 7](#_Toc165463663)

[Завдання 1.3 8](#_Toc165463664)

[Код завдання 1.3: 8](#_Toc165463665)

[central\_rectangles\_integration.m 8](#_Toc165463666)

[Результати виконання програм 8](#_Toc165463667)

[Завдання 1.4 10](#_Toc165463668)

[Код завдання 1.3: 10](#_Toc165463669)

[trapezoidal\_integration.m 10](#_Toc165463670)

[Результати виконання програми 11](#_Toc165463671)

[Завдання 1.5 12](#_Toc165463672)

[Код завдання 1.5: 12](#_Toc165463673)

[simpsons\_integration.m 12](#_Toc165463674)

[Результати виконання програми 13](#_Toc165463675)

[Завдання 1.6 14](#_Toc165463676)

[Код завдання 1.6: 14](#_Toc165463677)

[euler\_method.m 14](#_Toc165463678)

[Результати виконання програми 14](#_Toc165463679)

[Завдання 1.7 16](#_Toc165463680)

[Код завдання 1.7: 16](#_Toc165463681)

[runge\_kutta\_second\_order.m 16](#_Toc165463682)

[Результати виконання програми 17](#_Toc165463683)

[Завдання 1.8 18](#_Toc165463684)

[Код завдання 1.8: 18](#_Toc165463685)

[runge\_kutta\_third\_order.m 18](#_Toc165463686)

[Результати виконання програми 19](#_Toc165463687)

[Завдання 1.9 21](#_Toc165463688)

[Код завдання 1.9: 21](#_Toc165463689)

[runge\_kutta\_forth\_order.m 21](#_Toc165463690)

[Результати виконання програми 22](#_Toc165463691)

[Завдання 2.1 24](#_Toc165463692)

[Код завдання 2.1: 24](#_Toc165463693)

[Результати виконання програми 30](#_Toc165463694)

[Завдання 2.2 31](#_Toc165463695)

[Код завдання 2.2: 31](#_Toc165463696)

[IntegralsAndDifferentialsCalculator.mlapp 31](#_Toc165463697)

[Результати виконання програми 38](#_Toc165463698)

[**ВИСНОВОК** 39](#_Toc165463699)

# **МЕТА**

Інтеграл та інтегрування є невід'ємними та важливими елементами кількостей та функцій. Інтегрування тісно пов'язане з найважливішими методами аналізу та дослідження числових функцій - середніми, границями, безмежними, нескінченно великими кількостями, диференціалами, похідними і т. д. Звичайно, виникають два типи проблем, які відображають два значення інтегрування: геометричне та аналітико-алгебраїчне. Перше - знаходження площі плоскої фігури під довільною кривою (квадратура) та знаходження об'єму (кубічна ємність). Друге - обчислення загального значення певної змінної, яка приймає змінні значення відповідно до одиниць часу, довжини і т. д.

Диференціальні рівняння (ДР) - це галузь математики, яка вивчає теорію та методи розв'язання, які встановлюють бажану функцію та її похідні різних порядків одного аргументу (звичайні диференціальні) або кількох аргументів (часткові диференціальні рівняння).

Будь-які динамічні процеси описуються диференціальними рівняннями. Наприклад, системи керування польотом для різних коптерів, штучний інтелект у грі для керування об'єктами, деформація, руйнування, фазові переходи.

У цій роботі застосовуються деякі числові методи для обчислення інтегралів та вирішення диференціальних рівнянь.

# Завдання 1.1

Розробити програмне забезпечення для методу лівих прямокутників, організуйте програму у формі функції. Програмне забезпечення може бути написане будь-якою мовою програмування.

Вхідні параметри: f(x) - функція, інтеграл якої обчислюється, a, b - межі інтегрування, N - кількість кроків. Вихідні параметри: I - значення інтеграла.

## Код завдання 1.1:

### left\_rectangles\_integration.m

function I = left\_rectangles\_integration(f, a, b, N)

% Step size

h = (b - a) / N;

% Initialize integral value

I = 0;

% Iterate over each step

for i = 1:N

% Calculate x value for the left endpoint of the rectangle

x\_i = a + (i - 1) \* h;

% Evaluate function value at x\_i

f\_i = f(x\_i);

% Add area of rectangle to integral

I = I + f\_i \* h;

end

end

## Результати виконання програми

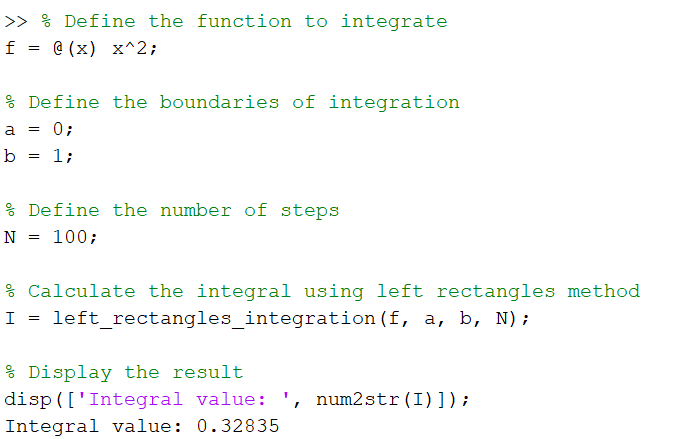


Рисунок 1.1 – «Розв’язок за методом лівих прямокутників»

Ця функція використовує метод лівих прямокутників для чисельного обчислення значення інтеграла від певної функції на певному відрізку. Ось як вона працює крок за кроком:

1. Розрахунок розміру кроку h:
   * Спочатку обчислюється розмір кроку h, який визначається як різниця між правою та лівою межами інтегрування, поділена на кількість кроків.
2. Ініціалізація змінної I:
   * Потім створюється змінна I, що представляє собою суму інтегральних значень, яку початково ініціалізується нулем.
3. Ітерація по кроках:
   * Функція проходить через кожен крок інтегрування від лівої до правої межі вказаного відрізка.
   * Для кожного кроку обчислюється значення функції в лівому кінці поточного прямокутника.
   * Потім обчислюється площа прямокутника за допомогою значення функції та розміру кроку, і ця площа додається до загальної суми інтегральних значень.
4. Повернення значення інтеграла I:
   * Після завершення ітерації значення інтеграла повертається як результат функції.

# Завдання 1.2

Розробити програмне забезпечення для методу правих прямокутників, організуйте програму у формі функції. Програмне забезпечення може бути написане будь-якою мовою програмування.

Вхідні параметри: f(x) - функція, інтеграл якої обчислюється, a, b - межі інтегрування, N - кількість кроків.

Вихідні параметри: I - значення інтеграла.

## Код завдання 1.2:

### right\_rectangles\_integration.m

function I = right\_rectangles\_integration(f, a, b, N)

% Step size

h = (b - a) / N;

% Initialize integral value

I = 0;

% Iterate over each step

for i = 1:N

% Calculate x value for the right endpoint of the rectangle

x\_i = a + i \* h;

% Evaluate function value at x\_i

f\_i = f(x\_i);

% Add area of rectangle to integral

I = I + f\_i \* h;

end

end

## Результати виконання програми

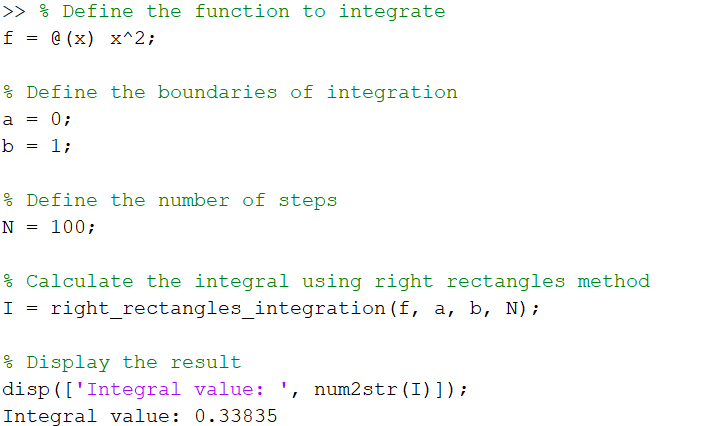


Рисунок 1.2 – «Обчислення right\_rectangles\_integration»

У цій функції реалізовано метод правих прямокутників для обчислення визначеного інтегралу.

Основні кроки роботи функції:

1. Обчислення розміру кроку h, який визначається як різниця між межами інтегрування a і b, поділена на кількість кроків N.
2. Ініціалізація змінної I, яка представляє суму площ прямокутників та є результатом інтегрування.
3. Цикл, в якому обчислюється значення інтегральної функції на кожному правому кінці прямокутника (x\_i), де x\_i обчислюється як a + i \* h.
4. Оцінка значення функції f на кожному x\_i.
5. Додавання площі поточного прямокутника до загальної суми інтегралу I.

Цей код може бути використаний для чисельного обчислення інтегралів методом правих прямокутників у програмних проектах, де потрібно здійснити чисельний аналіз або моделювання деяких фізичних процесів.

# Завдання 1.3

Розробити програмне забезпечення для методу центральних прямокутників, організуйте програму у формі функції. Програмне забезпечення може бути написане будь-якою мовою програмування.

Вхідні параметри: f(x) - функція, інтеграл якої обчислюється, a, b - межі інтегрування, N - кількість кроків.

Вихідні параметри: I - значення інтеграла.

## Код завдання 1.3:

### central\_rectangles\_integration.m

function I = central\_rectangles\_integration(f, a, b, N)

% Step size

h = (b - a) / N;

% Initialize integral value

I = 0;

% Iterate over each step

for i = 1:N

% Calculate x value for the midpoint of the rectangle

x\_i = a + (i - 0.5) \* h;

% Evaluate function value at x\_i

f\_i = f(x\_i);

% Add area of rectangle to integral

I = I + f\_i \* h;

end

end

## Результати виконання програм

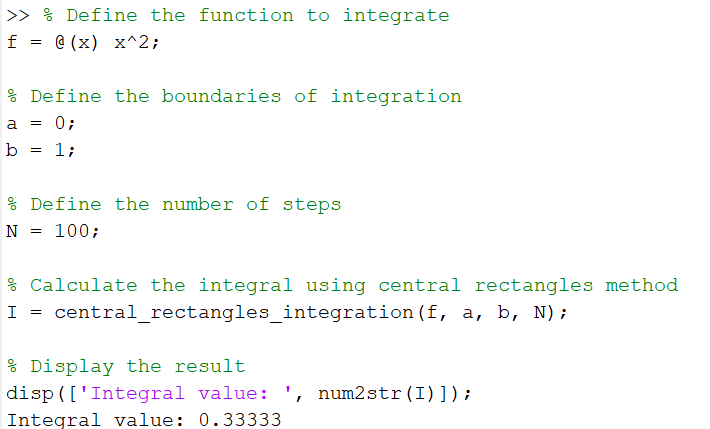


Рисунок 1.3 – «Обчислення central\_rectangles\_integration»

У цій функції реалізовано метод центральних прямокутників для обчислення визначеного інтегралу.

Основні кроки роботи функції:

1. Обчислення розміру кроку h, який визначається як різниця між межами інтегрування a і b, поділена на кількість кроків N.
2. Ініціалізація змінної I, яка представляє суму площ прямокутників та є результатом інтегрування.
3. Цикл, в якому обчислюється значення інтегральної функції в середині кожного прямокутника (x\_i), де x\_i обчислюється як a + (i - 0.5) \* h.
4. Оцінка значення функції f в кожній точці x\_i.
5. Додавання площі поточного прямокутника до загальної суми інтегралу I.

Цей код може бути використаний для чисельного обчислення інтегралів методом центральних прямокутників у програмних проектах, де потрібно здійснити чисельний аналіз або моделювання деяких фізичних процесі

# Початок форми

# Завдання 1.4

Розробити програмне забезпечення для трапецієвого методу, організуйте програму у формі функції. Програмне забезпечення може бути написане будь-якою мовою програмування.

Вхідні параметри: f(x) - функція, інтеграл якої обчислюється, a, b - межі інтегрування, N - кількість кроків.

Вихідні параметри: I - значення інтеграла.

## Код завдання 1.3:

### trapezoidal\_integration.m

function I = trapezoidal\_integration(f, a, b, N)

% Step size

h = (b - a) / N;

% Initialize integral value

I = 0;

% Evaluate function at endpoints

f\_a = f(a);

f\_b = f(b);

% Iterate over each step

for i = 1:(N-1)

% Calculate x value for current step

x\_i = a + i \* h;

% Evaluate function value at x\_i

f\_i = f(x\_i);

% Add area of trapezoid to integral

I = I + f\_i;

end

% Add contributions from endpoints

I = h \* (0.5 \* (f\_a + f\_b) + I);

end

## Результати виконання програми

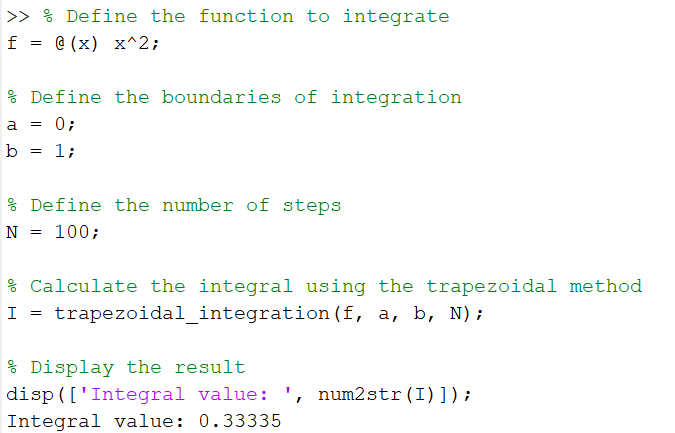


Рисунок 1.4 – «Обчислення trapezoidal\_integration»

У цій функції реалізовано метод трапецій для обчислення визначеного інтегралу.

Основні кроки роботи функції:

1. Обчислення розміру кроку h, який визначається як різниця між межами інтегрування a і b, поділена на кількість кроків N.
2. Ініціалізація змінної I, яка представляє суму значень функції у вузлах інтегрування.
3. Оцінка значень функції f у кожній вузловій точці крім крайніх точок a та b.
4. Додавання значень функції у вузлах інтегрування до загальної суми I.
5. Додавання внесків від крайніх точок a та b до загальної суми I, використовуючи формулу для методу трапецій.

Цей код може бути використаний для чисельного обчислення інтегралів методом трапецій у програмних проектах, де потрібно здійснити чисельний аналіз або моделювання деяких фізичних процесів.

# Завдання 1.5

Розробити програмне забезпечення для методу Сімпсона, організуйте програму у формі функції. Програмне забезпечення може бути написане будь-якою мовою програмування.

Вхідні параметри: f(x) - функція, інтеграл якої обчислюється, a, b - межі інтегрування, N - кількість кроків.

Вихідні параметри: I - значення інтеграла.

## Код завдання 1.5:

### simpsons\_integration.m

function I = simpsons\_integration(f, a, b, N)

% Step size

h = (b - a) / N;

% Initialize integral value

I = 0;

% Evaluate function at endpoints

f\_a = f(a);

f\_b = f(b);

% Add contributions from endpoints

I = I + f\_a + f\_b;

% Iterate over each step

for i = 1:(N-1)

% Calculate x value for current step

x\_i = a + i \* h;

% Evaluate function value at x\_i

f\_i = f(x\_i);

% Add contribution from current step

if mod(i,2) == 0

% If i is even

I = I + 2 \* f\_i;

else

% If i is odd

I = I + 4 \* f\_i;

end

end

% Multiply by h/3

I = (h / 3) \* I;

end

## Результати виконання програми

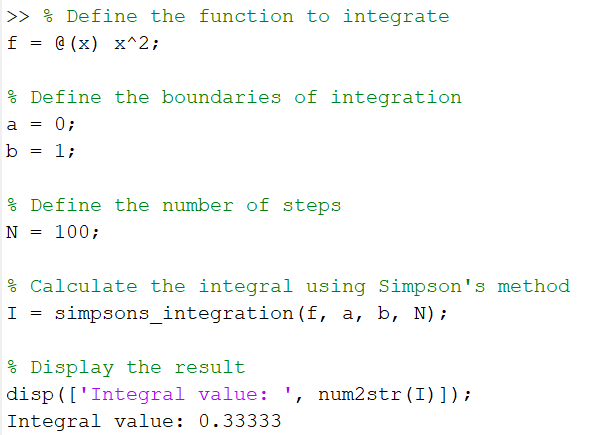


Рисунок 1.5 – «Обчислення simpsons\_integration.m»

У цій функції реалізовано метод Сімпсона для обчислення визначеного інтегралу.

Основні кроки роботи функції:

1. Обчислення розміру кроку h, який визначається як різниця між межами інтегрування a і b, поділена на кількість кроків N.
2. Ініціалізація змінної I, яка представляє суму значень функції у вузлах інтегрування.
3. Оцінка значень функції f у кожній вузловій точці крім крайніх точок a та b.
4. Додавання внесків від крайніх точок a та b до загальної суми I.
5. Додавання внесків від кожного проміжного кроку до загальної суми I в залежності від парності індексу. Якщо індекс парний, то внесок дорівнює двічі значенню функції, якщо непарний - чотири рази.
6. Після завершення циклу, загальна сума I множиться на h/3, що є кроком методу Сімпсона.

Цей код може бути використаний для чисельного обчислення інтегралів методом Сімпсона у програмних проектах, де потрібно здійснити чисельний аналіз або моделювання фізичних процесів.

# Завдання 1.6

Розробити програмне забезпечення для методу Ейлера, організуйте програму у формі функції. Програмне забезпечення може бути написане будь-якою мовою програмування.

Вхідні параметри: 𝑓(𝑥,𝑦) - функція, яка задається справа від рівняння; 𝑎 та 𝑏 - межі відрізка, де знаходиться розв'язок; 𝑦0​ - початкове значення функції; 𝑁 - кількість кроків.

Вихідні параметри: 𝑦 - масив значень функції на заданому відрізку.

## Код завдання 1.6:

### euler\_method.m

function [x, y] = euler\_method(f, a, b, y0, N)

% Step size

h = (b - a) / N;

% Initialize arrays to store x and y values

x = zeros(1, N+1);

y = zeros(1, N+1);

% Set initial values

x(1) = a;

y(1) = y0;

% Perform Euler's method

for i = 1:N

x(i+1) = x(i) + h;

y(i+1) = y(i) + h \* f(x(i), y(i));

end

end

## Результати виконання програми

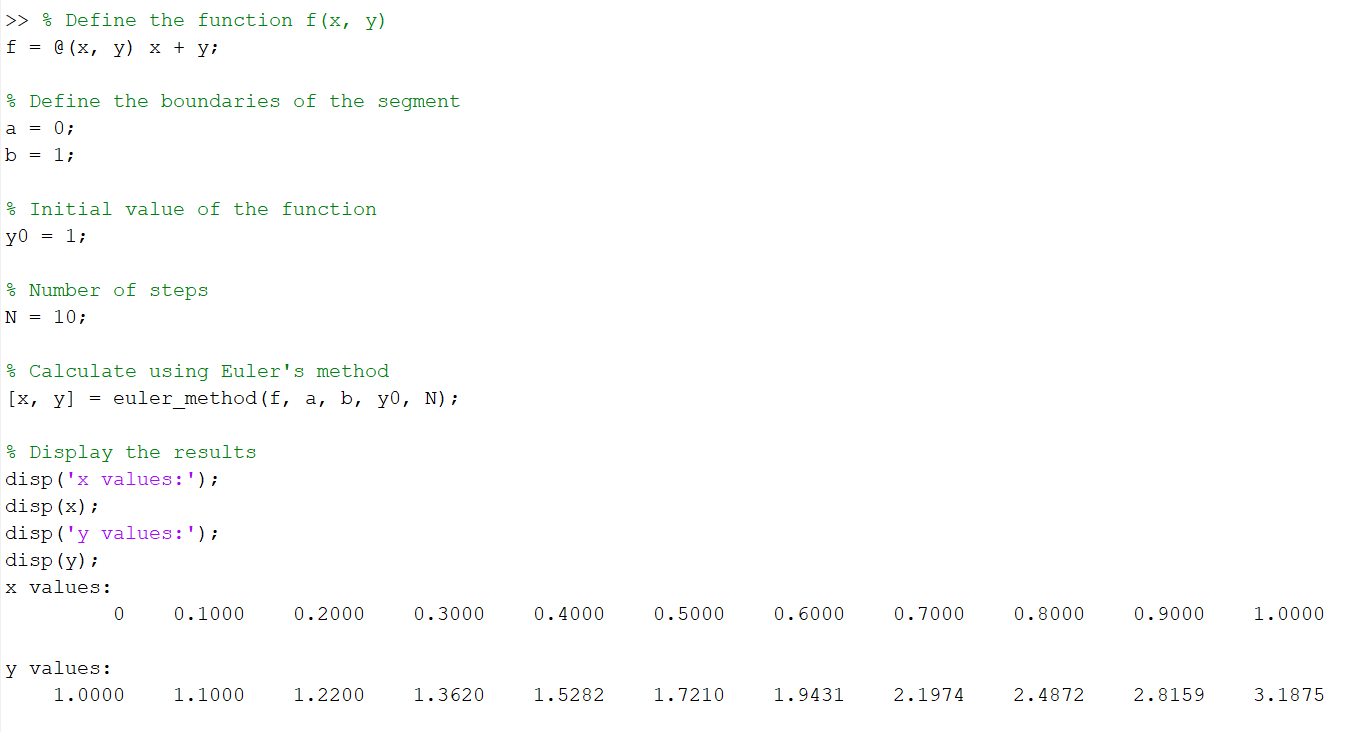


Рисунок 1.6 – «Обчислення euler\_method»

У цій функції реалізовано метод Ейлера для чисельного розв'язання звичайних диференціальних рівнянь (ЗДР) першого порядку.

Основні кроки роботи функції:

1. Обчислення розміру кроку h, який визначається як різниця між межами інтервалу a і b, поділена на кількість кроків N.
2. Ініціалізація масивів для зберігання значень x і y, де x - це значення незалежної змінної, а y - значення функції, яка задовольняє диференціальне рівняння.
3. Встановлення початкових значень x(1) і y(1).
4. Виконання методу Ейлера за допомогою циклу, де на кожному кроці обчислюється наступне значення x і використовується для обчислення відповідного значення y з використанням формули 𝑦𝑖+1=𝑦𝑖+ℎ⋅𝑓(𝑥𝑖,𝑦𝑖)*yi*+1​=*yi*​+*h*⋅*f*(*xi*​,*yi*​), де 𝑓(𝑥𝑖,𝑦𝑖) - функція, що задає диференціальне рівняння.
5. Після завершення циклу повертаються масиви x і y, які містять значення незалежної та залежної змінних відповідно на всіх кроках методу Ейлера.

Цей код може бути використаний для чисельного розв'язання різноманітних диференціальних рівнянь у програмних проектах, де необхідно моделювання динамічних процесів або аналіз систем з різними змінними.

# Завдання 1.7

Розробити програмне забезпечення для методу Рунге-Кутта другого порядку, організуйте програму у формі функції. Програмне забезпечення може бути написане будь-якою мовою програмування.

Вхідні параметри: 𝑓(𝑥,𝑦) - функція, яка задається справа від рівняння; 𝑎 та 𝑏 - межі відрізка, де знаходиться розв'язок; 𝑦0​ - початкове значення функції; 𝑁 - кількість кроків.

Вихідні параметри: 𝑦 - масив значень функції на заданому відрізку.

## Код завдання 1.7:

### runge\_kutta\_second\_order.m

function [x, y] = runge\_kutta\_second\_order(f, a, b, y0, N)

% Step size

h = (b - a) / N;

% Initialize arrays to store x and y values

x = zeros(1, N+1);

y = zeros(1, N+1);

% Set initial values

x(1) = a;

y(1) = y0;

% Perform Runge-Kutta second order method

for i = 1:N

k1 = h \* f(x(i), y(i));

k2 = h \* f(x(i) + h, y(i) + k1);

y(i+1) = y(i) + 0.5 \* (k1 + k2);

x(i+1) = x(i) + h;

end

end

## Результати виконання програми

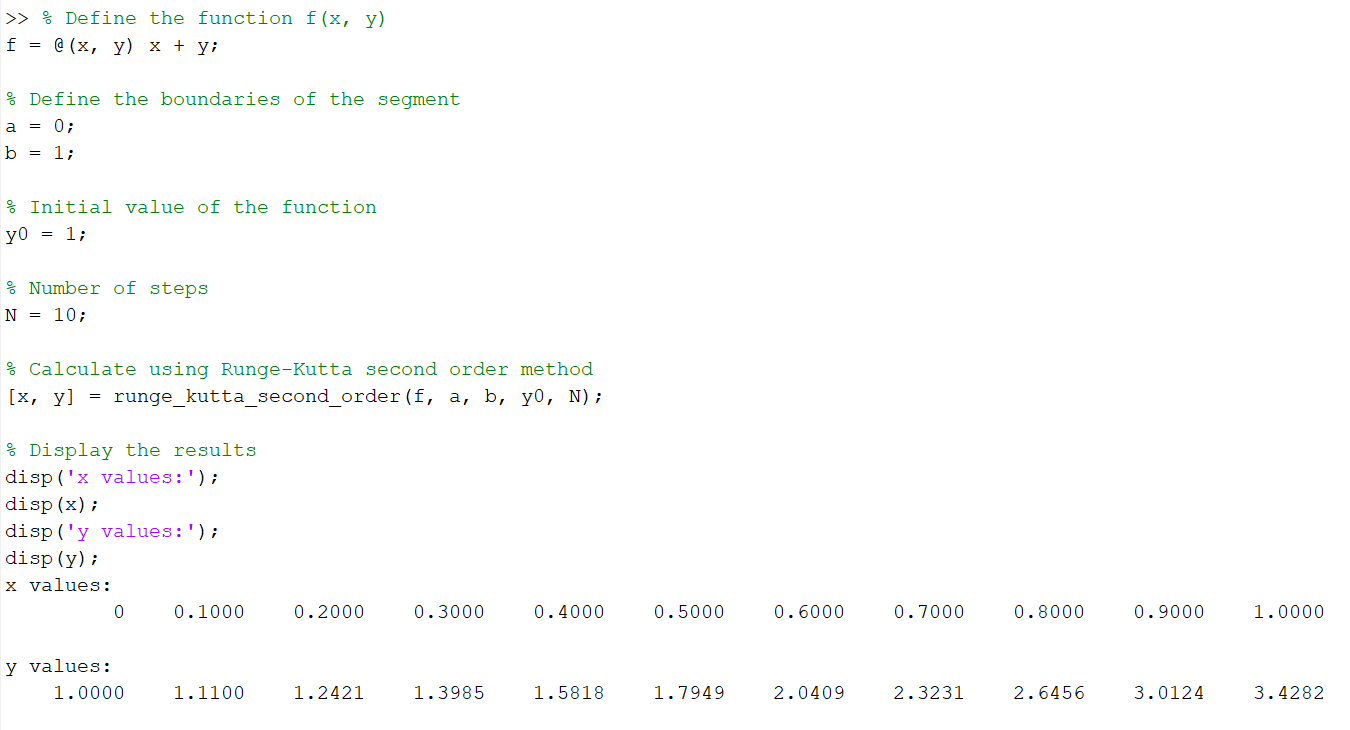


Рисунок 1.7 – «Обчислення runge\_kutta\_second\_order»

Ця функція реалізує метод другого порядку Рунге-Кутта для чисельного розв'язання звичайних диференціальних рівнянь (ЗДР) другого порядку.

1. Ініціалізація змінних: Функція ініціалізує крок ℎ та масиви 𝑥 та 𝑦, щоб зберігати значення аргументу і функції відповідно.
2. Встановлення початкових значень: Встановлюються початкові значення 𝑥 та 𝑦 для подальшого використання у циклі.
3. Виконання методу Рунге-Кутта другого порядку: Для кожного кроку циклу обчислюються коефіцієнти 𝑘1​ та 𝑘2 згідно з формулами методу Рунге-Кутта. Потім за допомогою цих коефіцієнтів обчислюється нове значення функції 𝑦*y* для наступного кроку 𝑥.
4. Оновлення значень аргументу 𝑥: Значення аргументу 𝑥 також оновлюються для наступного кроку.
5. Повернення результатів: Після завершення циклу повертаються масиви 𝑥 та 𝑦, що містять значення аргументу та відповідні значення функції на кожному кроці.

# Завдання 1.8

Розробити програмне забезпечення для методу Рунге-Кутта третього порядку, організуйте програму у формі функції. Програмне забезпечення може бути написане будь-якою мовою програмування.

Вхідні параметри: 𝑓(𝑥,𝑦) - функція, яка задається справа від рівняння; 𝑎 та 𝑏 - межі відрізка, де знаходиться розв'язок; 𝑦0​ - початкове значення функції; 𝑁 - кількість кроків.

Вихідні параметри: 𝑦 - масив значень функції на заданому відрізку.

## Код завдання 1.8:

### runge\_kutta\_third\_order.m

function [x, y] = runge\_kutta\_third\_order(f, a, b, y0, N)

% Step size

h = (b - a) / N;

% Initialize arrays to store x and y values

x = zeros(1, N+1);

y = zeros(1, N+1);

% Set initial values

x(1) = a;

y(1) = y0;

% Perform Runge-Kutta third order method

for i = 1:N

k1 = h \* f(x(i), y(i));

k2 = h \* f(x(i) + h/2, y(i) + k1/2);

k3 = h \* f(x(i) + h, y(i) - k1 + 2\*k2);

y(i+1) = y(i) + (k1 + 4\*k2 + k3) / 6;

x(i+1) = x(i) + h;

end

end

## Результати виконання програми

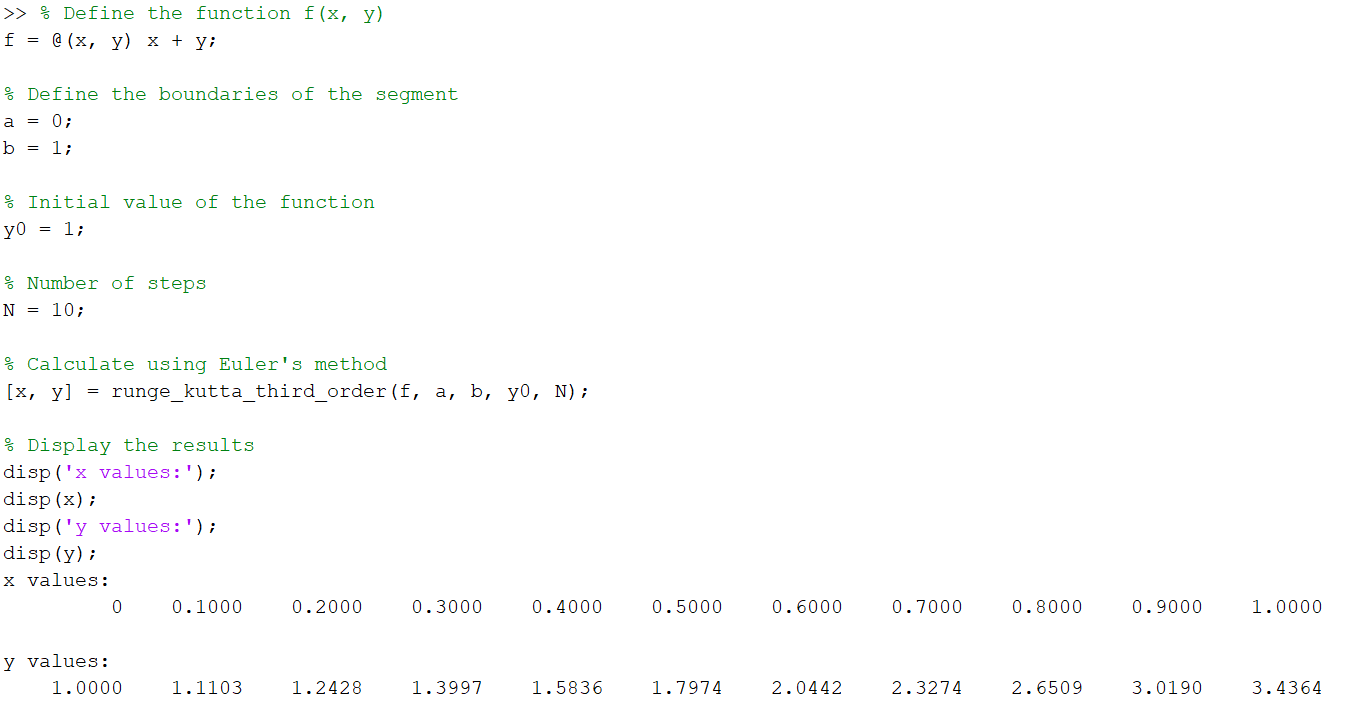


Рисунок 1.8 – «Обчислення runge\_kutta\_third \_order»

Ця функція реалізує метод Рунге-Кутта третього порядку для чисельного розв'язання звичайних диференціальних рівнянь (ЗДР) першого порядку. Ось детальний опис роботи цієї функції:

1. Вхідні параметри:

f: Функція, що визначає праву частину диференціального рівняння.

a, b: Межі відрізка, на якому шукається розв'язок.

y0: Початкове значення функції у точці x = a.

N: Кількість кроків, що використовуються для розбиття відрізка на підінтервали.

1. Вихідні параметри:

x: Масив, який містить значення аргументу x на кожному кроці.

y: Масив, який містить значення функції y(x) на відповідних кроках.

1. Опис алгоритму:

Обчислюємо розмір кроку h за формулою (b - a) / N.

Ініціалізуємо масиви x та y для збереження значень аргументу та функції відповідно.

Встановлюємо початкові значення x(1) = a та y(1) = y0.

Проводимо ітерації для кожного кроку від i = 1 до N.

На кожному кроці обчислюємо коефіцієнти k1, k2 та k3 за формулами методу Рунге-Кутта третього порядку.

Використовуючи обчислені коефіцієнти, знаходимо значення функції y на наступному кроці за формулою, що випливає з методу Рунге-Кутта.

Переходимо до наступного кроку, збільшуючи значення x на h.

Повертаємо масиви x та y як вихідні дані.

# Завдання 1.9

Розробити програмне забезпечення для методу Рунге-Кутта четвертого порядку, організуйте програму у формі функції. Програмне забезпечення може бути написане будь-якою мовою програмування.

Вхідні параметри: 𝑓(𝑥,𝑦) - функція, яка задається справа від рівняння; 𝑎 та 𝑏 - межі відрізка, де знаходиться розв'язок; 𝑦0​ - початкове значення функції; 𝑁 - кількість кроків.

Вихідні параметри: 𝑦 - масив значень функції на заданому відрізку.

## Код завдання 1.9:

### runge\_kutta\_forth\_order.m

function [x\_values, y\_values] = runge\_kutta\_fourth\_order(f, a, b, y0, N)

% Step size

h = (b - a) / N;

% Initialize arrays to store x and y values

x\_values = zeros(1, N+1);

y\_values = zeros(1, N+1);

% Set initial values

x\_values(1) = a;

y\_values(1) = y0;

% Perform fourth-order Runge-Kutta method

for i = 1:N

k1 = h \* f(x\_values(i), y\_values(i));

k2 = h \* f(x\_values(i) + h/2, y\_values(i) + k1/2);

k3 = h \* f(x\_values(i) + h/2, y\_values(i) + k2/2);

k4 = h \* f(x\_values(i) + h, y\_values(i) + k3);

y\_values(i+1) = y\_values(i) + (k1 + 2\*k2 + 2\*k3 + k4) / 6;

x\_values(i+1) = x\_values(i) + h;

end

end

## Результати виконання програми

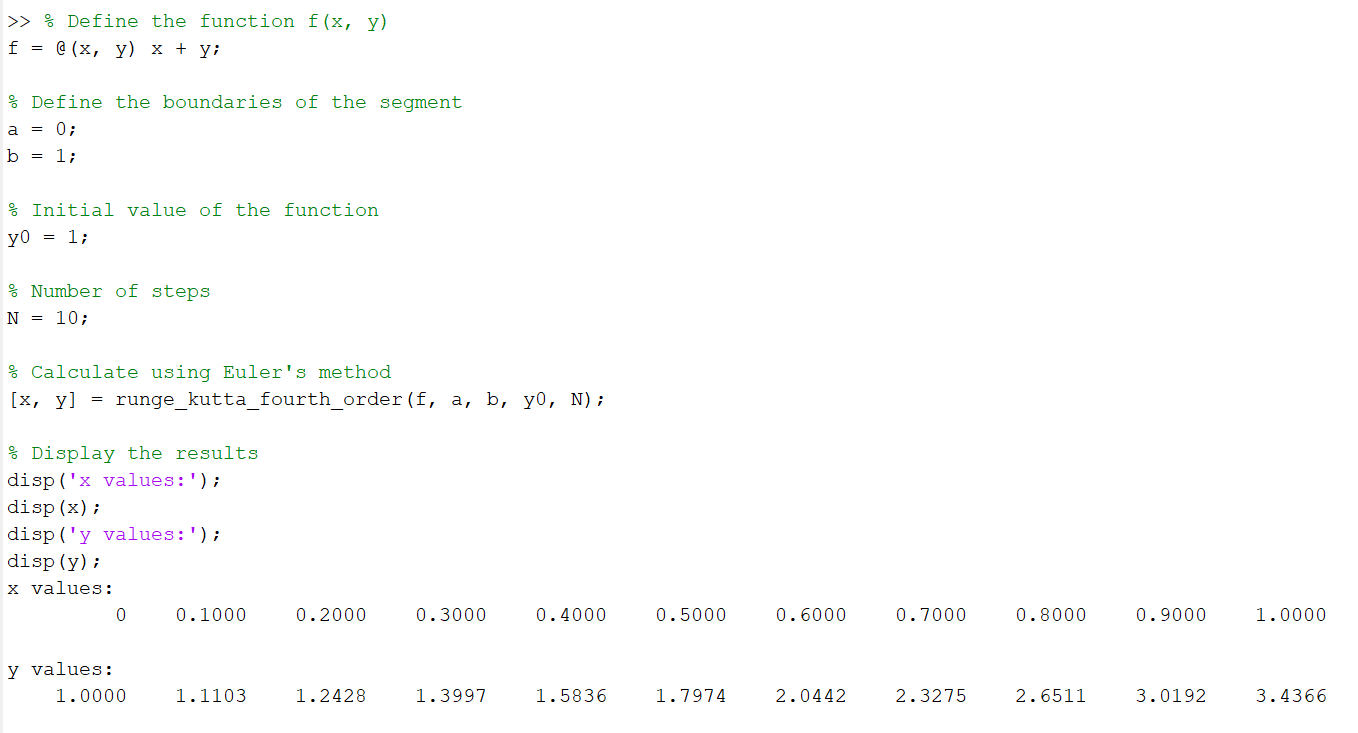


Рисунок 1.9 – «Обчислення runge\_kutta\_fourth \_order»

Ця функція реалізує метод Рунге-Кутта четвертого порядку для чисельного розв'язання звичайних диференціальних рівнянь (ЗДР) першого порядку. Ось детальний опис роботи цієї функції:

1. Вхідні параметри:

f: Функція, що визначає праву частину диференціального рівняння.

a, b: Межі відрізка, на якому шукається розв'язок.

y0: Початкове значення функції у точці x = a.

N: Кількість кроків, що використовуються для розбиття відрізка на підінтервали.

1. Вихідні параметри:

x\_values: Масив, який містить значення аргументу x на кожному кроці.

y\_values: Масив, який містить значення функції y(x) на відповідних кроках.

1. Опис алгоритму:

Обчислюємо розмір кроку h за формулою (b - a) / N.

Ініціалізуємо масиви x\_values та y\_values для збереження значень аргументу та функції відповідно.

Встановлюємо початкові значення x\_values(1) = a та y\_values(1) = y0.

Проводимо ітерації для кожного кроку від i = 1 до N.

На кожному кроці обчислюємо коефіцієнти k1, k2, k3 та k4 за формулами методу Рунге-Кутта четвертого порядку.

Використовуючи обчислені коефіцієнти, знаходимо значення функції y на наступному кроці за формулою, що випливає з методу Рунге-Кутта.

Переходимо до наступного кроку, збільшуючи значення x на h.

Повертаємо масиви x\_values та y\_values як вихідні дані.

# Завдання 2.1

Реалізуйте графічний інтерфейс користувача для програмного продукту. Програмний продукт призначений для вирішення двох завдань, тому користувач повинен вибрати обчислення визначеного інтегралу або розв'язання диференціального рівняння. Забезпечте можливість вибору будь-якого методу для вирішення вибраної проблеми. Вибір цільових функцій здійснюється зі списку, наданого розробником програмного продукту.

## Код завдання 2.1:

classdef IntegralsAndDifferentialsCalculator < matlab.apps.AppBase

% Properties that correspond to app components

properties (Access = public)

UIFigure matlab.ui.Figure

LowerboundEditField matlab.ui.control.NumericEditField

LowerboundEditFieldLabel matlab.ui.control.Label

UpperboundEditField matlab.ui.control.NumericEditField

UpperboundEditFieldLabel matlab.ui.control.Label

ProblemsListBox matlab.ui.control.ListBox

ListofproblemsListBoxLabel matlab.ui.control.Label

ResultTextArea matlab.ui.control.TextArea

OutputTextAreaLabel matlab.ui.control.Label

y0EditField matlab.ui.control.NumericEditField

y0startvalueEditFieldLabel matlab.ui.control.Label

StepEditField matlab.ui.control.NumericEditField

StepnumberEditFieldLabel matlab.ui.control.Label

SolveButton matlab.ui.control.Button

ProblemTypeButtonGroup matlab.ui.container.ButtonGroup

DifferentialButton matlab.ui.control.RadioButton

IntegralButton matlab.ui.control.RadioButton

MethodsDropDown matlab.ui.control.DropDown

TakeaMethodDropDownLabel matlab.ui.control.Label

end

% Callbacks that handle component events

methods (Access = private)

% Selection changed function: ProblemTypeButtonGroup

function ProblemTypeButtonGroupValueChanged(app, event)

% Get the selected problem type

problemType = app.ProblemTypeButtonGroup.SelectedObject.Text;

% Show/hide numeric fields based on problem type

if strcmp(problemType, 'Integral')

app.y0EditField.Visible = 'off'; % Hide y0 start value field

app.ProblemsListBox.Items = {'f(x)=exp(-x)', 'f(x)=sin(x)', 'f(x)=exp(-x^2)', 'f(x)=exp(-4x-x^3)'}; % Set list box items for integrals

elseif strcmp(problemType, 'Differential')

app.y0EditField.Visible = 'on'; % Show y0 start value field

app.ProblemsListBox.Items = {'y''=-xy', 'y''=-2xy', 'y''=-x(y^2)', 'y''=-2x(y^2)'}; % Set list box items for differentials

end

% Update dropdown menu based on problem type

if strcmp(problemType, 'Integral')

% Show only methods for solving integrals in dropdown

integralsMethods = {'Left Rectangle', 'Right Rectangle', 'Central Rectangle', 'Trapeziodal', 'Simpsons'};

app.MethodsDropDown.Items = integralsMethods;

elseif strcmp(problemType, 'Differential')

% Show only methods for solving differentials in dropdown

differentialsMethods = {'Euler', 'Runge-Kutta 2nd Order', 'Runge-Kutta 3rd Order', 'Runge-Kutta 4th Order'};

app.MethodsDropDown.Items = differentialsMethods;

end

end

% Button pushed function: SolveButton

function SolveButtonPushed(app, event)

% Get the selected problem from the ProblemsListBox

selected\_problem = app.ProblemsListBox.Value;

% Determine the type of problem (integral or differential)

problem\_type = app.ProblemTypeButtonGroup.SelectedObject.Text;

% Prompt the user to select the method

selected\_method = app.MethodsDropDown.Value;

% Parse the selected problem and extract parameters

if strcmp(problem\_type, 'Integral')

f = {

@(x) exp(-x), % f(x) = exp(-x)

@(x) sin(x), % f(x) = sin(x)

@(x) exp(-x^2), % f(x) = exp(-x^2)

@(x) exp(-4\*x - x^3) % f(x) = exp(-4x - x^3)

};

% Initialize the result string

result\_string = '';

% Iterate over each integral function

for i = 1:numel(f)

integral\_function = f{i};

I = left\_rectangles\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

% Perform integration using the selected method

switch selected\_method

case 'Left Rectangle'

% Solve integral problem using left rectangles method

I = left\_rectangles\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

case 'Right Rectangle'

% Solve integral problem using right rectangles method

I = right\_rectangles\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

case 'Central Rectangle'

% Solve integral problem using central rectangles method

I = central\_rectangles\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

case 'Trapezoidal'

% Solve integral problem using trapezoidal method

I = trapezoidal\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

case 'Simpsons'

% Solve integral problem using simpsons method

I = simpsons\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

end

% Append the result for the current integral and method to the result string

result\_string = [result\_string sprintf('Integral value for function %d using method %s: %f\n', i, selected\_method, I)];

end

% Update the ResultTextArea with the result string

app.ResultTextArea.Value = result\_string;

else

f = {

@(x, y) -x\*y, % y' = -xy

@(x, y) -2\*x\*y, % y' = -2xy

@(x, y) -x\*(y^2), % y' = -x(y^2)

@(x, y) -2\*x\*(y^2) % y' = -2x(y^2)

};

% Initialize variables for storing results

result\_string = '';

% Iterate over each differential function

for i = 1:numel(f)

differential\_function = f{i};

% Initialize variables for storing results for the current differential equation

x\_values = [];

y\_values = [];

% Solve differential problem using the selected method

switch selected\_method

case 'Euler'

% Solve differential problem using Euler's method

[x\_values, y\_values] = euler\_method(differential\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.y0EditField.Value, app.StepEditField.Value);

case 'Runge-Kutta (2nd order)'

% Solve differential problem using Runge-Kutta of the second order method

[x\_values, y\_values] = runge\_kutta\_second\_order(differential\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.y0EditField.Value, app.StepEditField.Value);

case 'Runge-Kutta (3rd order)'

% Solve differential problem using Runge-Kutta of the third order method

[x\_values, y\_values] = runge\_kutta\_third\_order(differential\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.y0EditField.Value, app.StepEditField.Value);

case 'Runge-Kutta (4th order)'

% Solve differential problem using Runge-Kutta of the forth order method

[x\_values, y\_values] = runge\_kutta\_fourth\_order(differential\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.y0EditField.Value, app.StepEditField.Value);

end

% Append the results for the current differential and method to the result string

result\_string = [result\_string sprintf('x values for differential %d using method %s: %s\n', i, selected\_method, num2str(x\_values))];

result\_string = [result\_string sprintf('y values for differential %d using method %s: %s\n\n', i, selected\_method, num2str(y\_values))];

end

% Display the result for all differentials and methods

app.ResultTextArea.Value = result\_string;

end

end

end

% Component initialization

methods (Access = private)

% Create UIFigure and components

function createComponents(app)

% Create UIFigure and hide until all components are created

app.UIFigure = uifigure('Visible', 'off');

app.UIFigure.Color = [1 1 1];

app.UIFigure.Position = [100 100 1386 808];

app.UIFigure.Name = 'MATLAB App';

% Create TakeaMethodDropDownLabel

app.TakeaMethodDropDownLabel = uilabel(app.UIFigure);

app.TakeaMethodDropDownLabel.HorizontalAlignment = 'right';

app.TakeaMethodDropDownLabel.Position = [584 655 84 22];

app.TakeaMethodDropDownLabel.Text = 'Take a Method';

% Create MethodsDropDown

app.MethodsDropDown = uidropdown(app.UIFigure);

app.MethodsDropDown.Items = {'Left Rectangle', 'Right Rectangle', 'Central Rectangle', 'Trapezoidal', 'Simpsons', 'Euler', 'Runge Kutta (2nd order)', 'Runge Kutta (3rd order)', 'Runge Kutta (4th order)'};

app.MethodsDropDown.Position = [683 655 120 22];

app.MethodsDropDown.Value = 'Left Rectangle';

% Create ProblemTypeButtonGroup

app.ProblemTypeButtonGroup = uibuttongroup(app.UIFigure);

app.ProblemTypeButtonGroup.SelectionChangedFcn = createCallbackFcn(app, @ProblemTypeButtonGroupValueChanged, true);

app.ProblemTypeButtonGroup.Title = 'Choose a problem to solve';

app.ProblemTypeButtonGroup.Position = [630 697 152 77];

% Create IntegralButton

app.IntegralButton = uiradiobutton(app.ProblemTypeButtonGroup);

app.IntegralButton.Text = 'Integral';

app.IntegralButton.Position = [11 31 62 22];

app.IntegralButton.Value = true;

% Create DifferentialButton

app.DifferentialButton = uiradiobutton(app.ProblemTypeButtonGroup);

app.DifferentialButton.Text = 'Differential';

app.DifferentialButton.Position = [11 9 79 22];

% Create SolveButton

app.SolveButton = uibutton(app.UIFigure, 'push');

app.SolveButton.ButtonPushedFcn = createCallbackFcn(app, @SolveButtonPushed, true);

app.SolveButton.BackgroundColor = [0.0745 0.6235 1];

app.SolveButton.FontColor = [1 1 1];

app.SolveButton.Position = [655 37 101 33];

app.SolveButton.Text = 'Solve all';

% Create StepnumberEditFieldLabel

app.StepnumberEditFieldLabel = uilabel(app.UIFigure);

app.StepnumberEditFieldLabel.HorizontalAlignment = 'right';

app.StepnumberEditFieldLabel.Position = [599 544 74 22];

app.StepnumberEditFieldLabel.Text = 'Step number';

% Create StepEditField

app.StepEditField = uieditfield(app.UIFigure, 'numeric');

app.StepEditField.Position = [688 544 100 22];

% Create y0startvalueEditFieldLabel

app.y0startvalueEditFieldLabel = uilabel(app.UIFigure);

app.y0startvalueEditFieldLabel.HorizontalAlignment = 'right';

app.y0startvalueEditFieldLabel.Position = [597 509 76 22];

app.y0startvalueEditFieldLabel.Text = 'y0 start value';

% Create y0EditField

app.y0EditField = uieditfield(app.UIFigure, 'numeric');

app.y0EditField.HandleVisibility = 'callback';

app.y0EditField.Position = [688 509 100 22];

% Create OutputTextAreaLabel

app.OutputTextAreaLabel = uilabel(app.UIFigure);

app.OutputTextAreaLabel.HorizontalAlignment = 'center';

app.OutputTextAreaLabel.Position = [685 328 41 22];

app.OutputTextAreaLabel.Text = 'Output';

% Create ResultTextArea

app.ResultTextArea = uitextarea(app.UIFigure);

app.ResultTextArea.HorizontalAlignment = 'center';

app.ResultTextArea.Position = [371 124 669 188];

% Create ListofproblemsListBoxLabel

app.ListofproblemsListBoxLabel = uilabel(app.UIFigure);

app.ListofproblemsListBoxLabel.HorizontalAlignment = 'right';

app.ListofproblemsListBoxLabel.Position = [551 427 90 22];

app.ListofproblemsListBoxLabel.Text = 'List of problems';

% Create ProblemsListBox

app.ProblemsListBox = uilistbox(app.UIFigure);

app.ProblemsListBox.Items = {'f(x)=exp(−x)', 'f(x)=sin(x)', 'f(x)=exp(−x^2)', 'f(x)=exp(−4x−x^3)'};

app.ProblemsListBox.Position = [655 401 180 74];

app.ProblemsListBox.Value = 'f(x)=exp(−x)';

% Create UpperboundEditFieldLabel

app.UpperboundEditFieldLabel = uilabel(app.UIFigure);

app.UpperboundEditFieldLabel.HorizontalAlignment = 'right';

app.UpperboundEditFieldLabel.Position = [599 613 74 22];

app.UpperboundEditFieldLabel.Text = 'Upper bound';

% Create UpperboundEditField

app.UpperboundEditField = uieditfield(app.UIFigure, 'numeric');

app.UpperboundEditField.Position = [688 613 100 22];

% Create LowerboundEditFieldLabel

app.LowerboundEditFieldLabel = uilabel(app.UIFigure);

app.LowerboundEditFieldLabel.HorizontalAlignment = 'right';

app.LowerboundEditFieldLabel.Position = [599 580 74 22];

app.LowerboundEditFieldLabel.Text = 'Lower bound';

% Create LowerboundEditField

app.LowerboundEditField = uieditfield(app.UIFigure, 'numeric');

app.LowerboundEditField.Position = [688 580 100 22];

% Show the figure after all components are created

app.UIFigure.Visible = 'on';

end

end

% App creation and deletion

methods (Access = public)

% Construct app

function app = IntegralsAndDifferentialsCalculator

% Create UIFigure and components

createComponents(app)

% Register the app with App Designer

registerApp(app, app.UIFigure)

if nargout == 0

clear app

end

end

% Code that executes before app deletion

function delete(app)

% Delete UIFigure when app is deleted

delete(app.UIFigure)

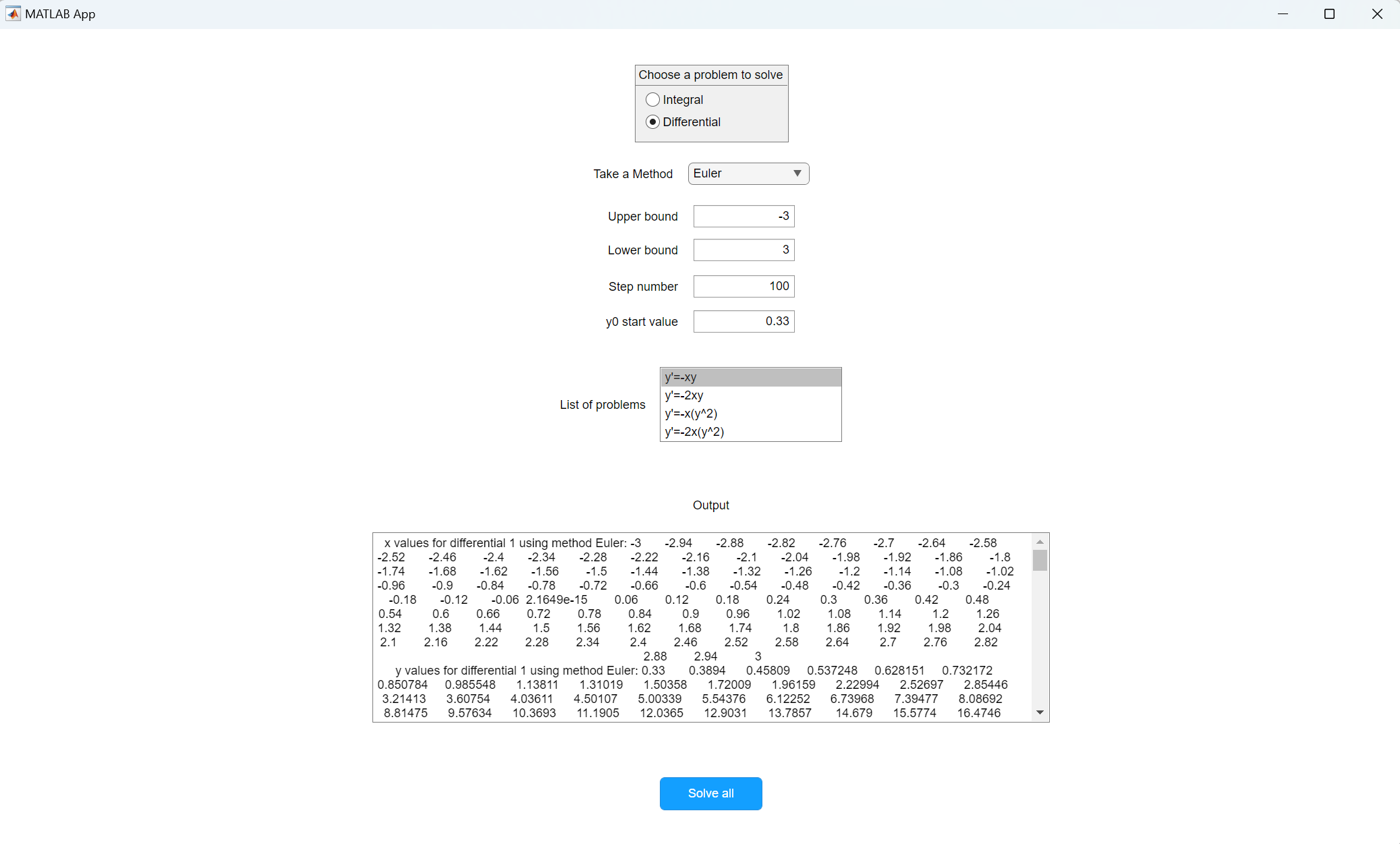
end

end

end

Програма передбачає розв’язання математичних завдань за допомогою вищеописаних методів використовуючи можливості написання графічного інтерфейсу в програмі MatLab за допомогою додатку App Designer. Даний продукт використовує заготовлені розробником інтеграли та диференціали і пропонує стандартні методи розв’язку для них. Застосунок вбачає елементи динамічної зміни інтерфейсу відповідно до вибору користувача. Обчислення відбувається після вибору користувачем необхідного йому методу, надання даних для розв’язку та натискання кнопки для початку обчислень та виведення результатів в вікно виводу.

## Результати виконання програми



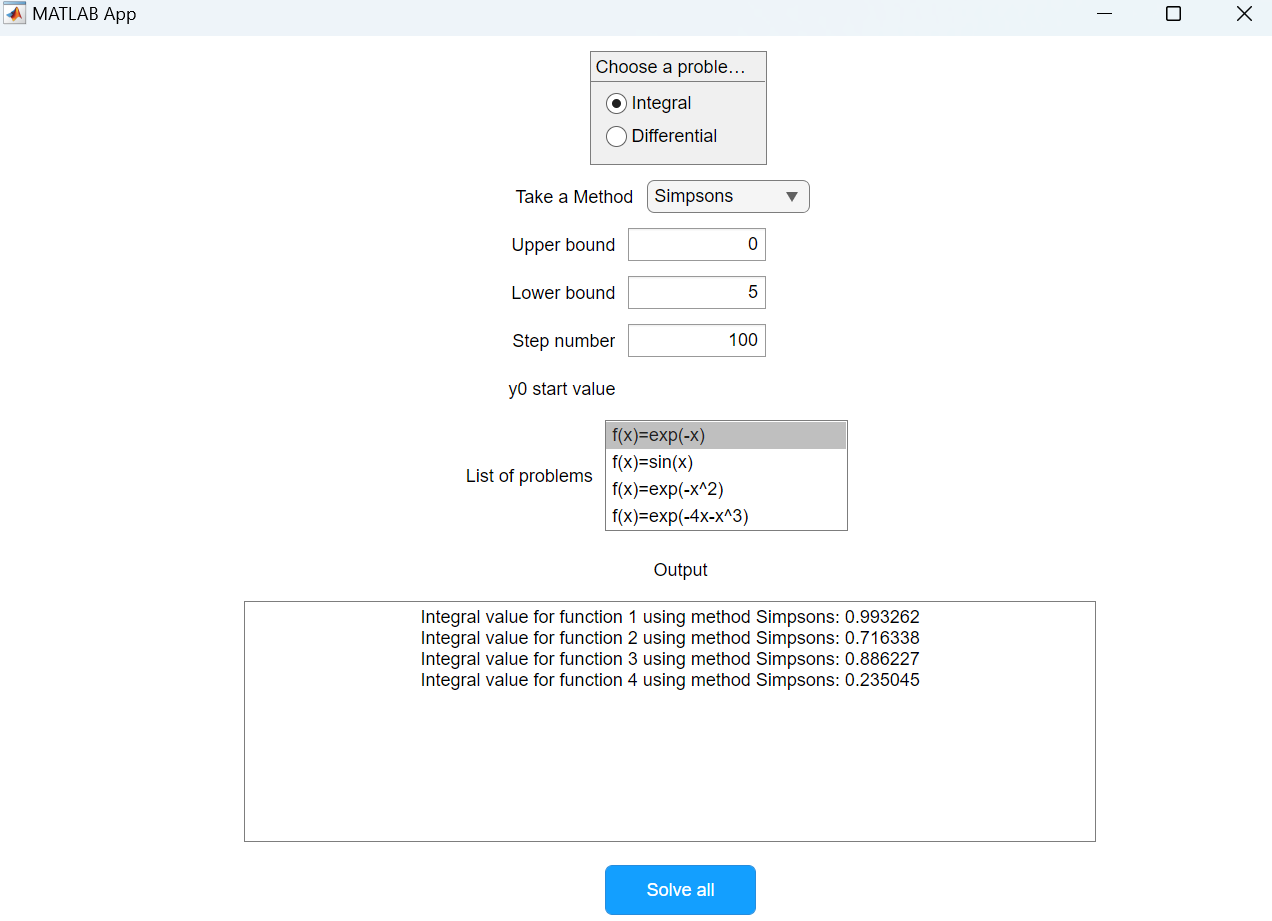


Рисунок 2.1 – «Результат виконання програми з граф. інтерфейсом»Початок форми

# Завдання 2.2

Модифікуйте і протестуйте розроблений програмний продукт. Для розв'язання диференціальних рівнянь надайте як табличний, так і графічний вивід результатів. Надайте зведену таблицю з результатами обчислень для визначеного інтегралу за п'ятьма методами, для однієї з вибраних функцій зі списку. Проаналізуйте результати. Зробіть висновки про роботу методів.

## Код завдання 2.2:

## IntegralsAndDifferentialsCalculator.mlapp

classdef IntegralsAndDifferentialsCalculator < matlab.apps.AppBase

% Properties that correspond to app components

properties (Access = public)

UIFigure matlab.ui.Figure

FirstproblemtableLabel matlab.ui.control.Label

Table matlab.ui.control.Table

LowerboundEditField matlab.ui.control.NumericEditField

LowerboundEditFieldLabel matlab.ui.control.Label

UpperboundEditField matlab.ui.control.NumericEditField

UpperboundEditFieldLabel matlab.ui.control.Label

ProblemsListBox matlab.ui.control.ListBox

ListofproblemsListBoxLabel matlab.ui.control.Label

ResultTextArea matlab.ui.control.TextArea

OutputTextAreaLabel matlab.ui.control.Label

y0EditField matlab.ui.control.NumericEditField

y0startvalueEditFieldLabel matlab.ui.control.Label

StepEditField matlab.ui.control.NumericEditField

StepnumberEditFieldLabel matlab.ui.control.Label

SolveButton matlab.ui.control.Button

ProblemTypeButtonGroup matlab.ui.container.ButtonGroup

DifferentialButton matlab.ui.control.RadioButton

IntegralButton matlab.ui.control.RadioButton

MethodsDropDown matlab.ui.control.DropDown

TakeaMethodDropDownLabel matlab.ui.control.Label

XYAxes matlab.ui.control.UIAxes

end

% Callbacks that handle component events

methods (Access = private)

% Selection changed function: ProblemTypeButtonGroup

function ProblemTypeButtonGroupValueChanged(app, event)

% Get the selected problem type

problemType = app.ProblemTypeButtonGroup.SelectedObject.Text;

% Show/hide numeric fields based on problem type

if strcmp(problemType, 'Integral')

app.y0EditField.Visible = 'off'; % Hide y0 start value field

app.ProblemsListBox.Items = {'f(x)=exp(-x)', 'f(x)=sin(x)', 'f(x)=exp(-x^2)', 'f(x)=exp(-4x-x^3)'}; % Set list box items for integrals

app.Table.ColumnName = {'Left Rectangles', 'Right Rectangles', 'Central Rectangles', 'Trapezoidal', 'Simpsons'};

app.XYAxes.Visible = "off";

elseif strcmp(problemType, 'Differential')

app.y0EditField.Visible = 'on'; % Show y0 start value field

app.ProblemsListBox.Items = {'y''=-xy', 'y''=-2xy', 'y''=-x(y^2)', 'y''=-2x(y^2)'}; % Set list box items for differentials

app.Table.ColumnName = {'X', 'Y'};

app.XYAxes.Visible = "on";

end

% Update dropdown menu based on problem type

if strcmp(problemType, 'Integral')

% Show only methods for solving integrals in dropdown

integralsMethods = {'Left Rectangle', 'Right Rectangle', 'Central Rectangle', 'Trapeziodal', 'Simpsons'};

app.MethodsDropDown.Items = integralsMethods;

elseif strcmp(problemType, 'Differential')

% Show only methods for solving differentials in dropdown

differentialsMethods = {'Euler', 'Runge-Kutta 2nd Order', 'Runge-Kutta 3rd Order', 'Runge-Kutta 4th Order'};

app.MethodsDropDown.Items = differentialsMethods;

end

end

% Button pushed function: SolveButton

function SolveButtonPushed(app, event)

% Get the selected problem from the ProblemsListBox

selected\_problem = app.ProblemsListBox.Value;

% Determine the type of problem (integral or differential)

problem\_type = app.ProblemTypeButtonGroup.SelectedObject.Text;

% Prompt the user to select the method

selected\_method = app.MethodsDropDown.Value;

% Parse the selected problem and extract parameters

if strcmp(problem\_type, 'Integral')

f = {

@(x) exp(-x), % f(x) = exp(-x)

@(x) sin(x), % f(x) = sin(x)

@(x) exp(-x^2), % f(x) = exp(-x^2)

@(x) exp(-4\*x - x^3) % f(x) = exp(-4x - x^3)

};

% Initialize the result string

result\_string = '';

% Initialize the result matrix for the integral values of the first method

integral\_values = zeros(1, 5); % 5 methods

% Select the first integral function

integral\_function = f{1};

% Iterate over each method

methods = {'Left Rectangle', 'Right Rectangle', 'Central Rectangle', 'Trapezoidal', 'Simpsons'};

for j = 1:numel(methods)

method = methods{j};

switch method

case 'Left Rectangle'

% Solve integral problem using left rectangles method

I = left\_rectangles\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

case 'Right Rectangle'

% Solve integral problem using right rectangles method

I = right\_rectangles\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

case 'Central Rectangle'

% Solve integral problem using central rectangles method

I = central\_rectangles\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

case 'Trapezoidal'

% Solve integral problem using trapezoidal method

I = trapezoidal\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

case 'Simpsons'

% Solve integral problem using simpsons method

I = simpsons\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

end

% Store the integral value in the result matrix

integral\_values(j) = I;

end

% Update the IntegralTable with the integral values

app.Table.Data = integral\_values;

% Iterate over each integral function

for i = 1:numel(f)

integral\_function = f{i};

I = left\_rectangles\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

% Perform integration using the selected method

switch selected\_method

case 'Left Rectangle'

% Solve integral problem using left rectangles method

I = left\_rectangles\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

case 'Right Rectangle'

% Solve integral problem using right rectangles method

I = right\_rectangles\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

case 'Central Rectangle'

% Solve integral problem using central rectangles method

I = central\_rectangles\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

case 'Trapezoidal'

% Solve integral problem using trapezoidal method

I = trapezoidal\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

case 'Simpsons'

% Solve integral problem using simpsons method

I = simpsons\_integration(integral\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.StepEditField.Value);

end

% Append the result for the current integral and method to the result string

result\_string = [result\_string sprintf('Integral value for function %d using method %s: %f\n', i, selected\_method, I)];

end

% Update the ResultTextArea with the result string

app.ResultTextArea.Value = result\_string;

else

f = {

@(x, y) -x\*y, % y' = -xy

@(x, y) -2\*x\*y, % y' = -2xy

@(x, y) -x\*(y^2), % y' = -x(y^2)

@(x, y) -2\*x\*(y^2) % y' = -2x(y^2)

};

% Initialize variables for storing results

result\_string = '';

% Initialize variables for storing X and Y values

all\_x\_values = {};

all\_y\_values = {};

% Iterate over each differential function

for i = 1:numel(f)

differential\_function = f{i};

% Initialize variables for storing results for the current differential equation

x\_values = [];

y\_values = [];

% Solve differential problem using the selected method

switch selected\_method

case 'Euler'

% Solve differential problem using Euler's method

[x\_values, y\_values] = euler\_method(differential\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.y0EditField.Value, app.StepEditField.Value);

case 'Runge-Kutta 2nd Order'

% Solve differential problem using Runge-Kutta of the second order method

[x\_values, y\_values] = runge\_kutta\_second\_order(differential\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.y0EditField.Value, app.StepEditField.Value);

case 'Runge-Kutta 3rd Order'

% Solve differential problem using Runge-Kutta of the third order method

[x\_values, y\_values] = runge\_kutta\_third\_order(differential\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.y0EditField.Value, app.StepEditField.Value);

case 'Runge-Kutta 4th Order'

% Solve differential problem using Runge-Kutta of the forth order method

[x\_values, y\_values] = runge\_kutta\_fourth\_order(differential\_function, app.UpperboundEditField.Value, app.LowerboundEditField.Value, app.y0EditField.Value, app.StepEditField.Value);

end

% Append the results for the current differential and method to the result string

result\_string = [result\_string sprintf('x values for differential %d using method %s: %s\n', i, selected\_method, num2str(x\_values))];

result\_string = [result\_string sprintf('y values for differential %d using method %s: %s\n\n', i, selected\_method, num2str(y\_values))];

% Store X and Y values for plotting

all\_x\_values{i} = x\_values;

all\_y\_values{i} = y\_values;

end

% Display the result for all differentials and methods

app.ResultTextArea.Value = result\_string;

% Update the table with X and Y values

app.Table.Data = [all\_x\_values{1}', all\_y\_values{1}'];

% Plot X and Y values on the XYAxes

cla(app.XYAxes); % Clear previous plot

hold(app.XYAxes, 'on');

for i = 1:numel(all\_x\_values)

plot(app.XYAxes, all\_x\_values{i}, all\_y\_values{i});

end

hold(app.XYAxes, 'off');

xlabel(app.XYAxes, 'X');

ylabel(app.XYAxes, 'Y');

title(app.XYAxes, 'Graph of Differential Equations');

legend(app.XYAxes, 'Differential 1', 'Differential 2', 'Differential 3', 'Differential 4');

end

end

end

% Component initialization

methods (Access = private)

% Create UIFigure and components

function createComponents(app)

% Create UIFigure and hide until all components are created

app.UIFigure = uifigure('Visible', 'off');

app.UIFigure.Color = [1 1 1];

app.UIFigure.Position = [100 100 1686 970];

app.UIFigure.Name = 'MATLAB App';

% Create XYAxes

app.XYAxes = uiaxes(app.UIFigure);

title(app.XYAxes, 'Table of first differential')

xlabel(app.XYAxes, 'X')

ylabel(app.XYAxes, 'Y')

zlabel(app.XYAxes, 'Z')

app.XYAxes.Position = [1083 535 562 377];

% Create TakeaMethodDropDownLabel

app.TakeaMethodDropDownLabel = uilabel(app.UIFigure);

app.TakeaMethodDropDownLabel.HorizontalAlignment = 'right';

app.TakeaMethodDropDownLabel.Position = [734 820 84 22];

app.TakeaMethodDropDownLabel.Text = 'Take a Method';

% Create MethodsDropDown

app.MethodsDropDown = uidropdown(app.UIFigure);

app.MethodsDropDown.Items = {'Left Rectangle', 'Right Rectangle', 'Central Rectangle', 'Trapezoidal', 'Simpsons', 'Euler', 'Runge Kutta (2nd order)', 'Runge Kutta (3rd order)', 'Runge Kutta (4th order)'};

app.MethodsDropDown.Position = [833 820 120 22];

app.MethodsDropDown.Value = 'Left Rectangle';

% Create ProblemTypeButtonGroup

app.ProblemTypeButtonGroup = uibuttongroup(app.UIFigure);

app.ProblemTypeButtonGroup.SelectionChangedFcn = createCallbackFcn(app, @ProblemTypeButtonGroupValueChanged, true);

app.ProblemTypeButtonGroup.Title = 'Choose a problem to solve';

app.ProblemTypeButtonGroup.Position = [768 865 152 77];

% Create IntegralButton

app.IntegralButton = uiradiobutton(app.ProblemTypeButtonGroup);

app.IntegralButton.Text = 'Integral';

app.IntegralButton.Position = [11 31 62 22];

app.IntegralButton.Value = true;

% Create DifferentialButton

app.DifferentialButton = uiradiobutton(app.ProblemTypeButtonGroup);

app.DifferentialButton.Text = 'Differential';

app.DifferentialButton.Position = [11 9 79 22];

% Create SolveButton

app.SolveButton = uibutton(app.UIFigure, 'push');

app.SolveButton.ButtonPushedFcn = createCallbackFcn(app, @SolveButtonPushed, true);

app.SolveButton.BackgroundColor = [0.0745 0.6235 1];

app.SolveButton.FontColor = [1 1 1];

app.SolveButton.Position = [764 47 160 43];

app.SolveButton.Text = 'Solve all';

% Create StepnumberEditFieldLabel

app.StepnumberEditFieldLabel = uilabel(app.UIFigure);

app.StepnumberEditFieldLabel.HorizontalAlignment = 'right';

app.StepnumberEditFieldLabel.Position = [749 706 74 22];

app.StepnumberEditFieldLabel.Text = 'Step number';

% Create StepEditField

app.StepEditField = uieditfield(app.UIFigure, 'numeric');

app.StepEditField.Position = [838 706 100 22];

% Create y0startvalueEditFieldLabel

app.y0startvalueEditFieldLabel = uilabel(app.UIFigure);

app.y0startvalueEditFieldLabel.HorizontalAlignment = 'right';

app.y0startvalueEditFieldLabel.Position = [747 671 76 22];

app.y0startvalueEditFieldLabel.Text = 'y0 start value';

% Create y0EditField

app.y0EditField = uieditfield(app.UIFigure, 'numeric');

app.y0EditField.HandleVisibility = 'callback';

app.y0EditField.Position = [838 671 100 22];

% Create OutputTextAreaLabel

app.OutputTextAreaLabel = uilabel(app.UIFigure);

app.OutputTextAreaLabel.HorizontalAlignment = 'center';

app.OutputTextAreaLabel.Position = [823 334 41 22];

app.OutputTextAreaLabel.Text = 'Output';

% Create ResultTextArea

app.ResultTextArea = uitextarea(app.UIFigure);

app.ResultTextArea.HorizontalAlignment = 'center';

app.ResultTextArea.Position = [509 130 669 188];

% Create ListofproblemsListBoxLabel

app.ListofproblemsListBoxLabel = uilabel(app.UIFigure);

app.ListofproblemsListBoxLabel.HorizontalAlignment = 'right';

app.ListofproblemsListBoxLabel.Position = [702 433 90 22];

app.ListofproblemsListBoxLabel.Text = 'List of problems';

% Create ProblemsListBox

app.ProblemsListBox = uilistbox(app.UIFigure);

app.ProblemsListBox.Items = {'f(x)=exp(−x)', 'f(x)=sin(x)', 'f(x)=exp(−x^2)', 'f(x)=exp(−4x−x^3)'};

app.ProblemsListBox.Position = [806 407 180 74];

app.ProblemsListBox.Value = 'f(x)=exp(−x)';

% Create UpperboundEditFieldLabel

app.UpperboundEditFieldLabel = uilabel(app.UIFigure);

app.UpperboundEditFieldLabel.HorizontalAlignment = 'right';

app.UpperboundEditFieldLabel.Position = [749 775 74 22];

app.UpperboundEditFieldLabel.Text = 'Upper bound';

% Create UpperboundEditField

app.UpperboundEditField = uieditfield(app.UIFigure, 'numeric');

app.UpperboundEditField.Position = [838 775 100 22];

% Create LowerboundEditFieldLabel

app.LowerboundEditFieldLabel = uilabel(app.UIFigure);

app.LowerboundEditFieldLabel.HorizontalAlignment = 'right';

app.LowerboundEditFieldLabel.Position = [749 742 74 22];

app.LowerboundEditFieldLabel.Text = 'Lower bound';

% Create LowerboundEditField

app.LowerboundEditField = uieditfield(app.UIFigure, 'numeric');

app.LowerboundEditField.Position = [838 742 100 22];

% Create Table

app.Table = uitable(app.UIFigure);

app.Table.ColumnName = {'Left Rectangles'; 'Right Rectangles'; 'Central Rectangles'; 'Trapezoidal'; 'Simpsons'};

app.Table.RowName = {};

app.Table.Position = [55 395 571 431];

% Create FirstproblemtableLabel

app.FirstproblemtableLabel = uilabel(app.UIFigure);

app.FirstproblemtableLabel.FontSize = 24;

app.FirstproblemtableLabel.Position = [238 844 205 31];

app.FirstproblemtableLabel.Text = 'First problem table';

% Show the figure after all components are created

app.UIFigure.Visible = 'on';

end

end

% App creation and deletion

methods (Access = public)

% Construct app

function app = IntegralsAndDifferentialsCalculator

% Create UIFigure and components

createComponents(app)

% Register the app with App Designer

registerApp(app, app.UIFigure)

if nargout == 0

clear app

end

end

% Code that executes before app deletion

function delete(app)

% Delete UIFigure when app is deleted

delete(app.UIFigure)

end

end

end

Даний код покращує роботу програми надаючи користувачу табличні значення та графічне представлення на графіку диференціалів

## Результати виконання програми

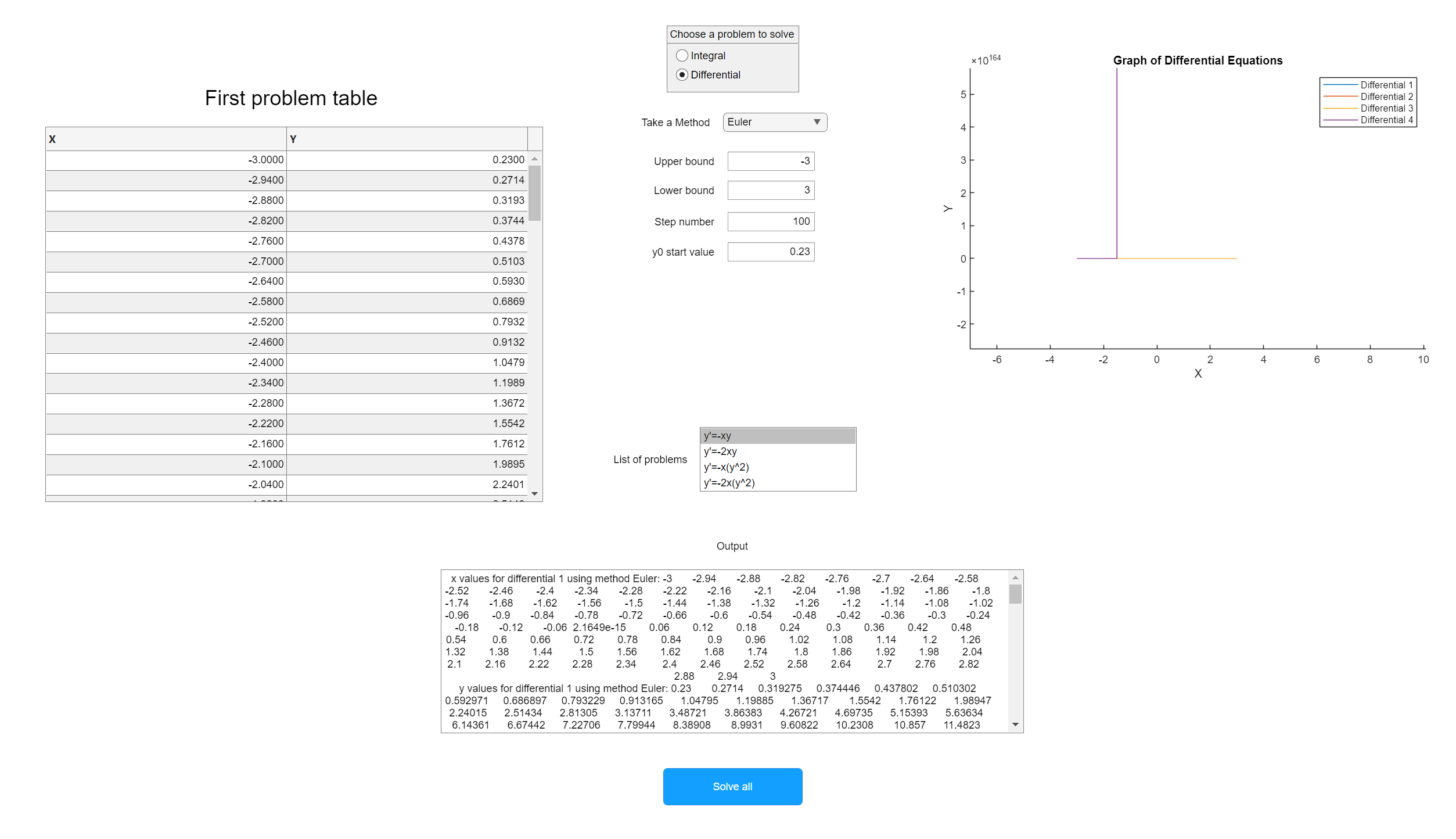
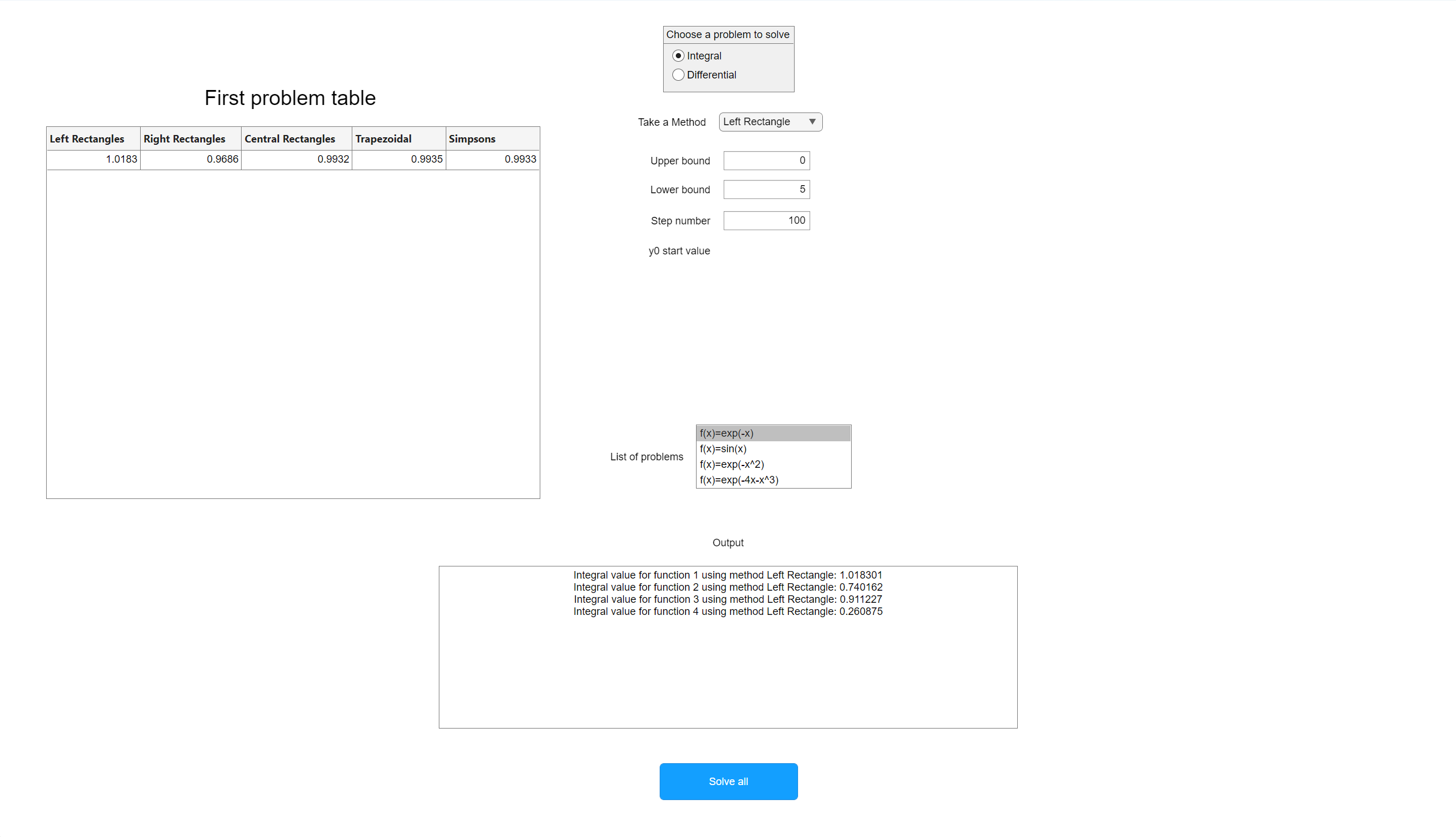


Рисунок 2.2 – «Результат виконання модифікованої програми з граф. інтерфейсом»Початок форми

# **ВИСНОВОК**

Інтеграл та інтегрування є важливими поняттями в математиці та науці. Інтегрування відображає два основні типи проблем: геометричні, які стосуються знаходження площі під кривою, і аналітико-алгебраїчні, які описують обчислення загального значення змінної, що змінюється згідно з різними параметрами.

Диференціальні рівняння дозволяють моделювати різноманітні динамічні процеси, такі як системи керування польотом коптерів, штучний інтелект у відеоіграх, деформація матеріалів тощо. У даній роботі використовуються числові методи для обчислення інтегралів та вирішення диференціальних рівнянь, що є важливими інструментами в аналізі та моделюванні різноманітних явищ.