Student Name: Mina Gerges
ID: 70040441
Computational Physics_ PHYS624
HW5: BVP Shooting method (RK4 with Secant)

UAEU كلية العلوم College of Science | جامعة الإمارات العربية المتحدة United Arab Emirates University

---

Solving Boundary Value Problems with "Shooting Method"

1. Introduction

The objective is to solve a second order differential eqaution for a boundary value problem using the shooting method with the dependence on RK4 and secant method.
The idea is to transform the 2nd Differential equation into a system of two first order differential equations to be solved as initial value problem.

Solve the following differential equation using shooting method:

$$U^{//} = -\frac{\pi^2}{4}(u+1)$$

with the following boundary conditions: U(0)=0 and U(1)=1. You can use RK4 method in combination with Secant Method to solve this and compare your results with analytical solution given in page 98 of the book. Make a comparison plot with details in short report fom.

Script structure

1. System of Equations:

- \`f1\` and \`f2\` functions represent the two equations in the system:
  - \`f1\` calculates the derivative of y1 with respect to x, represented as (dy1/dx = y2).
  - \`f2\` calculates the derivative of y2 with respect to x, defined by (dy2/dx = -pi^2/4 *(y1 + 1)).
- An additional \`analytical_solution\` function provides the analytical solution to the equation for comparison against numerical results.

3. Runge-Kutta 4th Order (RK4) Method:

The \`RK4\` function applies the RK4 algorithm to solve the system's initial value problem. It calculates values of y1 and y2 across the range of \(x \) from 0 to \`x_end\` using the step size \`dx=h=0.05\`.

4. Shooting Method with Secant Method Iteration:

This section refines the initial slope u'(0) using the \`shooting_method\` function to satisfy the boundary condition u(1) = 1.
This is achieved by:

- Error Minimization: The secant method stops once the calculated boundary value is within \`TOL\` of the target boundary condition (\`target\`).

Student Name: Mina Gerges
ID: 70040441
Computational Physics_ PHYS624
HW5: BVP Shooting method (RK4 with Secant)

UAEU كلية العلوم College of Science
جامعة الإمارات العربية المتحدة United Arab Emirates University

----

C++ Script

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <fstream>
#include <iomanip>

using namespace std;
#define PI 3.14159265358979323846
const double TOL = 1e-6;  // Tolerance
const int MAX_ITER = 100; // iterations for Secant Method
// Define the first order differential equations of the system
double f1(double x, double y1, double y2) {
    return y2;  // dy1/dx = y2 = z
}
double f2(double x, double y1, double y2) {
    return -PI * PI / 4 * (y1 + 1);  // dy2/dx = -π^2/4 * (y1 + 1)
}
// Analytical solution
double analytical_solution(double x) {
    return cos(PI * x / 2) + 2 * sin(PI * x / 2) - 1;
}
// Runge-Kutta 4th order (RK4) for solving the IVP, with output logging to file
void RK4(double y1_0, double y2_0, double x_end, double dx, double& y1_end, ofstream&
outfile) {
    double x = 0.0;
    double y1 = y1_0; // left Boundary
    double y2 = y2_0; // Guess of the derivative Z=y2
    // Write header for file with precision
    outfile << fixed << setprecision(4) << "x\t\t" << setprecision(5)
            << "numerical_u(x)\t" << setprecision(5) << "numerical_u'(x)\t"
            << setprecision(4) << "analytical_u(x)\n";
    while (x <= x_end) {
        outfile << setprecision(4) << x << "\t"
                << setprecision(5) << y1 << "\t"
                << setprecision(5) << y2 << "\t"
                << setprecision(4) << analytical_solution(x) << "\n";
        // Runge-Kutta calculations
        double k1_y1 = dx * f1(x, y1, y2);
        double k1_y2 = dx * f2(x, y1, y2);
        double k2_y1 = dx * f1(x + 0.5 * dx, y1 + 0.5 * k1_y1, y2 + 0.5 * k1_y2);
        double k2_y2 = dx * f2(x + 0.5 * dx, y1 + 0.5 * k1_y1, y2 + 0.5 * k1_y2);
        double k3_y1 = dx * f1(x + 0.5 * dx, y1 + 0.5 * k2_y1, y2 + 0.5 * k2_y2);
        double k3_y2 = dx * f2(x + 0.5 * dx, y1 + 0.5 * k2_y1, y2 + 0.5 * k2_y2);
        double k4_y1 = dx * f1(x + dx, y1 + k3_y1, y2 + k3_y2);
        double k4_y2 = dx * f2(x + dx, y1 + k3_y1, y2 + k3_y2);
        y1 += (k1_y1 + 2 * k2_y1 + 2 * k3_y1 + k4_y1) / 6.0;
        y2 += (k1_y2 + 2 * k2_y2 + 2 * k3_y2 + k4_y2) / 6.0;
        x += dx;
    }
    y1_end = y1;
}
// Secant Method to adjust initial slope y2_0
double shooting_method(double x_end, double dx, double target, ofstream& outfile) {
```

كلية العلوم
UAEU College of Science
جامعة الإمارات العربية المتحدة
United Arab Emirates University

```
    double y1_0 = 0.0;      // Initial condition u(0) = 0
    double y2_0 = 0.5;      // First guess for u'(0)
    double y2_1 = 5.0;      // Second guess for u'(0)
    double y1_end_0, y1_end_1;
    RK4(y1_0, y2_0, x_end, dx, y1_end_0, outfile);
    RK4(y1_0, y2_1, x_end, dx, y1_end_1, outfile);
    cout << "Iter\t y2_0\t\t y2_1\t\t y1_end\t\t Error\n";
    for (int iter = 0; iter < MAX_ITER; ++iter) {
        double error = y1_end_1 - target;
        cout << iter << "\t" << y2_0 << "\t" << y2_1 << "\t" << y1_end_1 << "\t" << error
<< endl;
        // Secant method iteration
        double y2_new = y2_1 - (y1_end_1 - target) * (y2_1 - y2_0) / (y1_end_1 -
y1_end_0);

        // Update previous guesses
        y2_0 = y2_1;
        y1_end_0 = y1_end_1;
        y2_1 = y2_new;

        // Run RK4 with new guess
        RK4(y1_0, y2_1, x_end, dx, y1_end_1, outfile);

        // Check if we reached the target within tolerance
        if (fabs(y1_end_1 - target) < TOL) {
            return y2_1;  // Successful convergence
        }
    }
    cerr << "Shooting method did not converge within maximum iterations." << endl;
    return y2_1;
}
int main() {
    double x_end = 1.0;   // Boundary at x = 1
    double dx = 0.05;      // Step size for RK4
    double target = 1.0;  // Boundary condition u(1) = 1
    // save results
    ofstream outfile("results22.txt");
    if (!outfile) {
        cerr << "Error opening file for writing." << endl;
        return 1;
    }

    // Shooting method
    double y2_0 = shooting_method(x_end, dx, target, outfile);
    cout << "Initial slope u'(0) found: " << y2_0 << endl;
    outfile.close();

    return 0;
}
```

Student Name: Mina Gerges
ID: 70040441
Computational Physics_ PHYS624
HW5: BVP Shooting method (RK4 with Secant)

Sample of script results

At u'(x)= 0.5

x	numerical_u(x)	numerical_u'(x)	analytical_u(x)
0.0000	0.00000	0.50000	0.0000
0.0500	0.02189	0.37522	0.1538
0.1000	0.03748	0.24812	0.3006

At u'(x)= 5

x	numerical_u(x)	numerical_u'(x)	analytical_u(x)
0.0000	0.00000	5.00000	0.0000
0.0500	0.24666	4.86134	0.1538
0.1000	0.48563	4.69272	0.3006
0.1500	0.71545	4.49515	0.4393

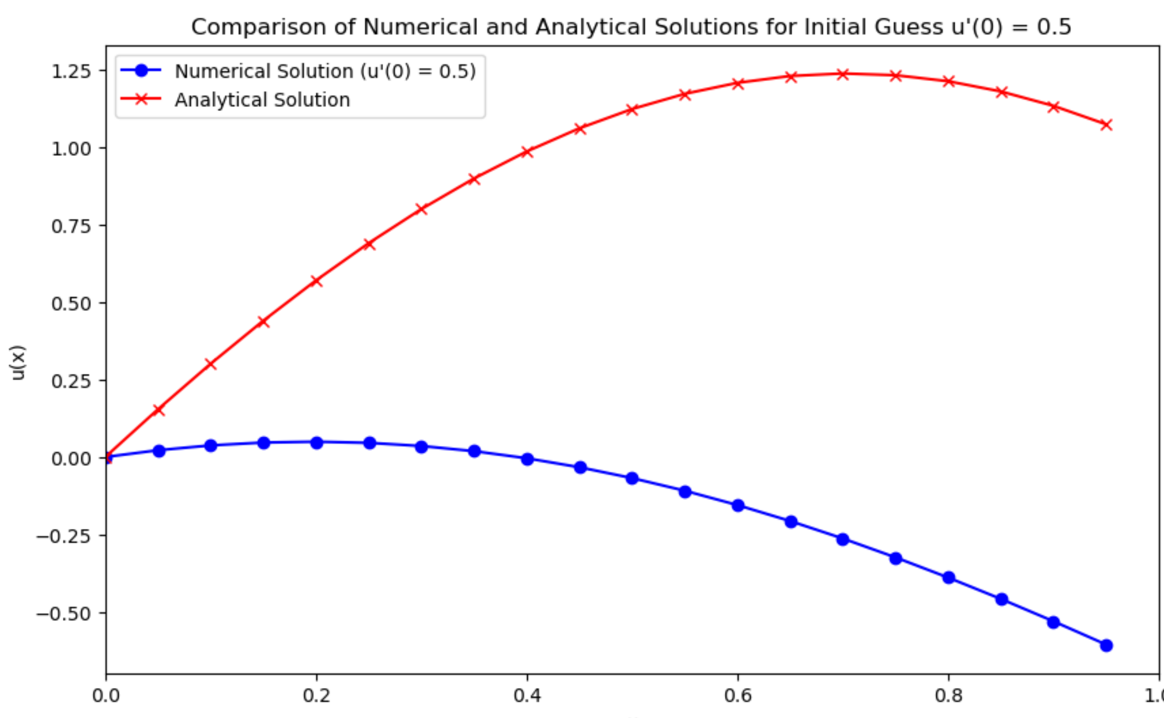
With secant method

At at u'(x)= 3.14159

x	numerical_u(x)	numerical_u'(x)	analytical_u(x)
0.0000	0.00000	3.14159	0.0000
0.0500	0.15384	3.00866	0.1538
0.1000	0.30056	2.85719	0.3006
0.1500	0.43926	2.68809	0.4393

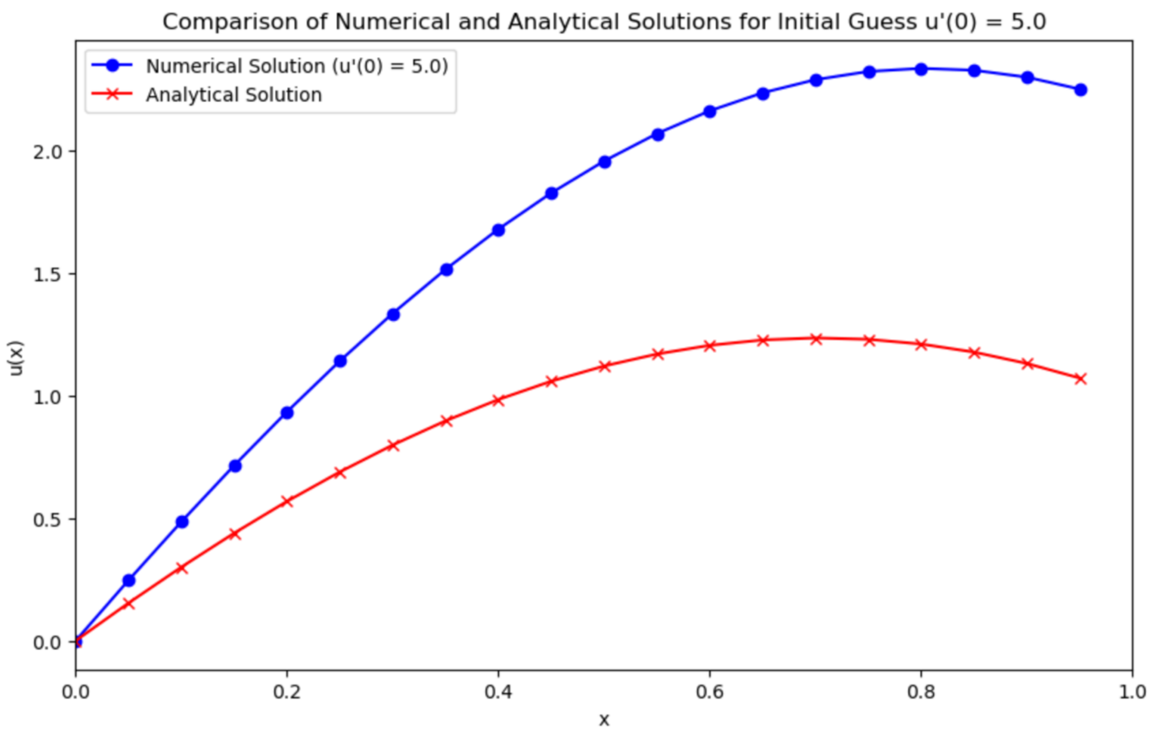
Plotting the results:

Plot1: Presents the comparsion of the numerical solution at the initial guess of the derivative u'(x)= 0.5 with the analytical solution.

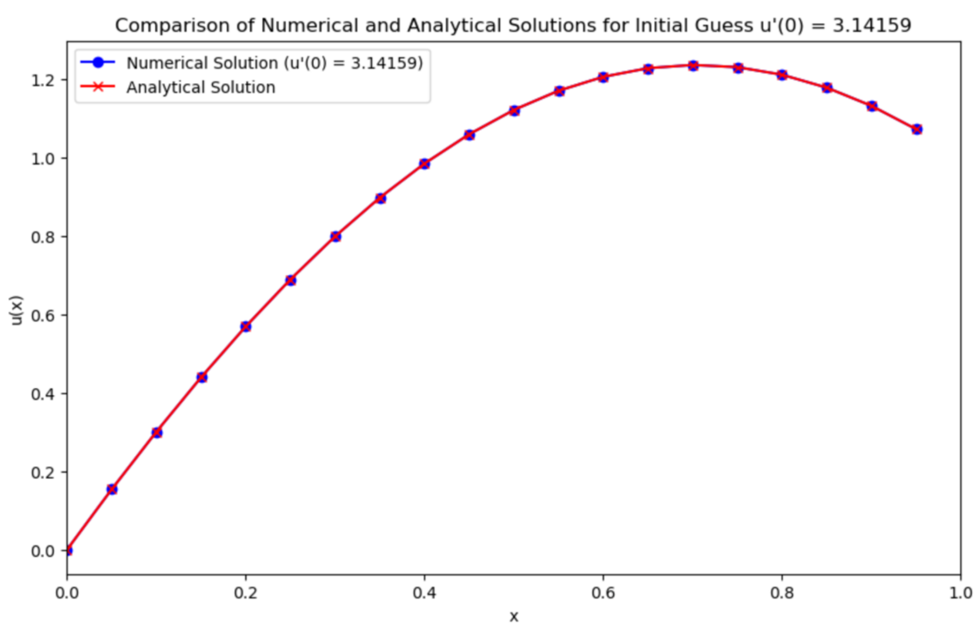


Student Name: Mina Gerges
ID: 70040441
Computational Physics_ PHYS624
HW5: BVP Shooting method (RK4 with Secant)

Plot2: Presents the comparsion of the numerical solution at the initial guess of the derivative u'(x)= 5 with the analytical solution.



Plot1: Presents the comparsion of the numerical solution at the initial guess of the derivative u'(x)= 3.14159 with the analytical solution.



END