Green's Function and ODE With IVP

Kaceson Tisdel

Responsibilities and Tasks

Not in a group so this was all me!

System Performance and Context Description

This project tests how well Python can handle graphing and solving a second-order differential equation. The code uses SciPy's RKF45 method to solve the homogeneous and non-homogeneous equations, then compares those results to Green's function in a graph. The graph shows that both of the methods produce nearly the same results, showing the system can do the calculations smoothly

Specific Problem Solved

The specific problems solved were

1) $y'' + 2y' + y = 2x; t \geq 0; y(0) = y'(0) = 0$
2) $y'' + y = x^2 \ ; \ t \geq 0; y(0) = y'(0) = 0$

These problems were to be solved by hand using three different methods: Green's function, undetermined coefficients, and variable parameters. Then compare the solutions to ensure they are all equal to make sure the work was done correctly. Then, Section 2 asks to solve Equation 1 by first finding and plotting the homogeneous part, then writing the part of the program to plot Green's function, and graph all the data.

Mathematical Approach

Shown in other attached document.

Implementation Approach

```
------------------------------------------------------------
DEFINE function for HOMOGENEOUS ODE system:
  Inputs: x, y
  Let y0 = y, y1 = y'
  Return [y1,  -2*y1 - y0]   # represents y'' + 2y' + y = 0

DEFINE function for NON-HOMOGENEOUS ODE system:
  Inputs: x, y
  Let y0 = y, y1 = y'
  Return [y1,  2*x - 2*y1 - y0]   # represents y'' + 2y' + y = 2x

------------------------------------------------------------
SOLVE both systems using RKF45 method:
  Call solve_ivp for each system over x=0→10
  Save results as sol_hom and sol_nonhom
```

DEFINE Green's function:

G(x,s) = (x − s) * exp(−(x − s)) if x ≥ s
         0   if x < s

DEFINE forcing function:

f(s) = 2*s   # same as the 2x term on the right side of ODE

FOR each x point in the grid:

Compute G(x, s) * f(s) for all s values

Integrate that product using the trapezoid rule

Append the result to the greens solution array

PLOT #1:

Plot sol_hom (homogeneous solution)

Label the graph and display

PLOT #2:

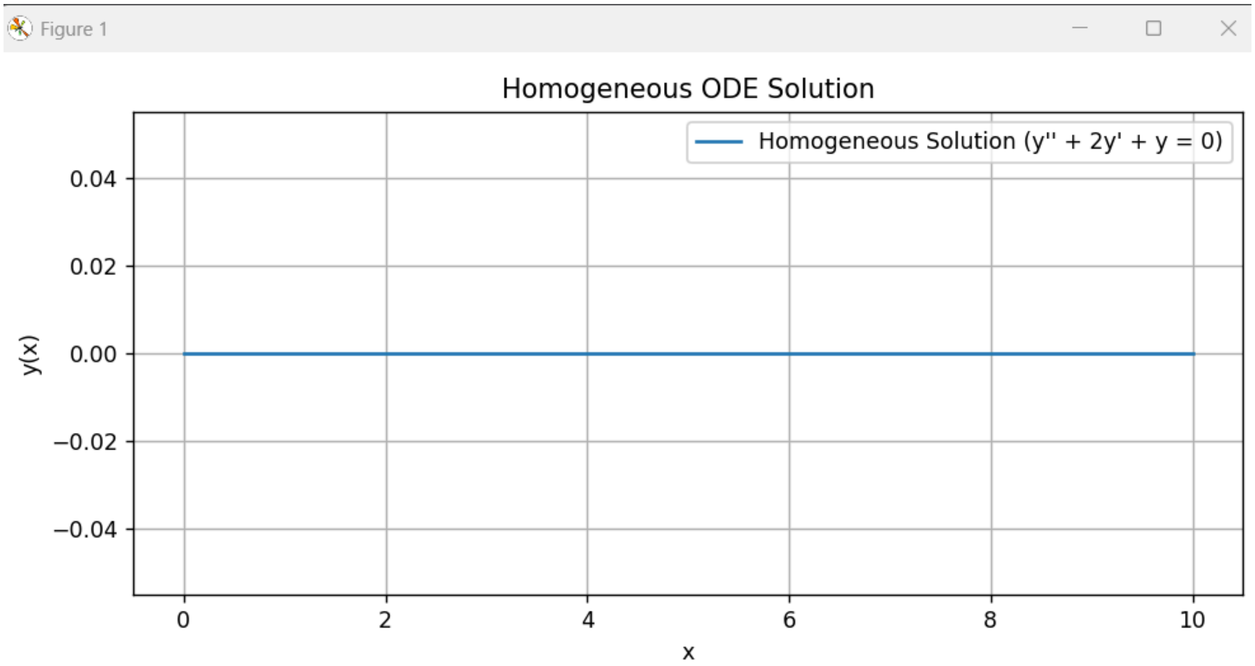
Plot sol_nonhom (numerical RKF45 result)

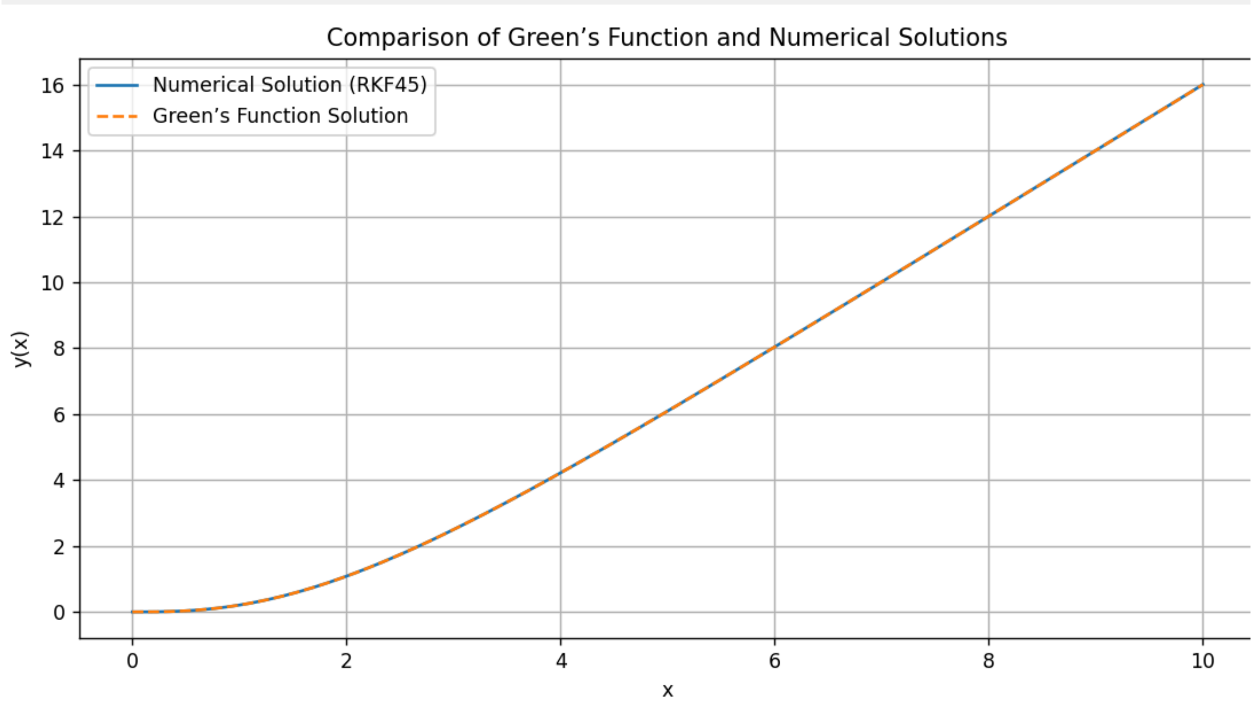
Plot greens_solution (Green's function result)

Add labels, legend, and display

Key Phases

```
# ========================================================================

sol_hom = solve_ivp(ode_system_homogeneous, x_span, y0, t_eval=x_points, method='RK45')
sol_nonhom = solve_ivp(ode_system_nonhomogeneous, x_span, y0, t_eval=x_points, method='RK45')

#Non-Homogeneous Equation

def greens_function(x, s):  1 usage
    return np.where(x >= s, (x - s) * np.exp(-(x - s)), 0.0)

def forcing(s):  1 usage
    return 2 * s  # f(s) = 2x (same as RHS)

greens_solution = []
for xi in x_points:
    integrand = greens_function(xi, x_points) * forcing(x_points)
    greens_solution.append(trapezoid(integrand, x_points))
greens_solution = np.array(greens_solution)
```





Refrences

Slides from the Padlet