Student Name: Mina Gerges
ID: 70040441
Computational Physics_ PHYS624
HW6: Pratial Differential Equations

Solving the Heat Diffusion Equation with the Explicit Method

1. Introduction

The objective is to solve the heat diffusion differential eqaution using the explicit method

Use steps sizes **(a)** $h = 0.1$ and $k = 0.0005$ and **(b)** $h = 0.1$ and $k = 0.01$ to approximate the solution to the heat equation

$$\frac{\partial u}{\partial t}(x,t) - \frac{\partial^2 u}{\partial x^2}(x,t) = 0, \quad 0 < x < 1, \quad 0 \le t,$$

with boundary conditions

$$u(0,t) = u(1,t) = 0, \quad 0 < t,$$

and initial conditions

$$u(x,0) = \sin(\pi x), \quad 0 \le x \le 1.$$

Compare the results at $t = 0.5$ to the exact solution

$$u(x,t) = e^{-\pi^2 t}\sin(\pi x).$$

The Equation used for the Explicit Scheme for PDE's:

$$u_{i,j+1} = \alpha\, u_{i-,j} + (1 - 2\alpha)u_{i\,,j} + \alpha\, u_{i+1,j}$$

Script structure

1. **Defining the analytical function**
2. Entering the requied data as user input for h, k, T, L.
   - h: the spatial step. = 0.1
   - k: the time step = {0.0005, 0.01}
   - T: the total simulation time.={0.5}
   - L: the second boundary condition. = 1
3. **Calculating alpha = k / (h * h)**
4. **Checking the stability condition if alpha >= 0.5, however the cod still runs.**

A message will appear in the terminal showing that the stability is not achieved.

5. **Creating two loops for time and position.**
6. **Saving data to (.dat) files, _____ online resources are dealing pretty much with dat files, therfore I tried that out.**

Student Name: Mina Gerges
ID: 70040441
Computational Physics_ PHYS624
HW6: Pratial Differential Equations

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

---

C++ Script

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <vector>
#include <fstream> // Include for file operations

using namespace std;
double h, k, T, L;
// Exact solution function
double exact_solution(double x, double t) {
    if (x == 0.0 || x == L) {
        return 0.0;
    }
    return exp(-M_PI * M_PI * t) * sin(M_PI * x);
}

int main() {
    // Get user inputs

    cout << "Enter the spatial step size (h): ";
    cin >> h;
    cout << "Enter the time step size (k): ";
    cin >> k;
    cout << "Enter the total simulation time (T): ";
    cin >> T;
    cout << "Enter the length of the rod (L): ";
    cin >> L;

    double alpha = k / (h * h);

    // Warn about stability condition but do not terminate
    if (alpha >= 0.5) {
        cerr << "Warning: Stability condition violated (alpha >= 0.5).
Results may be inaccurate.\n";
    }

    int nx = static_cast<int>(L / h) + 1;
    int nk = static_cast<int>(T / k) + 1;

    // Grid and time step indices
    int i, j;

    // Define the grid
    vector<vector<double> > u(nk, vector<double>(nx, 0.0));
```

```
    // Set initial and boundary conditions
    for (i = 0; i < nx; ++i) {
        double x = i * h;
        u[0][i] = sin(M_PI * x / L); // Initial condition
    }

    for (j = 1; j < nk; ++j) {
        u[j][0] = 0.0;          // Boundary at x=0
        u[j][nx - 1] = 0.0;     // Boundary at x=L
    }

    // Time integration loop (explicit scheme)
    for (j = 0; j < nk - 1; ++j) {
        for (i = 1; i < nx - 1; ++i) {
            u[j + 1][i] = alpha * u[j][i - 1] + (1 - 2 * alpha) * u[j][i]
+ alpha * u[j][i + 1];
        }
    }

    // Open a file to save the data
    ofstream outfile("Heat_Diffusion_distribution.dat");
    if (!outfile.is_open()) {
        cerr << "Error: Could not open file for writing!" << endl;
        return 1;
    }

    // Write data to file in a format suitable for plotting
    outfile << "x t u_exact u_numerical\n"; // Header
    for (j = 0; j < nk; ++j) {
        double t = j * k; // Current time
        for (i = 0; i < nx; ++i) {
            double x = i * h;
            double exact = exact_solution(x, t);
            outfile << x << " " << t << " " << exact << " " << u[j][i] <<
"\n";
        }
        outfile << "\n"; // Separate time steps with a blank line
    }

    outfile.close();
    cout << "Data saved to temperature_distribution.dat" << endl;

    // Output results for a few selected time steps (optional)
    cout << "Numerical and Exact Solutions for selected time steps:\n";
```

```
    cout << "------------------------------------------------------\n";
    for (j = 0; j < nk; j += nk / 10) { // Output 10 evenly spaced time
steps
        double t = j * k; // Current time
        cout << "Time = " << t << "\n";
        cout << "x          Numerical      Exact         Error\n";
        cout << "-----------------------------------------------\n";

        for (i = 0; i < nx; ++i) {
            double x = i * h;
            double exact = exact_solution(x, t);
            double error = abs(u[j][i] - exact);

            cout << setw(10) << x
                 << setw(15) << u[j][i]
                 << setw(15) << exact
                 << setw(15) << error << "\n";
        }
        cout << "-----------------------------------------------\n";
    }

    return 0;
}
```

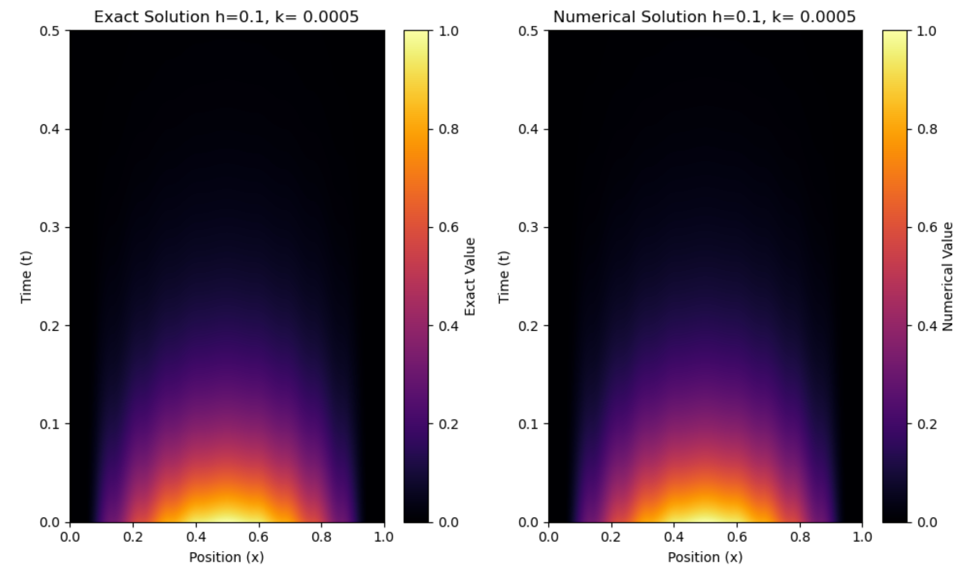
*****The script calcuates the numerical and the exact vlaue at each timestep and corresponding position.**

<u>Sample of script results</u>
At h= 0.1 and k=0.0005
Stability condition is achieved, where alpha =0.05

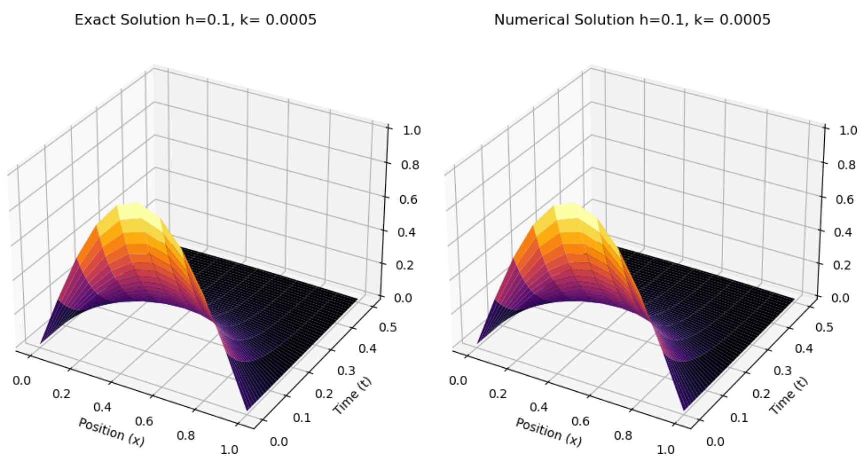
x	t	u_exact.	u_numerical
0	0.5	0	0
0.1	0.5	0.00222241	0.00228652
0.2	0.5	0.00422728	0.00434922
0.3	0.5	0.00581836	0.00598619
0.4	0.5	0.00683989	0.00703719
0.5	0.5	0.00719188	0.00739934
0.6	0.5	0.00683989	0.00703719
0.7	0.5	0.00581836	0.00598619
0.8	0.5	0.00422728	0.00434922
0.9	0.5	0.00222241	0.00228652
1	0.5	0	0

Plotting the results

Plot1: Presents the Heat diffusion map over spatial and temporal steps

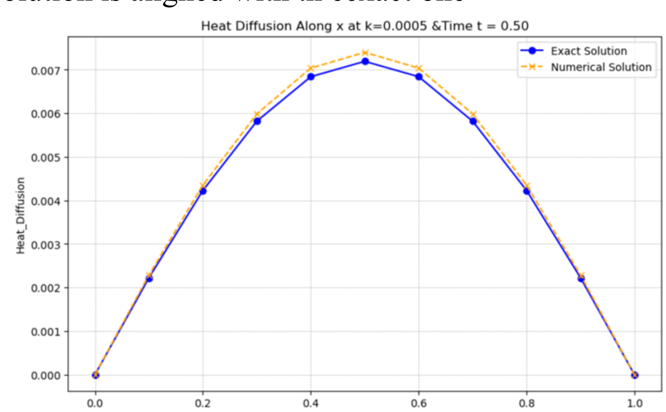


Plot2: Presents a surface plot of the same solution presented above.



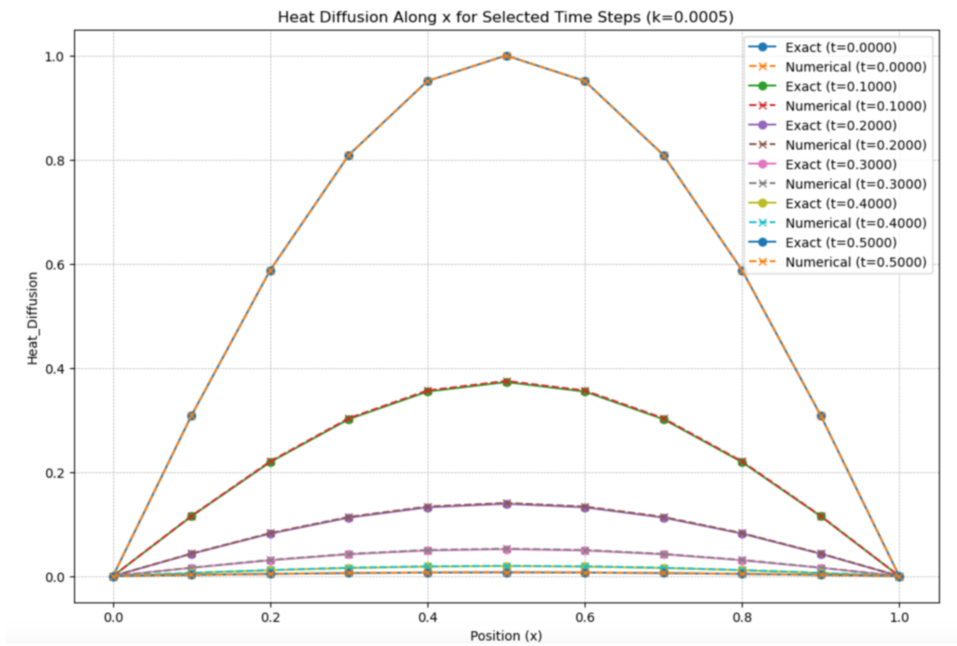
Plot3: Presents the heat distribution at time step=0.5 along the x-position with as step k=0.0005

The stability condition is clear here as the numerical solution is aligned with th eexact one

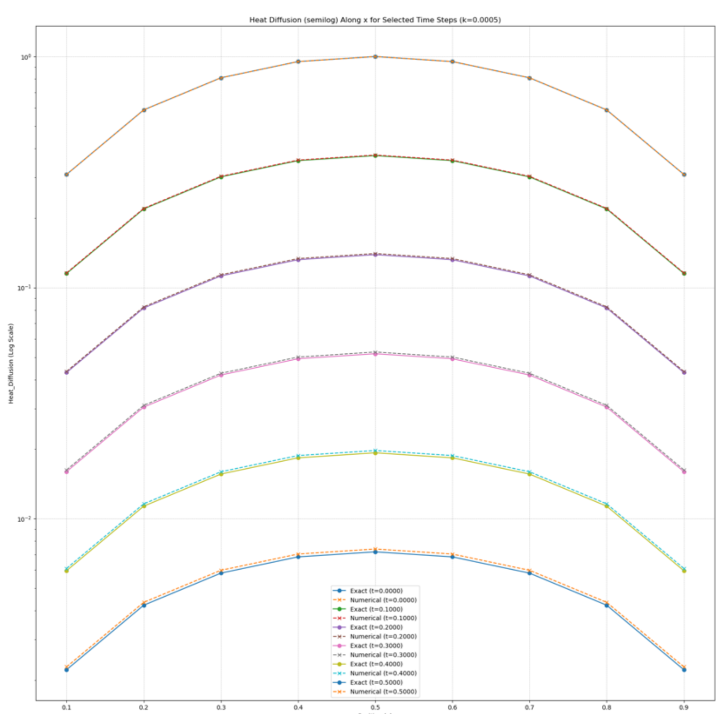


Student Name: Mina Gerges
ID: 70040441
Computational Physics_ PHYS624
HW6: Pratial Differential Equations

Plot4: presents a comparison of the numerical and the exact solution at different time steps.
Once again the stability is clear at all those time steps.



Plot5: Presents an overplot of the numerical and the exact solution in a semilog format for the heat diffusion.
This plots aims to clarify the color map(plot1) and the surface plot(plot2) mentioned above.
One cann easily understand the at the boundaries the function yields zero which is not included in the plot as log(0) is undefined, therfore I masked the results in position range((x_values >= 0.1) & (x_values <= 0.9))



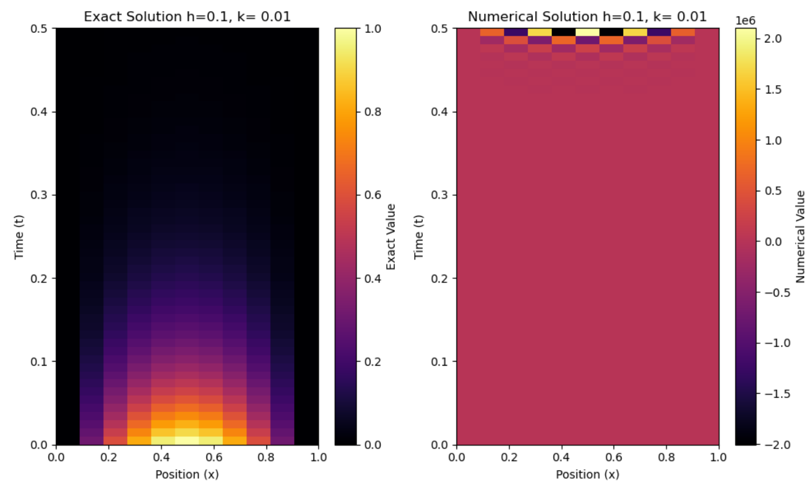
Student Name: Mina Gerges
ID: 70040441
Computational Physics_ PHYS624
HW6: Pratial Differential Equations

At h= 0.1 and k=0.01

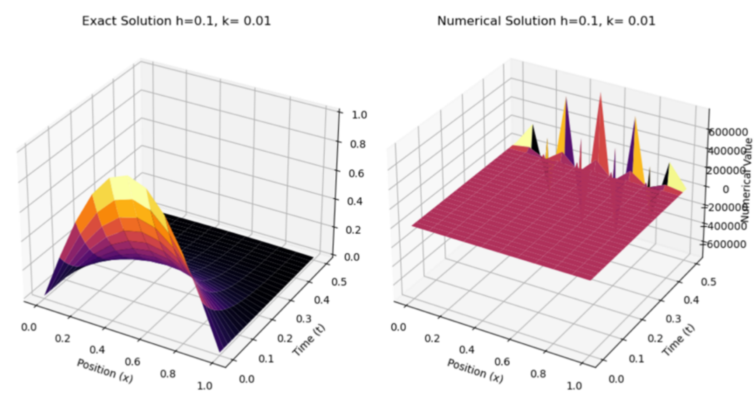
Stability condition is not achieved, where alpha =1

x	t	u_exact	u_numerical
0	0.5	0	0
0.1	0.5	0.00222241	657271
0.2	0.5	0.00422728	-1.24783e+06
0.3	0.5	0.00581836	1.71241e+06
0.4	0.5	0.00683989	-2.00552e+06
0.5	0.5	0.00719188	2.09995e+06
0.6	0.5	0.00683989	-1.98881e+06
0.7	0.5	0.00581836	1.68537e+06
0.8	0.5	0.00422728	-1.22079e+06
0.9	0.5	0.00222241	640558
1	0.5	0	0

Plot6: Presents the Heat diffusion map over spatial and temporal steps



Plot7: Presents a surface plot of the same solution presented above.



Plot8: Presents the heat distribution at time step=0.5 along the x-position with as step k=0.01

It is very clear that the solution is unstable here as the value of alpha = 1, therefore we cannot consider larger steps in the explicit scheme of solving PDE's.