Tutorial 2

https://github.com/Blink29/Numerical-Techniques → Assignment 2

Name: Paurush Kumar Roll Number: NA22B002

Problem Statement

We aim to solve a one-dimensional heat diffusion equation in steady state with the inclusion of source terms, represented as:

$$\alpha \frac{d^2T}{dx^2} + Q(x) = 0$$

Given α = 0.0001, Q(+0.25) = 1, Q(+0.75) = -1, Solve for T(0, 1) with Boundary Conditions as $dt/dx = 0$ at x =0, 1. (Dirichlet boundary condition).

We solve this equation numerically using:

1. **Gaussian Elimination** (Direct Method),
2. **Gauss-Seidel Iteration** (Iterative Method)

Mathematical Formulation

Using finite difference discretization for the second derivative:

$$\frac{d^2T}{dx^2} \approx \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2}$$

The governing equation becomes:

$$\alpha \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2} + Q_i = 0$$

Rearranging

$$T_{i-1} + T_{i+1} - 2T_i + \frac{\Delta x^2}{\alpha} Q_i = 0$$

This equation forms a tridiagonal matrix system, which we solve using the above methods.

Code

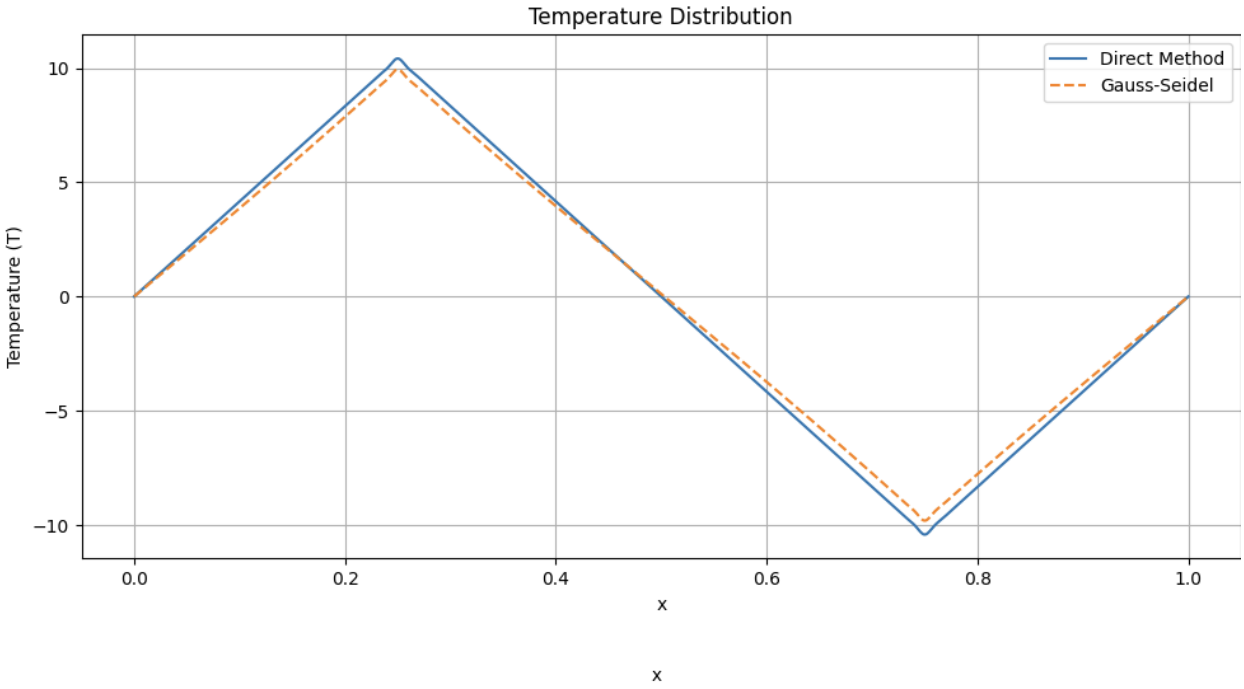
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d

# Parameters
alpha = 0.0001
L = 1.0
N = 120
dx = L / N

# Grid
x = np.linspace(0, L, N+1)

# Initialize arrays with new size
Q = np.zeros(N+1)
Q[int(0.25 * N)] = 1
Q[int(0.75 * N)] = -1

# Recreate coefficient matrix A and vector b with new size
A = np.zeros((N+1, N+1))
b = np.zeros(N+1)

# Fill A and b with finer discretization
for i in range(1, N):
    A[i, i-1] = alpha / dx**2
    A[i, i] = -2 * alpha / dx**2
    A[i, i+1] = alpha / dx**2
    b[i] = -Q[i]
```

```
# Apply boundary conditions
A[0, 0] = 1
A[N, N] = 1
b[0] = 0
b[N] = 0

# Direct method
def gaussian_elimination(A, b):
    n = len(A)
    # Make copies to avoid modifying original arrays
    A = A.copy()
    b = b.copy()
    x = np.zeros(n)

    # Forward elimination
    for i in range(n-1):
        # Find pivot
        pivot = A[i][i]

        # Eliminate column i
        for j in range(i+1, n):
            factor = A[j][i] / pivot
            for k in range(i, n):
                A[j][k] = A[j][k] - factor * A[i][k]
            b[j] = b[j] - factor * b[i]

    # Back substitution
    x[n-1] = b[n-1] / A[n-1][n-1]
    for i in range(n-2, -1, -1):
        sum_ax = 0
        for j in range(i+1, n):
            sum_ax += A[i][j] * x[j]
        x[i] = (b[i] - sum_ax) / A[i][i]

    return x
```

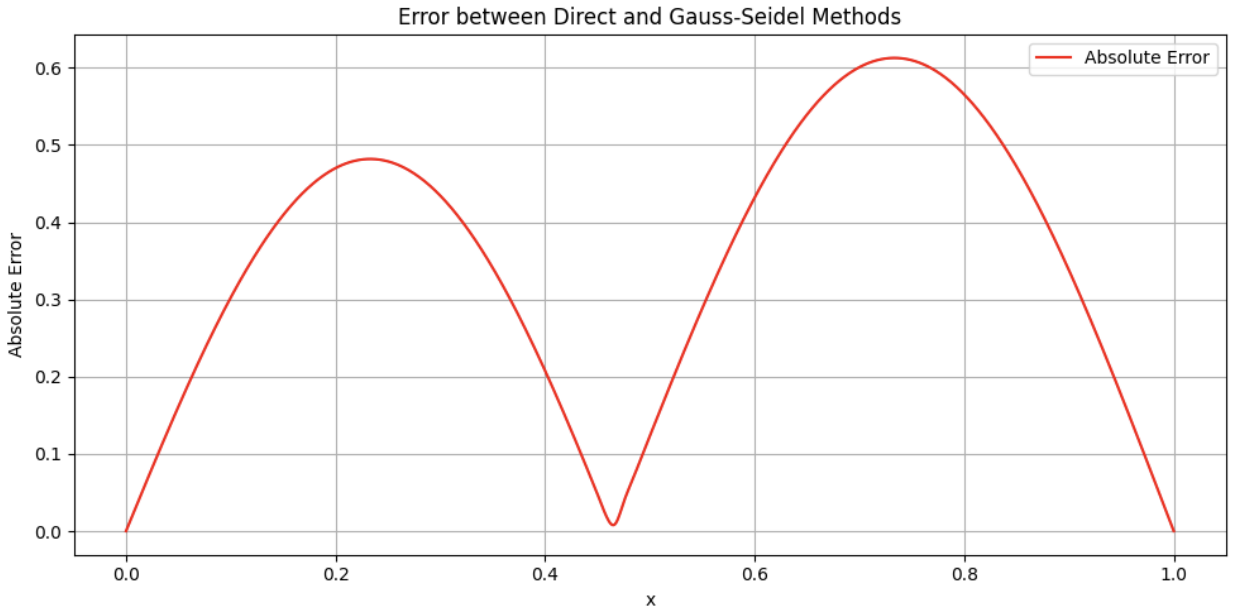
```
T_gauss_direct = gaussian_elimination(A, b)

# Gauss-Seidel method
def gauss_seidel(A, b, tol=1e-10, max_iter=1000):
    T = np.zeros_like(b)
    for _ in range(max_iter):
        T_old = T.copy()
        for i in range(len(T)):
            sigma = 0
            for j in range(len(T)):
                if j != i:
                    sigma += A[i, j] * T[j]
            T[i] = (b[i] - sigma) / A[i, i]
        if np.linalg.norm(T - T_old, ord=np.inf) < tol:
            break
    return T

T_gauss_seidal = gauss_seidel(A, b)

error = np.abs(T_gauss_direct - T_gauss_seidal)

x_fine = np.linspace(0, L, 1000)

# Create interpolation functions
T_direct_interp = interp1d(x, T_gauss_direct, kind='cubic')
T_gs_interp = interp1d(x, T_gauss_seidal, kind='cubic')
error_interp = interp1d(x, error, kind='cubic')

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 10))

# Plot temperature distribution
ax1.plot(x_fine, T_direct_interp(x_fine), label="Direct Metho
d")
ax1.plot(x_fine, T_gs_interp(x_fine), label="Gauss-Seidel", l
inestyle='--')
```

```
ax1.set_xlabel("x")
ax1.set_ylabel("Temperature (T)")
ax1.set_title("Temperature Distribution")
ax1.legend()
ax1.grid(True)

# Plot error
ax2.plot(x_fine, error_interp(x_fine), 'r-', label="Absolute
Error")
ax2.set_xlabel("x")
ax2.set_ylabel("Absolute Error")
ax2.set_title("Error between Direct and Gauss-Seidel Method
s")
ax2.legend()
ax2.grid(True)

plt.tight_layout()
plt.savefig("temperature_and_error.png")
plt.show()

print("Gaussian Direct Solution:", T_gauss_direct)
print("Gauss-Seidel Method Solution:", T_gauss_seidal)
print("Maximum Error:", np.max(error))
```

Results



x



Conclusion

- The **Direct Method** (Gaussian Elimination) provides an exact numerical solution within machine precision.
- The **Gauss-Seidel Method** converges iteratively and approximates the solution.
- Maximum Error between the two methods : 0.6125996139776451