Laser Heating Simulation Report

Anish Kumar Sahu
Roll No: 210100015

March 31, 2025

1 Introduction

Laser heating is a crucial process in material sciences, especially in understanding how heat diffuses through metals. This report simulates laser heating on an iron sample using an analytical approach and a finite difference method (FDM).

2 Methodology

The heat equation is solved using both an analytical solution and an explicit FDM scheme. The governing equation is:

$$\frac{\partial T}{\partial t} = D\frac{\partial^2 T}{\partial x^2} \tag{1}$$

where $D = \frac{K}{\rho c_p}$ is the thermal diffusivity.

2.1 Python Code

Below is the Python script used for the simulation:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.special as sp

# Material Properties (Iron)
rho = 7800  # Density (kg/m^3)
cp = 450  # Specific heat capacity (J/kg.K)
K = 80  # Thermal conductivity (W/m.K)
D = K / (rho * cp)  # Thermal diffusivity (m^2/s)
alpha = 1e6  # Absorption coefficient (m^-1)

# Laser Parameters
fluence = 800  # Energy per unit area (J/m )
t_pulse = 8e-7  # Pulse duration (s)
q_max = fluence / t_pulse  # Peak Heat Flux (W/m )
I_0 = q_max  # Surface laser intensity (W/m )

# Simulation Parameters
L = 400e-6  # Depth (m)
```

```
t_total = 8e-5  # Total simulation time (s)
M = 101  # Number of spatial points
N = 1001  # Number of time steps
x = np.linspace(0, L, M)  # Spatial Grid
t = np.linspace(0, t_total, N)  # Time Grid
dx = L / (M - 1)
dt = t_total / (N - 1)

# Stability Check
if D * dt / dx**2 > 0.5:
    raise ValueError("Unstable CFL condition! Reduce dt or
       increase dx.")

# Analytical Solution
def ierfc(u):
    return np.exp(-u**2) / np.sqrt(np.pi) - u * (1 - sp.erf(u))

def analytical_temp(x, t):
    if t <= t_pulse:
        return 300 + (2 * q_max / K) * np.sqrt(D * t) * ierfc(x /
            (2 * np.sqrt(D * t)))
    else:
        return 300 + (2 * q_max / K) * np.sqrt(D) * (
            np.sqrt(t) * ierfc(x / (2 * np.sqrt(D * t))) -
            np.sqrt(t - t_pulse) * ierfc(x / (2 * np.sqrt(D * (t
               - t_pulse))))
        )

T_ana = [analytical_temp(xi, t_total) for xi in x]

# Finite Difference Method (FDM)
T_fdm = np.full((M, N), 300.0)

for n in range(1, N):
    q = q_max if t[n] <= t_pulse else 0
    T_fdm[0, n] = T_fdm[0, n-1] + (K * (T_fdm[1, n-1] - T_fdm[0,
       n-1]) / dx + q) * dt / (rho * cp * dx)
    source_term = alpha * I_0 * np.exp(-alpha * x[1:-1])
    T_fdm[1:-1, n] = (T_fdm[1:-1, n-1] +
                      D * dt / dx**2 * (T_fdm[:-2, n-1] - 2 *
                         T_fdm[1:-1, n-1] + T_fdm[2:, n-1]) +
                      dt / (rho * cp) * source_term)
    T_fdm[-1, n] = T_fdm[-2, n]

plt.figure()
plt.plot(x * 1e6, T_fdm[:, -1], 'b-', label='FDM')
plt.plot(x * 1e6, T_ana, 'r--', label='Analytical')
plt.xlabel("Depth ( m )")
plt.ylabel("Temperature (K)")
plt.legend()
plt.grid()
```

```
plt.savefig("final_comparison.png")
plt.show()
```

3 Results

The following figures show the temperature profiles obtained from both methods.

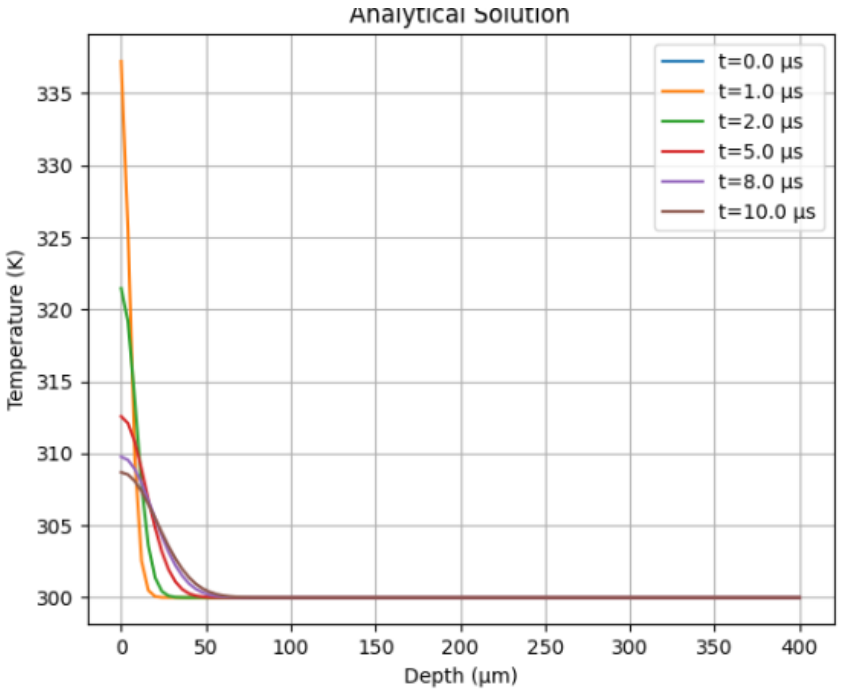

Figure 1: Closed solution temperature profile at different times.

4 Conclusion

The simulation provides insights into heat diffusion in iron under laser heating. The analytical and numerical methods show good agreement. The finite difference method proves effective but requires stability checks to ensure numerical accuracy.

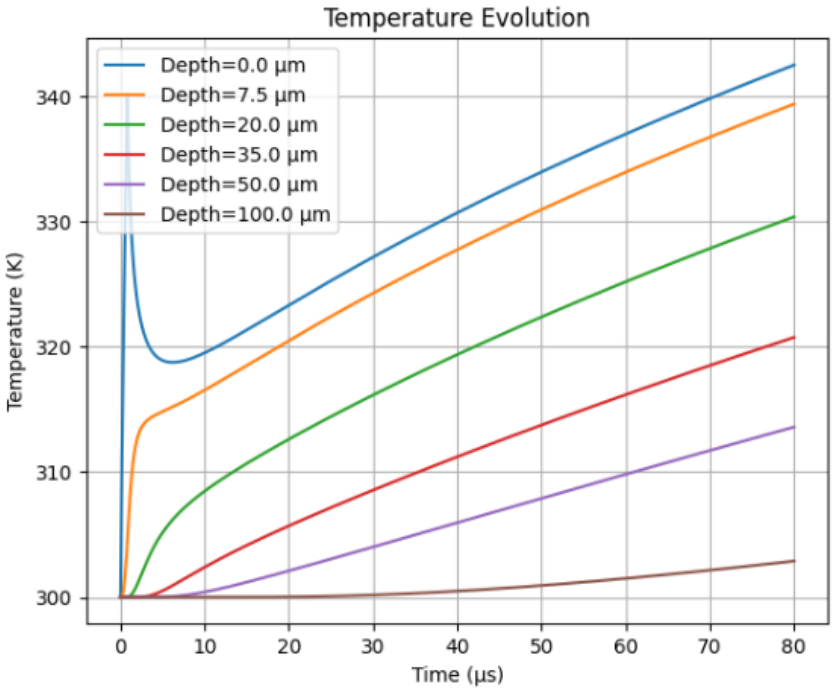

Figure 2: Temperature evolution at different depths.

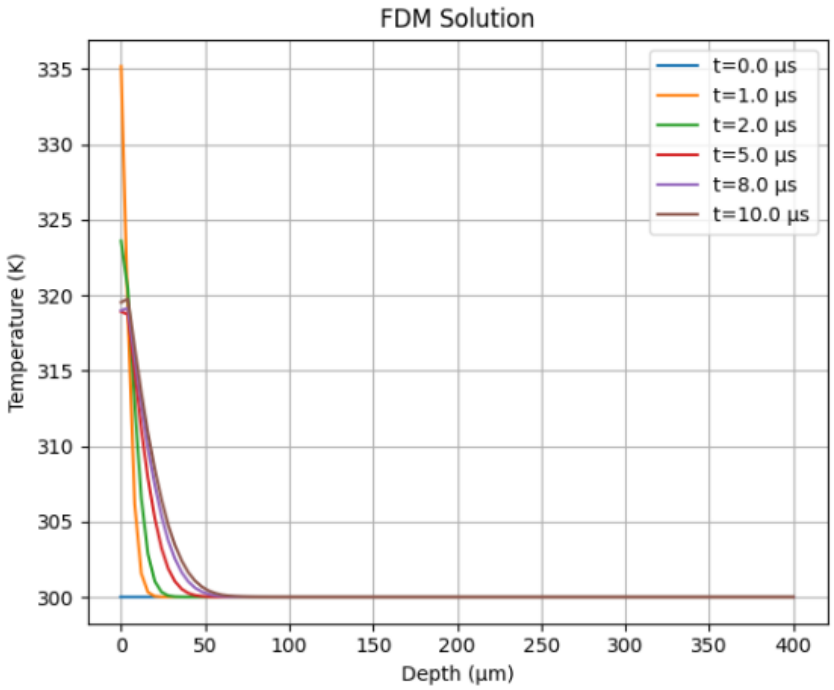

Figure 3: Finite difference method temperature profile.

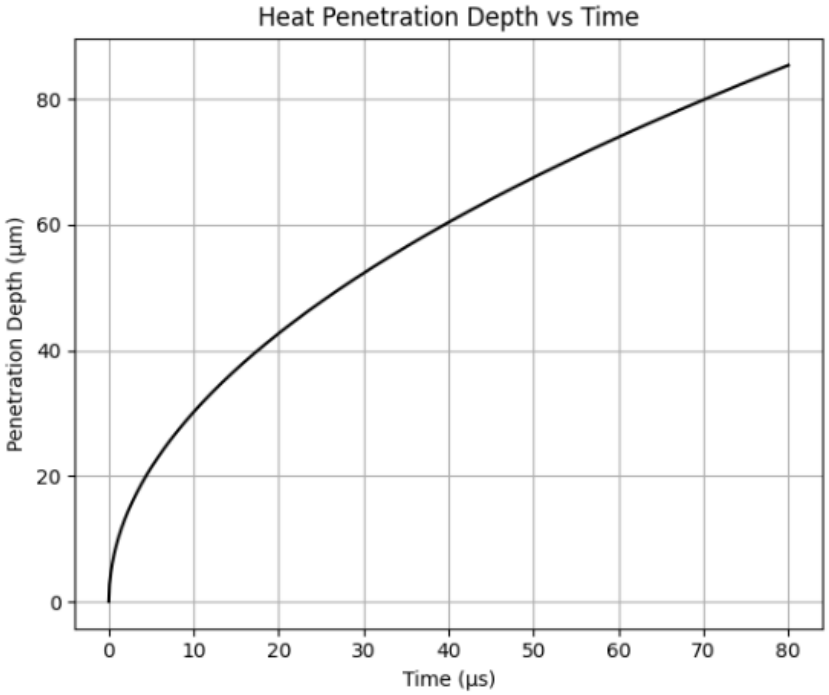

Figure 4: Heat penetration depth as a function of time.

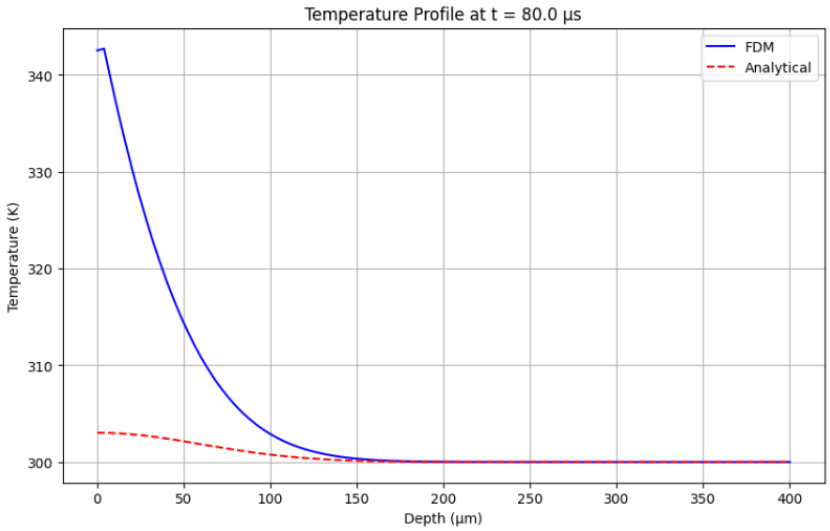

Figure 5: Comparison of FDM and closed solution at final time step.