Derivation and Numerical Solution of the Heat Conduction Equation with Detailed Code Explanation

Yazhuo Liu

March 1, 2025

Contents

1 Introduction

This document presents a comprehensive derivation of the heat conduction equation starting from the integral forms of the conservation laws, formation of the weak form of the partial differential equation (PDE) including boundary conditions, and implementation of the numerical solution using FEniCS with detailed line-by-line explanations of the code.

2 Derivation of the Heat Conduction Equation

2.1 Integral Conservation Laws

The conservation laws of mass, momentum, and energy are fundamental principles governing the behavior of physical systems. They can be expressed in integral form over a control volume V bounded by a closed surface S.

Material derivative :

$$\frac{D}{Dt}(*) = \frac{\partial}{\partial t}(*) + \vec{v} \cdot \nabla(*) \tag{1}$$

The material derivative is used to describe time rates of change for a given particle. The first term on RHS represents the self change, and the second term represents the flow/flux.

2.1.1 Conservation of Mass

The integral form of the conservation of mass is:

$$\frac{D}{Dt}\left(\int_V \rho \, dV\right) = 0 \tag{2}$$

$$\implies \frac{\partial}{\partial t}\int_V \rho \, dV + \int_S \rho \vec{v} \cdot \vec{n} \, dS = 0 \tag{3}$$

where:

- ρ is the density.
- $\vec{v}$ is the velocity vector.
- $\vec{n}$ is the outward-pointing unit normal vector on the surface S.

2.1.2 Conservation of Momentum

The integral form of the conservation of momentum is:

$$\frac{D}{Dt}\left(\int_V \rho \vec{v} \, dV\right) = \sum F_{\text{ext}} \tag{4}$$

$$\implies \frac{\partial}{\partial t}\int_V \rho \vec{v} \, dV + \int_S \rho \vec{v}(\vec{v} \cdot \vec{n}) \, dS = \int_S \vec{n} \cdot \vec{\sigma} \, dS + \int_V \rho \vec{g} \, dV \tag{5}$$

where:

- $\vec{\sigma}$ is the stress tensor.
- $\vec{g}$ is the body force per unit mass (e.g., gravity).

2.1.3 Conservation of Energy

The integral form of the conservation of energy is:

$$\frac{D}{Dt}(\int_V \rho e \, dV) = \dot{Q}_{\text{in}} + \dot{W}_{\text{in}} \tag{6}$$

$$\implies \frac{\partial}{\partial t} \int_V \rho e \, dV + \int_S \rho e \vec{v} \cdot \vec{n} \, dS = -\int_S \vec{q} \cdot \vec{n} \, dS + \int_V \dot{q} \, dV + \int_S (\vec{\sigma} \cdot \vec{v}) \cdot \vec{n} \, dS \tag{7}$$

where:

- e is the internal energy per unit mass.
- $\vec{q}$ is the heat flux vector.
- $\dot{q}$ represents volumetric heat sources.

2.2 Derivation of Differential Forms

To obtain the differential forms of the conservation equations, we apply the divergence theorem and consider the control volume to be fixed in space.

2.2.1 Conservation of Mass

Starting with the integral form:

$$\frac{\partial}{\partial t} \int_V \rho \, dV + \int_S \rho \vec{v} \cdot \vec{n} \, dS = 0 \tag{8}$$

Applying the divergence theorem:

$$\int_V \frac{\partial \rho}{\partial t} \, dV + \int_V \nabla \cdot (\rho \vec{v}) \, dV = 0 \tag{9}$$

Since the control volume V is arbitrary, the integrand must be zero:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \tag{10}$$

2.2.2 Conservation of Momentum

Starting with the integral form:

$$\frac{\partial}{\partial t} \int_V \rho \vec{v} \, dV + \int_S \rho \vec{v} (\vec{v} \cdot \vec{n}) \, dS = \int_S \vec{\sigma} \cdot \vec{n} \, dS + \int_V \rho \vec{g} \, dV \tag{11}$$

Applying the divergence theorem:

$$\int_V \frac{\partial}{\partial t}(\rho\vec{v})\,dV + \int_V \nabla\cdot(\rho\vec{v}\otimes\vec{v})\,dV = \int_V \nabla\cdot\vec{\sigma}\,dV + \int_V \rho\vec{g}\,dV \tag{12}$$

Simplifying:

$$\frac{\partial}{\partial t}(\rho\vec{v}) + \nabla\cdot(\rho\vec{v}\otimes\vec{v}) = \nabla\cdot\vec{\sigma} + \rho\vec{g} \tag{13}$$

2.2.3 Conservation of Energy

Starting with the integral form:

$$\frac{\partial}{\partial t}\int_V \rho e\,dV + \int_S \rho e\vec{v}\cdot\vec{n}\,dS = -\int_S \vec{q}\cdot\vec{n}\,dS + \int_V \rho\dot{q}\,dV + \int_S \vec{n}\cdot(\vec{\sigma}\cdot\vec{v})\,dS \tag{14}$$

Applying the divergence theorem:

$$\int_V \frac{\partial}{\partial t}(\rho e)\,dV + \int_V \nabla\cdot(\rho e\vec{v})\,dV = -\int_V \nabla\cdot\vec{q}\,dV + \int_V \rho\dot{q}\,dV + \int_V \nabla\cdot(\vec{\sigma}\cdot\vec{v})\,dV \tag{15}$$

Simplifying:

$$\frac{\partial}{\partial t}(\rho e) + \nabla\cdot(\rho e\vec{v}) = -\nabla\cdot\vec{q} + \dot{q} + \nabla\cdot(\vec{\sigma}\cdot\vec{v}) \tag{16}$$

2.3 Simplifying Under Assumptions

We make the following assumptions to simplify the equations:

1. **Stationary Medium:** $\vec{v} = 0$
2. **Constant Properties:** ρ, c, k are constants
3. **Negligible Viscous Dissipation:** $\nabla\cdot(\vec{\sigma}\cdot\vec{v}) = 0$

2.3.1 Simplified Conservation of Mass

With $\vec{v} = 0$:

$$\frac{\partial\rho}{\partial t} = 0 \tag{17}$$

Since ρ is constant, this equation is satisfied.

2.3.2 Simplified Conservation of Momentum

With $\vec{v} = 0$:

$$\nabla\cdot\sigma = \rho\vec{g} \tag{18}$$

This is the equilibrium equation. Typically, we assume the body forces is negligible ($\rho\vec{g} = 0$).

2.3.3 Simplified Conservation of Energy

With $\vec{v} = 0$ and $\nabla \cdot (\vec{\sigma} \cdot \vec{v}) = 0$:

$$\frac{\partial}{\partial t}(\rho e) = -\nabla \cdot \vec{q} + \dot{q} \tag{19}$$

2.4 Relating Internal Energy to Temperature

Assuming the internal energy per unit mass e is related to temperature T by:

$$e = c_p T \tag{20}$$

where c_p is the specific heat capacity at constant volume.
Differentiating with respect to time:

$$\frac{\partial}{\partial t}(\rho e) = \rho c_p \frac{\partial T}{\partial t} \tag{21}$$

2.5 Fourier's Law of Heat Conduction

Fourier's law relates the heat flux $\vec{q}$ to the temperature gradient:

$$\vec{q} = -k \nabla T \tag{22}$$

where k is the thermal conductivity.
Substituting Fourier's law into the simplified energy equation:

$$\rho c_p \frac{\partial T}{\partial t} = -\nabla \cdot (-k \nabla T) + \dot{q} \tag{23}$$

Simplifying:

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + \dot{q} \tag{24}$$

Assuming k is constant:

$$\rho c_p \frac{\partial T}{\partial t} = k \nabla^2 T + \dot{q} \tag{25}$$

2.6 Defining Thermal Diffusivity

Thermal diffusivity α is defined as:

$$\alpha = \frac{k}{\rho c_p} \tag{26}$$

Substituting α into the equation:

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T + \frac{\dot{q}}{\rho c_p} \tag{27}$$

This is the **heat conduction equation**.

2.7 Final Heat Conduction Equation

The general heat conduction equation in terms of temperature T is:

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + \dot{q} \tag{28}$$

This partial differential equation describes how temperature changes with time due to heat conduction in a stationary medium with constant properties.

3 Weak Formulation of the Heat Conduction Equation

3.1 Strong Form of the PDE

The heat conduction equation (strong form) is:

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + \dot{q} \quad \text{in} \quad \Omega \tag{29}$$

with boundary conditions:

$$T = \bar{T} \quad \text{on} \quad \partial\Omega_T \quad \text{(Dirichlet condition)} \tag{30}$$

$$-k \frac{\partial T}{\partial n} = -k \nabla T \cdot \mathbf{n} = \bar{q} \quad \text{on} \quad \partial\Omega_q \quad \text{(Neumann condition)} \tag{31}$$

Note: $\bar{q}$ is the heat flux going inside.

3.2 Formation of the Weak Form

To derive the weak form, we multiply both sides of the PDE by a test function $w \in V_0$ (space of admissible test functions) and integrate over the domain Ω:

$$\int_\Omega w \rho c_p \frac{\partial T}{\partial t} dV = \int_\Omega w \nabla \cdot (k \nabla T) dV + \int_\Omega w \dot{q} dV \tag{32}$$

3.2.1 Integration by Parts

We apply integration by parts to the right-hand side to reduce the order of differentiation on T:

$$\int_\Omega w \nabla \cdot (k \nabla T) dV = -\int_\Omega \nabla w \cdot (k \nabla T) dV + \int_{\partial\Omega} w (k \nabla T \cdot \mathbf{n}) dS \tag{33}$$

Substituting back:

$$\int_\Omega w \rho c_p \frac{\partial T}{\partial t} dV = -\int_\Omega \nabla w \cdot (k \nabla T) dV + \int_{\partial\Omega} w (k \nabla T \cdot \mathbf{n}) dS + \int_\Omega w \dot{q} dV \tag{34}$$

3.2.2 Incorporating Boundary Conditions

Since $w = 0$ on $\partial\Omega_T$ (Dirichlet boundary), the boundary integral reduces to:

$$\int_{\partial\Omega_q} w(k\nabla T \cdot \mathbf{n})dS \tag{35}$$

Using the Neumann boundary condition $k\nabla T \cdot \mathbf{n} = -\bar{q}$:

$$\int_{\partial\Omega_q} w(k\nabla T \cdot \mathbf{n})dS = -\int_{\partial\Omega_q} w\bar{q}dS \tag{36}$$

3.2.3 Final Weak Formulation

Collecting terms, the weak form is:

$$\underbrace{\int_{\Omega} w\rho c_p \frac{\partial T}{\partial t}dV}_{\text{Transient Term}} + \underbrace{\int_{\Omega} \nabla w \cdot (k\nabla T)dV}_{\text{Diffusion Term}} = \underbrace{\int_{\Omega} w\dot{q}dV}_{\text{Source Term}} - \underbrace{\int_{\partial\Omega_q} w\bar{q}dS}_{\text{Neumann Boundary}} \tag{37}$$

3.3 Function Spaces

- **Trial Function Space** V:

$$V = \{T \in H^1(\Omega) \,|\, T = \bar{T} \text{ on } \partial\Omega_T\} \tag{38}$$

- **Test Function Space** V_0:

$$V_0 = \{w \in H^1(\Omega) \,|\, w = 0 \text{ on } \partial\Omega_T\} \tag{39}$$

4 LPBF Heat Sources

4.1 Classic Double Ellipsoid Laser Heat Source Model

The double ellipsoid model divides the heat source into front and rear semi-ellipsoids, with the total power distribution $\dot{q} = \dot{q}_{\text{front}} + \dot{q}_{\text{rear}}$. The mathematical expression is:

$$\dot{q}(x,y,z) = \begin{cases} \frac{6\sqrt{3}f_1Q_0}{a_1bc\pi\sqrt{\pi}} \exp\left(-3\left(\frac{x^2}{a_1^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2}\right)\right) & \text{(Front semi-ellipsoid, } x \geq 0\text{)} \\ \frac{6\sqrt{3}f_2Q_0}{a_2bc\pi\sqrt{\pi}} \exp\left(-3\left(\frac{x^2}{a_2^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2}\right)\right) & \text{(Rear semi-ellipsoid, } x < 0\text{)} \end{cases} \tag{40}$$

Parameters:

- Q_0: Total laser power (W).
- f_1, f_2: Power distribution coefficients for the front and rear semi-ellipsoids, satisfying $f_1 + f_2 = 2$.

- a_1, a_2: Axial lengths along the laser scanning direction (x-axis) for the front and rear semi-ellipsoids (m).
- b: Transverse axial length perpendicular to the scanning direction (m).
- c: Depth-wise axial length (m).

4.1.1 Improvements for LPBF Process

- **Dynamic Coordinate Transformation**

  The laser scanning speed v significantly affects the melt pool shape. A fixed coordinate system is transformed into a **moving coordinate system** to dynamically track the heat source position:
  $$x' = x - vt, \quad y' = y, \quad z' = z$$
  In the moving coordinate system, the heat source model incorporates time t to reflect the transient behavior during scanning.

- **Anisotropic Axial Length Adjustment**

  The melt pool shape is influenced by material thermophysical properties (e.g., thermal conductivity, specific heat) and process parameters (e.g., power, scanning speed). The axial lengths a_1, a_2, b, c can be calibrated via experiments or simulations:
  $$a_1(t) = k_1 \frac{Q_0}{v(t)\rho c_p}, \quad a_2(t) = k_2 \frac{Q_0}{v(t)\rho c_p}, \quad b(t) = k_3 \frac{Q_0}{v(t)\rho c_p}, \quad c(t) = k_4 \frac{Q_0}{v(t)\rho c_p}$$
  where k_1, k_2, k_3, k_4 are empirical coefficients, ρ is density, and c_p is specific heat.

- **Thermal Property at Different Temperature**

  The thermal conductivity k of material differs significantly with respect to temperature. It can be expressed as a function of temperature or density:
  $$k(T) = k_{\text{bulk}} \cdot (1 + \alpha(T - T_{\text{melt}}))$$

4.2 Convective Heat Flux

$$\bar{q}_{\text{conv}} = h(T_\infty - T) \tag{41}$$

- h: Convective heat transfer coefficient (W/(m^2·K))
- T_∞: Ambient temperature (K)

4.3 Radiative Heat Flux

$$\bar{q}_{\text{rad}} = \varepsilon \sigma_{\text{SB}} \left(T_\infty^4 - T^4 \right) \tag{42}$$

- ε: Emissivity (dimensionless)
- σ_{SB}: Stefan-Boltzmann constant (W/(m^2·K^4))

5 Numerical Solution Using FEniCS

We will solve the heat conduction equation using FEniCS for a cubic domain of size $1000 \times 600 \times 300$ µm with the following boundary conditions:

- **Bottom Face (z = 0):** $T = 300\,\mathrm{K}$
- **Top Face (z = 300):** A moving double ellipsoid laser heat source.
- **Other Boundaries:** Convection and radiation to room temperature.

5.1 FEniCS Code with Line-by-Line Explanations

Below is the FEniCS code with detailed explanations for each part.

```
from fenics import *
import numpy as np
import os
from mpi4py import MPI
import glob

# Enable optimization for compilation
parameters["form_compiler"]["optimize"] = True
parameters["form_compiler"]["cpp_optimize"] = True
parameters["form_compiler"]["cpp_optimize_flags"] = "-O3 -ffast-math
    -march=native"
parameters["form_compiler"]["quadrature_degree"] = 2
parameters["form_compiler"]["representation"] = "uflacs"

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

if rank > 0: # Suppress output from non-master processes
    set_log_level(LogLevel.WARNING)

if rank == 0: # Only the master process creates the directory
    if not os.path.exists("results"):
        os.makedirs("results")
    else:
        # Clear existing files
        files = glob.glob('results/*')
        for f in files:
            os.remove(f)
comm.barrier() # Ensure all processes wait until the directory is
   created

# Material Properties
rho = Constant(2700.0)      # Density [kg/m^3]
cp = Constant(900.0)        # Specific heat capacity [J/(kg*K)]
```

```python
k_bulk = Constant(237.0)    # Base thermal conductivity [W/(m*K)]
alpha = Constant(1e-3)      # Thermal conductivity temperature coefficient [1/K]

# Boundary condition parameters
h = 10.0                    # Convection coefficient [W/(m^2*K)]
T_inf = 300.0               # Ambient temperature [K]
epsilon = 0.5               # Emissivity
sigma_SB = 5.67e-8          # Stefan-Boltzmann constant

# Double ellipsoid heat source parameters
Q0 = 100.0                  # Laser power [W]
v = 1.0                     # Scanning speed [m/s]
a1, a2, b, c = 50e-6, 50e-6, 50e-6, 50e-6  # Ellipsoid parameters
f1, f2 = 0.6, 1.4           # Power distribution coefficients

# Define computational domain (unit: meters)
Lx, Ly, Lz = 1000e-6, 600e-6, 300e-6
# mesh = BoxMesh(comm, Point(0, 0, 0), Point(Lx, Ly, Lz), 100, 60, 30)
mesh = BoxMesh(Point(0, 0, 0), Point(Lx, Ly, Lz), 100, 60, 30)

# Time parameters
t_total = Lx/v              # Total time [s]
dt = t_total/100            # Time step [s]
num_steps = int(t_total/dt)

# Define function space
V = FunctionSpace(mesh, 'P', 1)

# Define test functions and unknown functions
w = TestFunction(V)
T = Function(V)      # Temperature field at current time step
T_n = Function(V)     # Temperature field at previous time step

# Initial condition
T_n = interpolate(Constant(T_inf), V)  # Initial temperature field

# Boundary condition definitions
# Define boundary locations
def top(x, on_boundary):
    return near(x[2], Lz) and on_boundary

def bottom(x, on_boundary):
    return near(x[2], 0.) and on_boundary

def walls(x, on_boundary):
    left = near(x[0], 0.) and on_boundary
```

```
    right = near(x[0], Lx) and on_boundary
    front = near(x[1], 0.) and on_boundary
    back = near(x[1], Ly) and on_boundary
    return left | right | front | back

# Mark boundaries
boundaries = MeshFunction("size_t", mesh, mesh.topology().dim() - 1, 0)
Top = AutoSubDomain(top)
Top.mark(boundaries, 1)
Bottom = AutoSubDomain(bottom)
Bottom.mark(boundaries, 2)
Walls = AutoSubDomain(walls)
Walls.mark(boundaries, 3)
ds = Measure('ds', domain=mesh, subdomain_data=boundaries)  # Redefine the measure 'ds' with subdomains

bc = DirichletBC(V, Constant(300.0), bottom)
bcs = [bc]

# Temperature-dependent thermal conductivity
def thermal_conductivity(T):
    return k_bulk * (1 + alpha*(T - 300))

# Define volumetric heat source for laser scanning
class HeatSource(UserExpression):
    def __init__(self, position, velocity, Q0, f1, f2, a1, a2, b, c, t, **kwargs):
        super().__init__(**kwargs)

        # Validate that position is a vector of length 3
        if len(position) != 3:
            raise ValueError("Position must be a vector with exactly 3 elements (x, y, z).")

        # Validate that velocity is a vector of length 3
        if len(velocity) != 3:
            raise ValueError("Velocity must be a vector with exactly 3 elements (vx, vy, vz).")

        self.position = np.array(position, dtype=float)  # Initial position [x0, y0, z0]
        self.velocity = np.array(velocity, dtype=float)  # Velocity [vx, vy, vz]
        self.Q0 = Q0
        self.f1 = f1
        self.f2 = f2
        self.a1 = a1
```

```python
        self.a2 = a2
        self.b = b
        self.c = c
        self.t = t

    def eval(self, value, x):

        laser_center = self.position + self.velocity * self.t  # Laser center position

        # Relative position
        x_prime = x - laser_center

        # Calculate heat source intensity
        if x_prime[0] >= 0:
            coeff = 6*sqrt(3) * self.f1 * self.Q0/(self.a1*self.b*self.c*np.pi*sqrt(np.pi))
            exponent = -3*((x_prime[0])**2/self.a1**2 + x_prime[1]**2/self.b**2 + x_prime[2]**2/self.c**2)
        else:
            coeff = 6*sqrt(3) * self.f2 * self.Q0/(self.a2*self.b*self.c*np.pi*sqrt(np.pi))
            exponent = -3*((x_prime[0])**2/self.a2**2 + x_prime[1]**2/self.b**2 + x_prime[2]**2/self.c**2)

        value[0] = coeff * exp(exponent)

    def value_shape(self):
        return ()

# Create heat source object
position = [-a1, Ly/2, Lz]
velocity = [v, 0.0, 0.0]
q_dot = HeatSource(position, velocity, Q0, f1, f2, a1, a2, b, c, t=0, degree=1)

# Define radiative heat flux
q_bar_conv = h * (T_inf - T)  # Convective heat flux
q_bar_rad = epsilon * sigma_SB * (T_inf**4 - T**4)  # Radiative heat flux

# Define variational form
F = w * rho * cp * (T - T_n) / dt * dx \
    + inner(grad(w), thermal_conductivity(T)*grad(T))*dx \
    - w * q_dot * dx \
    + w * (q_bar_rad + q_bar_conv) * ds(3)

# Create nonlinear problem and solver
```

```
problem = NonlinearVariationalProblem(F, T, bc, J=derivative(F, T))
solver = NonlinearVariationalSolver(problem)
solver.parameters['newton_solver']['relaxation_parameter'] = 1.0
solver.parameters["newton_solver"]["linear_solver"] = "cg"
solver.parameters["newton_solver"]["maximum_iterations"] = 50


# Time-stepping loop ===========================================================
t = 0.0  # Initialize time

# Create file to save results
file = XDMFFile("results/temperature.xdmf")
file.parameters["flush_output"] = True
file.write(mesh)
file.write(T_n, t)


# Initialize heat source object (Note: time parameter must be initialized first)
q_dot.t = t

# Progress counter
progress_counter = 0

for step in range(num_steps):
    # Update time
    t += dt

    # Print progress information
    if rank == 0:
        print(f"\n======= Computing time step {step + 1}/{num_steps} [t = {t:.6f} s] =======")

    # Update heat source position (Critical step!)
    # --------------------------------------------------
    q_dot.t = t  # Update heat source time parameter

    # Solve nonlinear problem
    # --------------------------------------------------
    try:
        # Call solver
        solver.solve()

        # Check solution validity
        max_T = T.vector().max()
        min_T = T.vector().min()
        if max_T > 5000 or min_T < 0:
```

```
            print(f"Warning: Temperature solution out of physical range! Max temperature: {max_T} K, Min temperature: {min_T} K")
            break

    except Exception as e:
        print(f"Solving failed at time step {step}, error message:")
        print(str(e))
        break

    # Update solution from the previous time step
    # --------------------------------------------------
    T_n.assign(T)

    # Save results (every 10 steps)
    # --------------------------------------------------
    if step + 1 % 5 == 0:
        T.rename("Temperature", "Temperature")
        file.write(T, t)
        progress_counter += 1

# Save final result
file.close()
if rank == 0:
    print("\nComputation completed! Results saved to results/temperature.pvd")
```

Listing 1: FEniCS Code for Heat Conduction with Line-by-Line Explanation

6 Conclusion

Starting from the integral forms of the conservation laws, we derived the heat conduction equation and formulated its weak form suitable for finite element analysis. We then implemented a numerical solution using FEniCS, incorporating boundary conditions and a moving Gaussian heat source. The detailed derivations and line-by-line explanation of the code provide insights into setting up and solving PDEs using FEniCS.

7 References

- *Fundamentals of Fluid Mechanics* by BRUCE et al.
- *Fundamentals of Heat and Mass Transfer* by Bergman et al.
- *FEniCS Project Documentation*: `https://fenicsproject.org/`