Chapter 1

Introduction

1.1 Preface

The heat flow code is written to provide a dynamic simulation of heat flow in materials. This include non-equilibrium effects and insight from atomic scale models. It has been developed over a number of years at Exeter within the Hepplestone Reseach group and is maintained at our local git storage hub. Those wishing to use the code should be aware that their is a free academic license, but for industrial purposes a license is required.

1.2 Introduction

Fourier proposed the first form of heat conduction

$$\frac{dQ}{dt} = -\kappa A \frac{\Delta T}{\Delta x} \tag{1.1}$$

where Q is the amount of energy, T is temperature, κ is the thermal conductivity, A is the Area and x and t are length and time respectively.

This equation has formed the bedrock of heat transport processes in solids for centuries and when taken to its more advanced forms (See Chapter **??**) has provided mathematicians and scientists with lots of interesting aspects for research.

Chapter 2

Theory

2.1 Fourier

Fourier proposed the first form of heat conduction

$$\frac{dQ}{dt} = -\kappa A \frac{\Delta T}{\Delta x} \tag{2.1}$$

where Q is the amount of energy, T is temperature, κ is the thermal conductivity, A is the Area and x and t are length and time respectively.

This can be described as coming from the continuity equation.

$$\int J.da = \frac{dQ}{dt} \tag{2.2}$$

This is transformed to (via divergence theorem)

$$\int (\nabla.JdV) = \frac{dQ}{dt} \tag{2.3}$$

We rewrite the total energy in the system in terms of the heat capacity,

$$Q = \int \rho c(T - T_0)dV \tag{2.4}$$

$$\frac{dQ}{dT} = \int \rho c dV \tag{2.5}$$

$$\int (\nabla.JdV) = \int \rho \frac{dT}{dt} dV \tag{2.6}$$

But we define the conductivity as

$$\mathbf{J} = \kappa \nabla T \tag{2.7}$$

And this gives us the familiar 2nd order Fourier's law.

$$\rho c \frac{dT}{dt} = \nabla \kappa \nabla T \tag{2.8}$$

2.2 Cattaneo

The main issue with Fourier's law is that it requires 'spooky' action at a distance. This can be resolved by adding a correction. This has been discussed by Maxwell, Cattaneo and others. [**?**]. It takes the form

$$\rho c \frac{dT}{dt} + P + \tau_C \frac{d^2T}{dt^2} = \nabla \kappa \nabla T \tag{2.9}$$

Here T is the temperature, t is the time, τ_C is the catteno correction, P is the power density (power per unit volume), ρ is the mass density and c is the heat capacity per unit volume. κ is the thermal conductivity.

Chapter 3

Code implementation

3.1 Code form form of Fourier and cattaneo

Lets us restate the form of the equation tackled by heat flow.

$$\rho c \frac{dT}{dt} + Q + \tau_C \frac{d^2T}{dt^2} = \nabla \kappa \nabla T \tag{3.1}$$

Lets us restate the form of the equation tackled by heat flow.

$$\frac{dT}{dt} + \frac{Q}{\rho c} + \frac{1}{\rho c} \tau_C \frac{d^2T}{dt^2} = \frac{1}{\rho c} \nabla \kappa \nabla T \tag{3.2}$$

Our goal is represent the above in the form

$$H_{ij} T_i = S_j \tag{3.3}$$

where H_ij is the heatflow matrix (and sparse), T_j is the temperature vector containing the temperatures of all the different elements and

$$S_j = \frac{-T_{PP,j}}{(2\delta t)} - \frac{1}{\rho_i c_i} [B_j + Q_j + S_{cat}] \tag{3.4}$$

Here, S_j consists of the previous previous time step's temperatures $T_{PP,j}$, the boundary terms B_j and the cattaneo correction S_{cat}.

To this end, we expand the equation as

$$\frac{dT}{dt} + \frac{1}{\rho c} Q + \frac{1}{\rho c} \tau_C \frac{d^2T}{dt^2} = \frac{1}{\rho c} \nabla \kappa \nabla T \tag{3.5}$$

$$\frac{dT}{dt} + \frac{1}{\rho c} Q + \frac{1}{\rho c} \tau_C \frac{d^2T}{dt^2} = \hat{h}(\kappa, \mathbf{r}) T \tag{3.6}$$

We can first replace the terms with their vector equivalents, and expand dT/dt and d^2T/dt^2 with the EXPLICIT central difference expansions.

$$\frac{T_i - T_{i,PP}}{(2\delta t)} + \frac{Q_i}{\rho_i c_i} + \frac{\tau_C}{\rho_i c_i}\frac{T_i - 2T_{i,P} + T_{i,PP}}{(\delta t)^2} = \hat{h}(\kappa, \mathbf{r})T \tag{3.7}$$

We can gather the terms in Temperature together.

$$T_i\left(\frac{1}{(2\delta t)} + \frac{1}{c_i\rho_i}\frac{\tau}{(\delta t)^2}\right) - \frac{1}{2\delta t}T_{i,PP} + \frac{Q_i}{c_i\rho_i} + S_{cat} = \hat{h}(\kappa, \mathbf{r})T \tag{3.8}$$

where P_i and S_{cat} are defined below.
Now we expand the right hand side $\hat{h}(\kappa, \mathbf{r})T$.

$$\hat{h}(\kappa, \mathbf{r})T = \frac{1}{\rho c}\nabla\kappa\nabla T \tag{3.9}$$

3.1.1 H-matrix

The H-matrix has two terms

$$H_{ij} = H_{0,ij} + H_{p,ij} \tag{3.10}$$

where H_0 is the pure H-matrix and H_p is the periodic correction.
The right hand side of equation 3.1 or Eq. 3.9 is expressed as

$$\frac{1}{\rho c}\nabla\kappa\nabla T \approx \frac{1}{\rho c}\kappa\nabla^2 T \tag{3.11}$$

in heat flow (V0.2). In implicit finite difference, this is expressed as (uniform grid)

$$\begin{aligned}\hat{h}(\kappa, \mathbf{r})T = {} & \frac{1}{\rho_{ix,iy,iz}c_{ix,iy,iz}}\frac{1}{(\delta x)^2} \\ & [\left(\frac{\kappa_{ix+1,iy,iz}\kappa_{ix,iy,iz}}{\kappa_{ix+1,iy,iz} + \kappa_{ix,iy,iz}}\right)(T_{ix+1,iy,iz} - T_{ix,iy,iz}) \\ & - \left(\frac{\kappa_{ix-1,iy,iz}\kappa_{ix,iy,iz}}{\kappa_{ix-1,iy,iz} + \kappa_{ix,iy,iz}}\right)(T_{ix,iy,iz} - T_{ix-1,iy,iz}) \\ & + \left(\frac{\kappa_{ix,iy+1,iz}\kappa_{ix,iy,iz}}{\kappa_{ix,iy+1,iz} + \kappa_{ix,iy,iz}}\right)(T_{ix,iy+1,iz} - T_{ix,iy,iz}) \\ & - \left(\frac{\kappa_{ix,iy-1,iz}\kappa_{ix,iy,iz}}{\kappa_{ix,iy-1,iz} + \kappa_{ix,iy,iz}}\right)(T_{ix,iy,iz} - T_{ix,iy-1,iz}) \\ & + \left(\frac{\kappa_{ix,iy,iz+1}\kappa_{ix,iy,iz}}{\kappa_{ix,iy,iz+1} + \kappa_{ix,iy,iz}}\right)(T_{ix,iy,iz+1} - T_{ix,iy,iz}) \\ & - \left(\frac{\kappa_{ix,iy,iz-1}\kappa_{ix,iy,iz}}{\kappa_{ix,iy,iz-1} + \kappa_{ix,iy,iz}}\right)(T_{ix,iy,iz} - T_{ix,iy,iz-1})]\end{aligned} \tag{3.12}$$

However, this does not contain all the terms from the RHS of the expression, so and we express the H-matrix further using Eq.3.8 and Eq.3.12 as

$$\begin{aligned}
\hat{h}(\kappa, \mathbf{r})T = &\frac{1}{\rho_{ix,iy,iz}c_{ix,iy,iz}}\frac{1}{(\delta x)^2}\\
&[\left(\frac{\kappa_{ix+1,iy,iz}\kappa_{ix,iy,iz}}{\kappa_{ix+1,iy,iz}+\kappa_{ix,iy,iz}}\right)(T_{ix+1,iy,iz}-T_{ix,iy,iz})\\
&-\left(\frac{\kappa_{ix-1,iy,iz}\kappa_{ix,iy,iz}}{\kappa_{ix-1,iy,iz}+\kappa_{ix,iy,iz}}\right)(T_{ix,iy,iz}-T_{ix-1,iy,iz})\\
&+\left(\frac{\kappa_{ix,iy+1,iz}\kappa_{ix,iy,iz}}{\kappa_{ix,iy+1,iz}+\kappa_{ix,iy,iz}}\right)(T_{ix,iy+1,iz}-T_{ix,iy,iz})\\
&-\left(\frac{\kappa_{ix,iy-1,iz}\kappa_{ix,iy,iz}}{\kappa_{ix,iy-1,iz}+\kappa_{ix,iy,iz}}\right)(T_{ix,iy,iz}-T_{ix,iy-1,iz})\\
&+\left(\frac{\kappa_{ix,iy,iz+1}\kappa_{ix,iy,iz}}{\kappa_{ix,iy,iz+1}+\kappa_{ix,iy,iz}}\right)(T_{ix,iy,iz+1}-T_{ix,iy,iz})\\
&-\left(\frac{\kappa_{ix,iy,iz-1}\kappa_{ix,iy,iz}}{\kappa_{ix,iy,iz-1}+\kappa_{ix,iy,iz}}\right)(T_{ix,iy,iz}-T_{ix,iy,iz-1})]\\
&-T_{ix,iy,iz}\frac{1}{\rho_{ix,iy,iz}c_{ix,iy,iz}}\left(\frac{\rho_{ix,iy,iz}c_{ix,iy,iz}}{(2\delta t)}+\frac{\tau_C}{(\delta t)^2}\right)
\end{aligned} \tag{3.13}$$

which finally brings us to the H-matrix. We first use the notation $(ix, iy, iz) \equiv I$. We can do this using the logic loop which adds ix,iy,iz in the standard fashion.

```
$I = 0$
  do $ix$ =1, $nx$
      do $iy = 1, ny$
          do $iz = 1, nz$
              $I = I + 1$
          end do
        end do
  end do
```
(3.14)

This allows us to turn the 3D vector into a single vector quantity.

Hence, we can finally write our H-matrix. This we split here for ease of reading into two terms. First the central diagonal,

$$
\begin{aligned}
&H_{[(ix,iy,iz),(ix',iy',iz'),0]} = \\
&\qquad H_{II,0} = \frac{-1}{\rho_{ix,iy,iz} c_{ix,iy,iz}} \\
&\qquad\qquad \delta_{(ix,iy,iz),(ix',iy',iz')} \\
&\qquad\qquad [\left(\frac{\kappa_{ix+1,iy,iz}\kappa_{ix,iy,iz}}{\kappa_{ix+1,iy,iz} + \kappa_{ix,iy,iz}} + \frac{\kappa_{ix-1,iy,iz}\kappa_{ix,iy,iz}}{\kappa_{ix-1,iy,iz} + \kappa_{ix,iy,iz}} \right) \\
&\qquad\qquad + \left(\frac{\kappa_{ix,iy+1,iz}\kappa_{ix,iy,iz}}{\kappa_{ix,iy+1,iz} + \kappa_{ix,iy,iz}} + \frac{\kappa_{ix,iy-1,iz}\kappa_{ix,iy,iz}}{\kappa_{ix,iy-1,iz} + \kappa_{ix,iy,iz}} \right) \\
&\qquad\qquad + \left(\frac{\kappa_{ix,iy,iz+1}\kappa_{ix,iy,iz}}{\kappa_{ix,iy,iz+1} + \kappa_{ix,iy,iz}} + \frac{\kappa_{ix,iy,iz-1}\kappa_{ix,iy,iz}}{\kappa_{ix,iy,iz-1} + \kappa_{ix,iy,iz}} \right)] \\
&\qquad\qquad + \left(\frac{1}{(\delta t)} + \frac{\tau_C}{\rho_{ix,iy,iz} c_{ix,iy,iz} (\delta t)^2} \right)
\end{aligned}
\tag{3.15}
$$

and the off diagonal term, where $I \neq J$. Here we use one example, but it is cycled over all others.

$$
\begin{aligned}
&H_{[(ix+1,iy,iz),(ix,iy,iz),0]} = \\
&\qquad H_{IJ,0} = \frac{1}{\rho_{ix,iy,iz} c_{ix,iy,iz}} \left(\frac{\kappa_{ix+1,iy,iz}\kappa_{ix,iy,iz}}{\kappa_{ix+1,iy,iz} + \kappa_{ix,iy,iz}} \right)
\end{aligned}
\tag{3.16}
$$

and

$$
\begin{aligned}
&H_{[(ix-1,iy,iz),(ix,iy,iz),0]} = \\
&\qquad H_{IJ,0} = \frac{1}{\rho_{ix,iy,iz} c_{ix,iy,iz}} \left(\frac{\kappa_{ix-1,iy,iz}\kappa_{ix,iy,iz}}{\kappa_{ix-1,iy,iz} + \kappa_{ix,iy,iz}} \right)
\end{aligned}
\tag{3.17}
$$

with similar terms present for the y and z-axis, resulting in 6 off diagonal terms.

3.1.2 Periodicity and H-matrix

In a future release we will include the periodic correction $H_{p,ij}$ but for now it is not implemented nor discussed.

3.2 S-vector

The heat flow code has all the terms which are linear included in the so called "**S**" vector, shown in Eq. 3.4. It is broken into easy to understand components that can be coded in separately, the boundary vector **B**, the power vector **P**, the temperature evolution vector $\frac{1}{\delta t}\mathbf{T_p}$, and the cattaneo correction $\mathbf{S_{cat}}$. As in the

H-matrix, we can use contracted notation for (ix, iy, iz), where I is the index that loops over all cells. As this is a vector this is less complex to understand than the H-matrix so in general (with the exception of the boundary) we can just use I.

3.2.1 Temperature evolution vector

This is the easiest term to implement, arising from $\frac{dT}{dt}$. We approximate the time evolution with EXPLICIT dependence

$$\frac{dT}{dt} \approx \frac{T_i - T_{i,PP}}{2\delta t} \tag{3.18}$$

where $T_{i,PP}$ is the temperatures stored for each element I in the previous previous time step (Following central difference first derivative) and (δt) is our timestep size.

3.2.2 Power vector

The power vector represents input energy from an external source at a particular cell I in the simulation. We can have multiple cells if we wish.

$$P_I = \frac{1}{\rho_I C_I} Q_i \tag{3.19}$$

where Q_I is whether that element has a power in or not at that time step. We have implemented 5 different types of power in, and these are documented else where. By default, Q_i is zero.

3.2.3 Cattaneo correction

$$S_{cat} = \frac{\tau_C}{\rho_i c_i} \frac{T_{i,PP} - 2T_{i,P}}{(\delta t)^2} \tag{3.20}$$

where τ_C is the catteneo correction.

3.2.4 Boundary vector

This is the most involved term in the S vector. Our 3D box consists of $N_x \times N_y \times N_z$ cubes. We begin with the previous equation in the h-matrix

$$\begin{aligned}
\hat{h}(\kappa,\mathbf{r})T = &\frac{1}{\rho_{ix,iy,iz}c_{ix,iy,iz}}\frac{1}{(\delta x)^2}\\
&[\left(\frac{\kappa_{ix+1,iy,iz}+\kappa_{ix,iy,iz}}{2}\right)(T_{ix+1,iy,iz}-T_{ix,iy,iz})\\
&-\left(\frac{\kappa_{ix-1,iy,iz}+\kappa_{ix,iy,iz}}{2}\right)(T_{ix,iy,iz}-T_{ix-1,iy,iz})\\
&+\left(\frac{\kappa_{ix,iy+1,iz}+\kappa_{ix,iy,iz}}{2}\right)(T_{ix,iy+1,iz}-T_{ix,iy,iz})\\
&-\left(\frac{\kappa_{ix,iy-1,iz}+\kappa_{ix,iy,iz}}{2}\right)(T_{ix,iy,iz}-T_{ix,iy-1,iz})\\
&+\left(\frac{\kappa_{ix,iy,iz+1}+\kappa_{ix,iy,iz}}{2}\right)(T_{ix,iy,iz+1}-T_{ix,iy,iz})\\
&-\left(\frac{\kappa_{ix,iy,iz-1}+\kappa_{ix,iy,iz}}{2}\right)(T_{ix,iy,iz}-T_{ix,iy,iz-1})]\\
&-T_{ix,iy,iz}\frac{1}{\rho_{ix,iy,iz}c_{ix,iy,iz}}\left(\frac{\rho_{ix,iy,iz}c_{ix,iy,iz}}{(2\delta t)}+\frac{\tau_C}{(\delta t)^2}\right)
\end{aligned} \tag{3.21}$$

But take $ix = 1$ for a sample boundary. We could also consider, $iy = iz = 1$ but these are all the same.

Similarly, $ix = N_x$ results in a similar result but for the other boundaries in our cell.

So lets reexamine the previous equation.

$$\begin{aligned}
\hat{h}(\kappa,\mathbf{r})T = &\frac{1}{\rho_{1,iy,iz}c_{1,iy,iz}}\frac{1}{(\delta x)^2}\\
&[\left(\frac{\kappa_{2,iy,iz}+\kappa_{1,iy,iz}}{2}\right)(T_{2,iy,iz}-T_{1,iy,iz})\\
&-\left(\frac{\kappa_{BX,iy,iz}+\kappa_{1,iy,iz}}{2}\right)(T_{1,iy,iz}-T_{BX,iy,iz})\\
&+\left(\frac{\kappa_{1,iy+1,iz}+\kappa_{1,iy,iz}}{2}\right)(T_{1,iy+1,iz}-T_{1,iy,iz})\\
&-\left(\frac{\kappa_{1,iy-1,iz}+\kappa_{1,iy,iz}}{2}\right)(T_{1,iy,iz}-T_{ix,iy-1,iz})\\
&+\left(\frac{\kappa_{1,iy,iz+1}+\kappa_{1,iy,iz}}{2}\right)(T_{1,iy,iz+1}-T_{1,iy,iz})\\
&-\left(\frac{\kappa_{1,iy,iz-1}+\kappa_{1,iy,iz}}{2}\right)(T_{1,iy,iz}-T_{1,iy,iz-1})]\\
&-T_{1,iy,iz}\frac{1}{\rho_{1,iy,iz}c_{1,iy,iz}}\left(\frac{\rho_{ix,iy,iz}c_{ix,iy,iz}}{(2\delta t)}+\frac{\tau_C}{(\delta t)^2}\right)
\end{aligned} \tag{3.22}$$

We note all the terms in the y and z axis are identical to the previous case. We also observe that the final time evolution term is included in the diagonal of the h-matrix. Thus we can discard both terms as they are considered in the H-matrix.

$$\begin{aligned}\hat{h}(\kappa, \mathbf{r})T = & \frac{1}{\rho_{1,iy,iz}c_{1,iy,iz}}\frac{1}{(\delta x)^2} \\ & [\left(\frac{\kappa_{2,iy,iz}+\kappa_{1,iy,iz}}{2}\right)(T_{2,iy,iz}-T_{1,iy,iz}) \\ & -\left(\frac{\kappa_{BX,iy,iz}+\kappa_{1,iy,iz}}{2}\right)(T_{1,iy,iz}-T_{BX,iy,iz})]\end{aligned} \tag{3.23}$$

This makes the boundary term much more transparent, for $ix = 1$, the boundary element must be

$$B_{1,iy,iz} = \frac{-1}{\rho_{1,iy,iz}c_{1,iy,iz}}\frac{1}{(\delta x)^2}[\left(\frac{\kappa_{BX,iy,iz}+\kappa_{1,iy,iz}}{2}\right)(T_{BX,iy,iz})] \tag{3.24}$$

For the opposite side, we get (and it is symmetric)

$$B_{N_x,iy,iz} = \frac{-1}{\rho_{N_x,iy,iz}c_{N_x,iy,iz}}\frac{1}{(\delta x)^2}[\left(\frac{\kappa_{BX,iy,iz}+\kappa_{N_x,iy,iz}}{2}\right)(T_{BX,iy,iz})] \tag{3.25}$$

Thus we have for any boundary element an additive term. If iy and ix are 1 then we simply add both the Y-boundary and x-boundary terms together.

3.3 Handling at cell boundaries

The discretisation of the grid introduces a problem. At the interface between two materials there is a discrepancy in thermal conductivity values. Typically, the thermal conductivity of the originating cell is applied to calculate heat transfer; thus, heat moving from cell A to B will use A's conductivity, and vice versa. However, this cannot be fixed by using the destination thermal conductivity; this just leads to a reciprocal problem.

We can highlight this error by considering the steady state. Firstly we write the heat transfer rate q as described by Fourier's law

$$q = -\kappa\nabla T, \tag{3.26}$$

where κ is the thermal conductivity, and ∇T is the temperature gradient.

The conservation of energy together with the steady state condition requires that for two adjacent cells A and B the heat flux into A (q_{BA}) must be equal to the heat flux out of B (q_{AB}). This can be written

$$q_{AB} - q_{BA} = 0 \tag{3.27}$$

If we express these rates using Fourier's law, we have:

$$q_{AB} = -\kappa_{AB}\nabla T \tag{3.28}$$

$$q_{BA} = -\kappa_{BA}\nabla T \tag{3.29}$$

where κ_{AB} and κ_{BA} are the thermal conductivities for the heat transfer from A to B and B to A, respectively. It follows that

$$-\kappa_{AB}\nabla T + \kappa_{BA}\nabla T = 0 \tag{3.30}$$

and hence

$$\kappa_{AB} = \kappa_{BA} \tag{3.31}$$

κ is a material property and should only depend on what material is present (with the exception of temperature dependence). Therefore if this is true in the steady state it must always be true. We have thus proved that a coupled conductivity must be used to describe the interface conductance.

To determine an expression for the coupled conductivity we examine the thermal resistance, which is defined as

$$\Theta = \frac{\Delta x}{A\kappa}, \tag{3.32}$$

where Δx is the path length, A is the conducting area and κ is the thermal conductivity.

In series, the total thermal resistance is the sum

$$\Theta_{\text{total}} = \Theta_1 + \Theta_2 + \Theta_3 + \dots \tag{3.33}$$

when considering thermal resistances we can make an assumption that the discreet points of the mesh are space filling. That is to say that for 2 adjacent cells A and B, half the distance from one to the next will have the physical properties of point A and the other half will have the physical properties of point B. We

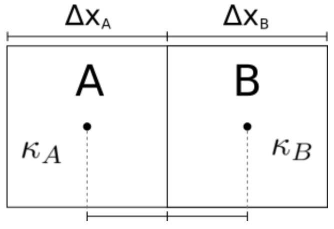

Figure 3.1: Two adjacent cells A and B each with it's own thermal conductivity and cell length.

approximate the heat as flowing from the center of cell A to the center of cell B as depicted in figure3.1. we can define the thermal resistance of each cell as

$$\Theta_A = \frac{\frac{1}{2}\Delta x_A}{a\ \kappa_A} \tag{3.34}$$

$$\Theta_B = \frac{\frac{1}{2}\Delta x_B}{a\ \kappa_B} \tag{3.35}$$

For two adjacent cells the area, a, must be the same for both. Now, in accordance with equation 3.33, we can simply add these terms to get the coupled thermal resistance.

$$\Theta_{AB} = \Theta_A + \Theta_B \tag{3.36}$$

We can now define the coupled conductivity as

$$\kappa_{AB} = \frac{\frac{1}{2}\Delta x_A + \frac{1}{2}\Delta x_B}{A\ \Theta_{AB}} \tag{3.37}$$

which substituting in and simplifying gives

$$\kappa_{AB} = \frac{(\Delta x_A + \Delta x_B)\ \kappa_A\ \kappa_B}{\Delta x_B\ \kappa_A + \Delta x_A\ \kappa_B} \tag{3.38}$$

and in the case of uniform grid this becomes

$$\kappa_{AB} = \frac{2\kappa_A \kappa_B}{\kappa_A + \kappa_B} \tag{3.39}$$

This is the harmonic average, which is commonly used to approximate thermal contact resistance.

3.4 Implementation choice

In the heat flow code, we have chosen to use the implicit finite difference method. The reason for this choice is best described as "best of all the choices available." The explicit method runs into significant issues with generating negative temperatures depending on the time step AND grid size. This is highly unsatisfactory problem. Similarly, the explicit method also doesn't simulate the steady state particularly well either, due to the need to iteratively solve. Finite element method would seem a natural choice, but the smoothness of the function often leads to highly unphysical results. This was observed regularly in Comsol (2021) simulations where a heat pulse would result in negative temperature zone immediately next to a region of extremely high temperature. This has not been observed to the authors knowledge experimentally and thus presented the grounds for rejection of a finite element approach.

3.5 Version history

3.5.1 0.1

The first iteration of heat flow built using the explicit method of finite difference. Ultimately, this version ran into numerous theoretical issues such as heat rates depending on the grid size and time steps as well as being unable to cope with non-uniform grids. An archive copy is maintained at Exeter, but is not available as there is so much wrong with it that their is little benefit to using it.

3.5.2 Central vs Forwards

Comparing the implementation of First of forward difference for temperature time derivative 3.2 and First order central difference, central difference shows significantly less deviation 3.3. Both systems are the same size, material properties with same number of time steps.

3.6 Matrix inversion methods

Frank will fill this in

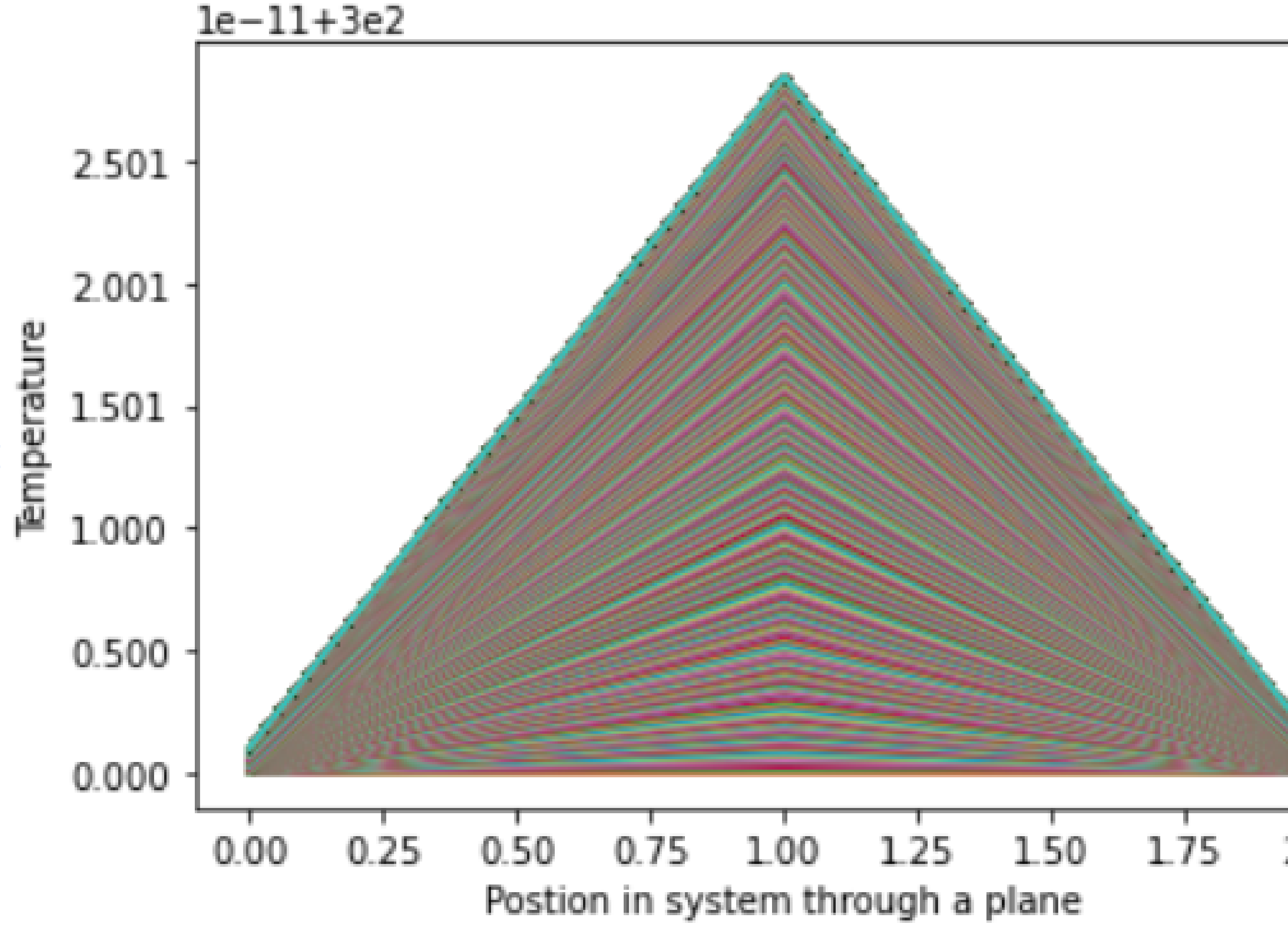

Figure 3.2: First order forward difference showing a greater and greater temperature gradient as the system evolves in an initial equilibrium state

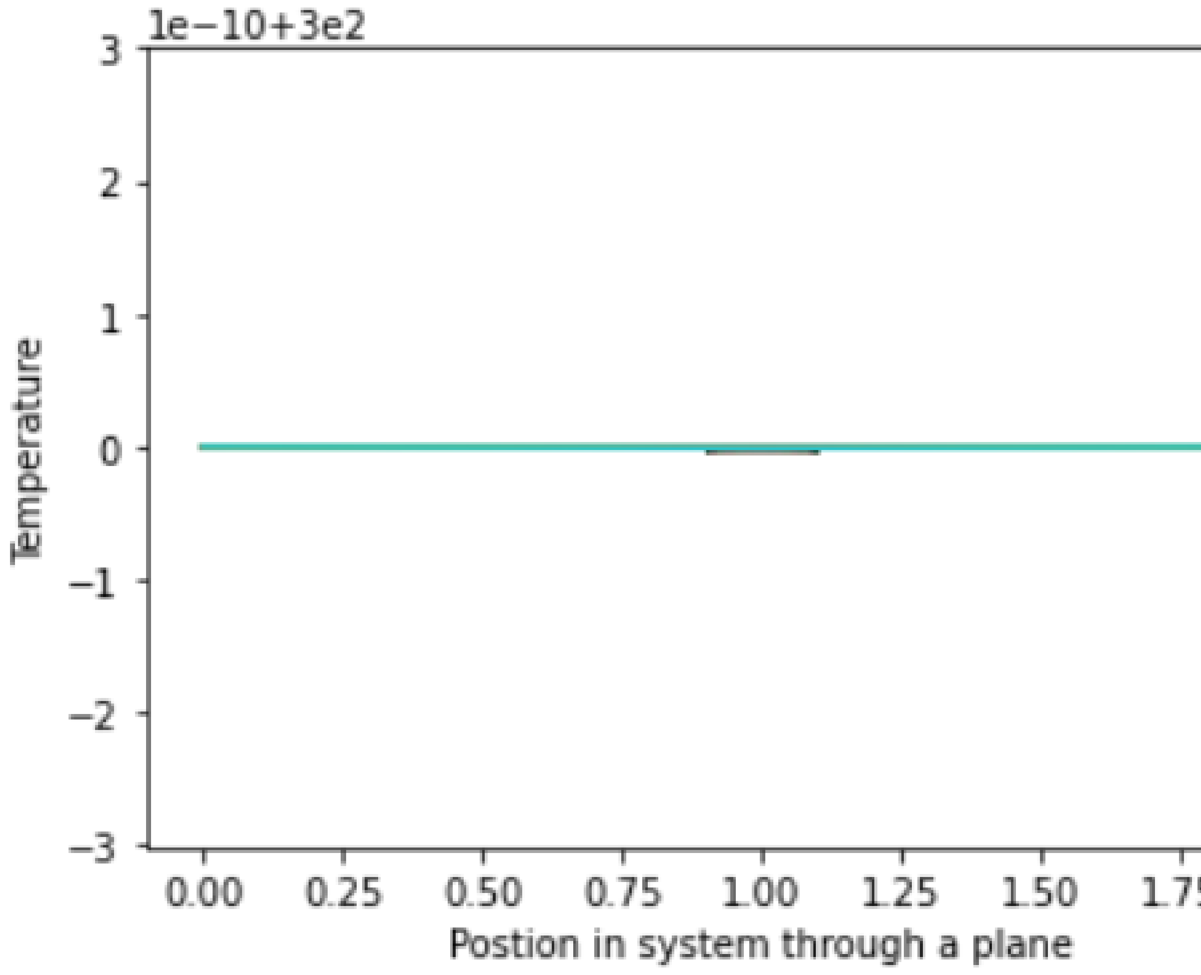

Figure 3.3: Central first order showing a negligible difference in each time evolution

Chapter 4

Input filesUtilities

4.1 Input files

Harry to fill in.

4.2 Parameters

4.3 Material Geometry

4.4 Optional: Material properties

Chapter 5

Utilities

5.1 Structure generator

Frank will fill this in

Chapter 6

Cases

6.1 Case One

Steady State, Uniform system, 1D, Three element system, No Heater.

6.1.1 Parameters

Table 6.1: Parameters

Parameter	**Value [a.u.]**
κ (Thermal conductivity)	1
ρ (Density)	1
C_v (Heat capacity)	1
Simulation cell size (δx, δy, δz)	1
Temperature of boundary	200
Thermal conductivity of boundary (x,y,z respectively)	1,0,0

6.1.2 Material Geometry

System geometry

Uniform system of parameters (6.1.1) in this case.

Heating

No heating in this case.

H Matrix

$$H = \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix} \tag{6.1}$$

S Matix

S Matrix is equal to the boundary matrix times -1 in this case.

$$B = \begin{bmatrix} 200 \\ 0 \\ 200 \end{bmatrix} \tag{6.2}$$

6.1.3 Result

$HT = -B$.

$$T = \begin{bmatrix} 200 \\ 200 \\ 200 \end{bmatrix} \tag{6.3}$$

6.2 Case Two

Steady State, Uniform system, 1D, Three element system, Heater in centre.

6.2.1 Parameters

Table 6.2: Parameters

Parameter	**Value [a.u.]**
κ (Thermal conductivity)	1
ρ (Density)	1
C_v (Heat capacity)	1
Simulation cell size (δx, δy, δz)	1
Temperature of boundary	200
Thermal conductivity of boundary (x,y,z respectively)	1,0,0
Heater Power density	10

6.2.2 Material Geometry

System geometry

Uniform system of parameters (6.2.1) in this case.

Heating

Heating geometry.

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \tag{6.4}$$

1 refers to the cell receiving a constant power. 0 refers to no power.

H Matrix

$$H = \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix} \tag{6.5}$$

S Matix

S Matrix is equal to the -1 times the boundary matrix(B) plus the heating matrix(Q) in this case.

$$B = \begin{bmatrix} 200 \\ 0 \\ 200 \end{bmatrix} \tag{6.6}$$

$$Q = \begin{bmatrix} 0 \\ 10 \\ 0 \end{bmatrix} \tag{6.7}$$

6.2.3 Result

$HT = -Q - B$.

$$T = \begin{bmatrix} 205 \\ 210 \\ 205 \end{bmatrix} \tag{6.8}$$

6.3 Case Three

Steady State, Heterogenous system, 1D, Three element system, No Heater.

6.3.1 Parameters

Table 6.3: Parameters

Parameter	**Value [a.u.]**
κ_1,κ_2 (Thermal conductivity's)	1,2
ρ (Density)	1
C_v (Heat capacity)	1
Simulation cell size (δx, δy, δz)	1
Temperature of boundary	200
Thermal conductivity of boundary (x,y,z respectively)	1,0,0

6.3.2 Material Geometry

System geometry

Heterogeneous system of parameters (6.4.1) in this case.

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \tag{6.9}$$

1 refers to κ_1, 2 refers to κ_2 of 6.4.1

Heating

Heating geometry.

$$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \tag{6.10}$$

0 refers to no cell power.

H Matrix

$$H = \begin{bmatrix} -\frac{7}{3} & \frac{4}{3} & 0 \\ \frac{4}{3} & -\frac{8}{3} & \frac{4}{3} \\ 0 & \frac{4}{3} & -\frac{7}{3} \end{bmatrix} \tag{6.11}$$

S Matrix

S Matrix is equal to the -1 times the boundary matrix(B) in this case.

$$B = \begin{bmatrix} 200 \\ 0 \\ 200 \end{bmatrix} \tag{6.12}$$

6.3.3 Result

$HT = -B$.

$$T = \begin{bmatrix} 200 \\ 200 \\ 200 \end{bmatrix} \tag{6.13}$$

6.4 Case Four

Steady State, Heterogenous system, 1D, Three element system, Heater in the centre.

Table 6.4: Parameters

Parameter	**Value [a.u.]**
κ_1,κ_2 (Thermal conductivity's)	1,2
ρ (Density)	1
C_v (Heat capacity)	1
Simulation cell size (δx, δy, δz)	1
Temperature of boundary	200
Thermal conductivity of boundary (x,y,z respectively)	1,0,0
Power density	10

6.4.1 Parameters

6.4.2 Material Geometry

System geometry

Heterogeneous system of parameters (6.4.1) in this case.

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \tag{6.14}$$

1 refers to κ_1, 2 refers to κ_2 of 6.4.1

Heating

Heating geometry.

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \tag{6.15}$$

0 refers to no cell power, 1 refers to constant cell power.

H Matrix

$$H = \begin{bmatrix} -\frac{7}{3} & \frac{4}{3} & 0 \\ \frac{4}{3} & -\frac{8}{3} & \frac{4}{3} \\ 0 & \frac{4}{3} & -\frac{7}{3} \end{bmatrix} \tag{6.16}$$

S Matrix

S Matrix is equal to the -1 times the boundary matrix(B) plus the heating matrix (Q) in this case.

$$B = \begin{bmatrix} 200 \\ 0 \\ 200 \end{bmatrix} \tag{6.17}$$

$$Q = \begin{bmatrix} 0 \\ 10 \\ 0 \end{bmatrix} \tag{6.18}$$

6.4.3 Result

$HT = -Q - B.$

$$T = \begin{bmatrix} 205 \\ 208.75 \\ 205 \end{bmatrix} \tag{6.19}$$

6.5 The Conjugate Gradient Method

The Conjugate Gradient (CG) method is a wonderful meeting of linear algebra and topology. It is an iterative method that can be used to exactly solve a set of n linear equations in at most n steps. In practice we usually cut off the iteration well before completion, based on the size of the residual.

The CG method has a special place in sparse systems due it only requiring matrix-vector multiplications. This means that you need not explicitly store or decompose the matrix; for large systems where the matrix is mostly zeros (sparse), the CG method can be significantly more memory-friendly (and efficient) than other direct solvers.

6.5.1 Exploring A Vector Topology

In practice we want to solve the set of equations which can be expressed in vector matrix notation as

$$Ax = b \tag{6.20}$$

where A is some symmetric, positive definite matrix (otherwise we must use the bi-conjugate gradient method), x is an unknown vector and b is a known vector.

In CG we we start form a slightly different place, but for a good reason. First we define an n-dimensional parabolic surface.

$$f(x) = \frac{1}{2}x^T Ax - b^T x \tag{6.21}$$

Here I am representing vectors as 1 by n matrices hence $b^T x = b \cdot x$; this makes the conversion to index notation more intuitive. Now, in order to minimise this function we take its derivative. In index notation this looks like

$$\frac{\partial f(x)}{\partial x_k} = \frac{\partial}{\partial x_k}\sum_j\sum_i\left(\frac{1}{2}x_i A_{ij} x_j - b_i x_i\right) \tag{6.22}$$

This is only going to result in a non-zero result when i or j are equale to k. This gives us:

$$\frac{\partial f(x)}{\partial x_k} = \frac{\partial}{\partial x_k}\sum_j\sum_i\frac{1}{2}A_{ij}(\delta_{ki}x_j + \delta_{kj}x_i) - \frac{\partial}{\partial x_k}\sum_i b_i x_i \tag{6.23}$$

Evaluating, all permutation we only get a term when either i or j is equal to k. The derivative can be further simplified using the properties of the Kronecker delta:

$$\frac{\partial f(x)}{\partial x_k} = \frac{1}{2}\sum_j A_{kj}x_j + \frac{1}{2}\sum_i A_{ik}x_i - b_k. \tag{6.24}$$

Now, given that the matrix A is symmetric, we have $A_{ik} = A_{ki}$. Using this property, the equation can be rewritten as:

$$\frac{\partial f(x)}{\partial x_k} = \sum_j A_{kj}x_j - b_k. \tag{6.25}$$

which in turn we can write in matrix form as,

$$\nabla f(x) = Ax - b \tag{6.26}$$

and we identify the minimum by setting the derivative to zero to get

$$Ax = b \tag{6.27}$$

and oh look where we are!

6.5.2 Minimising in Conjugate Directions

The Conjugate Gradient descent is a way to search for the function minimum by way of A-conjugate vectors. Each conjugate direction is generated independently without having to compute them all upfront. The wonderful aspect of these directions is that when you minimize the surface $f(x)$ along a conjugate direction, you get the exact minimum in that direction, and you never need to travel in that direction again. This is akin to minimizing a Cartesian space displacement by minimizing along orthogonal directions.

With the definition of the surface function given by equation 6.21, we can define a set of A conjugate vectors as

$$p^T A p_j = 0 \tag{6.28}$$

For an n by n matrix, A, there are n conjugate directions. Each conjugate direction acts as if orthogonal within a space transformed by A.

Since conjugate directions are interdependent, the choice of the first direction is arbitrary, and may as well take the steepest descent path. Now, given some direction, p, we can derive the minimum along that path with a starting position x. We can express this point on the surface as

$$f(x + \alpha p) \tag{6.29}$$

we can now differentiate with respect to α to find the minimisation along the conjugate direction p

$$\frac{df(x+\alpha p)}{d\alpha} = \frac{d}{d\alpha}\left(\frac{1}{2}(x+\alpha p)^T A(x+\alpha p) - b^T(x+\alpha p)\right) \tag{6.30}$$

Setting the derivative to zero and solving for α lets us find the step length for the given direction.

$$p^T A(x+\alpha p) - p^T b = 0 \qquad \Rightarrow \qquad \alpha = \frac{p^T r}{p^T A p}, \tag{6.31}$$

where $r = b - Ax$ is the residual.

The step size α ensures that $f(x)$ is minimized along the direction p. Moreover, once this minimization is done along p, the residual r will be orthogonal to p. This makes sense intuitively because if the surface is minimised in that direction no future movement in that direction is necessary, hence all future residuals will be, at least conjugate

We can demonstrate orthogonality by updating to $x_{k+1} = x_k + \alpha_k p_k$, the new residual r_{k+1} is given by:

$$r_{k+1} = b - A(x_k + \alpha_k p_k) \tag{6.32}$$

Now we can evaluate this to get

$$p_k^T r_{k+1} = p_k^T (b - A(x_k + \alpha_k p_k)) = p_k^T r_k - \alpha_k p_k^T A p_k \tag{6.33}$$

From equation 6.31b for α, we have $\alpha_k p_k^T A p_k = p_k^T r_k$, which in turn gives,

$$p_k^T r_{k+1} = 0 \tag{6.34}$$

So, the residual is orthogonal to the direction p after taking a step in the p direction to minimize $f(x)$. This ensures that no subsequent direction will undo the progress made in the p direction.

6.5.3 Generating Subsequent Conjugate Directions

Now that we have established the property of orthogonality between the conjugate direction p and the residual r, we can move forward to generating the subsequent conjugate directions.

Suppose we have the current conjugate direction p_k and the current residual r_k. The next conjugate direction p_{k+1} can be generated as follows,

$$p_{k+1} = r_k + \beta_k p_k \tag{6.35}$$

where β_k is a scalar that ensures p_{k+1} is conjugate to p_k with respect to matrix A. The value of β_k can be calculated using the formula:

$$\beta_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}} \tag{6.36}$$

This formula ensures that the new conjugate direction p_{k+1} is generated such that it remains conjugate to the previous direction p_k, while also being a descent direction for the function $f(x)$.

With each iteration, a new conjugate direction is generated, and the function $f(x)$ is minimized along this direction, bringing the solution vector x closer to the true solution of the system $Ax = b$. The iterative process continues until a stopping criterion, such as a maximum number of iterations or a tolerance on the size of the residual, is met.

6.5.4 Diagonal Preconditioning

one can (and we do) use preconditioning to speed up the convergence. This is done by transforming the problem into one with more favorable convergence criteria.

$$Ax = b \quad \Rightarrow \quad (M^{-1}A)x = M^{-1}b \tag{6.37}$$

the solution vector x remains unchanged and for a well chosen preconditioner matrix, M, the overall system is easier to solve.

For further reading see Matrix computations 3rd edition By Gene H. Golub, Charles F. Van Loan ISBN: 9780801854149, 0801854148

Bibliography