

COMPUTATIONAL METHODS

2D DIFFUSION EQUATION MODEL USING THE EXPLICIT
FINITE DIFFERENCE METHOD

Leon Knight
1909571

Table of Contents

1. Introduction to the problem under consideration	2
2. The constitutive equation of the problem	4
2.1 One dimensional Diffusion	4
2.2 Two-Dimensional Diffusion	5
3. Explicit finite difference form of the constitutive equation	6
3.1 Taylor series.....	6
3.2 Directional differences	7
3.3 Explicit Finite Difference scheme	8
3.4 Stability criterion.....	9
3.5 Grid spacing	9
3.6 Time Steps.....	10
4. The Initial and Boundary Conditions Implemented	11
4.1 Boundaries	11
4.2 Analytical Solution	11
5. Code Details and Explanations.....	12
6. Validation of the model	20
7. Sensitivity Analysis of the Code	21
7.1 Change in Time Step	21
7.1.1 Truncation errors	21
7.1.2 Round off errors.....	21
7.2 Change in Tolerance.....	23
7.3 Change in Grid Spacing	24
8. Main model's limitations	25
8.1 Round-off error	25
8.2 Truncation Error	25
8.3 Handling complex geometry	25
8.4 Stability concerns	25
8.5 Coding imperfections.....	25
9. Suggested improvements and future developments	26
9.1 ADI	26
9.2 FEM.....	26
9.3 BEM.....	26
10. Conclusion	27
11. References.....	28
Appendix (MATLAB Script)	30

1. Introduction to the problem under consideration

The objective of this report is to use the MATLAB software and applying the Explicit Finite Difference Method (EFDM), to model and calculate the temperature distribution through an aluminium plate over a set amount of time until thermal equilibrium has been reached. The software will allow me to monitor the temperature changes in the domain (centre) of the plate. I then aim to validate the code against an analytical method (Fourier's Series) of the solution to enable assurance that the software model is providing results that are both accurate and reliable.

The Aluminium plate being tested (figure 1) has the following material values and its associated letter value used in the code:

Length: $L=6\text{m}$
 Width: $W=6\text{m}$
 Conductivity: $K=143\text{ W/c.C}$
 Density: $D=2.8 \times 10^3\text{ Kg/m}^3$
 Specific Heat: $C=795\text{ J/Kg.C}$

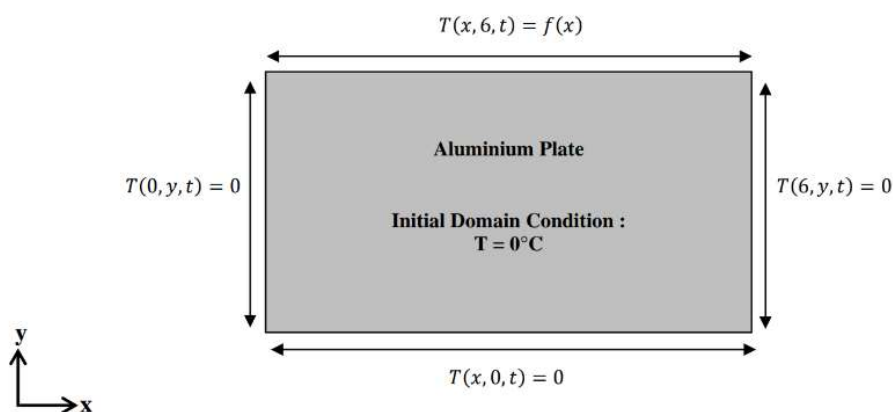


Fig. 1 (Marotin, 2023)

The temperature (T) of the plate, under ideal conditions, could use the two-dimensional heat diffusion equation. This would be calculated using a consistent density (ρ), uniform specific heat (C_p), and regular thermal conductivity (k) values. This would also require no internal heat or thermal sources coming from the plate itself.

The Explicit Finite Difference Method (EFDM) is a numerical method that can be used to help predict transient heat conduction using modelling techniques that provide visual outputs.

This report's objective will include:

- Reviewing the general basis of the heat equation.
- Exploring the numerical method of calculating the temperature distribution and then validating this code against an analytical method (Fourier's Series).
- Analysis of the code and the results it produces, discussing its limitations.
- Discuss the evaluation of this modelling method.
- Explore improvements

During this report, I will create an equation to measure the temperature distribution when heat is applied on the top edge of the material until thermal equilibrium has been reached (Appendix 1). A grid structure (mesh) will be created on the plate model, creating nodes which will help to define the different location temperatures. The temperature figure for each of the temperature nodes in the grid on the aluminium plate come in the form of $T(x,y,t)$, as shown on the material edge temperature values in figure 1. When the code is run, each temperature node at each passing time step will be referenced in this format. The coding will need to provide a user input to define the mesh size/number of nodes. The output of this code will produce distribution graphs and result figures. From these outputs I will then evaluate how the user inputs, Grid Spacing and Time Steps, can affect the outcomes of the code.

There are a few assumptions that have been made during this report:

- Thickness of the aluminium is negligible.
- Initial temperature of the material is 0°C
- Temperature of the edges will remain constant.
- A heat source of 50°C will be applied to the North boundary.
- Cooling device will be located on the other three edges.

2. The constitutive equation of the problem

The evolution of thermodynamics, mathematical analysis, and the study of diffusion processes are linked to the history of the heat diffusion equation, sometimes known as the "heat equation." The Mathematician Joseph Fourier is often credited as the person who helped birth the modern climate in science. Fourier's theories originated in the 18th century where he proclaimed that the Earth itself would be as cold as the sky were it not for the fact that the earth is able to retain heat in some compacity (Monks, 2022).

From this, Fourier developed the Fourier series which looked at a recurring function as an infinite sum of sine and cosine functions. This took form in the works he produced whilst investigating the heating of an infinite rectangular solid material. The idea is to break down complex functions into a sum of simpler functions (Fourier, J. 2009). This ability to break down these complex equations have since been used to aid in predicting the thermal diffusion of finite materials such as the aluminium plate I am investigating.

2.1 One dimensional Diffusion

The thermal diffusion equation for one dimensional forms of diffusion can be expressed in the following format:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

With the values equate to:

T = the Temperature

t = the Time

x = Spatial coordinate

$\frac{\partial T}{\partial t}$ = the rate of change of temperature in respect of time

$\frac{\partial^2 T}{\partial x^2}$ = the second spatial derivative of temperature in respect of spatial coordinates.

α = the Alpha, is the thermal diffusivity of the material. α can be defined as $\frac{k}{\rho c_p}$ where k is thermal conductivity ρ is the density, and c_p is the specific heat.

2.2 Two-Dimensional Diffusion

However, as the requirement of this report is to calculate over a two-dimensional surface, the equation will need to account for the second dimension. To achieve this, a second set of spatial derivative needs to be added to the calculation. This is expressed in the following format:

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

(Physlab.org, 2016)

The addition of the $\frac{\partial^2 T}{\partial y^2}$ to the equation means that we can work out the diffusion in both the x and now the y plains.

When the alpha is defined in the sum of its parts, the formula can then be written in the following:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = \frac{\rho c_p}{k} * \frac{\partial T}{\partial t}$$

3. Explicit finite difference form of the constitutive equation

3.1 Taylor series

The Taylor series is a technique used to illustrate a mathematical function when infinite mathematical terms are added together. Brook Taylor was a mathematician who introduced the idea in the early 1700s. In calculus and mathematical analysis, the Taylor series is a helpful tool when it comes to approximating functions. (*Maplesoft.com, 2023*). Calculus and mathematical analysis both use the Taylor series method.

The Taylor series in terms of function $f(x)$ when talking about a point, such as $x = x_0$, can be viewed as such:

$$f(x) = f(x_0) + f^{(1)}(x_0) \frac{(x - x_0)^1}{1!} + f^{(2)}(x_0) \frac{(x - x_0)^2}{2!} + f^{(3)}(x_0) \frac{(x - x_0)^3}{3!} + \dots$$

where:

$f(x_0)$ = Function at point $x = x_0$

$f^{(1)}(x_0)$ = 1st derivative at $x = x_0$

$f^{(2)}(x_0)$ = 2nd derivative at $x = x_0$

$f^{(3)}(x_0)$ = 3rd derivative at $x = x_0$ (*testbook.com, 2023*)

The Taylor series can go on infinitely, depending on the number of terms required. However, this formula can be written in a compact form, such as:

$$f(x) = \sum_{n=0}^{\infty} f^{(n)}(x_0) * \frac{(x - x_0)^n}{n!}$$

With n representing the number of derivatives in the equation and $n!$ Referring to the factorial of n .

$f^{(n)}(x_0)$ refers to the derivative of f evaluated to the point of x_0

Having this compact formula will greatly reduce the time it takes to input the data request into a system. Rather than inputting every derivative manually, such as the original equation, you can input a singular line in a code.

3.2 Directional differences

The derivative of a function at a point can be calculated by using different numerical methods. These are the Forward, backward and central differences.

The forward difference is determined by calculating the slope of the secant line that passes between the coordinates (x_0) and $(x_0 + h)$ where h is a step size (m).

The backward difference acts similarly to the forward, just with negative steps. For example, the secant line would pass through the coordinates (x_0) and $(x_0 - h)$ where h is a step size (m).

The central difference uses both previous differences with the aim to gain a more accurate approximate of the derivative. So, this would be based off the function at $(x_0 + h)$ and $(x_0 - h)$.

To work out the forward difference, you will need to complete the following steps:

$$f(x) = f(x_0) + f^{(1)}(x_0) \frac{(x - x_0)^1}{1!} + f^{(2)}(x_0) \frac{(x - x_0)^2}{2!} + f^{(3)}(x_0) \frac{(x - x_0)^3}{3!} + \dots$$

↓ Swap all $x - x_0$ values for h

$$f(x_0 + h) = f(x_0) + f^{(1)}(x_0) \frac{h^1}{1!} + f^{(2)}(x_0) \frac{h^2}{2!} + f^{(3)}(x_0) \frac{h^3}{3!} + \dots$$

↓ right side is divided by h value.

$$f(x_0 + h) - f(x_0) = \frac{f^{(1)}(x_0) * h^1 + f^{(2)}(x_0) \frac{h^2}{2!} + f^{(3)}(x_0) \frac{h^3}{3!}}{h}$$

↓ take h value over.

$$\frac{f(x_0 + h) - f(x_0)}{h} = f^{(1)}(x_0) * h^1 + f^{(2)}(x_0) \frac{h^2}{2!} + f^{(3)}(x_0) \frac{h^3}{3!}$$

↓ Make $f^{(1)}(x_0)$ the subject.

$$f^{(1)}(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h}$$

The backwards difference has the same process; however, the result will have a slight difference in that the positive h value will now be negative. You will end up with the following:

$$f^{(1)}(x_0) \approx \frac{f(x_0 - h) - f(x_0)}{h}$$

The central difference will instead have both the positive and negative values in its final equation. The h value will also now be $2h$ to account for both directions.

$$f^{(1)}(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h}$$

University of Manitoba, 2023)

3.3 Explicit Finite Difference scheme

Explicit finite difference schemes can be used for approximating the solution of partial differential equations (PDEs). This method is especially useful when dealing with time-dependent issues. Problems where the temperature distribution changes over an area during a length of time, such as heat conduction, are often addressed using it.

The formula given in this assignment, and as mentioned previously, was the following equation:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = \frac{1}{\alpha} * \frac{\partial T}{\partial t}$$

(Marotin, 2023)

To work out the second derivative approximations, the central difference method can be used. Where the x axis has been swapped out for the East and West directions and the y axis have been swapped out for the North and South. This results in the following equations:

x axis at point P

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_E^t - 2.T_P^t + T_W^t}{h^2}$$

y axis at point P

$$\frac{\partial^2 T}{\partial y^2} = \frac{T_N^t - 2.T_P^t + T_S^t}{h^2}$$

The 1st derivative would therefore become:

$$\frac{\partial T}{\partial t} = \frac{T_P^{t+\Delta t} - T_P^t}{\Delta t}$$

With these equations solved, they can be substituted into the thermal diffusion equation to give:

$$T_P^{t+\Delta t} = Fo (T_E^t + T_W^t + T_N^t + T_S^t - \left(4 - \frac{1}{Fo}\right) \cdot T_P^t)$$

(A.Marotin, 2023)

Δt refers to the time step.

The **Fo** in the equation relates to the Fourier number. This can be represented in the equation format:

$$Fo = \frac{\alpha \cdot \Delta t}{h^2}$$

According to Thermopedia.com (2011), In heat transfer problems, the Fourier number (Fo) is a dimensionless number that is commonly used in the context of transient (time-dependent) heat conduction. It describes how a material's rate of thermal energy storage and rate of heat conduction are compared.

The Fourier number can help to indicate the rates of heat transfer a material is experiencing. For instance: a low number would indicate that there is a slow rate of heat transfer. Equally, a high number would indicate a speedier heat transfer. (Mechcontent.com,2020)

3.4 Stability criterion

Stability criteria are conditions that must be met in numerical methods, this helps to ensure that the results remain within the confines of the boundary limitations; thus, preventing any numerical instability.

An issue that can lead to inaccuracies and unreliable outputs is the potential for large roundoff errors. This can happen due to the numerical solution rounding the results at each time step when trying to simplify the overall equation. This can have a substantial impact on the numerical simulations' stability requirement, particularly when the errors are able to accumulate over a vast number of time steps. The greater the number of steps, the more opportunity there is for an accumulation of error to occur. If the amplification factor fluctuates due to round-off errors, it can push the solution outside stable ranges. This increase in round-offs can make the disparity between the numerical system results and that of the true solution of the model far greater.

A method that can be used to help mitigate this is to run the formula with smaller step sizes. Although previously mentioned that more steps mean more opportunity for error, if the step sizes themselves are reduced, the impact of each step would be far smaller than that of larger time steps. This could help to produce a more reasonably accurate output from the software, provided higher levels of precision are used (Olver, P.J., 2008). The adverse effect would be that the formula would take longer and require more processing power for the programme to run.

For this reason, the Fourier number is used due to it being a dimensionless quantity. For this to work, it is important that the Fourier number is calculated to ensure the stability of the equation. If the value of this number is outside of this quantity range the code will fail to produce accurate results.

Considering the grid spacing and the time step, the suitable value for the Fourier number to ensure stability for this code is as below:

$$0 < F_0 \leq \frac{1}{4} \quad \text{in 2D}$$

3.5 Grid spacing

The spatial domain of the aluminium plate will be split evenly across the surface area. The grid spacing should also be made small enough to get an accurate reading, but large enough that the amount of grid points is reasonable. If there are too few locations, the results that are returned will not provide enough data to be considered accurate enough to properly portray the thermal diffusion. If there are too many locations, although an accurate representation of the thermal diffusion will be produced, the amount of time and processing power it would take the MATLAB equipment to run it would be unreasonable.

As the length of the material is 6m x 6m. The number of nodes travelling in the x and y directions should therefore be spread out in locations (in meters) divisible by 6. For example, grid spaces of 1m would allow for 6 nodes to travel in the x and y dimensions, provided a total of 36 nodes.

3.6 Time Steps

The temporal domain for the code has a similar situation as the grid spacing. Too small of a time step and you will get a highly accurate result, however, it could take unreasonably long to allow the software to run. Too large of a time step, and time jumps could miss important stages of the heat transfer in the model. This would provide an inaccurate set of results. Clearly, there is a balance between the quality of the results and how reasonable of a wait time there should be to receive the data. Software and Hardware limitations should also be taken into consideration as the equipment may not have the processing power to safely run the code.

For example, if the software dictated that a set of parameters would take up to a day for equilibrium to be reached in the model: too low of a time step could mean that the software could take several hours to run a single test, whereas a reasonable number could have informed you of this in the space of a few minutes.

4. The Initial and Boundary Conditions Implemented

4.1 Boundaries

The initial boundaries set for this analysis activity include the following:

- Dimensions are set for a 6m x 6m area.
- Nodes to be analysed are to fit within equal spaces within the 6m x 6m space.
- All edges are to have an initial temperature set at 0°C

Constant boundary conditions that come after

- The temperatures on the West, East and South ends are to be set at 0°C.
- The North edge is to be heated from an initial temperature of 0°C at a rate as per $f(x)$ until it reaches a maximum temperature of 50°C.
- A Dirichlet boundary condition shown in Figure 2 is set. This describes the temperature variables' values at a certain border point. In this example, the north boundary $F(x)$ has the below rules put into place on different sections of the edge. If a node is more than 2/3rds from zero, the $F(x)$ value becomes $-25x + 150$. Less than 2/3rds it becomes $12.5x$.

$$f(x) = \begin{cases} 12.5x & \text{if } 0 \leq x \leq \left(\frac{2}{3} \times 6\right) \\ -25x + 150 & \text{if } \left(\frac{2}{3} \times 6\right) < x \leq 6 \end{cases}$$

4.2 Analytical Solution

As mentioned in chapter 2, the Fourier series is an analytical method to solving heat transfer problems. This approach works especially well for scenarios with precise boundary conditions or periodic difficulties. So, when dealing with problems that cover the temperature distribution that fluctuates over time and area, such as conduction in a 1D rod or 2D plate, the Fourier series approach can be used.

This approach yields an accurate solution to heat transfer issues, particularly when the geometry and boundary conditions are simple. However, it is difficult to apply to more complicated geometries or non-homogeneous materials. This method also assumes that ideal conditions are met during the process, this may not be the case when put into a real-world scenario.

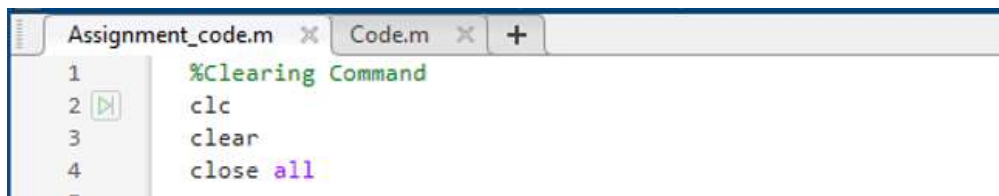
To complete this method, you will need to use the following steps:

- Set the boundary and initial conditions: examples include dimensions and temperature values.
- Variable separation: Assume the solution consists of spatial and temporal functions.
- Resolve the Original Differential Equations (ODEs) in terms of area and time: Solve every ODE that results from the process of separation.
- Fourier series expansion: The answer can be represented using the Fourier sine series (or cosine, depending on the boundary conditions).
- Apply initial conditions: Find the Fourier coefficients using the original temperature distribution.
- Construct the solution: The general solution is formed by combining the spatial and temporal solutions.

5. Code Details and Explanations

Below is a series of screenshots of the code created to run the program that will simulate the thermal diffusion of the aluminium plate.

At the start of the script, lines 2-4 are used to clear the command window of all existing programmes running. This ensures that the program is focused only on the current code and the operation window only shows the relevant information to the user.



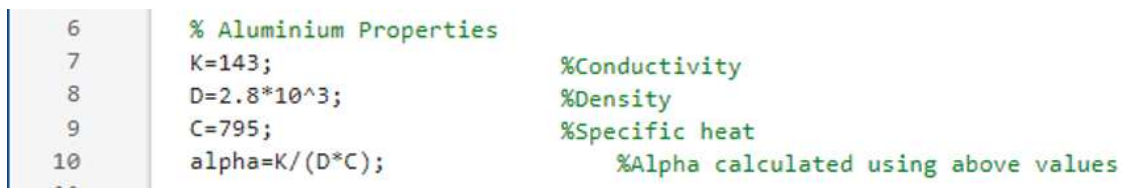
```

1 %Clearing Command
2 clc
3 clear
4 close all

```

Fig. 3 Code, lines 1-4

Lines 7- 10 are a list of the properties of the aluminium plate. This information acts as a reference hub for the rest of the code.



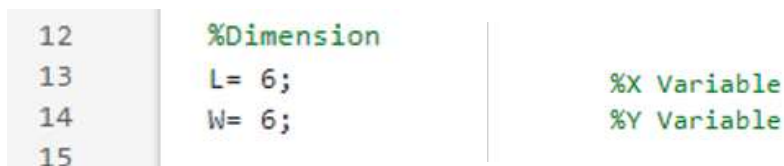
```

6 % Aluminium Properties
7 K=143; %Conductivity
8 D=2.8*10^3; %Density
9 C=795; %Specific heat
10 alpha=K/(D*C); %Alpha calculated using above values

```

Fig. 4 Code, lines 5-1

Like the previously mentioned lines, lines 13 and 14 are used as reference data that give the dimensions of the plate.



```

12 %Dimension
13 L= 6; %X Variable
14 W= 6; %Y Variable
15

```

Fig. 5 Code, lines 12-14

Line 17 will bring up a prompt to the user, asking them to input the grid spacing value.

The while code during Lines 18 to 21 will ensure that the input by the user is a figure divisible by, but not including, six. If the number is invalid, lines 19 & 20 will produce the error message and request a new figure is entered. This will continue to loop each time an invalid figure is inputted until a valid value has been selected.

```

16      %Grid Spacing
17      h=input("Enter dimesnion for grid spacing, 1,2 or 3m:");
18      while mod(L,h)~=0 || h<=0 || h>=W
19          fprintf('Grid value is Invalid. ');
20          h=input("Please Enter valid grid spacing value: ");
21      end

```

Fig. 6 Code, lines 16-21

Line 24 will now prompt the user to input a time step value.

Line 25 tells the code what the appropriate Fourier value range should be.

While code between lines 26 and 28 will continue to loop each time an invalid figure is inputted until a valid value has been selected.

```

23      %Time Steps
24      dt=input("Enter the value of time steps in seconds: "); %User Input for time step value
25      Fo=(alpha*dt)/(h^2); % Fourier Number
26      while Fo<=0 || Fo>0.25
27          fprintf("Time step is invalid.");
28          dt=input("Please Enter valid time step value: ");
29      end

```

Fig. 7 Code, lines 23-29

The lines in figure 8 define the rows and column values to fit within the matrix.

```

31      %Defining rows and columns
32      NR=W/h+1;
33      NC=L/h+1;

```

Fig. 8 Code, lines 31-33

Line 37 is used to zero out the initial grid spacing.

```

36      %Creating initial T1
37      T1=zeros(NR,NC);

```

Fig. 9 Code, lines 36-37

This section of the code sets the initial boundary conditions of the matrix. Line 41 sets the locations of the nodes depending on what the input the user original gave.

Lines 42 to 46 use an 'if' function to implement the values that were shown in figure 2. Lines 41& 42 indicate if a node is Less than 2/3rds from 'd' the $F(x)$ value becomes $12.5x$. Lines 44 and 45 indicate that if 'd' is more than 2/3rds from zero, the $F(x)$ value becomes $-25x + 150$.

```

39      % Amend Bounrary Conditions Of Initial Matrix
40      for c=1:NC
41          d=(c-1)*h;
42          if d<=2/3*L
43              T1(1,c)=12.5*d;
44          else
45              T1(1,c)= -25*d+150;
46          end
47      end
48

```

Fig. 10 Code, lines 39-47

This section of the code is used to create a second matrix from the first. Line 51 gives the tolerance value for the equilibrium, currently at a value of 0.001. Line 52 indicates the maximum value difference that can occur before an error occurs. Line 53 sets the initial count to zero.

```

49      %Create T2 Matrix from T1 Matrix with temp calculations
50      T2=T1;
51      Tol=0.001;
52      Dif=Tol+1;
53      Count=0;
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2
```

This 'while loop' between lines 56 and 62 is there to tell the program to keep running until thermal equilibrium has been reached. Line 60 is particularly important as it sets the Taylor series formula for the program to follow. Line 63 will help the program to determine once equilibrium has been reached.

```

55 %While loop set up to repeat calculations until the difference between T2 and
56 while Dif>Tol
57     T1=T2;
58     for G=2:NR-1
59         for H=2:NC-1
60             T2(G,H)=Fo*(T1(G-1,H)+T1(G,H-1)+T1(G,H+1)+T1(G+1,H)-(4-(1/Fo))*T1(
61                 end
62         end
63     Dif=max(max(abs(T2-T1)));

```



```

55 T1 (Err) is less than 0.01 and therefore close to equilibrium.
56 %Condition to initiate/continue loop
57
58 %Dimension to ignore perimeter nodes
59 %Dimension to ignore perimeter nodes
60 (G,H)); %Taylor series formula used to calculate T1 values at the next time step
61
62
63 %Formula to determine if thermal equilibrium has been reached

```

Fig. 12 Code, lines 55-63

This section of code is used to tell the program to create and display a graph with the results. May of the commands below will indicate ways to personalise the graph. For instance; Line 68 indicates the colour scheme of the graph. Lines 69 to 71 are commands detailing the description and location of a colour bar that will go beside the graph. Lines 67, 72 and 73 are all commands involving labelling of the graph.

```

65 figure(1)
66 contourf(flipud(T2))
67 title ('Temperature Distribution of plate');
68 colormap ('parula')
69 c=colorbar;
70 c.Location = 'southoutside';
71 c.Label.String = 'Temperature';
72 xlabel('X distance in meters');
73 ylabel('Y distance in meters');
74 pause(0.2)
75 Count=Count+1;
76 end

```



```

65
66 %Command to create graph
67 %Title for Fig1
68 %Color system used for figure 1
69 %Attaches color bar to figure 1, representing temp
70
71
72
73
74 %Pauses after each step to update graph to visually show the change along plate
75 % Update the amount of cycles completed
76

```

Fig. 13 Code, lines 65-76

The next set of commands are used to calculate the duration it would take the aluminium plate to reach thermal equilibrium based upon the inputs made by the user. Depending on the scaling of time, the if commands will give the results in the most appropriate units of time. The options start from seconds and scale up to days. Line 97 then commands the code to display the result.

```

78 % Thermal equilibrium time
79 eq=Count*dt; %Calculate the time taken to reach equilibrium by multiplying
80
81 if eq<60
82     fprintf ('It took %g seconds to reach thermal equilibrium\n', eq)
83
84 elseif eq>=60 && eq<3600
85     eqM=eq/60;
86     fprintf ('It took %g minutes to reach thermal equilibrium\n', eqM)
87
88 elseif eq>=3600 && eq<86400
89     eqh=(eq/60)/60;
90     fprintf ('It took %g hours to reach thermal equilibrium\n', eqh)
91
92 elseif eq>=86400
93     eqD=((eq/60)/60)/24;
94     fprintf ('It took %g days to reach thermal equilibrium\n', eqD)
95 end
96
97 fprintf ('(%g seconds)\n', eq)
98
78
79 the count by the time step
80
81 %Will display in seconds
82 % Display amount time taken to reach equilibrium in seconds
83
84 %Will display in minutes
85
86 % Display amount time taken to reach equilibrium in minutes
87
88 %Will display in hours
89
90 % Display amount time taken to reach equilibrium in hours
91
92 %Will display in days
93
94 % Display amount time taken to reach equilibrium in days
95
96
97 %Time in seconds for analysis
98

```

Fig. 14 Code, lines 78-97

Lines 101 to 110 gives the parameters for the analytical solution following the Fourier Series method. This is what will calculate the heat distribution along each of the nodes across each row and column. G and H both represent positions in the grid along the x and y axis, respectively.

Line 102 zero's out the starting values within the dimensions (NR,NC).

Line 107 has the function 'calculate_analytical_temperature(x, y, num_terms)' which is used for each point on the grid, which calculates the temperature at that point using the analytical solution.

Line 110 tells the programme to store the results in the T_analytic matrix at position (G,H). An additional command 'rot90' has been included to rotate the results table by 90degrees in order to get the desired orientation.

```

99 % Analytical Solution Calculation
100 % Calculate the analytical solution
101 num_terms = 100; % Number of terms in the series to include
102 T_analytic = zeros(NR, NC);
103 for G = 1:NR
104     for H = 1:NC
105         x = (G - 1) * h;
106         y = (H - 1) * h;
107         T_analytic(G, H) = calculate_analytical_temperature(x, y, num_terms);
108     end
109 end
110 T_analytic = rot90(T_analytic); % Rotate matrix 90 degrees counterclockwise for proper orientation
111

```

Fig. 15 Code, lines 99-110

These next two lines command the software to display the results of the analytical calculations completed in Lines 101 and 110.

```

112 % Display the analytical solution in the command window
113 disp('Analytical Temperature Distribution at Thermal Equilibrium:');
114 disp(T_analytic);
115

```

Fig. 16 Code, lines 112-114

Line 117 gets the program to calculate the difference between the results of the first program and the analytical calculations. Line 118 and 119 command the program to display these results.

```

116 % Calculate the difference between analytical and numerical solutions
117 difference = T1 - T_analytic; % Compute the difference between solutions
118 disp('Difference Between Numerical and Analytical solutions:');
119 disp(difference);
120

```

Fig. 17 Code, lines 116-119

The set of commands in figure 18 are duplicated of those during lines 65 to 76 with the exception of line 123 which indicates the graph be for the analytic results.

```

121 % Visualization of the analytical solution (Figure 2)
122 figure(2)
123 contourf(flip(T_analytic)) % Rotate by 90 degrees (clockwise) and then flip
124 colormap("parula")
125 colorbar % Keep the colorbar on the right side
126 c=colorbar;
127 c.Location = 'southoutside';
128 c.Label.String = 'Temperature';
129 clim([0 50]); % Set colorbar limits (Replace 'min_value' and '
130 xlabel('X distance in meters');
131 ylabel('Y distance in meters'); % Adjust the label accordingly
132 title('Analytical Temperature Distribution');

121
122
123 vertically
124
125
126 %Attaches color bar to figure 1, representing temp
127
128
129 max_value' with your desired range)
130
131
132

```

Fig. 18 Code, lines 121-132

The set of commands in figure 18 are duplicated of those during lines 65 to 76 with the exception of line 136 which indicates the graph be for the difference results.

```

134 % Visualization of the difference (Figure 3)
135 figure(3)
136 contourf(flip(difference)) % Flip matrix vertically for cor
137 colormap("parula") % Use 'jet' colormap for cl
138 colorbar % Add colorbar to denote value r
139 c=colorbar;
140 c.Location = 'southoutside';
141 c.Label.String = 'Temperature';
142 clim([-0.005 2]); % Set colorbar limits (Replace '
143 xlabel('X distance in meters');
144 ylabel('Y distance in meters');
145 title('Difference Between Numerical and Analytical Solutions');

134
135
136 rect orientation
137 arity
138 anges
139 %Attaches color bar to figure 1, representing temp
140
141
142 min_value' and 'max_value' with your desired range)
143
144
145

```

Fig. 19 Code, lines 134-145

These final segments take the analytical results and create a temperature distribution chart.

```
147 % Function to calculate the analytical temperature distribution
148 function T_analytical = calculate_analytical_temperature(x, y, num_terms)
149     T_analytical = 0;
150     for n = 1:num_terms
151         term = (sin(2*n*pi/3) / (n^2 * sinh(n*pi))) * sin(n*pi*x/6) * sinh(n*pi*y/6);
152         T_analytical = T_analytical + term;
153     end
154     T_analytical = (450/pi^2) * T_analytical;
155 end
```

Fig. 20 Code, lines 147-155

6. Validation of the model

The model and code can be validated by comparing the outcomes of the code's distribution and the analytical solution in various situations. Below are three sets of graphs resulting from the code. Fig.21 shows the numerical results, Fig.22 shows the analytical, and Fig.23 shows the comparable differences between the previous two charts.

Fig.23 clearly shows that there are some discrepancies between the two methods, particularly in one spot. However, the graph does not indicate that the difference between the two is significant, with the highest difference being roughly 1.8°C.

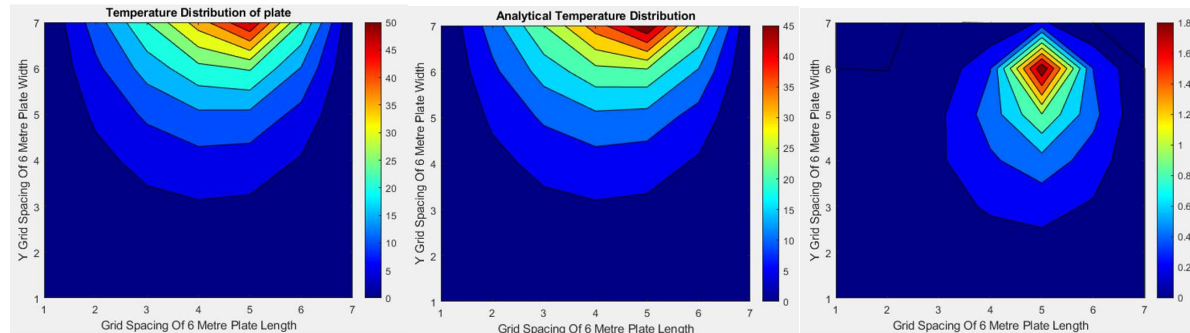


Fig.21 Numerical Result

Fig.22 Analytical Result

Fig.23 Comparison of Results

This test was conducted with a grid space of 1 metre and a time step of 800 with the tolerance 0.001. To better understand the true validation of the code, a second and third test should be implemented but with some changes to a couple of the factors. Altering the values for grid spacing and time step values would help to produce a set of results that come from a spectrum of inputs to see if an overall pattern forms between them.

If the overall results from the three sets of tests come with similar outcomes, then it can be assumed that the model, although with some margin of error, can be validated as a suitable substitute to completing the code through manual methods.

Using this code will therefore provide the user with a means to test different parameters for the heat distribution of the aluminium plate in a much more time efficient manner.

7. Sensitivity Analysis of the Code

To analyse the sensitivity of the code used in the test, it was run on numerous occasions to test three variables: Time Step, Tolerance and Grid Spacing. To review how each variable would affect the results, I inputted the results into a table and a graph.

7.1 Change in Time Step

In this set of results, I chose to alter just the time step value. This value dictates how many steps in seconds to skip for each reading. The higher the number of time steps, the greater the number of samples the programme needs to take. This also means that it also takes longer for the coding to reach equilibrium.

Looking at the graph (Fig. 24), the general impact of using a different value for the size of time steps is quite vast.

The results with the fewer time steps illustrate that the code will reach equilibrium much earlier than time steps in smaller increments. The differences in where thermal equilibrium is reached begins to plateau around the 600 Time step value. Time steps higher than this continues to take longer to reach equilibrium, but at a much more gradual pace. If tests were to be completed with greater numbers of time steps, the result differences would likely be negligible.

The values below 600 Time steps could be reaching equilibrium earlier due to the time steps being so large, equilibrium is possibly being met sometime between the final two large steps, and the coding assumes the final step is where equilibrium is met.

These results could be affected by factors such as truncation errors and potential round-off errors.

7.1.1 Truncation errors

Truncation errors occur when an infinite process is approximated by a finite one. In numerical methods, this often happens when we approximate derivatives or integrals, or when we stop summing terms in a series after a certain point. These errors are introduced because the exact mathematical operation is replaced by an approximation that can be computed by a machine.

Methods of reducing truncation errors can involve decreasing the grid spacing or time step, using higher-order approximations or increasing the number of terms used in the equation (Syrakos, A. 2015).

7.1.2 Round off errors

Because computer arithmetic stores numbers with a finite number of digits, round-off errors arise from this limitation in precision. With greater amounts of time steps, round-off errors are more likely to accumulate when more operations are carried out, especially in lengthy simulations. This can lead to greater accuracy disparities by the end. Fewer operations lessen the possibility of round-off errors building up. However, if the time step is too big, truncation errors may occur.

To summaries, Fewer time steps can lead to the risk of instability and have an increased likelihood of truncation errors. However, more time steps may give a more accurate solution, but it is more likely to incur round-off errors.



Fig 24. Graph of time step variable against thermal Equilibrium

Grid Spacing h (fixed)	Time Step	Tolerance (Fixed)	Thermal Equilibrium
0.75	1000	0.0001	238000
0.75	800	0.0001	232000
0.75	600	0.0001	224400
0.75	400	0.0001	213200
0.75	200	0.0001	194000
0.75	100	0.0001	174200

Fig 25. Table of results for time step variable

7.2 Change in Tolerance

In this set of results, I chose to alter just the tolerance value. This value dictates how precise the results need to be for each reading. As you would think, the graph below shows that the more constraint the tolerances are, the longer it took to reach equilibrium. If the tolerance is made larger, the programme can run much faster. However, this comes at the cost of having results that are much less accurate. However, smaller tolerance values do not necessarily equate to a more accurate depiction of thermal equilibrium.



Fig 26. Graph of Tolerance variable against thermal Equilibrium

Grid Spacing h (fixed)	Time Step (Fixed)	Tolerance	Thermal Equilibrium
1	500	0.1	29000
1	500	0.001	89500
1	500	0.0001	155500
1	500	0.00001	222000
1	500	0.000001	288000
1	500	0.0000001	354500

Fig 27. Table of results for tolerance variable

7.3 Change in Grid Spacing

The grid spacing value dictates how much space is between each node for each reading in meters. The more nodes in the area, the greater the number of opportunities to pick up temperature changes in the material. The results reveal that generally, the more space you put in between the nodes, the longer the equilibrium takes to reach. This could be because each node is having to pick up more area of changes, meaning that they pick up more temperature changes per node.

An outlier to the results is when the grid spacing is at 2 meters. The reasoning for this is unclear. There could be an issue with the model that is producing this result. The graph does show that the gradient of the line does begin to taper off as the grid spaces get smaller than 1.5 meters. This could suggest that after getting smaller than 0.75, it may not make a significant difference to the thermal equilibrium result.

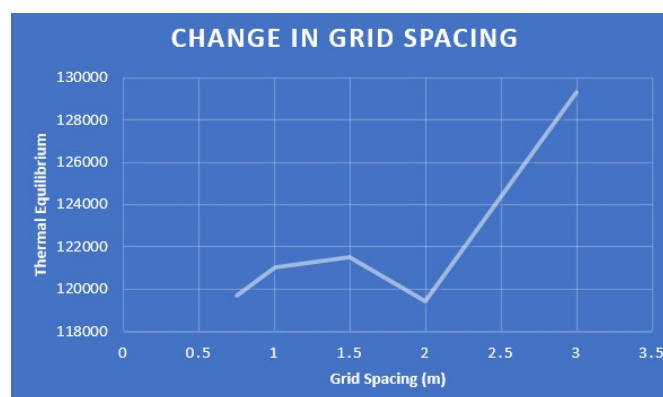


Fig 28. Graph of Grod Spacing variable against thermal Equilibrium

Grid Spacing h	Time Step (Fixed)	Tolerance (Fixed)	Thermal Equilibrium
3	150	0.0001	129300
2	150	0.0001	119400
1.5	150	0.0001	121500
1	150	0.0001	121050
0.75	150	0.0001	119700
0.5	150	0.0001	119100

Fig 29. Table of results for Grid Spacing variable

8. Main model's limitations

There are inherent limitations when developing numeric models. Such limitations include:

8.1 Round-off error

As discussed earlier at point 7.1.1, The limited precision with which computers represent and handle numbers is the source of round-off errors. These mistakes frequently compound over time, especially in extensive or protracted simulations.

Accuracy can be an issue when precision is limited. The small differences in temperature values may not be captured accurately. This error is also prevalent in longer simulations where errors are allowed to stack over time.

8.2 Truncation Error

As discussed at point 7.1.2, Truncation error happens when a discrete mathematical process is used to approximate a continuous one, such as approximating derivatives using finite differences or truncating an infinite series (like a Fourier series).

A grid that uses larger step sizes can lead to significant truncation errors, meaning temperature distributions may be less accurate.

8.3 Handling complex geometry

Many numerical methods, including finite difference methods, work well with simple, structured grids. Complex geometries necessitate more advanced techniques such as finite element methods (FEM) or boundary element methods (BEM).

Complex geometries would need the mesh to have more time and care spent on them to prevent any inaccuracies. This can also be quite taxing on the software to run. Boundary conditions will also become more complex as they will have more factors that will need to be considered before performing the test. (Lewis, R.W, 2004)

8.4 Stability concerns

Large time steps or incorrect grid spacing can produce unphysical results, like temperature oscillations or divergence, which can cause stability problems.

8.5 Coding imperfections

It can be challenging to model thermal equilibrium when complex boundary conditions, transient effects, or materials with different properties are present. Any errors in the coding can lead to the result providing unrealistic temperature distributions or incorrect heat flow solutions may be displayed if the numerical model does not adequately capture the physics of thermal equilibrium.

9. Suggested improvements and future developments

9.1 ADI

Alternating Direction Implicit (ADI) is an alternative method to solving heat transfer equations. ADI switches between implicit steps in several spatial directions, it preserves stability while enabling larger time steps. It achieves this by dividing the larger problem into multiple one-dimensional implicit problems that are computationally simpler to solve, the ADI method can successfully solve the equation with more ease this way (Namiki, T. 2002). ADI is unconditionally stable for linear problems, allowing larger time steps without sacrificing stability.

9.2 FEM

FDM is simple and efficient but has limitations in handling complex geometries. Alternatives to this could be the Finite Element Method (FEM) or the Boundary Element Method (BEM).

FEM can easily model complex geometries, such as curved surfaces and irregular domains, because it makes use of unstructured meshes. The mesh can also be refined in areas with steep gradients, increasing accuracy, thanks to adaptive mesh refinement (AMR).

9.3 BEM

BEM simplifies the problem by solving only on the boundary rather than the entire volume, reducing the computational domain for certain classes of problems. BEM reduces the problem dimensionality by one, for example, a 3D problem becomes 2D. This helps reduce the computational cost for problems by using simpler boundary conditions.

10. Conclusion

Through this assignment, I have effectively reviewed the fundamentals of the heat equation, investigated numerical techniques for determining temperature distribution, verified the results using Fourier's Series, an analytical solution, and critically examined the output of the numerical code.

The outcomes compared between the Numerical and Analytical methods confirmed that although there were some discrepancies between the two methods, the differences were minor with a difference maximum of 1.8°C in only a small area. However, to maintain accuracy in the future, certain limitations like mesh size and time step constraints need to be carefully managed to prevent inaccuracies such as truncation and round-off errors from occurring.

For straightforward heat transfer problems, EFDM works well due to its simple to use and implement. However, alternative numerical methods such as the Finite Element Method (FEM) offer greater accuracy and flexibility when handling complex geometries or varying material properties.

The combination of analytical and numerical methods creates an effective toolkit for resolving heat transfer issues. However, it's crucial to keep in mind that the software operates under the assumption of ideal circumstances and in terms of theoretical models. The model is useful for examining and testing theories, but it should be used with caution in practical applications.

11. References

- Davies, T.W. (2023) *Fourier number*, *THERMOPEDIA*. Available at: <https://www.thermopedia.com/content/782/> (Accessed: 20 January 2024).
- Fourier, J. (2009) 'The Analytical Theory of Heat', in *CHAPTER III - Propagation of Heat in an infinite rectangular solid*. Cambridge: Cambridge University Press, pp. 131–212. Available at: <https://www.cambridge.org/core/books/abs/analytical-theory-of-heat/propagation-of-heat-in-an-infinite-rectangular-solid/90F0233A00B4802CA7DA3042A7B46647> (Accessed: 05 September 2024).
- Lewis, R.W., Nithiarasu, P. and Seetharamu, K.N. (2004) *Fundamentals of the finite element method for heat and fluid flow*. PP. 300. Available at: https://cdn.preterhuman.net/texts/science_and_technology/physics/Fluid_Mechanics/Fundamentals%20Of%20The%20Finite%20Element%20Method%20For%20Heat%20And%20Fluid%20Flow%20-%20Lewis,%20Nithiarasu,Seetharamu.pdf (Accessed: 05 September 2024).
- Maplesoft.com (2023) *Taylor series*, *Maplesoft*. Available at: <https://www.maplesoft.com/ns/math/taylor-series.aspx> (Accessed: 21 January 2024).
- Marotin, A. (2023) *Computational Methods Assessment Specification*.
- Monks, K. (2022). *Fourier's Heat Equation and the Birth of Fourier Series* *Fourier's Heat Equation and the Birth of Fourier Series*. [online] Available at: https://digitalcommons.ursinus.edu/cgi/viewcontent.cgi?article=1008&context=triumphs_differentials. (Accessed: 5 September 2024)
- Namiki, T. (2002) *A new FDTD algorithm based on alternating-direction implicit method* | *IEEE Journals & Magazine | IEEE Xplore*, *A new FDTD algorithm based on alternating-direction implicit method*. Available at: <https://ieeexplore.ieee.org/document/795075/> (Accessed: 05 September 2024).
- Olver, P.J. (2008) *Finite difference methods for partial differential equations*, *Numerical Analysis Lecture Notes*. Pp. 191. Available at: https://www-users.cse.umn.edu/~olver/num_/lnp.pdf (Accessed: 05 September 2024).
- Pratik and Dikov, V. (2023) *Fourier number: Definition, formula, significance (heat transfer)*, *Mech Content*. Available at: <https://mechcontent.com/fourier-number/> (Accessed: 22 January 2024).
- Physlab (2023) *The Thermal Diffusion Equation*, *The Thermal Diffusion Equation - PhysLab*. Available at: <https://physlab.org/wp-content/uploads/2016/04/Doc.pdf> (Accessed: 20 January 2024).
- Syrakos, A. and Goulas, A. (2015) *ESTIMATE OF THE TRUNCATION ERROR OF FINITE VOLUME DISCRETISATION OF THE NAVIER-STOKES EQUATIONS ON COLOCATED GRIDS*. Department of Mechanical Engineering, Aristotle University of Thessaloniki, Greece, PP. 1. Available at: <http://arxiv.org/pdf/1508.02733> (Accessed: 05 September 2024).
- Testbook.com (2022) *Taylor series: Formula, theorem with proof method & examples*, *Testbook*. Available at: <https://testbook.com/maths/taylor-series> (Accessed: 19 January 2024).
- University of Manitoba (2022) *Numerical Differentiation - University of Manitoba*, *University of Manitoba*. Available at:

<https://home.cc.umanitoba.ca/~farhadi/Math2120/Numerical%20Differentiation.pdf>
(Accessed: 20 January 2024).

Appendix (MATLAB Script)

```
%Clearing Command
clc
clear
close all

% Aluminium Properties
K=143;           %Conductivity
D=2.8*10^3;      %Density
C=795;          %Specific heat
alpha=K/(D*C);   %Alpha calculated using above values

%Dimension
L= 6;           %X Variable
W= 6;           %Y Variable

%Grid Spacing
h=input("Enter dimesnion for grid spacing, 1,2 or 3m:"); %User Input for grid
spacing value
while mod(L,h)~=0 || h<=0 || h>=W
    fprintf('Grid value is Invalid. ');
    h=input("Please Enter valid grid spacing value: ");
end

%Time Steps
dt=input("Enter the value of time steps in seconds: "); %User Input for time step
value
Fo=(alpha*dt)/(h^2); % Fourier Number
while Fo<=0 || Fo>0.25
    fprintf("Time step is invalid.");
    dt=input("Please Enter valid time step value: ");
end

%Defining rows and columns
NR=W/h+1;
NC=L/h+1;

%Creating initial T1
T1=zeros(NR,NC);

% Amend Boundary Conditions Of Initial Matrix
for c=1:NC
    d=(c-1)*h;
    if d<=2/3*L
        T1(1,c)=12.5*d;
    else
        T1(1,c)= -25*d+150;
    end
end

%Create T2 Matrix from T1 Matrix with temp calculations
T2=T1;
Tol=0.001; % Tolerance for equilibrium
Dif=Tol+1; % Initial condition for max value in difference matrix (Dif) for
initial condition to be false.
Count=0;
```

```

%While loop set up to repeat calculations until the difference between T2 and T1
(Dif) is less than 0.01 and therefore close to equilibrium.
while Dif>Tol          %Condition to initiate/continue loop
    T1=T2;
    for G=2:NR-1        %Dimension to ignore perimeter nodes
        for H=2:NC-1    %Dimension to ignore perimeter nodes
            T2(G,H)=Fo*(T1(G-1,H)+T1(G,H-1)+T1(G,H+1)+T1(G+1,H)-(4-
(1/Fo))*T1(G,H));      %Taylor series formula used to calculate T1 values at the
next time step
        end
    end
    Dif=max(max(abs(T2-T1)));
%Formula to determine if thermal equilibrium has been reached

figure(1)
contourf(flipud(T2))
%Command to create graph
title('Temperature Distribution of plate'); %Title for Fig1
colormap('parula')          %Colour system used for figure 1
c=colorbar;                 %Attaches colour bar to figure 1, representing temp
c.Location = 'southoutside';
c.Label.String = 'Temperature';
xlabel('X distance in meters');
ylabel('Y distance in meters');
pause(0.2) %Pauses after each step to update graph to visually show the change
along plate
Count=Count+1;              %Update the amount of cycles completed
end

% Thermal equilibrium time
eq=Count*dt; %Calculate the time taken to reach equilibrium by multiplying the
count by the time step

if eq<60                    %Will display in seconds
    fprintf('It took %g seconds to reach thermal equilibrium\n', eq)
% Display amount time taken to reach equilibrium in seconds

elseif eq>=60 && eq<3600    %Will display in minutes
    eqM=eq/60;
    fprintf('It took %g minutes to reach thermal equilibrium\n', eqM)
% Display amount time taken to reach equilibrium in minutes

elseif eq>=3600 && eq<86400 %Will display in hours
    eqh=(eq/60)/60;
    fprintf('It took %g hours to reach thermal equilibrium\n', eqh)
% Display amount time taken to reach equilibrium in hours

elseif eq>=86400            %Will display in days
    eqD=((eq/60)/60)/24;
    fprintf('It took %g days to reach thermal equilibrium\n', eqD)
% Display amount time taken to reach equilibrium in days
end

fprintf('(%g seconds)\n', eq) %Time in seconds for analysis

% Analytical Solution Calculation
% Calculate the analytical solution
num_terms = 100;           % Number of terms in the series to include
T_analytic = zeros(NR, NC);

```

```

for G = 1:NR
    for H = 1:NC
        x = (G - 1) * h;
        y = (H - 1) * h;
        T_analytic(G, H) = calculate_analytical_temperature(x, y, num_terms);
    end
end
T_analytic = rot90(T_analytic); % Rotate matrix 90 degrees counterclockwise for
proper orientation

% Display the analytical solution in the command window
disp('Analytical Temperature Distribution at Thermal Equilibrium:');
disp(T_analytic);

% Calculate the difference between analytical and numerical solutions
difference = T1 - T_analytic; % Compute the difference between solutions
disp('Difference Between Numerical and Analytical solutions:');
disp(difference);

% Visualization of the analytical solution (Figure 2)
figure(2)
contourf(flip(T_analytic)) % Rotate by 90 degrees (clockwise) and then flip
vertically
colormap("parula")
colorbar % Keep the colorbar on the right side
c=colorbar; %Attaches color bar to figure 1, representing temp
c.Location = 'southoutside';
c.Label.String = 'Temperature';
clim([0 50]); % Set colorbar limits (Replace 'min_value' and
'max_value' with your desired range)
xlabel('X distance in meters');
ylabel('Y distance in meters'); % Adjust the label accordingly
title('Analytical Temperature Distribution');

% Visualization of the difference (Figure 3)
figure(3)
contourf(flip(difference)) % Flip matrix vertically for correct orientation
colormap("parula") % Use 'jet' colormap for clarity
colorbar % Add colorbar to denote value ranges
c=colorbar; %Attaches color bar to figure 1, representing temp
c.Location = 'southoutside';
c.Label.String = 'Temperature';
clim([-0.005 2]); % Set colorbar limits (Replace 'min_value' and
'max_value' with your desired range)
xlabel('X distance in meters');
ylabel('Y distance in meters');
title('Difference Between Numerical and Analytical Solutions');

% Function to calculate the analytical temperature distribution
function T_analytical = calculate_analytical_temperature(x, y, num_terms)
    T_analytical = 0;
    for n = 1:num_terms
        term = (sin(2*n*pi/3) / (n^2 * sinh(n*pi))) * sin(n*pi*x/6) *
sinh(n*pi*y/6);
        T_analytical = T_analytical + term;
    end
    T_analytical = (450/pi^2) * T_analytical;
end

```