STOCHASTIC SOLUTION TO SOLVE THERMAL CONDUCTION AND BOUNDARY LAYER PROBLEMS

by

Ashish S. Pujari

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Mechanical Engineering

Charlotte

2019

Approved by:

_____
Dr. Russell G. Keanini

_____
Dr. Peter T. Tkacik

_____
Dr. Yogendra P. Kakad

ABSTRACT

ASHISH S. PUJARI. Stochastic Solution to solve thermal conduction and boundary layer problems. (Under the direction of DR. RUSSELL G. KEANINI)

The following thesis describes the concept, application and result obtained on applying stochastic calculus to solve thermal conduction and boundary layer problems. Ito's lemma is the framework in obtaining the desired results. Different cases provided different results and every one of them was compared with the cases in analytical solution pertaining to the respective governing parameters.

The solutions obtained through the stochastic process appears to reasonably conform with their analytical counterparts. There are small regions of anomaly, near the boundary, where the difference in temperature is highly noticeable. A few methods to tackle these regions of anomaly are mentioned in the later chapters.

Considering that the results are reasonably accurate and observing its potential strength to solve thermal boundary layers with precision and at less computation expense, stochastic solutions would be greatly useful in the direction of mesh-free analysis of thermal transport problems.

DEDICATION

I'd like begin this thesis by paying my utmost respect to my parents and family for primarily shaping me in the position I am now. I am much grateful for everything they did and I'll be forever indebted.

ACKNOWLEDGEMENTS

After dedicating my thesis to my parents and family, I'd like to extend a humble thanks to my advisor Dr. Russell G. Keanini, without whom this whole thesis wouldn't have been possible.

I'd also like say thank you to all the faculty and friends involved, directly and indirectly, with this thesis for providing me with paths leading to the completion of the thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

The field of stochastic calculus is one which, of late, is being applied to departments in and out of engineering. The aspect of sampling data when a fixed equation could not predict the behaviour and making sense out of it is a major factor behind the optimisation of stochastic calculus. Methods involving stochastic calculus has been implemented in the realm of atmospheric boundary layers [1-6], ground water [7-13] and marine transport [14] problems. The objective in these areas of work is to find a degree of conformity in the continuum models, which showcases scalar transport, and stochastic differential equations that govern the random motion of these particles called random walkers.

The term random walker is well-known. Imagine a particle, in one dimension, having N number of steps and an equal chance of going left or right on every step. At the end of the steps, the particle may or may not be displaced from its original position. Such is the nature of random walk.

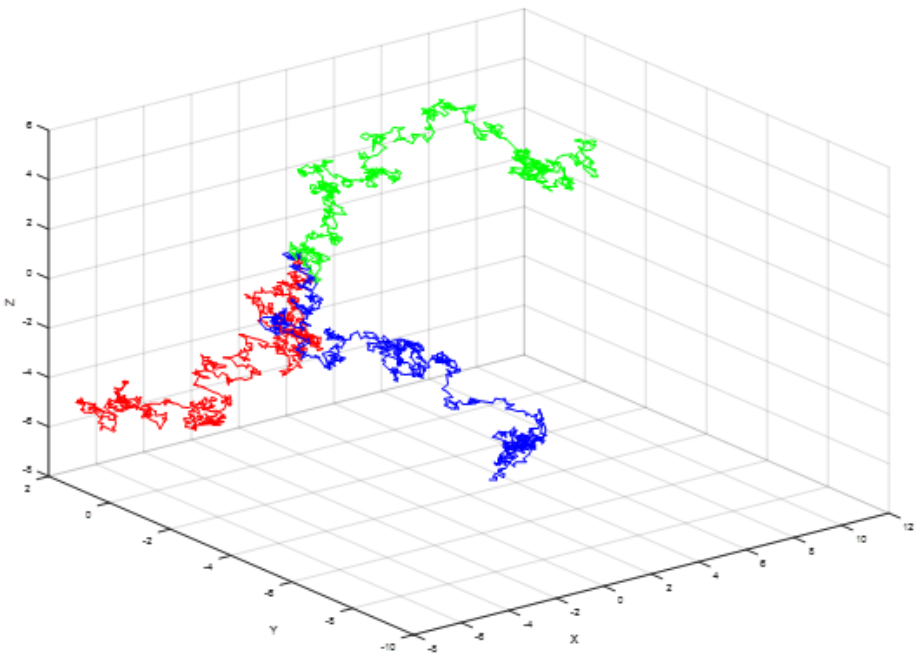

Figure 1.1: 3-D Physical Representation of Particles Undergoing Random Walk.

Figure 1.1 depicts 3 particles undergoing random walk in a 3-D space. This 1000 step random walk is the closest physical depiction of a stochastic solution incorporated in this research.

In the current paper, a two-dimensional motion of these random particles will be governed by a stochastic differential equation named after Kiyosi Ito, also known as Ito's Lemma:

$$d\vec{x}(t) = \vec{b}dt + \hat{a}d\vec{w} \quad (1)$$

Here, the particle's trajectory as a scalar quantity is represented by $\vec{x}(t)$, $\vec{b}$ represents the drift velocity, dt as the time increment, $\hat{a}$ as the quantity related to the diffusion tensor a, such that $\hat{a}a^T = a$, and $d\vec{w}$ as the multidimensional Wiener process.

Ito's lemma has been named after the Japanese mathematician Kiyosi Ito. Apart from engineering, its application extends in the mathematical finance realm[24], while

most notably being known for its role in derivation of the Black-Scholes equation[25].

For the domain covered by this paper, we assume a scalar term ϕ, such that $\phi = \phi(x, \tilde{t})$ and is twice differentiable in x while being only once differentiable in t. Ito's lemma is applied here to form a general solution by taking expectations over the complete continuum field[20]. The following is the result of the same:

$$\begin{aligned}\phi(x,\tilde{t}) = E_{x,\tilde{t}} \textstyle\int_{\tilde{t}}^{T} [local + advective + diffusive\ transport\ terms]dt + \\ E_{x,\tilde{t}}[Local\ Martingales] + E_{x,\tilde{t}}[Boundary\ Values\ Sampled] + \\ E_{x,\tilde{t}}[Initial\ Values\ Sampled]\end{aligned} \quad (2)$$

Here, $E_{x,\tilde{t}}$ term denotes the expectation observed at the solution point $x, \tilde{t}$ at different conditions. For the conditions mentioned in this paper, it is observed that the transport term sums to zero when the time goes from $\tilde{t}$ to T. Martingales are a pattern of random variables, which at any time, would yield an equal conditional expectation value for the successive observation as was obtained during the preceding observation. In the boundary of this research, expected values of local martingale terms tend to zero. Simply put, martingales would have an impact over the solution in presence of any source or sink for the heat transfer. The absence of any such inlets or outlets thus reduces martingales to zero.

This leave us with two terms, expectation observed for the boundary values sampled and the expectation observed for the initial sampled values. The sum of the former added with the average of the latter, hence, provides us the local scalar value at the solution point $x, \tilde{t}$.

A detailed methodology is explained in the latter section of this thesis. But based on the preliminary observation, it is evident that stochastic method does not involve the idea of meshes and contests to provide us with a more computationally inexpensive method to solve thermal boundary layer problems. But a much closer attention must be paid to the aspect that appropriate boundary condition is a certain way of

obtaining highly conformable solutions.

CHAPTER 2: OBJECTIVE

There has been tremendous work in the realm of boundary layer problems. Most notable of them is computational fluid dynamics (CFD). CFD employs data structures and numerical analysis to solve multiple problems related with fluid flow. But on a larger scale, this process tends to operate at the mercy of a powerful processor, bereft of which the computation becomes a time-consuming process.

The objective of this research is to develop a system that yields a stochastic solution for a thermal boundary layer profile. This solution will be developed with the usage of C++ (refer Appendix A). The results obtained will be compared using the results obtained through analytical solutions. An error and percentage error analysis is implemented to provide a better insight about the conformity of the stochastic solution to the analytical solution.

The governing theory and the specifications regarding test conditions will be provided in the latter sections. The ultimate objective, hence, is to provide a useful, mesh-free solution to solve thermal transport problems.

CHAPTER 3: RANDOM WALKER THEORY

From equation (2), we came to a conclusion that the expectations of 2 terms tend to zero, leaving us with the expectations of boundary and initial values sampled. The absence of a heat source or sink helps us take away the expectations originating from any martingales in the problem. The steady state nature of this problem helps us eliminate the expectations of local, advective and diffusive transport terms. Owing to the steady state nature of the same problem, expectations of the initial values sampled tend to zero as well, leaving us with values obtained from expectations of the boundary values sampled.

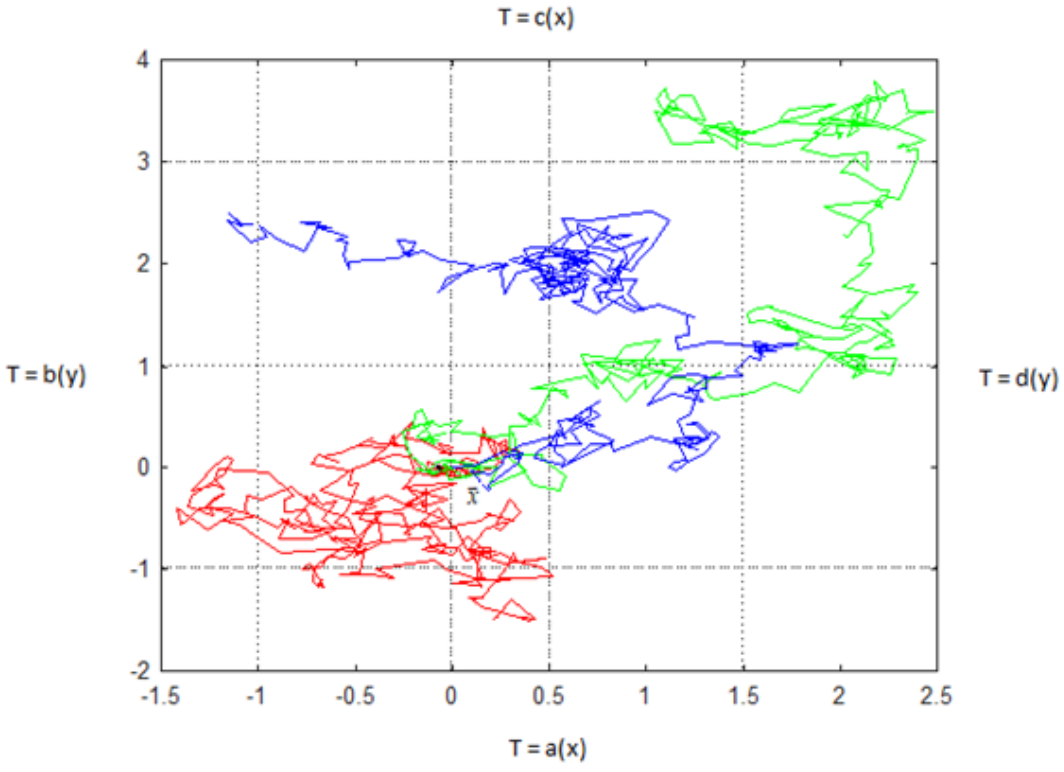

Figure 3.1: Motion of Random Walkers inside the 2-D Solution Domain.

Figure 3.1 helps provide a general depiction of the nature of the solution domain. The four sides are at certain temperatures; a(x), b(y), c(x) and d(y). Random walkers are released as a swarm from one solution point on the domain, labelled $\vec{x}$ here. The primary driving engine for these random walkers for the current research is the Ito's lemma (equation (1)):

$$d\vec{x}(t) = \vec{b}dt + \hat{a}d\vec{w} \quad (1)$$

Of which the specifications are the same as described in the Introduction section.

To solve this zeroth order partial differential equation, Equation (1) had to be discretized in order to be programmable. The new discretized equation hence looks like:

$$(x, \tilde{t})_{new} = (x, \tilde{t})_{old} + b * ds + \sqrt{2 * D * ds} * distribute(out) \quad (3)$$

From equation (4), $(x, \tilde{t})$ is the solution coordinate, b stands for the flow velocity, ds depicts the time step size incorporated and D represents diffusivity of the fluid. Now in order to understand the terms *distribute* and *out*, a small detour around the multidimensional wiener process is required.

Referring [26], it can be noted that the central limit theorem enables $d\vec{w}$ to have a normal distribution and be represented by its mean and standard deviation. It can be noted that the mean for the same is zero and the variance is directly proportional to the square root of time step size i.e. the length of an interval. This helps reduce $d\vec{w}$ to $w * \sqrt{ds}$, where w is a standard normal variable with variance of 1 and mean of 0. distribute, as a result, is a component of this normal distribution function with mean of 0 and variance of 1. To incorporate the random behavior, time since epoch is used as a seed for an industry standard psuedo - random number generator, mt19937_64, which gives an output *out*.

From equation (1), $\hat{a}.d\vec{w}$ is a random process that represents the diffusion process whereas $\vec{b}.ds$ is a deterministic process, all of which contributes towards random walker

displacement.

For a purely conduction problem, the drift velocity b here reduces to zero, hence leaving the solution to be completely governed by the diffusion process $\hat{a}.d\vec{w}$. Much about it is discussed in the latter chapters.

Now for cases where drift velocity b does not reduce to zero, the determination of random walker trajectory is obtained by diffusion and drift superposition. As mentioned previously, much about it is discussed in the later chapters.

Coming back to the crux of the problem, equation (2) was obtained with the usage of A. Friedman (1975) [15], and can be applied to any continuum scalar field. In this research, the scalar variable is temperature at any given point in the solution domain.

$$T(\tilde{x}) = E_{\tilde{x}}[Boundary\ Conditions\ Sampled]$$

Expanding which provides,

$$T(\tilde{x}) = \frac{1}{N}\sum_{i=1}^{N_t} c(x_i) + \frac{1}{N}\sum_{i=1}^{N_b} a(x_i) + \frac{1}{N}\sum_{i=1}^{N_l} b(y_i) + \frac{1}{N}\sum_{i=1}^{N_r} d(y_i) \quad (4)$$

$$N = N_t + N_b + N_l + N_r \quad (5)$$

Where N_t is the number of random walkers impinged on the top boundary, N_b being the number of random walkers impinged to the bottom boundary, and N_l and N_r being the number of random walkers sticking to the left and the right boundary respectively. As is evident through figure 3.1, $c(x_i), a(x_i), b(y_i), d(y_i)$ are the respective temperatures of top, bottom, left and right boundaries.

It can be clearly noted that the solution obtained through this method is governed by a few simple steps. Release random walkers from a solution point in the domain, track their trajectories, obtained their individual temperatures as they hit the boundary and utilize equation (4) to solve for temperature at that solution point in the domain.

CHAPTER 4: ANALYTICAL SOLUTION AND TEST PARAMETERS

For the research here, 4 different test cases will cover different aspects of governing properties that impact the trajectory of random walker and hence will help to provide a better comparison between the analytical and stochastic solutions under different conditions. Two broader category are the conditions under no flow and the rest with flow in the solution domain. All of the stochastic solutions are developed using C++ (refer Appendix A). The results obtained are discussed in the latter chapter.

4.1 No Flow Condition

To better understand a no flow condition, imagine a solid box that has a heated surface. Here, the heat will diffuse from the hotter surface to the cooler region hence forming a temperature gradient profile. The drift velocity will be zero and the problem will be under steady state condition. Referring S. Kakac ET. Al. (1993) [21], Appendix B provides a much elaborate derivation of the analytical solution used to obtain a temperature gradient profile when a heated surface in a solid box undergoes conduction.

$$T_n(x,y) = \sum_{n=1}^{\infty} \frac{2T_s}{w.\sinh(l.\lambda_n)} \left[\frac{1-\cos(w.\lambda_n)}{\lambda_n}\right] \sin(x.\lambda_n)\sinh(y.\lambda_n) \quad (6)$$

$$\lambda_n = \frac{n\pi}{w} \quad (7)$$

$$n = 1, 2, 3....$$

Equation (6), with the usage of MATLAB (refer Appendix D), helps provide a temperature gradient profile in the solid box undergoing conduction. T_s, w and l are the heated surface temperature, width and the length of the domain. x and y are the coordinates within the solution domain. Figure 4.1 provides a schematic

representation of the solution domain. Positive x is in the left hand direction whereas positive y is in the downwards direction.

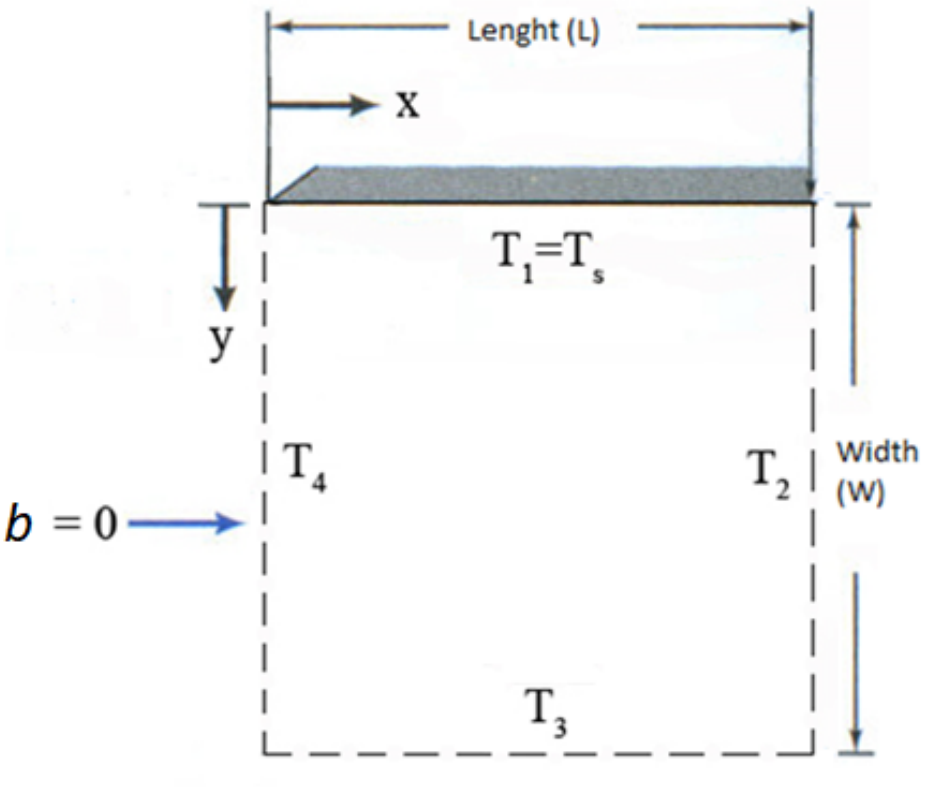

Figure 4.1: Solution Domain and Parameters for Case A.

Labelling this case as Case A, the following input parameters were utilized to obtain the desired result:

Table 4.1: Case A Parameters.

Case A	*Analytical Solution*	*Stochastic Solution*
Length	0.1 m	0.1 m
Width	0.03 m	0.03 m
$T2 = T3 = T4 = T_\infty$	0^oC	0^oC
$T_1 = T_s$	100^oC	100^oC
Drift Velocity (b)	0 m/s	0 m/s
Fluid Viscosity (μ)	4.79 x $10^{-7}m^2/s$	4.79 x $10^{-7}m^2/s$
Thermal Diffusivity (α)	6.7 x$10^{-5}m^2/s$	6.7 x$10^{-5}m^2/s$

4.2 With Flow Condition

To understand conditions where flow impacts the temperature gradient, the flat-plate problem provides an easier outlook to a much complex problem.

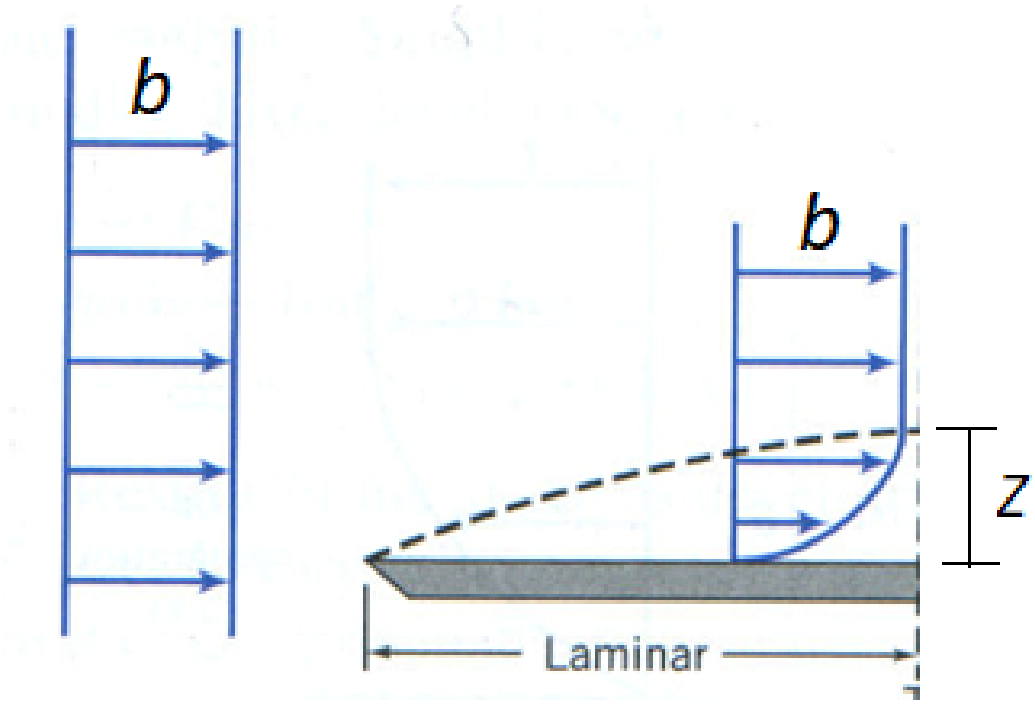

Figure 4.2: Flow Over A Flat Plate.

Figure 4.2 [27] here showcases the formation of boundary layer when a heated flat plat is introduced in a free stream. The solid dashed lines depicts the predicted formation of the thermal gradient profile. This profile is a result of the convective heat transfer occurring between the heated surface and the fluid.

Assuming no-slip, the velocity reaches 0 m/s right at the surface of the plate, whereas the stream right above it is slowed due to viscosity. Eventually as you go higher and higher, the respective streamlines overcome this viscous setback and attain free-stream velocity at a certain distance Z from the flat plate.

As was in the previous case, referring to S. Kakac et. Al. (1993) [21], Appendix C provides a complete derivation for the analytical solution to solve thermal boundary layer problems. Same as before, the problem will be solved under a steady state condition.

$$\frac{T(x,y)-T_\infty}{T_s-T_\infty} = 1 - 2\left[\frac{y}{\sqrt{\frac{12x\alpha}{b}}}\right] + \left[\frac{y}{\sqrt{\frac{12x\alpha}{b}}}\right]^2 \quad (8)$$

Here T_s and T_∞ are the surface and surrounding temperature respectively. b is the drift velocity. At the start of the main section, there was mention of 4 test cases. The first one was when drift velocity equals zero. The other three cases involve different parameters under the 'with flow condition'.

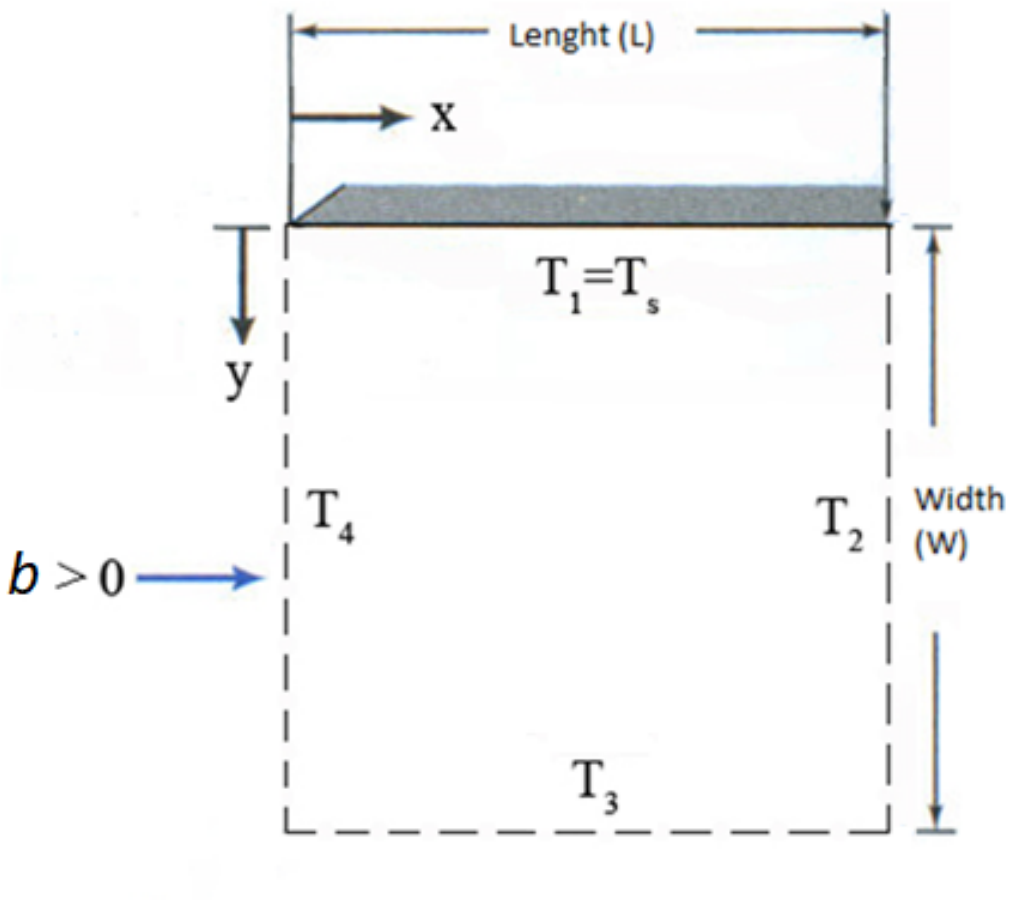

Figure 4.3: Solution Domain and Parameters for Case B, C and D.

Figure 4.3 provides a schematic representation of the solution domain for remaining three cases. Positive x is in the left hand direction whereas positive y is in the downwards direction. Flow b is in the positive x direction. Also, $T2 = T3 = T4 = T_\infty$

.

4.2.1 Case B

The following table provides with the input parameters used for both analytical and stochastic solution to obtain results for this case.

Table 4.2: Case B Parameters.

Case B	*Analytical Solution*	*Stochastic Solution*
Length	0.1 m	0.1 m
Width	0.03 m	0.03 m
$T2 = T3 = T4 = T_\infty$	200^{o}C	200^{o}C
$T_1 = T_s$	300^oC	300^oC
Drift Velocity (b)	0.02 m/s	0.02 m/s
Fluid Viscosity (μ)	4.79 x $10^{-7}m^2/s$	4.79 x $10^{-7}m^2/s$
Thermal Diffusivity (α)	6.7 x$10^{-5}m^2/s$	6.7 x$10^{-5}m^2/s$

4.2.2 Case C

Case C is different from Case B in the aspect that the surface and surrounding temperatures are raised, essentially increasing the difference between them.

Table 4.3: Case C Parameters.

Case C	*Analytical Solution*	*Stochastic Solution*
Length	0.1 m	0.1 m
Width	0.03 m	0.03 m
$T2 = T3 = T4 = T_\infty$	300^{o}C	300^{o}C
$T_1 = T_s$	600^oC	600^oC
Drift Velocity (b)	0.02 m/s	0.02 m/s
Fluid Viscosity (μ)	4.79 x $10^{-7}m^2/s$	4.79 x $10^{-7}m^2/s$
Thermal Diffusivity (α)	6.7 x$10^{-5}m^2/s$	6.7 x$10^{-5}m^2/s$

4.2.3 Case D

Case D focuses towards finding a solution with an increased drift velocity. In contrast with Case B, this case increases the drift velocity roughly five times.

Table 4.4: Case D Parameters.

Case D	*Analytical Solution*	*Stochastic Solution*
Length	0.1 m	0.1 m
Width	0.03 m	0.03 m
$T2 = T3 = T4 = T_\infty$	200^{o}C	200^{o}C
$T_1 = T_s$	300^oC	300^oC
Drift Velocity (b)	0.09 m/s	0.09 m/s
Fluid Viscosity (μ)	4.79 x $10^{-7}m^2/s$	4.79 x $10^{-7}m^2/s$
Thermal Diffusivity (α)	6.7 x$10^{-5}m^2/s$	6.7 x$10^{-5}m^2/s$

CHAPTER 5: RESULTS AND ERROR ANALYSIS

The solution domain was divided into 160 x 150 grids. After obtaining 160 x 150 temperatures within the domain, the function *imagesc* was utilised on MATLAB to provide with an image of the temperature gradient profile with scaled colours. Along with it, an error analysis was done to provide a clearer visual in the percentages of conformity of the stochastic solution to the analytical solution. This was done with another function in MATLAB named *contourf*, which provides a filled contours of any matrix.

5.1 Case A

Figure 5.1 depicts a temperature profile obtained through the analytical solution. Figure 5.2 is the temperature profile obtained using the stochastic solution. A quick visual analysis shows that there is some degree of error that occurs near the top left and top right boundaries. Apart from that, both the solution look visually conforming.

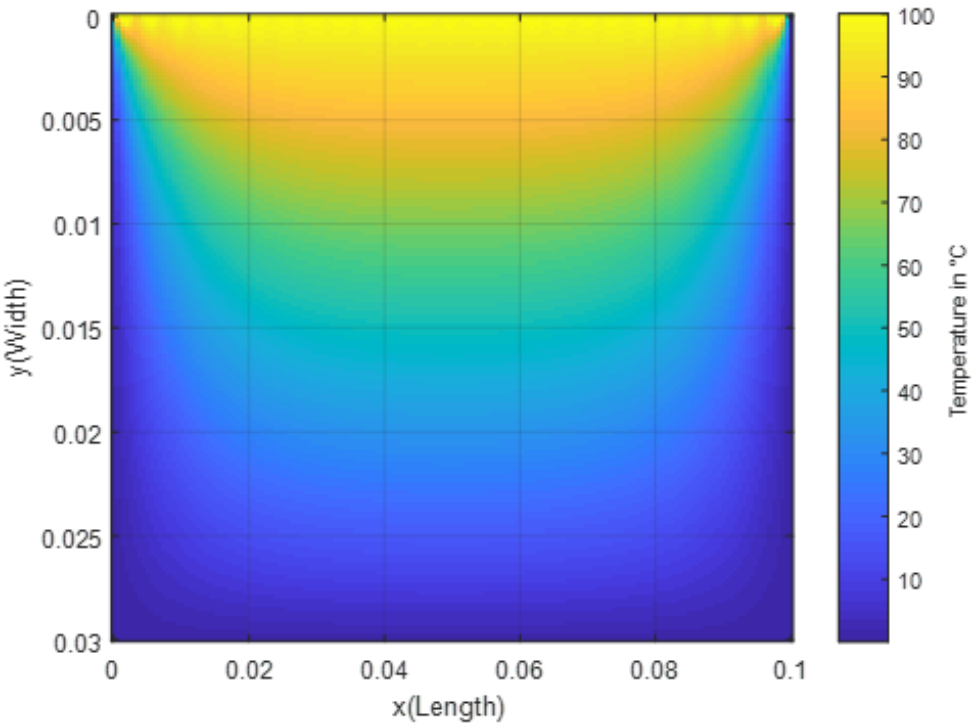

Figure 5.1: Analytical Temperature Profile (Case A).

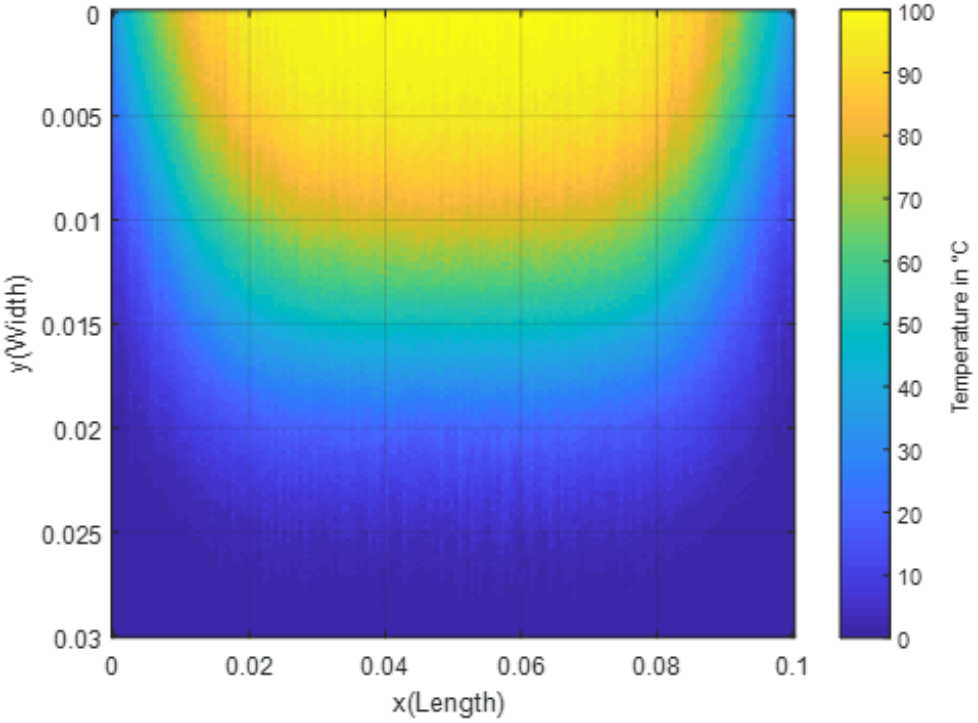

Figure 5.2: Stochastic Temperature Profile (Case A).

But such a visual analysis has its share of errors. To better understand the solution, figure 5.3 showcases the percentage error between the analytical and stochastic solution for Case A.

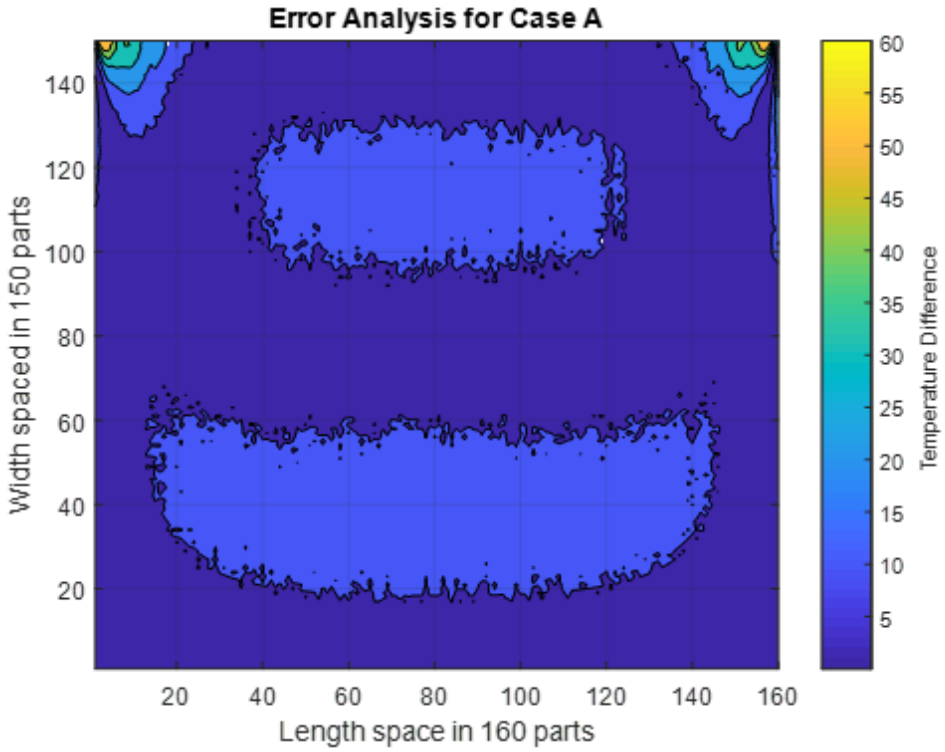

Figure 5.3: Calculated Error for Case A.

With the aid of the colour bar, it can be asserted that for the most part the error is roughly 0%-5% followed by sections of 5%-10% error. Near the top left and top right boundaries, as was indicated previously, a very small section shoots up to 60% error followed by smaller regions of 40%, 30% and 20% error. In contrast to the solution domain, these regions are very small. Regardless, to obtain complete conformity these regions should be taken to special consideration.

5.2 Case B

Figure 5.4 here shows a temperature profile obtained through the analytical solution for Case B. Figure 5.5 is the temperature profile obtained using the stochastic solution. A quick visual analysis shows that there is some degree of error that occurs near the

top left and the bottom right boundaries. Apart from that, both the solution look visually conforming.

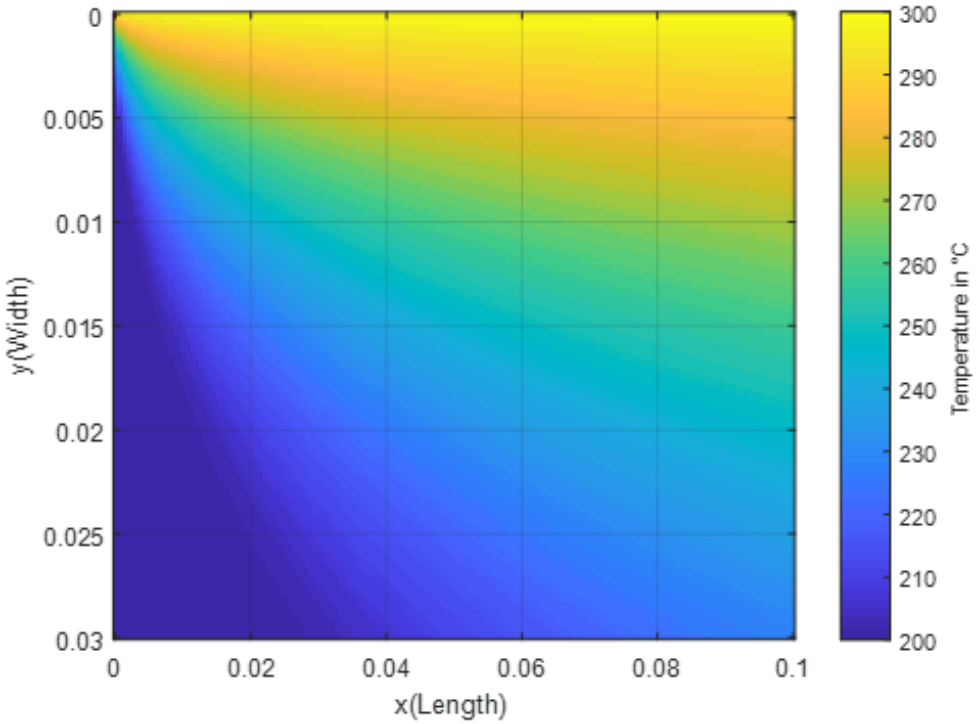

Figure 5.4: Analytical Temperature Profile (Case B).

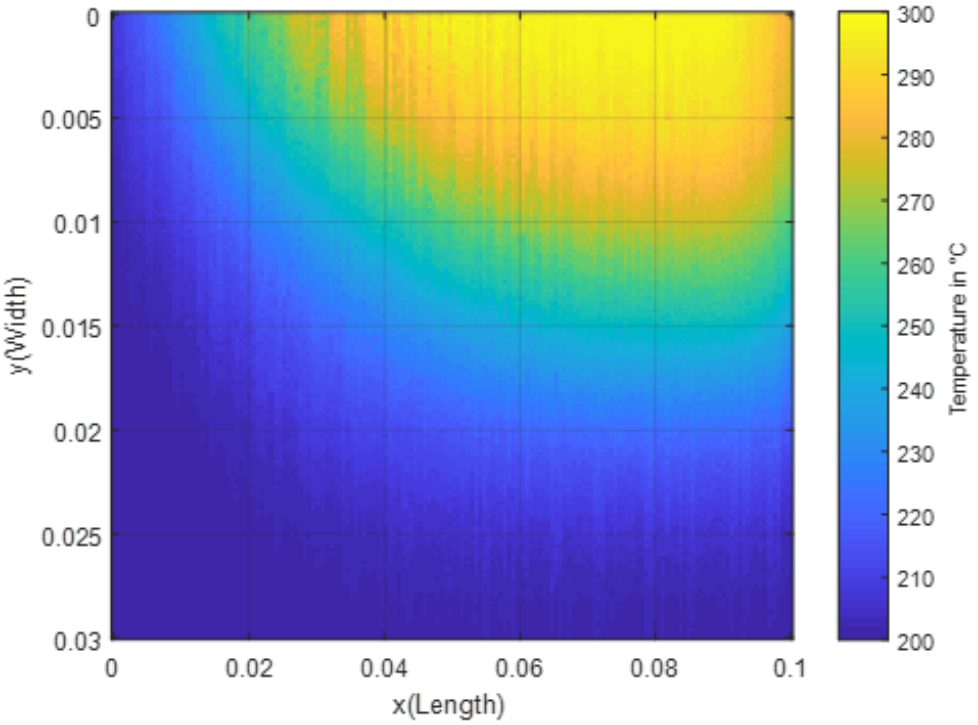

Figure 5.5: Analytical Temperature Profile (Case B).

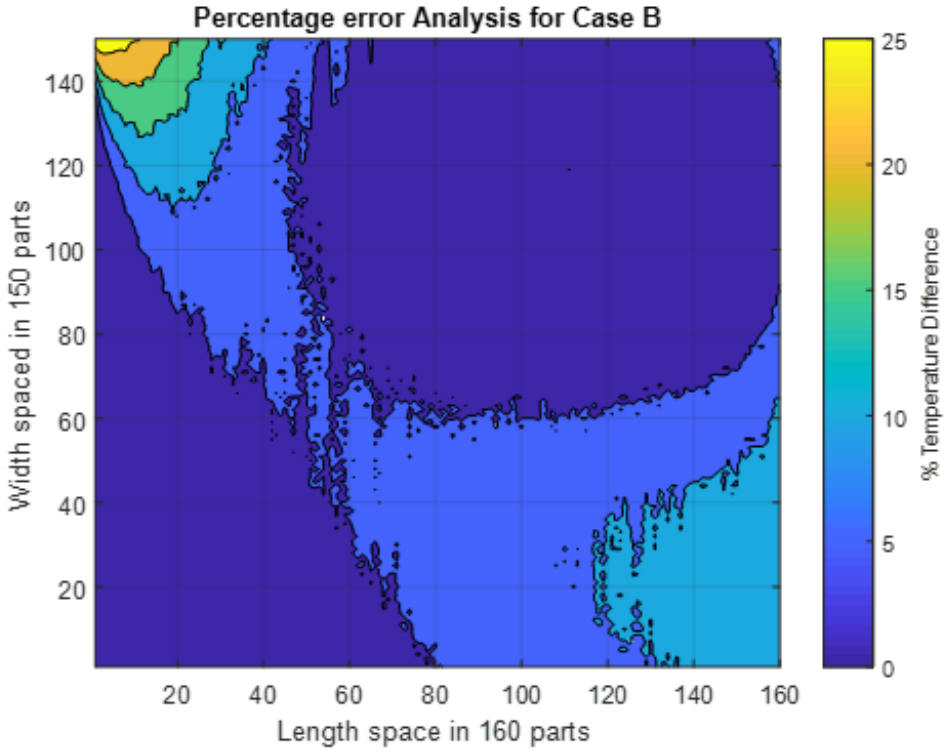

Figure 5.6: Calculated Error for Case B.

Figure 5.6 provides us with the contour plot for the percentage difference of the solutions obtained through the aforementioned steps. For the most part, the percentage error lies in the range of 0%-5%. But the two suspected regions of conformity has error shooting up to 25% (top left boundary) and roughly 10%-15% (bottom right boundary). As was mentioned in the previous case, special attention is required to monitor the behaviour near these boundaries.

5.3 Case C

Figure 5.7 here shows a temperature profile obtained through the analytical solution for Case C. Figure 5.8 is the temperature profile obtained using the stochastic solution for the same case. As was the situation in Case B, the visual analysis showcases the same type of anomaly observed near the top left and the bottom right boundaries. For the rest of the domain, both the solutions looks marginally agreeable.

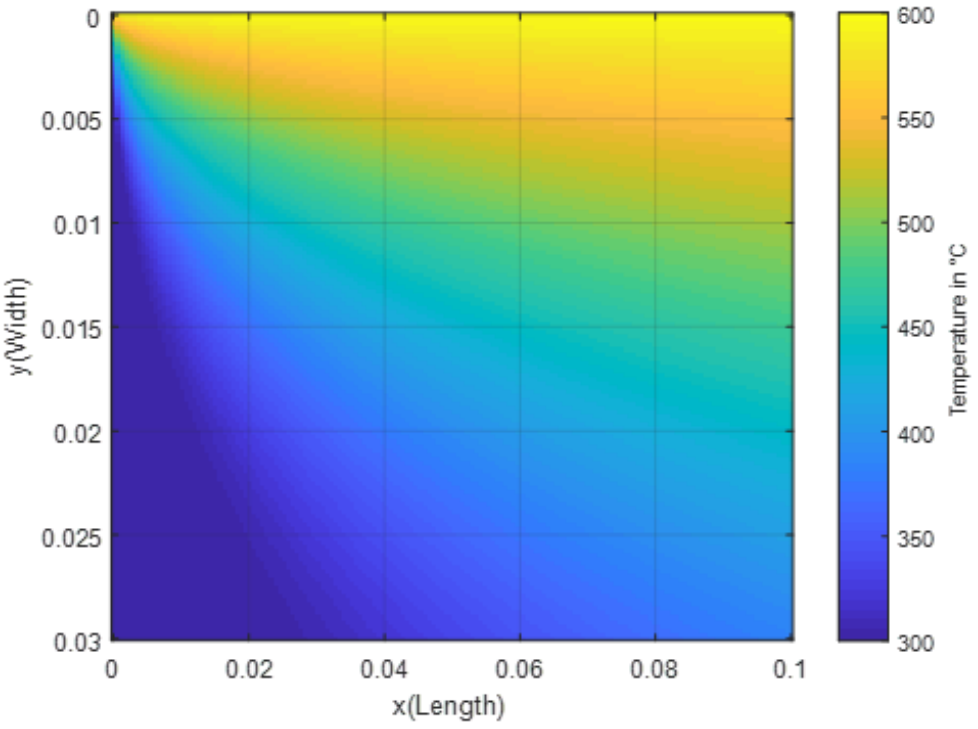

Figure 5.7: Analytical Temperature Profile (Case C).

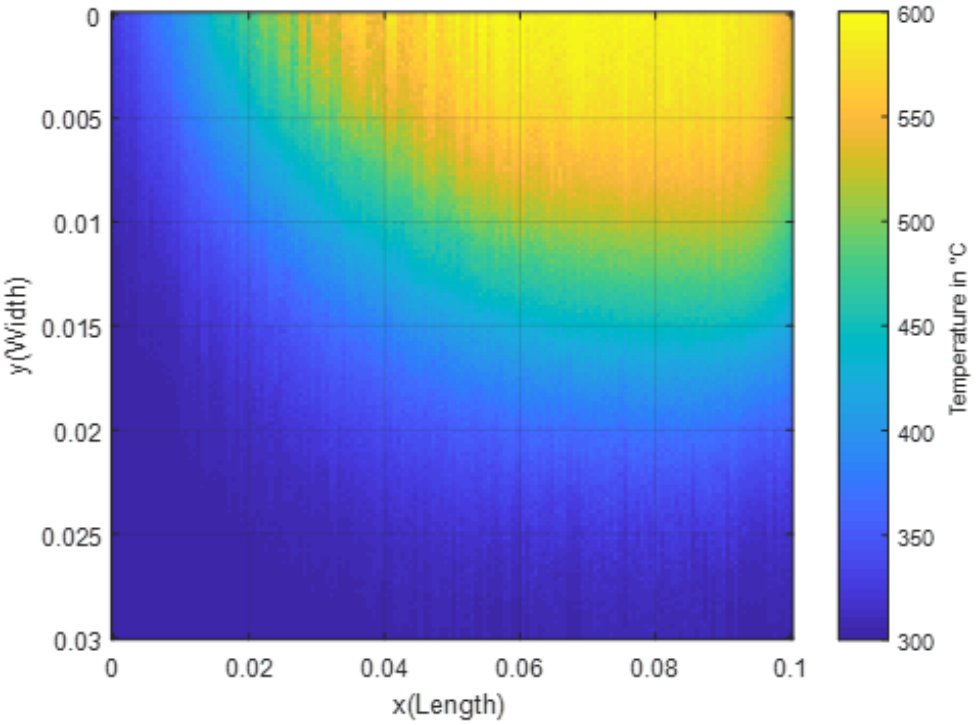

Figure 5.8: Analytical Temperature Profile (Case C).

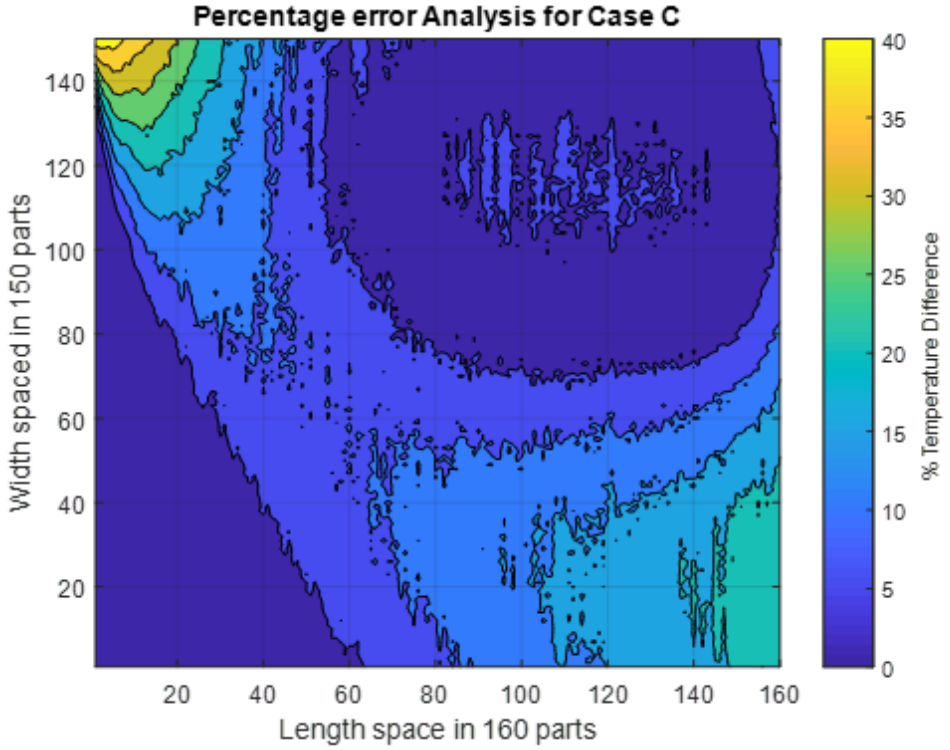

Figure 5.9: Calculated Error for Case C.

The percentage error analysis for Case C is depicted in Figure 5.9. For a good amount of the solution domain, the error is around 0%-10%. As we go closer to the top left boundary, the error percentage gradually increases from 10%, to 20%, to 25% and ultimately reaching 40% at a very small region. Moving right towards the bottom right boundary, the maximum difference in temperature shoots up to roughly 20%. This further strengthens the need to pay more attention to these two boundaries.

5.4 Case D

Figure 5.10 displays the temperature profile obtained through the usage of analytical solution for Case D. Figure 5.11, likewise, is the temperature profile obtained using the stochastic solution. It was different from Case B in that it had a drift velocity almost 5 times of the former. One of the random walker's trajectory governing factors was the time step size. Case D is different in the sense that all the previous cases utilised a time step size of 0.2 whereas this uses 0.07. The result in doing this is visually apparent. Figure 5.11 showcases that the lower the time step size, the more *hotter surface* value it samples hence overshooting or oversampling the values close to the right side boundary, away from the hot surface.

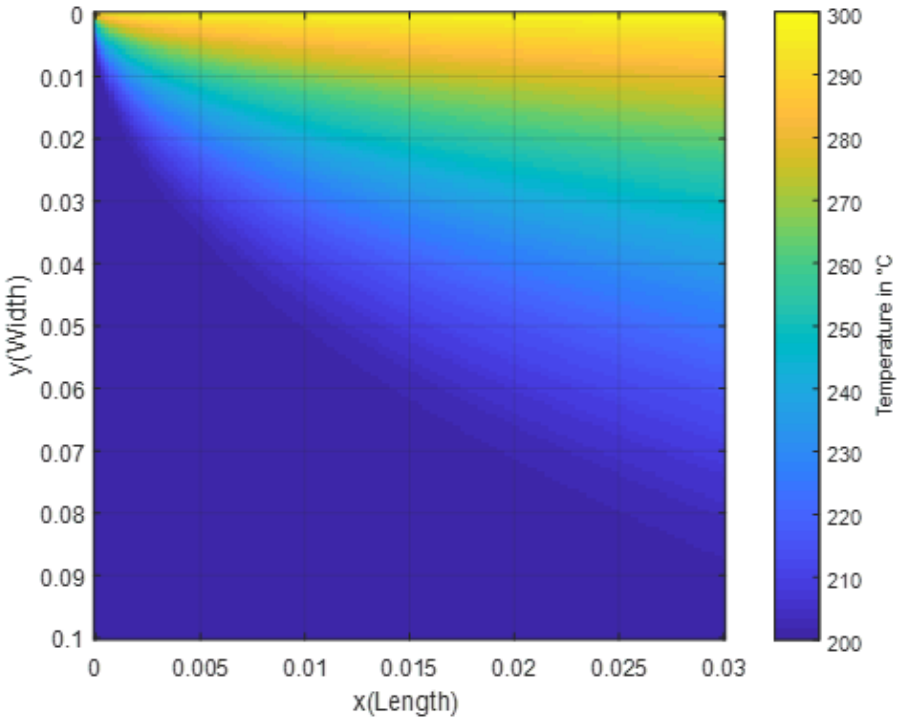

Figure 5.10: Analytical Temperature Profile (Case D).

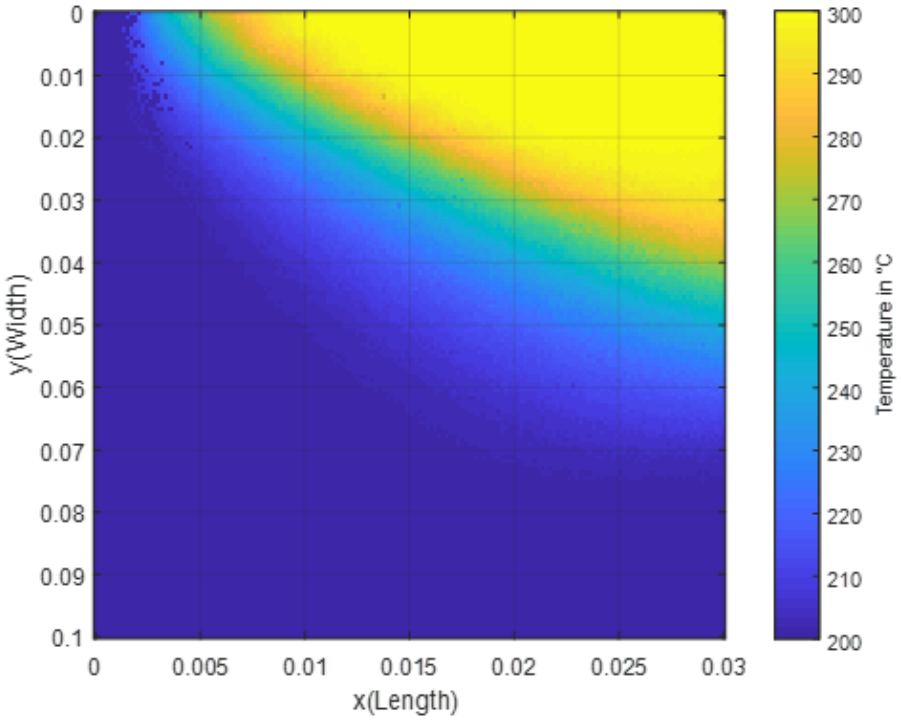

Figure 5.11: Analytical Temperature Profile (Case D).

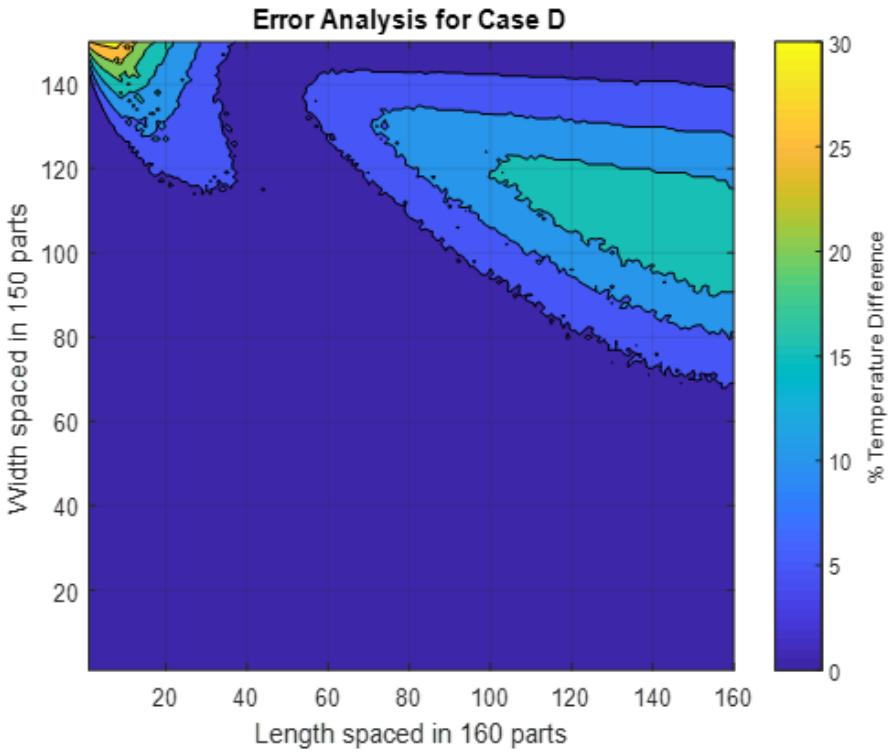

Figure 5.12: Calculated Error for Case D.

Figure 5.12 provides us the with the error analysis for the same. One boundary that remains the constant region of anomaly is the top left region where the temperature difference shoots up to 40%, albeit at a very small section. Another noticeable trend is the rise of the temperature difference to 20% between the top and middle right boundary. This is region where values were over sampled. For an appreciable part of the domain, the error is around 0%-5%.

CHAPTER 6: CONCLUSIONS

The error analysis in the previous section helps provide us with a better idea of the rate of conformity of the stochastic solution with the analytical solution. It can be said that the stochastic solution reasonably maps the analytical solution.

The regions of noted error are near the top left and right boundaries (Case A), top left and bottom right boundaries (Case B and C), and top left and right middle boundary (Case D). Errors here range from 4%-40%. It is to be noted that selection of time-step size plays an important role in the behaviour of random walkers. As was in Case D, a small time-step size (ts = 0.07) made the random walkers more inclined towards the heated surface than the other boundary. As can be noted in Case B and C, the time step size is not as smaller as in contrast with Case D (ts = 0.2) hence providing an error analysis graph much different than what was anticipated.

It is to be observed that increasing the number of random walkers, roughly beyond 10000, would not yield the desired result. Increasing grid sizes is another way that may be implemented to obtain a conformable result, but referring Appendix F even grid spacing of up to 1024 x 1024 does little to mitigate the error at previously noted regions of anomalies.

It is apparent that aside from the previously mentioned regions, the percentage error elsewhere for all the cases are well under 5-10%. These results, hence, helps narrow the focus towards the small regions of anomalies. One of the powerful approaches, especially when the random walker reaches close to the boundary is mentioned in the said paper [28]. It is an approach for future research.

REFERENCES

[1] M. Grzywinski and A. Sluzalec, Int. J. Heat Mass Transfer, Vol. 43, pp. 4003 (2000).

[2] S. Chevalier and O. Banton, J. Hydrology, Vol. 222, pp. 129 (1999).

[3] H. V. Nguyen et al., J. Hydrology, Vol. 215, pp. 188 (1999).

[4] B. Berkowitz and H. Scher, Transport Porous Media, Vol. 42, pp. 2411 (2002).

[5] J. D. McKinley, Comp. Geotechnics, Vol. 24, pp. 29 (1999).

[6] H. Trantham and D. Durnford, J. Containment Hydrol., Vol. 36, pp. 377 (1999).

[7] C.-I Hseih et al., Bound.-Layer Meteor. Vol. 109, pp. 113 (2003).

[8] O. Ditlevsen, Prob. Engrg. Mech., Vol. 18, pp. 97 (2003).

[9] A. M. Reynolds, Quart. J. Roy. Meteorolog. Soc. Vol. 125, pp. 1735 (1999).

[10] P. Franzese, Atmospheric Environment, Vol. 37, pp. 1691 (2003).

[11] N. Kljun et al., Bound.-Layer Meteor., Vol. 103, pp. 205 (2002).

[12] O. Kurbanmuradov and K. Sabelfeld, Bound-Layer Meteor., Vol. 97, pp. 191 (2000).

[13] Y. M. Gusey et al., J. Hydrol., Vol. 206, pp. 281 (1998).

[14] A. M. Reynolds, Physica D, Vol. 172, pp. 124 (2002).

[15] A. Friedman, Stochastic Differential Equations and Applications, Vol. 1 (Academic, New York, 1975).

[16] R. G. Keanini, Random walk methods for scalar transport problems subject to Dirichlet, Neumann, and mixed boundary conditions (2006).

[17] F. P. Incropera and D. P. DeWitt, Introduction to Heat Transfer, Fourth Edition (John Wiley & Sons, New York, 2002).

[18] Y. A. Cengel and J. M. Cimbala, Fluid Mechanics Fundamentals and Applications (McGraw-Hill, New York, 2006).

[19] W. M. Kays and M. E. Crawford, Convective Heat and Mass Transfer (McGraw-Hill, New York, 1980).

[20] V. S. Arpaci and P. S. Larsen, Convection Heat Transfer, (Prentice Hall, New York, 1984).

[21] S. Kakac and Y. Yener, Heat Conduction, Third Edition (Taylor & Francis, Washington DC, 1993).

[22] R. W. Fox et al., Introduction to Fluid Mechanics, Sixth Edition (John Wiley & Sons, New York, 2004).

[23] W. H. Press et al., Numerical Recipes in Fortran 77 The Art of Scientific Computing, Second Edition, Vol. 1 (Cambridge University Press, 1992).

[24] Paul Wilmott et al., The mathematics of financial derivatives: a student introduction (Cambridge University Press, 1995)

[25] Fischer Black and Myron Scholes, The Pricing of Options and Corporate Liabilities, Journal of Political Economy (University of Chicago Press, 1973)

[26] O. Yu. Kulchitski and D. F. Kuznetsov, The unified Taylor-Ito expansion, Journal of Mathematical Sciences, Vol. 99, No. 2 (Springer Link, 2000)

[27] Nathan James Andreu, Stochastic Solutions to Thermal Conduction and Thermal Boundary Layer Problems (2005)

[28] Aaron M. Lattanzi et al., A fully-developed boundary condition for the random walk particle tracking method, International Journal of Heat and Mass Transfer (Elsevier, 2018)

[29] Russell Keanini, Green's function-stochastic methods framework for probing nonlinear evolution problems: Burger's equation, the nonlinear Schrodinger's equation, and hydrodynamic organization of near-molecular-scale vorticity, Annals of Physics (2011)

APPENDIX A: C++ CODE FOR RANDOM WALKER SOLUTION

```
#include <math.h>
#include <fstream>
#include <random>
#include <chrono>
#include <iostream>
#include <algorithm>
#include <thread>
#include <boost/multi_array.hpp>
using namespace std;
using namespace boost;
//////Class For Trajectory Determination//////
class path {
long double diff;
multi_array<long double, 1> local, drift;
multi_array<long double, 3> pos;
multi_array<long double, 2> w;
vector<int> sortedi(int q, int a, int b)
{
vector<int> indices(b-a);
for (int n = a; n < b; n++)
indices[n-a] = n;
auto greater_weight = [q,this] (int n, int m) -> bool { return (w[q][n] > w[q][m]);
};
sort(indices.begin(), indices.end(), greater_weight);
return indices;
}
```

```
public:
int Nb=0;
int Nl=0;
int Nr=0;
int Nt=0;
const int dimen, rw, nfr;
const long double maxt, steps, deltas;
const multi_array<long double, 1> bound;
path(int dimen, int rw, long double maxt, long double steps, long double deltas,
const multi_array<long double, 1> bound) : dimen(dimen), rw(rw), maxt(maxt),
steps(steps),nfr((int)round(maxt/deltas)), deltas(deltas), bound(bound), local(extents[dimen]),
drift(extents[dimen]), pos(extents[(int)round(maxt/deltas)+1][rw][dimen]),
w(extents[(int)round(maxt/deltas)+1][rw])
{
if (abs(maxt/deltas - nfr) > 0.0001) throw invalid_argument("maxt is not a mul-
tiple of delta s");
if (abs(deltas/steps - (int)(deltas/steps+0.5)) > 0.0001) throw invalid_argument("delta
s is not a multiple of steps");
}
//////This function runs the random walker//////
void runrw(const multi_array<long double, 1>& local, const multi_array<long
double, 1>& drift, long double diff) {
this->local = local;
this->drift = drift;
this->diff = diff;
fill_n(w.data(), w.num_elements(), 1);
for (int n = 0; n < rw; n++)
```

```
pos[0][n] = local;
const int nsteps = (int)((maxt/steps)+0.5);
auto lambd = [&] (int a, int b) -> void {
vector<bool> dead(b-a);
for (int n = a; n < b; n++) {
dead[n-a]=false;
}
normal_distribution<long double> distribute(0, 1);
multi_array<long double, 2>
prev_coord(extents[b-a][dimen]); int NN = 0;
for (int n = a; n < b; n++)
{
prev_coord[n-a] = pos[0][n];
}
multi_array<bool, 1> absorbed(extents[b-a]);
fill_n(absorbed.data(), absorbed.num_elements(), false);
vector<int> sorted_indices = sortedi(0, b, a);
for (int stp = 1; stp <= nsteps; stp++)
{
unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
mt19937_64 out(seed);
for (int n = a; n < b; n++)
{
if (dead[n-a] == true)
break;
for (int i = 0; i < dimen;i++)
{
```

```
//////Governing Equation//////
prev_coord[n-a][i] = prev_coord[n-a][i] + drift[i]*steps + sqrt(2*diff*steps)*distribute(out);
if (prev_coord[n-a][i] <= 0 || prev_coord[n-a][i] >= bound[i]) {
if (i==0 && prev_coord[n-a][i] <= 0) {
Nl++; break;
} else if (i==0 && prev_coord[n-a][i] >= bound[i]) {
Nr++; break;
} else if (i==1 && prev_coord[n-a][i] <= 0) {
Nb++; break;
} else if (i ==1 && prev_coord[n-a][i] >= bound[i]) {
Nt++; break;
}
absorbed[n-a] = true;
break; }
NN = Nl+Nr+Nt+Nb;
if (NN >= rw) break;
}
if (NN >= rw) break;
}
if (NN >= rw) break;
}
};
int nthreads = 12;
int Nperthd = rw/nthreads;
int rem = rw % nthreads;
thread* thrds = new thread[nthreads];
for (int i = 0; i < nthreads; i++)
```

```
{
thrds[i] = thread(lambd, i<rem ? i*Nperthd+i : i*Nperthd+rem,
i<rem ? (i+1)*Nperthd+i+1 : (i+1)*Nperthd+rem);
}
for (int i = 0; i < nthreads; i++)
{
thrds[i].join();
}
delete[] thrds;
}
};
//////Main Function//////
int main()
{
int q1,q2,q3,q4,k1,k2,k3,k4;
int f; f = 0;
int c; c= 0;
multi_array<long double, 1> boundary(extents[2]), point(extents[2]), driftv(extents[2]),
dx(extents[2]);
boundary[0] = 0.1;
boundary[1] = 0.03;
driftv[0] = -0.09;
driftv[1] = 0;
long double data[160][150];
for (int i = 0; i < 160; i++)
{
for (int j = 0; j < 150; j++)
```

```
{
data[i][j]=0;
}
}
int l;
l = 0;
for(point[0] = 0.000625; point[0] <= 0.1; point[0] = point[0] + 0.000625)
{
c = 0;
for(point[1] = 0.0002; point[1] <= 0.03; point[1] = point[1] + 0.0002)
{
path pt(2, 10000, 5, 0.05,0.05, boundary);
pt.runrw(point, driftv, 6.7e-5);
q1=0;
q2=0;
q3=0;
q4=0;
k1 = pt.Nb;
k2 = pt.Nt;
k3 = pt.Nr;
k4 = pt.Nl;
u = k1+k2+k3+k4;
for(int op = 0;op < k1; op++) q4 = q4 + 300;  //Hotter surface temperature.
Change temp. accordingly
for(int io = 0;io < k2; io++) q1 = q1 + 200;  //Ambient surrounding temperature.
Change temp. accordingly
for(int pl = 0;pl < k3; pl++) q2 = q2 + 200;  //Ambient surrounding temperature.
```

```
Change temp. accordingly
   for(int ik = 0;ik < k4; ik++) q3 = q3 + 200;  //Ambient surrounding temperature.
Change temp. accordingly
   data[l][c] = (q4+q1+q2+q3)/u;
   c++;
   cout«"Nb = "«k1«" | Nt = "«k2«" | Nr = "«k3«" | Nl = "«k4«"";
   }
   f = f + 1;
   cout«f«"%";
   l++;
   }
   ofstream value("finald6.dat");
   for (int i = 0; i < 160; i++)
   {
   for (int j = 0; j < 150; j++)
   {
   value « data[i][j] « ' ';
   }
   value « '\n';
   }
   value.close();
   return 0;
   }
```

APPENDIX B: DETAILED CASE A ANALYTICAL SOLUTION DERIVATION

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

$$x(0) = 0$$

$$x(b) = 0$$

$$y(0) = 0$$

$$y(h) = T_s$$

$$T(x,y) = X(x)Y(y)$$

$$YX'' + XY'' = 0$$

$$\frac{X''}{X} + \frac{Y''}{Y} = 0$$

$$\frac{X''}{X} = -\frac{Y''}{Y} = \lambda^2$$

$$X''+X\lambda^2 = 0$$

$$Y''-Y\lambda^2 = 0$$

$$X(x) = A\cos(\lambda.x) + B\sin(\lambda.x)$$

Applying boundary condition $x(0) = 0$:

$$A = 0$$

Applying boundary condition $x(b) = 0$:

$$0 = sin(\lambda.b)$$

$$\lambda_p = \frac{p.\pi}{b}$$

$$p = 1,\ 2,\ 3,.....,50,.....$$

$$Y(y) = C\sinh(\lambda.y) + D\cosh(\lambda.y)$$

Applying boundary condition $y(0) = 0$:

$$D = 0$$

$$T = X.Y = B\sin(\lambda.x).C\sinh(\lambda.y)$$

$$T_p = \sum_{p=1}^{\infty} E\sin(\lambda_p.x).\sinh(\lambda_p.y)$$

Resulting in:

$$T_p.\int_0^b \sin(\lambda_q.x)dx = \left[\sum_{p=1}^{\infty} E\sin(\lambda_p.x).sinh(\lambda_p.y)\right].\int_0^b sin(\lambda_q.x)dx$$

Applying boundary condition $y(h) = T_s$:

$$T_s.\int_0^b \sin(\lambda_q.x)dx = \left[\sum_{p=1}^{\infty} E\sin(\lambda_p.x).sinh(\lambda_p.h)\right].\int_0^b sin(\lambda_q.x)dx$$

Yields:

$$T_s\left[\frac{1-\cos(b.\lambda)}{\lambda}\right] = E.\left(\frac{b}{2}\right).\sinh(h.\lambda)$$

And thus finally giving us:

$$T_p(x,y) = \sum_{p=1}^{\infty} \frac{2T_s}{b.\sinh(h.\lambda_p)}\left[\frac{1-\cos(b.\lambda_p)}{\lambda_p}\right]\sin(x.\lambda_p)\sinh(y.\lambda_p)$$

Source: [27]

APPENDIX C: DETAILED CASE B, C AND D ANALYTICAL SOLUTION DERIVATION

The derivation starts with the basic energy equation assuming steady, incompressible, laminar flow with constant fluid properties and also knowing that $\frac{dp}{dx} = 0$

$$u.\frac{\partial T}{\partial x} + \nu.\frac{\partial T}{\partial y} = \partial.\frac{\partial^2 T}{\partial x^2} + \frac{\nu}{c_p}\Big(\frac{\partial u}{\partial y}\Big)$$

Further reducing the energy equation by considering the flow to be parallel and neglecting viscous dissipation:

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

Using integral method and applying boundary conditions:

$$T(x,0) = T_s$$
$$\lim_{y\to\infty} T(x,y) = T_\infty$$
$$T(0,y) = T_\infty$$

We now define the thermal boundary layer thickness, $\delta(x)$, such that:

$$T(x,\delta) = T_s$$
$$\frac{\partial T(x,\delta)}{\partial x} = 0$$

As a result, the fluid now has a free stream temperature for $y > \delta(x)$. Integrating the left-hand side and treating u as a constant (since we are assuming Pr«1):

$$\int_0^{\delta(x)} \frac{\partial^2 T}{\partial y^2} dy = \frac{u}{\alpha}\int_0^{\delta(x)} \frac{\partial T}{\partial x} dy$$

$$\int_0^{\delta(x)} \frac{\partial^2 T}{\partial y^2} dy = \left.\frac{\partial T}{\partial y}\right|_{y=\delta} - \left.\frac{\partial T}{\partial y}\right|_{y=0} = -\left.\frac{\partial T}{\partial y}\right|_{y=0}$$

Using Leibnitz rule to rewrite the right-hand side:

$$\int_0^{\delta(x)} \frac{\partial T}{\delta x} dy = \frac{d}{dx}\int_0^{\delta(x)} T(x,y)dy - T(x,\delta)\frac{d\delta}{dy} = \frac{d}{dx}\int_0^{\delta(x)} T(x,s)dy - T_\infty \frac{d\delta}{dy}$$

This yields:

$$-\left.\frac{\partial T}{\partial y}\right|_{y=0} = \frac{u}{\alpha}\left[\frac{d}{dx}\int_0^{\delta(x)} T(x,s)dy - T_\infty \frac{d\delta}{dy}\right]$$

Assuming that the temperature distribution *T(x,y)* can be represented over $0 \leq y \leq \delta(x)$ by a second-degree polynomial in the form:

$T(x,y) = a + by + cy^2$

Solving for a, b, and c yields:

$a = T_s$

$b = \frac{2(T_\infty - T_s)}{\delta}$

$c = \frac{-(T_\infty - T_s)}{\delta^2}$

Therefore:

$$\frac{T(x,y) - T_\infty}{T_s - T_\infty} = 1 - 2\left(\frac{y}{\delta}\right) + \left(\frac{y}{\delta}\right)^2$$

And hence:

$$\delta\frac{d\delta}{dx} = \frac{6\alpha}{u}$$

Since $\delta(0) = 0$:

$$\delta = \sqrt{\frac{12\alpha x}{u}}$$

After substitution, we have:

$$\frac{T(x,y) - T_\infty}{T_s - T_\infty} = 1 - 2\left(\frac{y}{\sqrt{\frac{12\alpha x}{u}}}\right) + \left(\frac{y}{\sqrt{\frac{12\alpha x}{u}}}\right)^2$$

where u is equal to the free stream velocity.

Source: [27]

APPENDIX D: MATLAB CODE FOR ANALYTICAL SOLUTION (CASE A)

```
T=100;          //Heated surface temperature
b = 0.1; h = 0.03;          //Box dimensions (breadth x height)
a = 1; l = 1;
tyy = zeros(160,150);
for p = 0.000625:0.000625:0.1
a = 1;
for m = 0.0002:0.0002:0.03
for n = 1:50
tyy(l,a) = tyy(l,a)+((2*T)/(b*sinh(h*(n*pi/b))))*((1-cos(b*(
n*pi/b)))/(n*pi/b))*sin(p*(n*pi/b))*sinh(m*(n*pi/b));
end
a = a+1;
end
l = l+1;
end
l = 1;
for p = 0.000625:0.000625:0.1
a = 1;
for m = 0.0002:0.0002:0.03
if(tyy(l,a)>100)
tyy(l,a) = 100;
end
a = a+1;
end
l = l+1;
end
```

```
figure(1)
imagesc([0 0.3], [0.1 0], tyy');
grid; colorbar;
```

APPENDIX E: MATLAB CODE FOR ANALYTICAL SOLUTION (CASE B,C,D)

```
U1 = 0.02;        // u(Flow Velocity) = 0.09 for Case D
T = 300;         // T(Heated surface Temperature) = 600 for Case C
To = 200;        // To(Ambient Temperature) = 300 for Case C
h = 0.03; b = 0.1;        // Box dimensions (breadth x height)
ert = 4.97 * 10^-7;       //FluidViscosity
wer = 6.7 * 10^-5;        //ThermalDiffusivity
ky =0;
c = 5.48;
i = 1;
for x = 0.000625:0.000625:b
ky = ky+1; kt = 0;
for y = 0.0002:0.0002:h
kt = kt+1;
if (x==0)
u = u1;
opdd(ky,kt) = To;
else
if (y==0)
u = 0;
opdd(ky,kt) = T;
else
u = u1;
f = sqrt(12*wer*x/u);
if(y>f)
opdd(ky,kt) = To;
else
```

```
opdd(ky,kt) = To+((T-To)*(1-(2*(y/f))+(y/f)^2));
end
end
end
end
end
figure(2)
imagesc([0 b], [0 h], opdd')
grid;
xlabel('x(Length)');
ylabel('y(Width)');
colorbar
//For percent error analysis
tro = zeros(160,150);
for i = 1:160
for j = 1:150
tro(i,j) = dt(i,j)/opdd(i,j) * 100;        // dt = abs(opdd - (stochastic solution))
end
end
```

APPENDIX F: ANALYTICAL AND STOCHASTIC SOLUTIONS WITH INCREASED GRIDS 1024 x 1024 (CASE A,B,C,D)

F.1 Case A

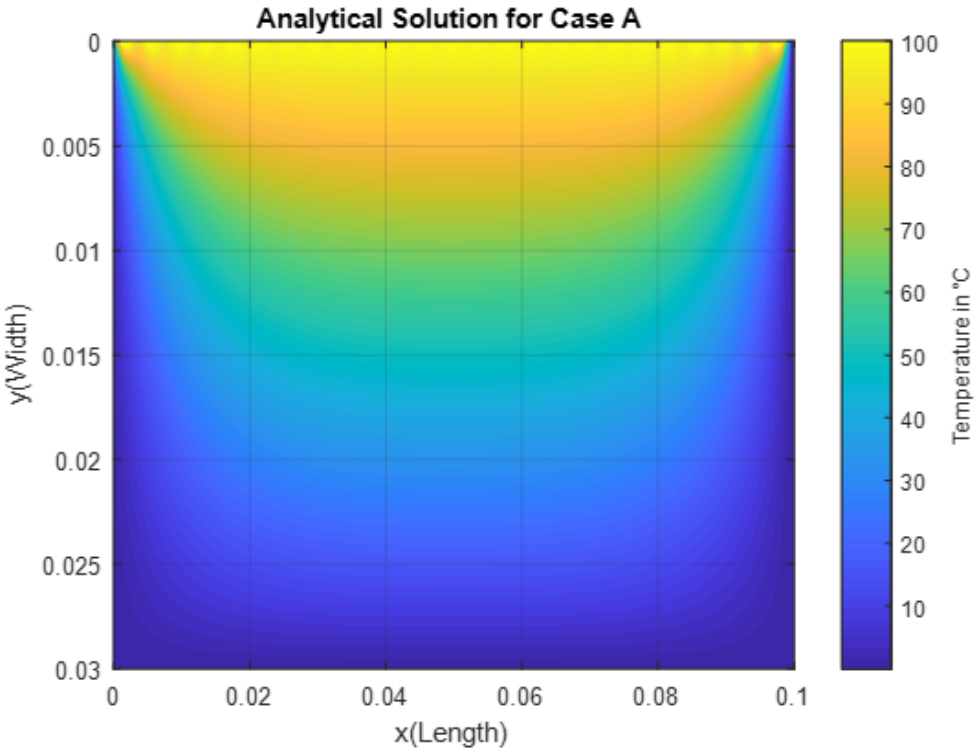

Figure F.1: Analytical Temperature Profile (Case A).

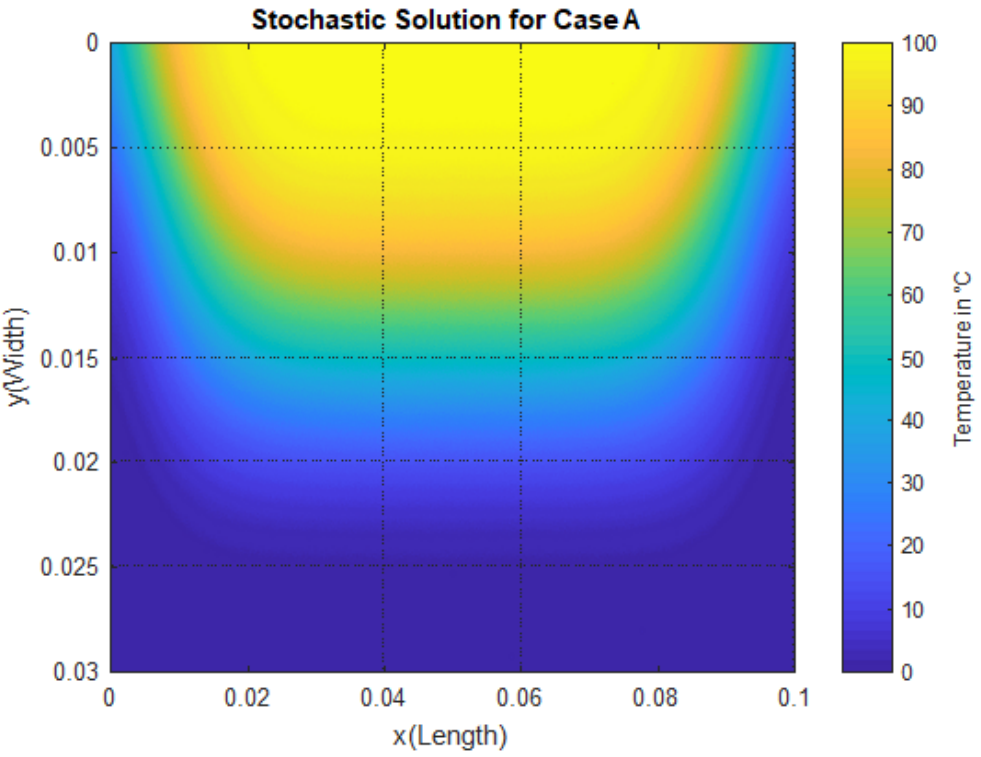

Figure F.2: Stochastic Temperature Profile (Case A).

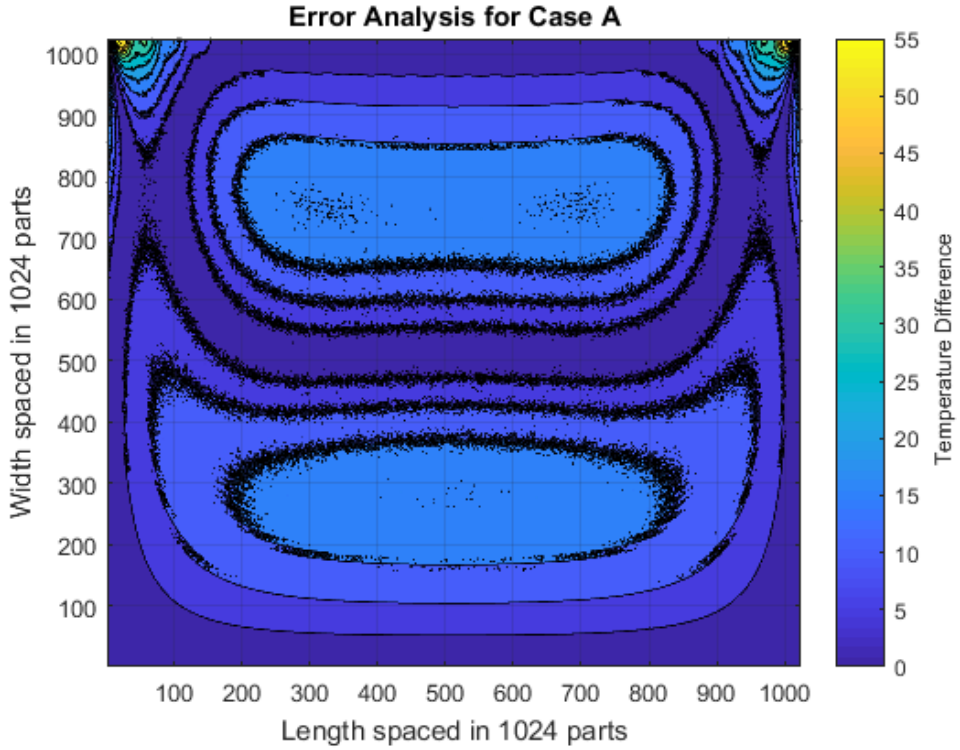

Figure F.3: Temperature Difference (Case A).

F.2 Case B

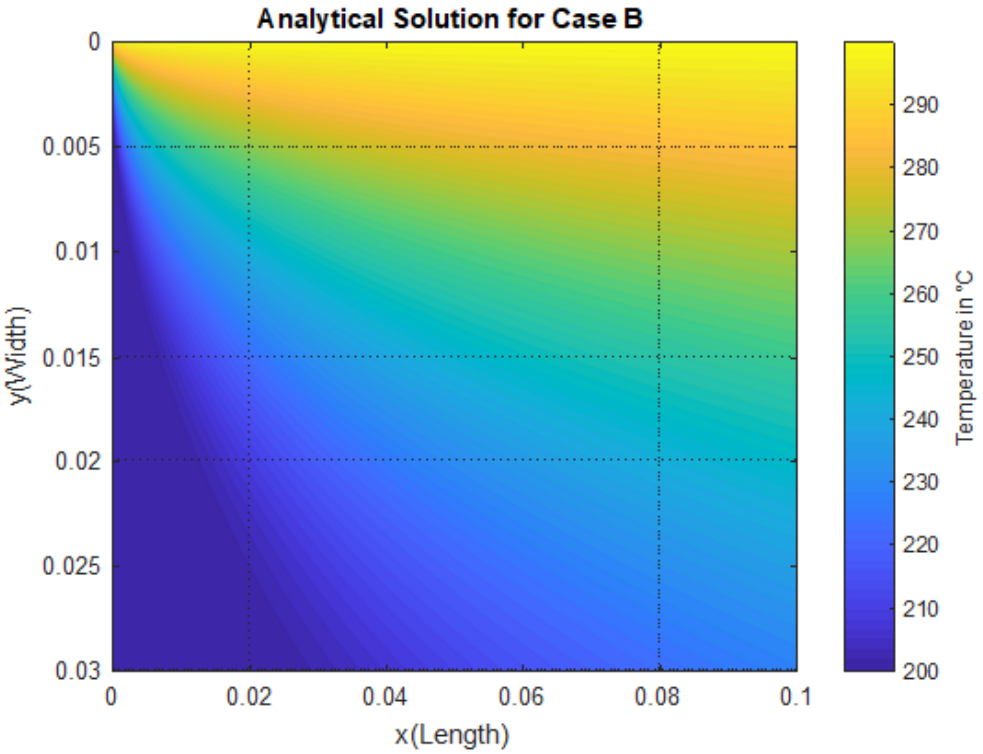

Figure F.4: Analytical Temperature Profile (Case B).

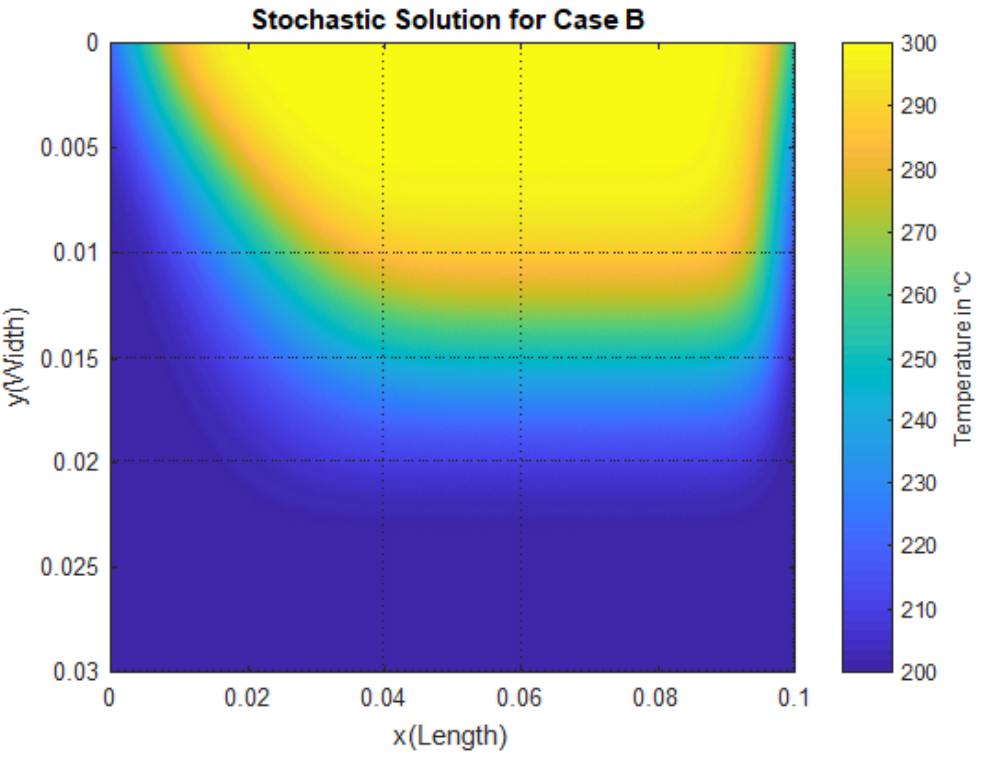

Figure F.5: Stochastic Temperature Profile (Case B).

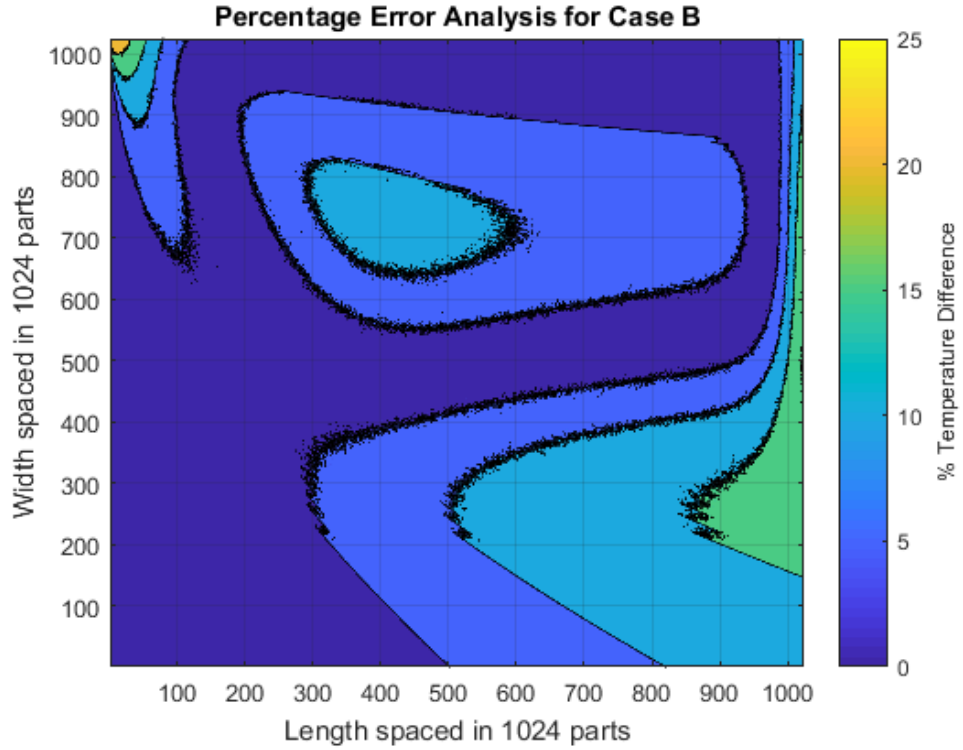

Figure F.6: Percentage Temperature Difference (Case B).

F.3 Case C

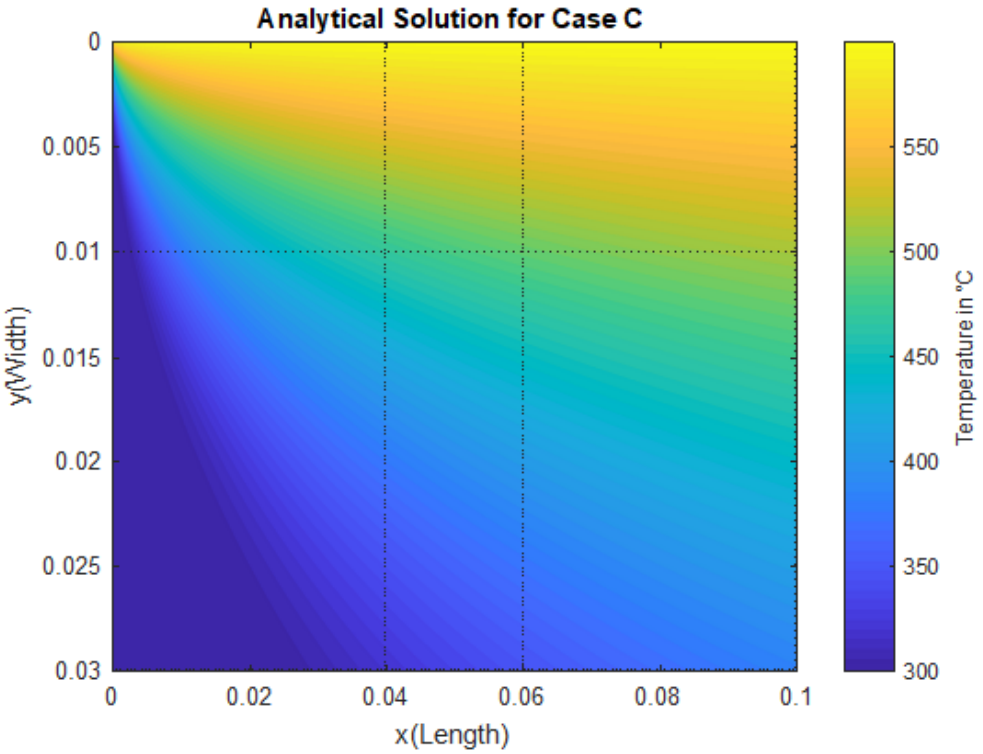

Figure F.7: Analytical Temperature Profile (Case C).

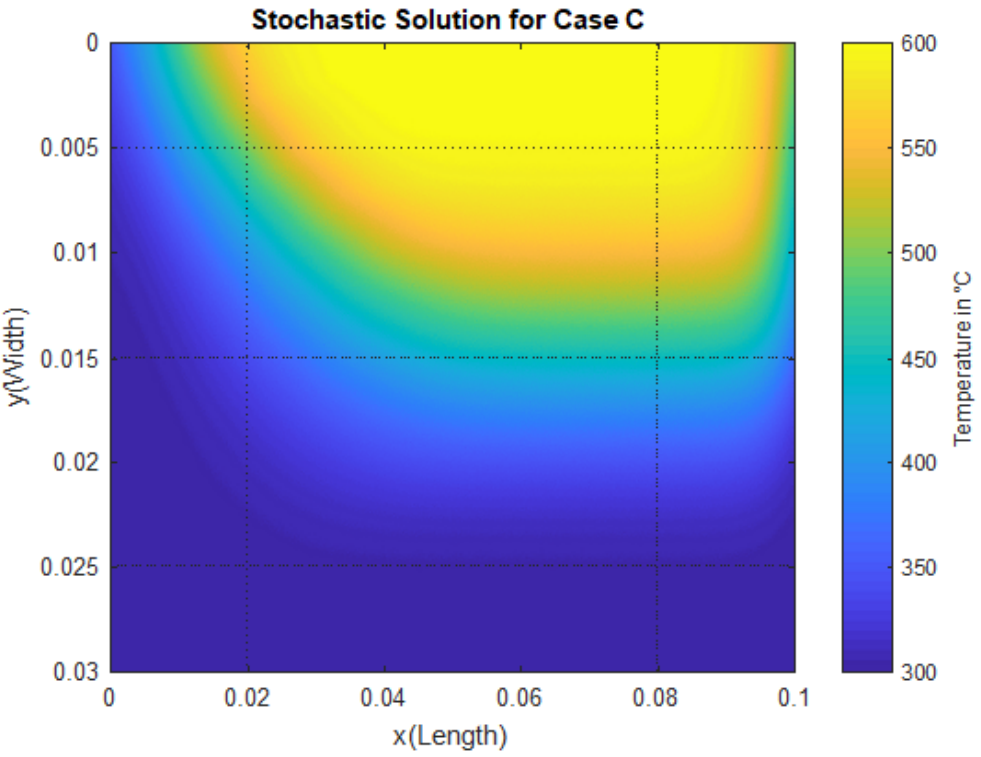

Figure F.8: Stochastic Temperature Profile (Case C).

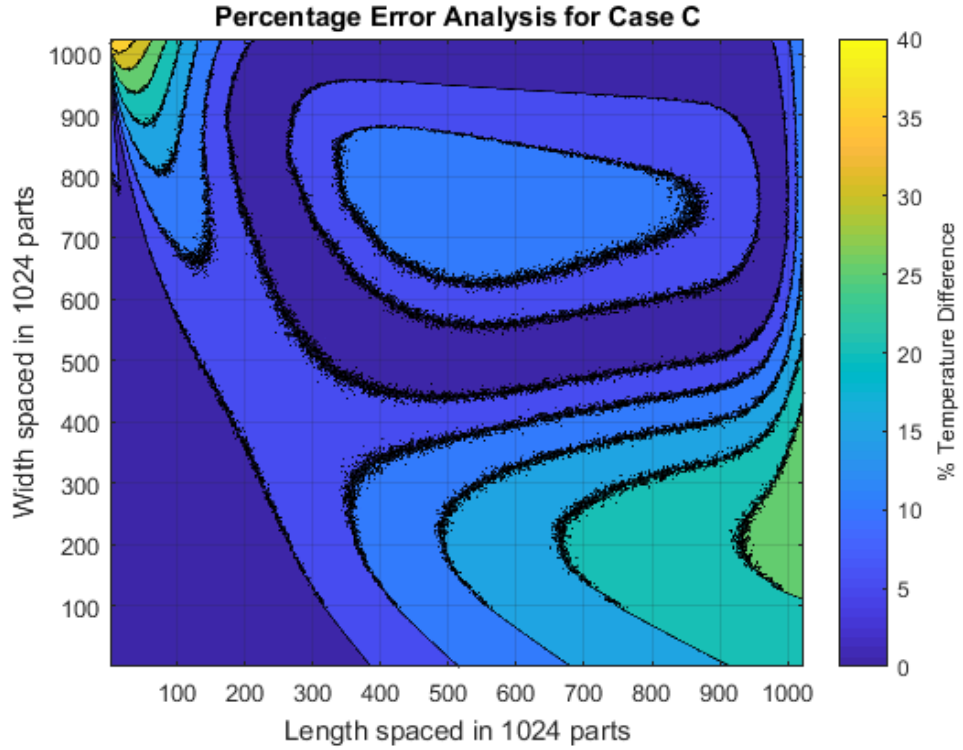

Figure F.9: Percentage Temperature Difference (Case C).

F.4 Case D

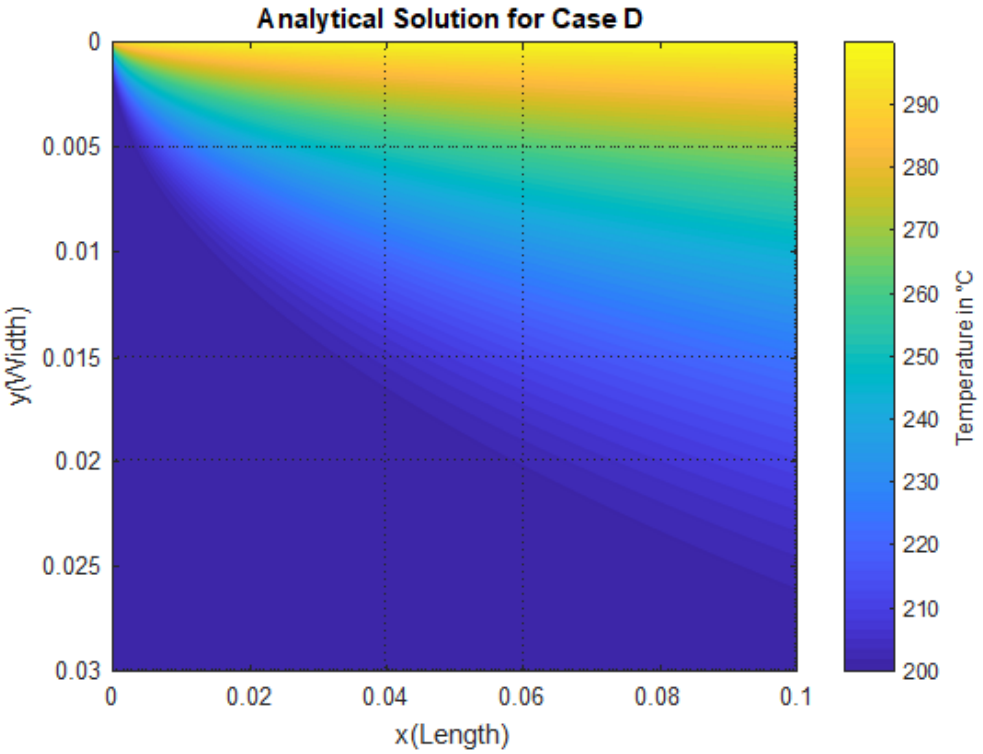

Figure F.10: Analytical Temperature Profile (Case D).

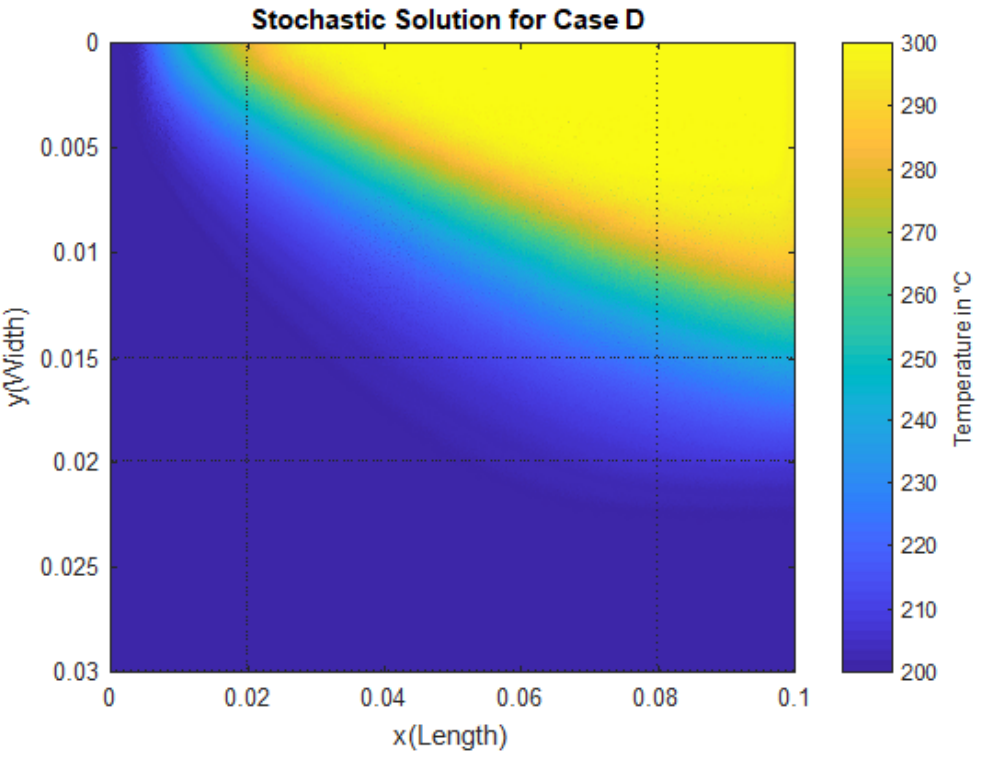

Figure F.11: Stochastic Temperature Profile (Case C).

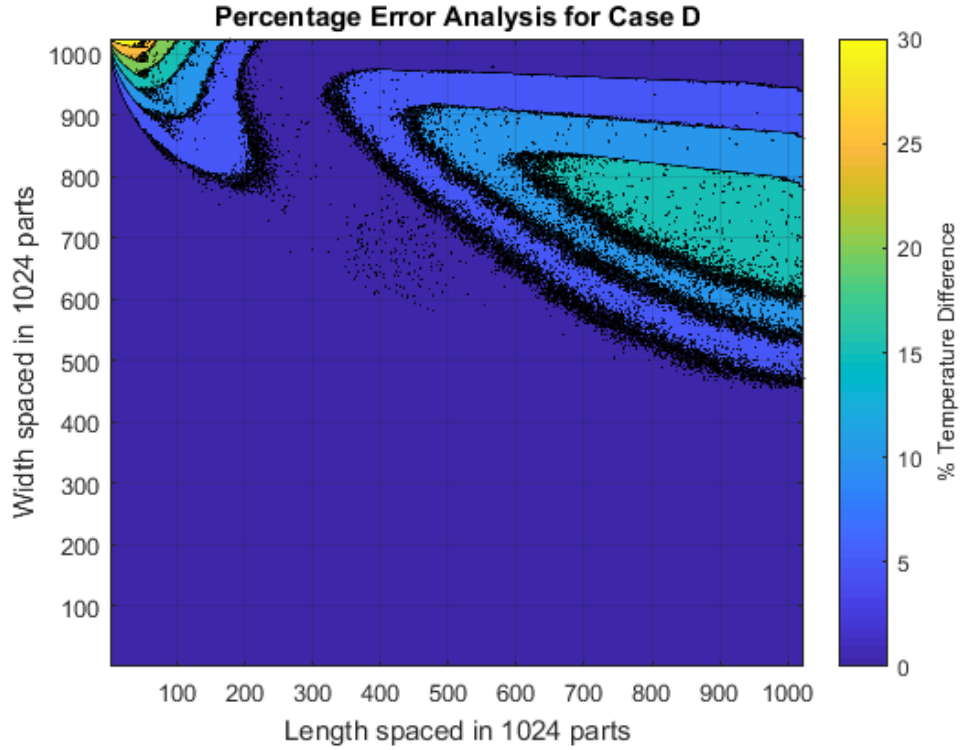

Figure F.12: Percentage Temperature Difference (Case C).