

**MXB226**

**Case Study Project**

**Notes: fix bolded equations**

**Oliver Canvin Tyler Ellwood Nikkissja Leo Alexander Ludford-Kidd**

**Table of Contents**

**1. Executive Summary**

# 2. Introduction

# 2.1 Purpose of the report

## 2.2 Issues to be discussed and their significance

# 3. Discussion

# 3.1 Mathematical Model

### 3.1.1 Laplace’s equation in discretised form

### 3.1.2 Solving for nodes

### 3.1.2.1 Solving for Node 1

# 3.1.2.2 Solving for ED boundary nodes

### 3.1.2.3 Solving for AE boundary nodes

### 3.1.2.3 Solving for Node 29

### 3.1.2.4 Solving for Node 24

### 3.1.2.5 Solving for Node 33

### 3.1.2.6 Solving for interior nodes

### 3.2 Direct Methods

### 3.2.1 Cholesky Decomposition

### 3.2.2 Packed Storage

### 3.2.3 Banded Storage

### 3.2.4 Sparse Storage

**3.3 Iterative Methods**

### 3.3.1 Jacobi Method

### 3.3.2 Gauss-Siedel Method

### 3.3.3 Successive Over-Relaxation Method

### 3.3.4 Conjugate Gradient Method

### 3.3.5 Compressed Sparse Row Storage

**3.4 Node Re-orderings**

# 4. Results

## 4.1 Heat Distribution

### 4.2 Ambient Temperature Investigation

**4.3 Method Performance**

### 4.3.1 Memory required

### 4.3.2 Required tolerance and iteration count

### 4.3.3 Floating Point Operations counts and runtimes

### 4.3.4 Effect of relaxation factor in SOR

# 5. Conclusions

# 6. References

**1. Executive Summary**

# This report modelled and solved the steady-state heat distribution in a custom electronic component. This component experiences temperatures range from 40°C to 70°C. The interior section of this component has a maximum temperature of 65.5°C and a minimum temperature of 43.3°C.

# This custom electronic component has a sensitive point at (0.03, 0.03). This point will remain within the functionable 50°C to 55°C range if the ambient temperature the component is exposed to remains within a -30°C and 55°C range.

# Five different storage methods were used when generating and solving this mathematical model. The recommended storage method is ….

# 2. Introduction

## 2.1 Purpose of the report

The purpose of this report is to investigate the steady-state heat distribution in a newly designed custom electronic component and provide its performance specifications. A mathematical model for the heat distribution in the component has been developed and solved using numerical strategies. This report also investigates the efficiencies of data storage methods by comparing their memory usage, run times and number of floating point operations.

The schematic of the electronic component is shown in Figure 1. The placement of the component within the electronic device results in different temperatures along the boundaries of the component. The boundary AB (red) maintains perfect thermal contact with another component, which has a known temperature of 70°C. The boundary CD (purple) also maintains perfect thermal contact with another component, which has a known temperature of 40°C. The boundary AED (green) is thermally insulated, and the boundary BC (blue) is exposed to the ambient air temperature.

Chart, line chart

Description automatically generated

*Figure X: Schematic of electronic component.*

## 2.2 Issues to be discussed and their significance

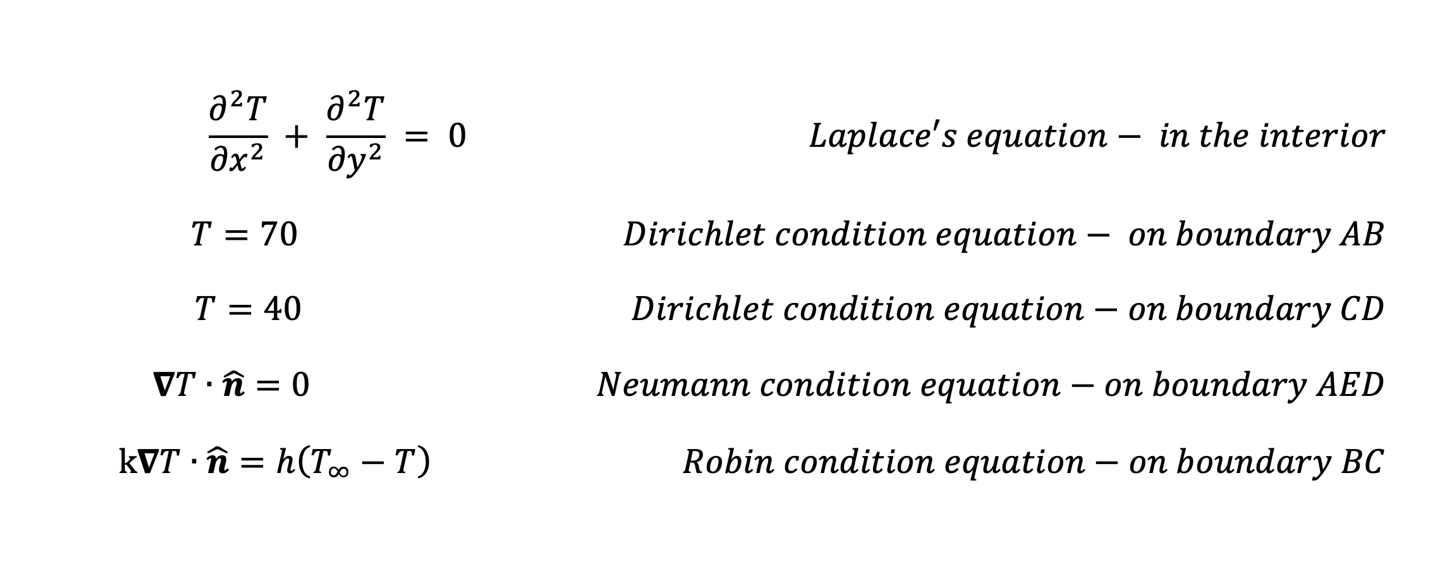
The custom electronic component is to be marketed globally, so the component needs to meet specific performance specifications for this to occur. The component must maintain a temperature between 50°C and 55°C at the point (0.03, 0.03). This point (orange) is shown on Figure X. The component will not function properly if the sensitive point is outside of this range of temperatures. This report will investigate the ambient temperatures that the electronic component can be exposed to and maintain a functional temperature at the sensitive point.

## 

# 3. Discussion

## 3.1 Mathematical Model

The mathematical model for steady-state heat distribution, in the absence of sources or sinks, is given by Laplace’s equation, . Let *)* represent the temperature of the electronic component at the point *()*. The mathematical model for the component is as follows:



### The thermal conductivity, *k* = 3 *Wm-1C-1*, and heat transfer coefficient, *h* = 20 *Wm-1C-1* are known and remain constant throughout the investigation. Initially, the ambient temperature assumed as *T*∞ = 20°*C*.

### The electrical component model is converted to a matrix problem by discretising the surface area domain and forming finite difference equations. The domain is discredited by converting the continuous surface area into discrete counterparts using a mesh of squares. A node is assigned to each corner of the mesh, and numbered as shown in Figure X. The nodes along boundaries AB and CD are under prescribed temperature (Dirichlet) boundary conditions, meaning the temperature at those nodes will remain constant, therefore those nodes are not assigned a number. All nodes along the AB boundary remain a constant 70°*C* and all nodes along the CB boundary remain a constant 40°*C*.

### 

*Figure X: Discretisation mesh with assigned nodes.*

### 3.1.1 Laplace’s equation in discretised form

### To solve the mathematical model of heat distribution, Laplace’s equation needs to be converted to a discretised form. The second order partial derivatives in Laplace’s equation are approximated using a second order central difference approximation, derived from the Taylor series expansion, as follows:

### Where the nodes are assigned as shown in Figure X.

### Diagram Description automatically generated

*Figure X: Explanation of node index notation*

### Substituting (1) and (2) into Laplace’s equation:

### In this case , since the mesh is square, the equation can be arranged to:

### Equation (3) is the discretised form of Laplace’s equation and is used when solving for each node.

### 3.1.2 Solving for nodes

### Chart, line chart Description automatically generated

*Figure X: Schematic defining interior nodes (blue) and unit normals for boundary nodes.*

### 3.1.2.1 Solving for Node 1

### Node 1 is under insulated boundary conditions, so is solved using the Neumann condition equation. This equation requires a unit normal, as shown in green in Figure X.

### Let node 1 = u1, node 2 = u2, node 7 = u7, and let uW be the ghost node west of node 1 and uS be the ghost node south of node 1, as shown in Figure X.

### 

*Figure X: Insulated boundary node 1 showing ghost nodes in grey*

### Applying Equation (3) to this node gives:

### Solve for uW and uS using the insulated (Neumann condition) boundary equation, .

### 

*Figure X: Unit normal for Node 1*

### Substituting (5) into (4).

### 3.1.2.2 Solving for ED boundary nodes

### The nodes along the ED boundary (2, 3, 4, 5 and 6) are under insulated boundary conditions, so are solved using the Neumann condition equation. This equation requires a unit normal, as shown in orange in Figure X.

### Using Node 3 as an example, let node 3 = u3, node 2 = u2, node 4 = u4, node 9 = u9, and let uS be the ghost node south of node 3, as shown in Figure X.

### A picture containing chart Description automatically generated

*Figure X: Example of insulated ED boundary node showing ghost node in grey*

### Applying Equation (3) to this node gives:

### Solve for uS using the insulated (Neumann condition) boundary equation, .

### A picture containing shape Description automatically generated

*Figure X: Unit normal for ED boundary node*

### Substituting (7) into (6).

### 3.1.2.3 Solving for AE boundary nodes

### The nodes along the AE boundary (7, 13, 19, 25 and 30) are under insulated boundary conditions, so are solved using the Neumann condition equation. This equation requires a unit normal, as shown in purple in Figure X.

### Using Node 13 as an example, let node 13 = u13, node 7 = u7, node 14 = u14 and node 19 = u19, and let uW be the ghost node west of node 13, as shown in Figure X.

### Timeline Description automatically generated

*Figure X: Example of insulated AE boundary node showing ghost node in grey*

### Applying Equation (3) to this node gives:

### Solve for uW using the insulated (Neumann condition) boundary equation, .

### A picture containing shape Description automatically generated

*Figure X: Unit normal for AE boundary node*

### Sub (9) into (8).

### 3.1.2.3 Solving for Node 29

### Node 29 is under convective boundary conditions, so is solved using the Robin condition equation. This equation requires a unit normal, as shown in blue in Figure X.

### Let node 29 = u29, node 28 = u28, node 23 = u23, and let uE be the ghost node east of node 29 and uN be the ghost node north of node 29, as shown in Figure X.

### Chart Description automatically generated

*Figure X: Convective boundary node 29 showing ghost nodes in grey*

### Applying Equation (3) to this node gives:

### Solve for uE and uN using the convective (Robin condition) boundary equation, .

### 

*Figure X: Unit normal for Node 29*

### Substituting (11) into (10) and substituting in constants:

### 3.1.2.4 Solving for Node 24

### Node 24 is under convective boundary conditions, so is solved using the Robin condition equation. This equation requires a unit normal, as shown in red in Figure X.

### Let node 24 = u24, node 23 = u23, node 18 = u18, and let uN be the ghost node north of node 33, as shown in Figure X. The node east of node 24 is known to be 40°C.

### Chart Description automatically generated

*Figure X: Convective boundary node 24 showing ghost node in grey*

### Applying Equation (3) to this node gives:

### Solve for uN using the convective (Robin condition) boundary equation, .

### Diagram Description automatically generated

*Figure X: Unit normal for Node 24*

### Substituting (13) into (12).

### Substituting in *a* and *b* and constants.

### 3.1.2.5 Solving for Node 33

### Node 33 is under convective boundary conditions, so is solved using the Robin condition equation. This equation requires a unit normal, as shown in yellow in Figure X.

### Let node 33 = u33, node 32 = u32, node 28 = u28, and let uE be the ghost node east of node 33, as shown in Figure X. The node north of node 33 is known to be 70°C.

### 

*Figure X: Convective boundary node 33 showing ghost node in grey*

### Applying Equation (3) to this node gives:

### Solve for uE using the convective (Robin condition) boundary equation, .

### Diagram Description automatically generated

*Figure X: Unit normal for Node 33*

### Substituting (15) into (14):

### Substituting in *a* and *b* and constants:

### 3.1.2.6 Solving for interior nodes

### The interior nodes, as shown in blue in Figure X, are not under any boundary conditions, so the interior nodes are solved using the standard Laplace’s equation.

### Using Node 15 as an example, let node 15 = u15, node 9 = u9, node 13 = u13, node 16 = u16, and node 21 = u21, as shown in Figure X.

### 

*Figure X: Example of interior node*

### Applying Equation (3) to this node gives:

### 3.2 Direct Methods

### 3.2.1 Cholesky Decomposition

### Cholesky decomposition is a direct method for solving linear systems. To have a unique solution when using the Cholesky decomposition method the matrix must be symmetric positive definite. Therefore, the matrix A that was formed to solve this model had to be symmetric positive definite.

### 3.2.2 Packed Storage

### Packed storage involves the non-zero portion of the matrix being packed by rows into a one-dimensional array.

### 3.2.3 Banded Storage

Banded storage is a storage method that compacts large banded systems by storing diagonals as rows in a band matrix. It is shown that the storage method is very efficient, with it being the second smallest file out of the storage methods that were tested, with a file size of 1848 bytes. This is because, unlike some other storage methods, it only stores a smaller section of the original matrix. Band storage only stores the diagonals of a banded system as the rows of the band matrix, and not any of the zeros of the top right or bottom left.

This method results in a matrix with the same number of columns as the original and rows equal to the bandwidth of the original matrix plus 1 if it is symmetric positive definite. The bandwidth of the matrix *A* is 6 and because it is an SPD, the band form will have 7 rows resulting is a decrease of 26 rows over the original matrix.

### 3.2.4 Sparse Storage

Sparse storage in MATLAB works by assigning a row and column position to each non-zero value stored in a matrix A, starting from the first row, and proceeding down. MATLAB’s symamd() function for a SPD matrix returns the permutation vector p such that A(p,p) tends to have a sparser Cholesky factor than A. What this meant for this task is that vectors A and b were permuted, the temperatures x were solved for, and then the permutation on x was undone to get the solution vector.

**3.3 Iterative Methods**

Iterative methods are another numerical technique that can be used to evaluate linear systems of the form

Where,

These methods are typically implemented for sufficiently large values of . Unlike the direct methods discussed in the previous section iterative methods do not need the coefficient matrix to be modified or factorised in anyway. Instead, iterative methods use a linear operator, an initial approximation of the vector and a measurement of the resulting error (the residual), to generate a sequence of vectors that approach the true value of . That is

,

The first three iterative methods that were used to evaluate the discretised steady state system of interest fall within the class of stationary iterative methods, with the last iterative method being apart of the Krylov subspace iterative methods. Both classes can be defined as

,

Where is the linear operator acting on the *k*-th vector of the sequence. The error of at the *k-*th for a given linear system that has an exact solution

,

For each iteration of the method, it can be shown that for some matrix the error evolves by in accordance with the following equation.

The matrix is known as the iterative matrix. An iterative method will convergent for any iterative matrix that has a spectral radius smaller than the unity, that is

Iterative methods also take advantage of the intrinsic property of splitting a matrix and expressing it as a either a sum or difference of matrices, such as

Where is an easily invertible matrix, and with the basis that has now been established, iterative methods can now be formally defined as

**,**

**, ,**

The stationary iterative methods that were used in for this project are based on the splitting

Where is diagonal portion of **,** is strictly the lower triangular portion and is strictly the upper triangular portion of the matrix .

### 3.3.1 Jacobi Method

The simplest of the iterative methods that were used is the Jacobi method. The iterative method is name after the German mathematician Carl Gustav Jacob Jacobi. Given that the linear system has a coefficient matrix that is square and diagonally dominant, the iterative method solves for each diagonal element and returns a new approximation of .

Which does conform to the formal definition of an iterative method, with and . Since this is an iterative method, it follows that the method will converge if , and while the error is still given by the same equation defined in the previous section, however, by looking at the following equation

And upon taking the norms of the equation

It can be seen that the smaller is the faster the Jacobi method will converge.

The matrix equation that defines the Jacobi method can also be represented as the following component-wise formula:

Using the component-wise formula a basic algorithm can be formed.

|  |
| --- |
| **Algorithm: Jacobi Method.** |
| **Input: TOL (tolerance), Maximum of iterations M**  **Output:** approximation of the vector **x** and if the method has converged or not  **While**    **For** to **do**    **For** to **do**    **End**    **End**      **End** |

By looking at the algorithm above it can be seen that the inner two loops dictate the overall efficiency of the algorithm as they depend on the dimensions of the matrix which can be much larger than the number of maximum iterations. The algorithm might also converge before the maximum iterations. On line 9 there are two floating point operations (FLOPS), there is also one FLOP in line 11. This leads to:

### This how many floating-point operations are done for each iteration of the two inner for loops. From this summation it is easy to see that the Jacobi method has a scalability and time complexity of and since iterative methods don’t need to factorise or modify the coefficient matrix the Jacobi method also has a storage scalability as .

### 3.3.2 Gauss-Siedel Method

### 3.3.3 Successive Over-Relaxation Method

### 3.3.4 Conjugate Gradient Method

### 3.3.5 Compressed Sparse Row Storage

**3.4 Node Re-orderings**

# 4. Results

## 4.1 Heat Distribution

## 

# The steady-state heat distribution in the custom electronic component is shown in Figure X. The boundary of the component experiences a minimum temperature of 40°C and a maximum temperature of 70°C. When the ambient temperature is 20°C, the interior of the component experiences a minimum temperature of 43.3°C and a minimum temperature of 65.5°C.

*Figure X: Steady-state heat distribution in custom electronic component*

### 4.2 Ambient Temperature Investigation

### 

*Figure X: Effect of Ambient Temperature of Component at (0.03, 0.03)*

The custom electronic component has a sensitive point at (0.03, 0.03) that must remain between 50°C and 55°C. To ensure the electronic component is functional globally, it was investigated what ambient temperature is allowable for the point (0.03, 0.03) to remain functional, as shown in Figure X. The component can experience temperatures from as low as -30°C to as high as 55°C. This covers most, if not all, of the world in terms of daily average temperatures. The component will therefore be functional globally and should be marketed as such.

**4.3 Method Performance**

The performance of each storage method was compared using memory, as shown in Figure X, floating point operation, as shown in Figure X, and run time, as shown in Figure X.

The runtime to solve a linear system using band storage was around third fastest out of the eight methods tested. Taking an average of 20 trials, the time it took to solve the linear system was 0.0003 seconds. It can be this fast because there are less rows to perform the Cholesky factorization, backward substitution, and forward substitution on. Band storage also has the least number of flops out of all the methods tested and has a fill-in percentage of 0.1915.

Using the sparse method, it took an average of 0.0156 seconds to run, had a total of 5207 floating point operations, and the resulting sparse A matrix took up 2528 bytes. In addition, while the bandwidth of the sparse A matrix was 6, the bandwidth of the permuted matrix was 29, and that the fill-in of the permuted matrix was -1.

### 

### Chart, bar chart Description automatically generated

*Figure X: Amount of memory used by each storage type*

### 

*Figure X: Number of floating-point operations for each storage type*

### 

*Figure X: Run time for each storage type*

# 5. Conclusions

# This report discussed the steady-state heat distribution in a custom electronic component. The component can be exposed and function in most global temperatures. The recommended storage method is ….

# 6. References

# Farquhar, M. (2022). *MXB226 Computational Methods 1: Direct methods for linear systems* [Chapter notes]. Blackboard.

# <https://blackboard.qut.edu.au/bbcswebdav/pid-10046319-dt-content-rid-58090098_1/courses/MXB226_22se2/Chapter1.pdf>

# Farquhar, M. (2022). *MXB226 Computational Methods 1: Data structures and algorithms for structured linear systems* [Chapter notes]. Blackboard. <https://blackboard.qut.edu.au/bbcswebdav/pid-10046321-dt-content-rid-58090099_1/courses/MXB226_22se2/Chapter2.pdf>

# Farquhar, M. (2022). *MXB226 Computational Methods 1: Iterative methods for linear systems* [Chapter notes]. Blackboard.

# <https://blackboard.qut.edu.au/bbcswebdav/pid-10046322-dt-content-rid-58116101_1/courses/MXB226_22se2/Chapter3_2022.pdf>

# Buttle, N. (2022). *MXB226 Computational Methods 1: Prac 5 Visualisation* [Practical notes]. Blackboard.

# <https://blackboard.qut.edu.au/bbcswebdav/pid-9997685-dt-content-rid-59395030_1/courses/MXB226_22se2/MXB226%20Prac%205.pdf>