The cooling of a mono-crystalline bar

Patricio Dias, M.J.; Mattheij, R.M.M.; ten Thije Boonkkamp, J.H.M.

Published: 01/01/2005

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):
The Cooling
of a Mono-Crystalline Bar

Miguel Patrício    Robert M.M. Mattheij
J.H.M. ten Thije Boonkkamp
## Contents

1 Introduction 1

2 The Poisson Problem 1
   2.1 Formulation of the problem 1  
   2.2 Pseudo-Unsteady approach 3

3 Numerical Solution 5
   3.1 Some notes on the discretization 5  
   3.2 An approach 6

4 Consistency and Stability 10
   4.1 Consistency 10  
   4.2 Stability 12

5 Numerical results 15

6 The Cooling of a Mono-Crystalline Bar 18
   6.1 Introduction 18  
   6.2 Formulation of the problem 18  
   6.3 Pseudo-Unsteady approach 19

7 Numerical Solution 20

8 Numerical results 24
1 Introduction

The Poisson equation arises in several domains of Science, modelling phenomena from Physics and Chemistry. In many applications it is convenient to use this equation in cylindrical coordinates, instead of the usual cartesian coordinates. Also in some cases the analytical solution of the Poisson equation associated to boundary conditions may be represented by a series of functions. However, generally speaking, this series present a slow convergence.

The goal of this work is to determine the numerical solution of the Poisson equation, associated to some boundary conditions and defined in a system of cylindrical coordinates. We present this problem in Section 2, where we also refer to the ADI method which allows us to obtain a related numerical scheme.

In Section 3 we analyse some aspects related to the discretizations we will perform and include the necessary equations that constitute an algorithm for obtaining the desired numerical solution.

In this context, it is also important to study some properties essential to obtaining an efficient numerical solution. This is done in Section 4.

Finally, in Section 5 we include some numerical results that illustrate the application of the referred algorithm to the problem presented.

The final part of this work, Sections 6 to 8, is dedicated to the presentation of a problem from practice, the Cooling of a Mono-Crystalline Bar. We aim to obtain the numerical solution of this problem. Our approach was similar to the one we presented in the previous sections for the Poisson Problem.

2 The Poisson Problem

2.1 Formulation of the problem

Let us consider the Poisson problem in cylindrical coordinates \((r, \theta, z)\)

\[
\nabla^2 u = f(r, z),
\]

where

\[
f(r, z) = \begin{cases} 
-\frac{1}{r}e^{-z} \sin(r) & r \neq 0 \\
0 & r = 0 
\end{cases}.
\]

The Laplace operator is given by

\[
\nabla^2 u = \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial u}{\partial r} \right) + \frac{1}{r^2} \frac{\partial u}{\partial \theta^2} + \frac{\partial^2 u}{\partial z^2},
\]
whilst the domain of interest will be

\[ \Gamma = \{(r, \theta, z) : 0 \leq r \leq \frac{\pi}{2}, 0 \leq \theta \leq 2\pi, 0 \leq z \leq L\}. \]

We are dealing with a specific problem — which typically arises from Physics — in which cylindrical symmetry is verified, and therefore the second term in (3) vanishes, allowing us to write

\[ \nabla^2 u = \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial u}{\partial r} \right) + \frac{\partial^2 u}{\partial z^2}. \]  

Moreover, the same reason makes it unnecessary to consider the whole domain \( \Gamma \). We may instead work only with \( \overline{\Omega} \), which corresponds, for a fixed value of the angle \( \theta \), to one half of a rectangular section of the cylinder \( \Gamma \), and is given by

\[ \{(r, z) : 0 \leq r \leq \frac{\pi}{2}, 0 \leq z \leq L\}. \]

On \( \overline{\Omega} \) we define the problem given by equation (1) and the following boundary conditions: see Figure 1-

\[ u(r, 0) = \cos(r); \quad 0 \leq r \leq \frac{\pi}{2} \]
\[ u(r, L) = e^{-L} \cos(r); \quad 0 \leq r \leq \frac{\pi}{2} \]
\[ u\left(\frac{\pi}{2}, z\right) = 0; \quad 0 < z < L \]

(5)

to which we add the symmetry condition at \( r = 0 \)

\[ \frac{\partial u}{\partial r}(0, z) = 0; \quad 0 < z < L. \]  

(6)

The solution of the problem (1)-(5)-(6), which is represented in Figure 2, is known "a priori", namely

\[ u(r, z) = e^{-z} \cos(r), \quad (r, z) \in \overline{\Omega}. \]  

(7)
2.2 Pseudo-Unsteady approach

For the computation of the solutions of many stationary problems such as the one described above, one may consider the solution as the result of a temporary evolution of some transient process. We shall use this to solve the previous problem, by considering instead

\[
\frac{\partial u}{\partial t} = \nabla^2 u - f(r, z); \quad (r, z) \in \Omega \\
u(r, 0, t) = \cos(r); \quad 0 \leq r \leq \frac{\pi}{2} \\
u(r, L, t) = e^{-L} \cos(r); \quad 0 \leq r \leq \frac{\pi}{2} \\
\frac{\partial u}{\partial r}(0, z, t) = 0; \quad 0 < z < L \\
u(\frac{\pi}{2}, z, t) = 0; \quad 0 < z < L \\
u(r, z, 0) = u^0(r, z)
\]

where the initial condition \( u^0 \) is an initial guess for the final solution.

We shall then work out a time-dependent numerical method to deal with (8), which will be said to be a pseudo-unsteady method, in the sense that its numerical solution in the limit as \( t \to \infty \) does not actually depend on \( t \).

Now, in order to compute such a numerical solution, we cover \( \overline{\Omega} \) with a rectangular grid of grid size \( \Delta r \) in the \( r \)-direction and \( \Delta z \) in the \( z \)-direction, with \( M + 1 \) and \( N + 1 \) lines, respectively. Including the boundary points, we will then have exactly \((M + 1) \times (N + 1)\) grid points \((r_j, z_k)\), where we
Define
\[ r_j = (j - 1)\Delta r, \quad j = 1, \ldots, M + 1; \]
\[ z_k = (k - 1)\Delta z, \quad k = 1, \ldots, N + 1. \]

We denote the approximation of \( u(r_j, z_k, t) \) by \( u_{j,k}(t) \). By ordering the points lexicographically we define the time dependent solution vector as
\[
[u_{1,2}(t), \ldots, u_{M,2}(t)|u_{1,3}(t), \ldots, u_{M,3}(t)| \ldots |u_{1,N}(t), \ldots, u_{M,N}(t)]^T. \tag{9}
\]
This leads via the method of lines to the ODE system
\[
\frac{du}{dt} = \tilde{\mathbf{A}}u + \tilde{f},
\]
where the block matrix \( \tilde{\mathbf{A}} \) is a sparse matrix with 5 non-zero diagonals and \( \tilde{f} \) contains the source term and the contributions from the boundary conditions.

To prevent us from being subjected to the strong stability restrictions that would occur if we applied an explicit method and to avoid having to deal with the complexity (in terms of the number or elementary operations) associated to the usage of an implicit method, we will use the ADI method instead. This alternative still combines low complexity with good stability properties.

The idea is to split the matrix \( \tilde{\mathbf{A}} \) into a matrix \( \mathbf{A}_r \) and a matrix \( \mathbf{A}_z \), derived from discretising respectively the first and second terms in (4), so that \( \tilde{\mathbf{A}} = \mathbf{A}_r + \mathbf{A}_z \). The scheme we obtain reads
\[
\begin{align*}
\mathbf{u}^{(l)} - \frac{1}{2}\Delta t\mathbf{A}_r\mathbf{u}^{(l)} &= \mathbf{u}^{(n)} + \frac{1}{2}\Delta t\mathbf{A}_z\mathbf{u}^{(n)} + \frac{1}{2}\Delta t\tilde{f}; \\
\mathbf{u}^{(n+1)} - \frac{1}{2}\Delta t\mathbf{A}_z\mathbf{u}^{(n+1)} &= \mathbf{u}^{(l)} + \frac{1}{2}\Delta t\mathbf{A}_r\mathbf{u}^{(l)} + \frac{1}{2}\Delta t\tilde{f},
\end{align*}
\tag{10,11}
\]
where \( \Delta t \) is the time step and we denote by \( u_{j,k}^{n} \) the approximation of \( u(r_j, z_k, n\Delta t) \), for \( n = 0, 1, 2, \ldots \).

The “implicit” direction is used for the discretisation of the \( r \)-step first and then for the discretisation of the \( z \)-step. Indeed, we need to perform two steps per ADI iteration and here \( l \) represents the intermediate step \( n + \frac{1}{2} \) in between time steps \( n \) and \( n + 1 \). Let us now rewrite our scheme as
\[
\begin{align*}
(I - \frac{1}{2}\Delta t\mathbf{A}_r)\mathbf{u}^{(l)} &= (I + \frac{1}{2}\Delta t\mathbf{A}_z)\mathbf{u}^{(n)} + \frac{1}{2}\Delta t\tilde{f}; \\
(I - \frac{1}{2}\Delta t\mathbf{A}_z)\mathbf{u}^{(n+1)} &= (I + \frac{1}{2}\Delta t\mathbf{A}_r)\mathbf{u}^{(l)} + \frac{1}{2}\Delta t\tilde{f}.
\end{align*}
\tag{12,13}
\]
Since both \((I - \frac{1}{2\tau}A_r)\) and \((I - \frac{1}{2\tau}A_z)\) are tridiagonal, each of these matrices allows for a simple and cheap LU-decomposition. On the other hand, however, we should note that these matrices have each a dimension of \([M \times (N-1)] \times [M \times (N-1)]\), which would make it impossible to manipulate them in a computer program when taking fine grids, if we were to deal with them directly. Fortunately, it is possible to bypass this problem, and we will later show how to do so.

3 Numerical Solution

3.1 Some notes on the discretization

There are some details we should make clear before getting our hands on the actual computations. First of all, generally speaking, when we want to approximate the second order derivative term \(\frac{d}{dx}(a(x)\frac{du}{dx})\) we may do so the following way:

\[
\frac{d}{dx}(a(x)\frac{du}{dx}) = \frac{1}{h}[a(x + \frac{h}{2})(u(x + h) - u(x)) - a(x - \frac{h}{2})(u(x) - u(x - h))]
\]

\[
= \frac{1}{h^2}[a(x + h/2)u(x + h) - (a(x + h/2) + a(x - h/2))u(x)]
\]

We may now apply this to our context in order to discretize the first term in (4):

\[
\frac{1}{r} \frac{\partial}{\partial r}(r \frac{\partial u}{\partial r}) = \frac{1}{\Delta r^2} [\alpha_{j} u_{j-1,k} - 2u_{j,k} + \beta_{j} u_{j+1,k}],
\]

where for \(j = 2, \ldots, M + 1,\)

\[
\alpha_{j} := \frac{r_{j} - \frac{1}{2} \Delta r}{r_{j}} \text{ and } \beta_{j} := \frac{r_{j} + \frac{1}{2} \Delta r}{r_{j}}.
\]

As a second remark, we note that extra care must be taken when \(r = 0.\) In this case, we will expand \(u(r)\) around \(r = 0,\) supressing any further dependence on other variables, to obtain
\[ u(r) = u(0) + \frac{1}{2}r^{2}\frac{d^{2}u}{dr^{2}}(0) + O(r^{3}), \quad (15) \]

where we use the fact that \( \frac{du}{dr} = 0 \). By inserting this expansion into

\[ \nabla_{r}^{2}u = \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial u}{\partial r} \right) \]

we find

\[ \nabla_{r}^{2}u = \frac{1}{r} \frac{d}{dr} \left( r \frac{d u}{d r} \right) = 2\frac{d^{2}u}{dr^{2}}(0) + O(r). \]

Once again using (15) at \( r = \Delta r \) we can derive a difference approximation for \( \frac{d^{2}u}{dr^{2}}(0) \) and subsequently we find the \( O(r) \) approximation

\[ \nabla_{r}^{2}u = \frac{4}{\Delta r^{2}}(u(\Delta r) - u(0)). \]

As a final note, we can naturally prescribe the boundary conditions:

\[ z = 0 \ (k = 1); \quad u_{j,1} = \cos(r_{j}); \]
\[ z = L \ (k = N + 1); \quad u_{j,N+1} = e^{-L} \cos(r_{j}); \]
\[ r = \frac{\pi}{2} \ (j = M + 1); \quad u_{M+1,k} = 0; \quad (17) \]

### 3.2 An approach

Let us now analyse the steps of the \textit{ADI} method. Firstly, we take only the \( r \)-derivative implicitly, as in (10). Hence, by taking central differences in the \( z \)-direction and using (14) we get

\[ u_{j,k}^{(l)} - \frac{\Delta t}{2\Delta z^{2}} [u_{j,k-1} - 2u_{j,k} + u_{j,k+1}]^{(l)} = u_{j,k}^{(n)} + \frac{\Delta t}{2} \sin(r_{j}) - e^{-z_{k}}, \quad (18) \]

for \( j = 2, \ldots, M \) and \( k = 2, \ldots, N \). The approximation described in (16) allows us to obtain the equation for \( j = 1 \) and \( k = 2, \ldots, N \):

\[ u_{1,k}^{(l)} - \frac{\Delta t}{2\Delta z^{2}} [-4u_{1,k} + 4u_{2,k}]^{(l)} = u_{1,k}^{(n)} + \frac{\Delta t}{2} e^{-z_{k}}. \]

\[ (19) \]
We now have to solve (12), where $A_r$ and $A_z$ are the block matrices given by

$$A_r = \frac{1}{\Delta r^2} \begin{pmatrix}
A & & & \\
& A & & \\
& & A & \\
& & & A
\end{pmatrix}$$

and

$$A_z = \frac{1}{\Delta z^2} \begin{pmatrix}
-2 & 1 & & & \\
& -2 & 1 & & \\
& 1 & -2 & 1 & & \\
& & 1 & -2 & 1 & \\
& & & 1 & -2 & \\
& & & & 1 & -2
\end{pmatrix}.$$
As we already mentioned, working with matrices so large would pose a problem. So, instead of working with the vector of unknowns

\[ [u^{(l)}_{1,2}, \ldots, u^{(l)}_{M,2}, u^{(l)}_{1,3}, \ldots, u^{(l)}_{M,3}, \ldots, u^{(l)}_{1,N}, \ldots, u^{(l)}_{M,N}]^{T}, \]

we will solve the system line by line. Hence, for each \( k = 2, \ldots, N \), we will deal with a linear system of the form

\[
(I - \frac{1}{2} \frac{\Delta t}{\Delta r^2} A) \begin{bmatrix} u^{(l)}_{1,k} \\ u^{(l)}_{2,k} \\ \vdots \\ u^{(l)}_{M,k} \end{bmatrix} = b^{(n)}_j + \vec{f}_k,
\]

where \( b^{(n)}_k \) is the \( M \times 1 \) vector, dependent on \( k \) and \( n \), given by

\[
b^{(n)}_k = \begin{bmatrix} u^{(n)}_{1,k} \\ u^{(n)}_{2,k} \\ \vdots \\ u^{(n)}_{M,k} \end{bmatrix} + \frac{1}{2} \frac{\Delta t}{\Delta z^2} \left( \begin{bmatrix} u^{(l)}_{1,k-1} \\ u^{(l)}_{2,k-1} \\ \vdots \\ u^{(l)}_{M,k-1} \end{bmatrix} - 2 \begin{bmatrix} u^{(l)}_{1,k} \\ u^{(l)}_{2,k} \\ \vdots \\ u^{(l)}_{M,k} \end{bmatrix} + \begin{bmatrix} u^{(l)}_{1,k+1} \\ u^{(l)}_{2,k+1} \\ \vdots \\ u^{(l)}_{M,k+1} \end{bmatrix} \right),
\]

and

\[
\vec{f}_k = \frac{1}{2} \Delta t e^{-z_k} \begin{bmatrix} \frac{\sin(r_2)}{r_2} \\ \vdots \\ \frac{\sin(r_M)}{r_M} \end{bmatrix}.
\]

This will allow us to perform the first half step of each iteration and compute, for each \( k = 2, \ldots, N \), the vector

\[
\begin{bmatrix} u^{(l)}_{1,k} \\ u^{(l)}_{2,k} \\ \vdots \\ u^{(l)}_{M,k} \end{bmatrix}.
\]
which is no more than a split part of the solution vector (21). Finally we take the $z$-derivative implicitly, as in (11). This time we will solve the system column by column, which can be done by splitting (21) into the vectors (for $j = 1, \ldots, M$)

$$
\begin{bmatrix}
  u_{j,2}^{(n+1)} \\
  u_{j,3}^{(n+1)} \\
  \vdots \\
  u_{j,N}^{(n+1)}
\end{bmatrix}
$$

We then write

$$
u_{j,k}^{(n+1)} = \frac{1}{2} \Delta t \frac{\Delta z^2}{2} [u_{j,k-1} - 2u_{j,k} + u_{j,k+1}]^{(n+1)}
= u_{j,k}^{(l)} + \frac{1}{2} \Delta t \frac{\Delta z^2}{2} [\alpha_j u_{j-1,k} - 2u_{j,k} + \beta_j u_{j+1,k}]^{(l)} + \frac{1}{2} \Delta t \frac{\sin(r_j)}{r_j} e^{-z_k}
$$

(23)

for $j = 2, \ldots, M$ and $k = 2, \ldots, N$. Proceeding like before we get, for $j = 1$,

$$
u_{1,k}^{(n+1)} = \frac{1}{2} \Delta t \frac{\Delta z^2}{2} [u_{1,k-1} - 2u_{1,k} + u_{1,k+1}]^{(n+1)}
= u_{1,k}^{(l)} + \frac{1}{2} \Delta t \frac{\Delta z^2}{2} [-4u_{1,k} + 4u_{2,k}]^{(l)} + \frac{1}{2} \Delta t e^{-z_k}.
$$

(24)

Thus, for $j = 1, \ldots, M$, we will have to solve the system

$$
(I - \frac{1}{2} \Delta t \frac{\Delta z^2}{2} C) \begin{bmatrix}
  u_{j,2}^{(n+1)} \\
  u_{j,3}^{(n+1)} \\
  \vdots \\
  u_{j,N}^{(n+1)}
\end{bmatrix} = d_j^{(l)} + \tilde{g}_k,
$$

(25)

where $C$ is a $(N-1) \times (N-1)$ matrix and $d_j^{(l)}$ an $(N-1) \times 1$ vector. These are given by

$$
C = \begin{bmatrix}
  -2 & 1 & & & & \\
  1 & -2 & 1 & & & \\
  & \ddots & \ddots & \ddots & \ddots & \\
  & & 1 & -2 & 1 & \\
  & & & 1 & -2 & 1
\end{bmatrix}
$$
and

\[
d_j^{(l)} = \begin{bmatrix}
u_{j,2} \\ u_{j,3} \\ \vdots \\ u_{j,N}
\end{bmatrix}^{(l)} + \frac{1}{2} \frac{\Delta t}{\Delta \tau^2} \begin{pmatrix}
\alpha_j & u_{j-1,2} & u_{j-1,3} & \cdots & u_{j-1,N} \\
u_{j,2} & u_{j,3} & \cdots & \cdots & u_{j,N} \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
u_{j,N} & u_{j+1,2} & u_{j+1,3} & \cdots & u_{j+1,N}
\end{pmatrix}^{(l)},
\]

for \( j = 2, \ldots, M \). Moreover, when we take \( j = 1 \),

\[
d_1^{(l)} = \begin{bmatrix}
u_{1,2} \\ u_{1,3} \\ \vdots \\ u_{1,N}
\end{bmatrix}^{(l)} + \frac{1}{2} \frac{\Delta t}{\Delta \tau^2} \begin{pmatrix}
-4 & u_{1,2} & u_{1,3} & \cdots & u_{1,N} \\
u_{2,2} & u_{2,3} & \cdots & \cdots & u_{2,N} \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
u_{1,N} & u_{2,2} & u_{2,3} & \cdots & u_{2,N}
\end{pmatrix}^{(l)}.
\]

On the other hand,

\[
\tilde{g}_j = \frac{1}{2} \frac{\Delta t \sin(r_j)}{r_j} \begin{bmatrix}
e^{-z_2} \\ e^{-z_3} \\ \vdots \\ e^{-z_{N-1}} \\ e^{-z_N}
\end{bmatrix} + \frac{1}{2} \frac{\Delta t}{\Delta \tau^2} \begin{bmatrix}
\cos(r_j) \\ 0 \\ \vdots \\ 0 \\ e^{-L \cos(r_j)}
\end{bmatrix}.
\]

To summarize, we can establish an algorithm to deal with (10) and (11) which basically consists of (22) and (25).

4 Consistency and Stability

4.1 Consistency

We will now study the consistency of the difference scheme (12)-(13). For the sake of simplicity, let’s rewrite (18) and (23) respectively as

\[
(1 - \frac{1}{2} \rho \tau L^2) u^{(l)} + (1 + \frac{1}{2} \rho \tau L^2) u^{(n)} + \frac{\Delta t \sin(r_j)}{r_j} e^{-z_k} = (26)
\]

and

\[
(1 - \frac{1}{2} \rho \tau L^2) u^{(n+1)} + (1 + \frac{1}{2} \rho \tau L^2) u^{(l)} + \frac{\Delta t \sin(r_j)}{r_j} e^{-z_k}, \quad (27)
\]
where we define

\[
\begin{align*}
\rho_r &= \frac{\Delta t}{\Delta r^2}; \\
\rho_z &= \frac{\Delta t}{\Delta z^2}; \\
L_r^{(n)}[u_{j,k}] &= \alpha_j u_{j-1,k}^{(n)} - 2u_{j,k}^{(n)} + \beta_j u_{j+1,k}^{(n)}; \\
L_z^{(n)}[u_{j,k}] &= u_{j,k-1}^{(n)} - 2u_{j,k}^{(n)} + u_{j,k+1}^{(n)}. 
\end{align*}
\]

If we multiply both sides of (26) by \((1 + \frac{1}{2}\rho_r L_r^2)\), we get

\[
(1 + \frac{1}{2}\rho_r L_r^2)(1 - \frac{1}{2}\rho_r L_r^2)u_{j,k}^{(l)} = (1 + \frac{1}{2}\rho_r L_r^2)(1 + \frac{1}{2}\rho_z L_z^2)u_{j,k}^{(n)} \\
+ (1 + \frac{1}{2}\rho_r L_r^2)\frac{\Delta t \sin(r_j)}{r_j} e^{-z_k}
\]

We also multiply both sides of (27), this time by \((1 - \frac{1}{2}\rho_r L_r^2)\):

\[
(1 - \frac{1}{2}\rho_r L_r^2)(1 - \frac{1}{2}\rho_z L_z^2)u_{j,k}^{(n+1)} = (1 - \frac{1}{2}\rho_r L_r^2)(1 + \frac{1}{2}\rho_r L_r^2)u_{j,k}^{(l)} \\
+ (1 - \frac{1}{2}\rho_r L_r^2)\frac{\Delta t \sin(r_j)}{r_j} e^{-z_k}.
\]

The last two expression yield

\[
(1 - \frac{1}{2}\rho_r L_r^2)(1 - \frac{1}{2}\rho_z L_z^2)u_{j,k}^{(n+1)} = (1 + \frac{1}{2}\rho_r L_r^2)(1 + \frac{1}{2}\rho_z L_z^2)u_{j,k}^{(n)} \\
+ \Delta t \frac{\sin(r_j)}{r_j} e^{-z_k}.
\]

We can now divide this last equality by \(\Delta t\) to obtain the equivalent expression, after some simplifications

\[
\frac{u_{j,k}^{n+1} - u_{j,k}^{n}}{\Delta t} = \frac{L_r^2}{2\Delta r^2}(u_{j,k}^{n+1} + u_{j,k}^{n}) + \frac{L_z^2}{2\Delta z^2}(u_{j,k}^{n+1} + u_{j,k}^{n}) \\
- \frac{\Delta t}{4} \frac{L_r^2 L_z^2}{\Delta r^2 \Delta z^2}(u_{j,k}^{n+1} + u_{j,k}^{n}) + \frac{\sin(r_j)}{r_j} e^{-z_k}.
\]

Now all that remains to be done is to use the Taylor expansion on this last expression and show that
\[ -\frac{\Delta t}{4} \frac{L_x^2 L_z^2}{r^2} \frac{\partial^2}{\partial z^2} (u_{j,k}^{n+1} - u_{j,k}^n) = \Delta t [O(\Delta t) + O(\Delta r^2) + O(\Delta z^2)]. \]

Hence the two step difference scheme (12)-(13) is second order accurate in \( \Delta t, \Delta r \) and \( \Delta z \), for \( j = 2, \ldots, M \). We note that the reasoning we presented is also valid for \( j = 1 \), except for the \( O(r) \) approximation we got in (16). This means that overall we will only have first order accuracy in \( \Delta r \).

### 4.2 Stability

Let us start the study of the stability of the scheme (12)-(13) by looking into \( \| (I - \frac{1}{2} \Delta t A_r)^{-1} (I + \Delta t A_z) \| \), where we choose the 2-norm. First of all, we claim that the square \( M \times M \) matrix \( A \), given by (20), is diagonalizable. Indeed, it may be written as

\[
A = D^{-1} S D,
\]

where \( D \) is the diagonal matrix given by

\[
\begin{bmatrix}
1 & \sqrt{8} & \sqrt{16} & \ldots & \sqrt{8(M-1)} \\
\sqrt{8} & \sqrt{16} & \ldots & \sqrt{8(M-1)} \\
\sqrt{16} & \ldots & \ldots & \ldots & \ldots \\
\sqrt{8(M-1)} & \ldots & \ldots & \ldots & \ldots \\
\end{bmatrix}
\]

and \( S \) is the symmetric matrix

\[
\begin{bmatrix}
-4 & \sqrt{2} & \ldots & \ldots & \ldots \\
\sqrt{2} & -2 & \frac{3}{4} \sqrt{2} & \ldots & \ldots \\
\ldots & \frac{3}{4} \sqrt{2} & -2 & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \frac{2(M-2)+1}{2(M-1)} \sqrt{\frac{M-1}{M-2}} & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & -2 \\
\end{bmatrix}
\]

Then clearly \( A \) is diagonalizable. Moreover, the Gerschgorin Circle theorem assures us that its eigenvalues are non-positive, which will prove to be an important feature. However, for now let \( W \) be the invertible matrix such that

12
Let us build a larger matrix \( \tilde{W} \), composed of such transformation matrices:

\[
\tilde{W} = \begin{bmatrix}
W & & \\
& W & \\
& & \ddots \\
& & & W
\end{bmatrix},
\]

which we use to transform \((I - \frac{1}{2} \Delta t A_r)\) by doing

\[
\tilde{W}^{-1} (I - \frac{1}{2} \Delta t A_r) \tilde{W} = \begin{bmatrix}
\Lambda_r & & \\
& \Lambda_r & \\
& & \ddots \\
& & & \Lambda_r
\end{bmatrix},
\]

where \(\Lambda_r = I - \frac{1}{2} \Delta t A\). We now obtain

\[
\tilde{W}^{-1} (I - \frac{1}{2} \Delta t A_r)^{-1} \tilde{W} (I + \frac{1}{2} \Delta t A_z) \tilde{W} = \begin{bmatrix}
\Lambda_r^{-1} & & \\
& \Lambda_r^{-1} & \\
& & \ddots \\
& & & \Lambda_r^{-1}
\end{bmatrix} (I + \Delta t A_z)
\]

which follows from the simple structure of the matrix \(A_z\). Next, we can bring the split matrix \(A_z\) onto tridiagonal form, like \(A_r\), if we renumber the unknowns along vertical grid lines. This renumbering corresponds to a permutation of the rows and columns of \(A_z\), and we can formally define a tridiagonal matrix \(B_z\) by

\[
B_z = \tilde{P}^{-1} A_z \tilde{P},
\]

where \(\tilde{P}\) is the correspondent permutation matrix. Applying the same permutations to the matrix in (28), we obtain
\[
\tilde{P}^{-1}\tilde{W}^{-1}(I - \frac{1}{2}\Delta t A_r)^{-1}\tilde{W}\tilde{P}\tilde{P}^{-1}\tilde{W}^{-1}(I + \frac{1}{2}\Delta t A_z)\tilde{W}\tilde{P} = \tilde{P}^{-1} \begin{bmatrix}
\Lambda_r^{-1} & & \\
& \Lambda_r^{-1} & \\
& & \ddots \\
& & & \Lambda_r^{-1}
\end{bmatrix} \tilde{P}(I + \frac{1}{2}\Delta t B_z).
\]

(29)

It is simple to see then that \(\tilde{P}\) transforms the matrix with the \(\Lambda_r^{-1}\) blocks into another diagonal matrix, namely

\[
\begin{bmatrix}
(1 - \frac{1}{2}\Delta t v_1)^{-1}I & & \\
& (1 - \frac{1}{2}\Delta t v_2)^{-1}I & \\
& & \ddots \\
& & & (1 - \frac{1}{2}\Delta t v_M)^{-1}I
\end{bmatrix}.
\]

Since the 2-norm of a matrix does not change when pre- or postmultiplied by orthogonal (permutation) matrices, it is easy to see that

\[
\|(I - \frac{1}{2}\Delta t A_r)^{-1}(I + \frac{1}{2}\Delta t A_z)\|_2 = \max_{1\leq j\leq M, 1\leq k\leq N} \left| \frac{1 - 2\frac{\Delta t}{\Delta x^2} \sin^2\left(\frac{\pi}{2} z_k\right)}{1 - \frac{1}{2}\Delta t v_j} \right|.
\]

Finally, for the stability of our method, we must investigate the matrix \(K\) and its similar form \(L\), given respectively by

\[
K := (I - \frac{1}{2}\Delta t A_z)^{-1}(I + \frac{1}{2}\Delta t A_r)(I - \frac{1}{2}\Delta t A_r)^{-1}(I + \frac{1}{2}\Delta t A_z)
\]

and

\[
L := (I - \frac{1}{2}\Delta t A_z)K(I - \frac{1}{2}\Delta t A_z)^{-1}.
\]

Actually, we need to transform \(L\) first, like we did above in (29) and work with the matrix \(\tilde{L}\) say, then we easily obtain

\[
\|L\|_2 = \|\tilde{L}\|_2 = \max_{1\leq j\leq M, 1\leq k\leq N} \left| \frac{1 - 2\frac{\Delta t}{\Delta x^2} \sin^2\left(\frac{\pi}{2} z_k\right)}{1 - \frac{1}{2}\Delta t v_j} + \frac{1 + \frac{\Delta t}{\Delta x^2} \sin^2\left(\frac{\pi}{2} z_k\right)}{1 + \frac{\Delta t}{\Delta x^2} \sin^2\left(\frac{\pi}{2} z_k\right)} \right|,
\]

which is obviously bounded by 1, independently of the grid sizes (recall that the eigenvalues \(\{v_j\}_{j=1, \ldots, M}\) are non-positive). Although \(L\) may be a skew transformation of \(K\), this does not affect the stability as such.
5 Numerical results

In order to properly implement in Matlab the method described above, we are required to establish a stoppage condition, ie, we have to know just when we have performed a sufficient number of iterations. Recall that we are dealing with a pseudo-unsteady method which was designed for (8), when indeed we want to solve the time-independent problem (1)-(5)-(6). It is then natural to stop the iterative (convergent) process when there is virtually no time dependency, ie, when

$$\left\| \frac{du}{dt} \right\| \approx 0.$$  (30)

Besides taking the last consideration into account, we also need to define the initial guess function $u^0$ which appears in (8). We expect that a better initial choice (in the sense that it is closer to the actual solution) would allow a faster convergence.

Here we include three different initial functions, represented by $u_1^0$, $u_2^0$ and $u_3^0$, shown respectively in Figures 3, 4 and 5. For each case we compute the numerical solutions using the technique described in Section 3.

![Figure 3: Initial guess function $u_1^0(r, z)$.](image)

The first guess function arises from the fact that in this particular case we actually know the solution to our problem- given in (7) - which can be approximated by the function expressed by

$$u_1^0(r, z) = \max[(1 - z + \frac{z^2}{2} - \frac{z^3}{3!}) \times (1 - \frac{r^2}{2}), 0]$$

for small values of $z$ and which coincides with 0 elsewhere, except at the
boundaries, where for all our choices of starting functions the conditions
given at (17) are imposed.

For the previous guess function we used the fact that the exact solution is
known, which is something that cannot be expected to happen in a non-
academical problem. If also in this case we did not have that \textit{a priori}
knowledge, we could define the initial guess function \( u^0 \) which coincides
with 1 everywhere except at the boundaries.

Finally, the last initial guess function \( u^3 \) is obtained by taking 0 everywhere
except in the boundaries, where again the conditions given by (17) are held.
In the numerical results we will present, we take \( L = 10 \) and consider that
we attain the stoppage condition described by (30) when \( n \) is large enough
so that
\[ \frac{\| u^{n+1} - u^n \|_\infty}{\Delta t} < 10^{-3}. \]  

(31)

The results obtained when we choose \( u_0^1 \) are shown in Table 1.

<table>
<thead>
<tr>
<th>\Delta t = 10^{-2}</th>
<th>| u_\infty - u_e |<em>\infty \quad u_e - u_0^1 |</em>\infty</th>
<th>M = N = 20</th>
<th>M = N = 30</th>
<th>M = N = 40</th>
<th>M = N = 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>N. Iterations</td>
<td>42</td>
<td>1.606 \times 10^{-1}</td>
<td>1.889 \times 10^{-1}</td>
<td>1.738 \times 10^{-1}</td>
<td>2.019 \times 10^{-1}</td>
</tr>
<tr>
<td>N. Iterations</td>
<td>57</td>
<td>2.563 \times 10^{-3}</td>
<td>1.161 \times 10^{-3}</td>
<td>6.676 \times 10^{-4}</td>
<td>4.298 \times 10^{-4}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>\Delta t = 10^{-3}</th>
<th>| u_\infty - u_e |<em>\infty \quad u_e - u_0^1 |</em>\infty</th>
<th>M = N = 20</th>
<th>M = N = 30</th>
<th>M = N = 40</th>
<th>M = N = 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>N. Iterations</td>
<td>208</td>
<td>2.344 \times 10^{-3}</td>
<td>1.007 \times 10^{-3}</td>
<td>4.983 \times 10^{-4}</td>
<td>2.735 \times 10^{-4}</td>
</tr>
<tr>
<td>N. Iterations</td>
<td>207</td>
<td>2.338 \times 10^{-3}</td>
<td>1.005 \times 10^{-3}</td>
<td>4.940 \times 10^{-4}</td>
<td>2.715 \times 10^{-4}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>\Delta t = 10^{-4}</th>
<th>| u_\infty - u_e |<em>\infty \quad u_e - u_0^1 |</em>\infty</th>
<th>M = N = 20</th>
<th>M = N = 30</th>
<th>M = N = 40</th>
<th>M = N = 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>N. Iterations</td>
<td>2070</td>
<td>2.338 \times 10^{-3}</td>
<td>1.005 \times 10^{-3}</td>
<td>4.940 \times 10^{-4}</td>
<td>2.715 \times 10^{-4}</td>
</tr>
<tr>
<td>N. Iterations</td>
<td>2065</td>
<td>2.338 \times 10^{-3}</td>
<td>1.005 \times 10^{-3}</td>
<td>4.940 \times 10^{-4}</td>
<td>2.715 \times 10^{-4}</td>
</tr>
</tbody>
</table>

Table 1: Results obtained for \( u_1^0 \).

Here \( \| u_\infty - u_e \|_\infty \) represents the infinity norm of the difference between the discrete exact solution to our problem \( u_e \) and \( u_0^1 \), for different values of \( M \) and \( N \). Also, for the same range of values, and when \( \Delta t \) varies, we show the error, ie, the maximum norm of the difference between the numerical solution \( u_\infty \) computed by our method and the exact solution of the problem. Finally, we include the number of iterations needed to obtain (31).

In Figure 6 we show the numerical solution \( u_\infty \).

In figure 6 we show the numerical solution obtained for \( \Delta t = 10^{-2} \), when we take \( M = N = 20 \).
By now taking $u^0_2$ and $u^0_3$ as our initial guess functions, we get the results shown respectively in Table 2 and Table 3.

<table>
<thead>
<tr>
<th>$\Delta t = 10^{-2}$</th>
<th>$|u_\infty - u_0|_\infty$</th>
<th>$N. Iterations$</th>
<th>$M = N = 20$</th>
<th>$M = N = 30$</th>
<th>$M = N = 40$</th>
<th>$M = N = 50$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>9.999 x 10^{-1}</td>
<td>9.999 x 10^{-1}</td>
<td>9.999 x 10^{-1}</td>
<td>9.999 x 10^{-1}</td>
</tr>
<tr>
<td>$\Delta t = 10^{-3}$</td>
<td></td>
<td></td>
<td>2.564 x 10^{-3}</td>
<td>1.161 x 10^{-3}</td>
<td>6.676 x 10^{-4}</td>
<td>4.298 x 10^{-4}</td>
</tr>
<tr>
<td>$\Delta t = 10^{-4}$</td>
<td></td>
<td></td>
<td>2.691 x 10^{-3}</td>
<td>1.287 x 10^{-3}</td>
<td>7.844 x 10^{-4}</td>
<td>5.498 x 10^{-4}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\Delta t = 10^{-2}$</th>
<th>$|u_\infty - u_0|_\infty$</th>
<th>$N. Iterations$</th>
<th>$M = N = 20$</th>
<th>$M = N = 30$</th>
<th>$M = N = 40$</th>
<th>$M = N = 50$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>6.065 x 10^{-1}</td>
<td>7.165 x 10^{-1}</td>
<td>7.788 x 10^{-1}</td>
<td>8.187 x 10^{-1}</td>
</tr>
<tr>
<td>$\Delta t = 10^{-3}$</td>
<td></td>
<td></td>
<td>2.392 x 10^{-3}</td>
<td>1.042 x 10^{-3}</td>
<td>5.347 x 10^{-4}</td>
<td>3.041 x 10^{-4}</td>
</tr>
<tr>
<td>$\Delta t = 10^{-4}$</td>
<td></td>
<td></td>
<td>2.330 x 10^{-3}</td>
<td>1.002 x 10^{-3}</td>
<td>4.910 x 10^{-4}</td>
<td>2.687 x 10^{-4}</td>
</tr>
</tbody>
</table>

Table 2: Results obtained for $u^0_2$.

<table>
<thead>
<tr>
<th>$\Delta t = 10^{-2}$</th>
<th>$|u_\infty - u_0|_\infty$</th>
<th>$N. Iterations$</th>
<th>$M = N = 20$</th>
<th>$M = N = 30$</th>
<th>$M = N = 40$</th>
<th>$M = N = 50$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.326 x 10^{-3}</td>
<td>9.974 x 10^{-4}</td>
<td>4.858 x 10^{-4}</td>
<td>2.645 x 10^{-4}</td>
</tr>
<tr>
<td>$\Delta t = 10^{-3}$</td>
<td></td>
<td></td>
<td>2.326 x 10^{-3}</td>
<td>9.974 x 10^{-4}</td>
<td>4.858 x 10^{-4}</td>
<td>2.645 x 10^{-4}</td>
</tr>
<tr>
<td>$\Delta t = 10^{-4}$</td>
<td></td>
<td></td>
<td>2.326 x 10^{-3}</td>
<td>9.974 x 10^{-4}</td>
<td>4.858 x 10^{-4}</td>
<td>2.645 x 10^{-4}</td>
</tr>
</tbody>
</table>

Table 3: Results obtained for $u^0_3$.

Due to the values of the errors obtained, the graphical representations for the solutions obtained when taking $u^0_2$ or $u^0_3$ are similar to Figure 6.

From the comparison of these Tables we conclude that the errors obtained do not depend much on the choice we made for the initial value. However, the computational effort is smaller for $u^0_1$ and $u^0_3$, as would be expected, since they are closer to the actual solution than $u^0_2$. 

18
6 The Cooling of a Mono-Crystalline Bar

6.1 Introduction

We now look into the problem of the cooling of a mono-crystalline bar, taken from practice. We approach this problem in a similar way to what we did with the previous one in sections in Sections 2 to 5. Here we will merely present the model and the numerical results we obtained.

A common way to grow large silicon mono-crystals is to pull the crystal very slowly from a melt. The temperatures at which these processes take place are usually rather high. Silicon, for example, solidifies at 1693 K. This is why the crystal will lose most of its heat through radiation. At the same time, the radiation from the surface is small when compared to the conduction in the material. In order to investigate the typical temperature variation along the crystalline bar we consider the following model.

6.2 Formulation of the problem

Consider a long cylindrical bar defined by \( 0 \leq z \leq L \), \( 0 \leq r \leq a \), and \( 0 \leq \theta \leq 2\pi \), where \((r, \theta, z)\) are cylindrical coordinates. The bar is assumed to be long enough for a semi-infinite approximation. The bar is heated from one end \( z = 0 \) where the melting front is situated, whilst it is cooled by radiation from the other sides \( r = a \) - see Figure 7.

At these high temperatures, cooling by natural convection of the surrounding air is very small compared to radiation. The melting front will be considered planar, although in reality it is slightly curved. The mono-crystal is slowly pulled away with constant velocity \( V \) in positive \( z \)-direction. The heat flow in and out of the bar is in equilibrium so we have a steady state. Inside the bar (ie, in \( \Omega = \{0 < z < L, 0 < r < a\} \)) we have the stationary convective heat equation for the temperature \( T \):

\[
\rho C_p V \frac{\partial T}{\partial z} = \kappa \nabla^2 T,
\]  

(32)
where the Laplace operator is given by (4). This can be rewritten as

$$\frac{\partial T}{\partial z} = \gamma \nabla^2 T,$$

(33)

where $\gamma = \frac{\kappa}{\rho C_p V}$. Here $\rho$ and $C_p$ are nearly constant for the temperatures considered and represent, respectively, the crystal density and the specific heat. We have assumed that $\kappa$, which is in general the temperature dependent thermal conductivity (typically inversely proportional to the temperature), is a constant. To complete the problem formulation, we have the following boundary conditions to quantify the heating, radiation and the symmetry condition at the axis:

$$z = 0, \quad T = T_m;$$
$$z = L, \quad \frac{\partial T}{\partial z} = 0;$$
$$r = 0, \quad \frac{\partial T}{\partial r} = 0;$$
$$r = a, \quad \frac{\partial T}{\partial r} = \varphi(T^4 - T_0^4);$$

where $\varphi = -\frac{\sigma}{\kappa}$.

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>$2.33 \times 10^3$</th>
<th>$T_m$</th>
<th>$1.693 \times 10^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_p$</td>
<td>$9.22 \times 10^2$</td>
<td>$\epsilon$</td>
<td>$6.6 \times 10^{-1}$</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>$2.16 \times 10^1$</td>
<td>$T_0$</td>
<td>$3.0 \times 10^2$</td>
</tr>
<tr>
<td>$a$</td>
<td>$1.15 \times 10^{-2}$</td>
<td>$V$</td>
<td>$1.67 \times 10^{-5}$</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>$5.6704 \times 10^{-8}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Typical parameter values of silicon.

The temperature of the environment $T_0$ represents the heating by radiation from the surroundings back into the crystal. The universal constant $\sigma$ in the law of radiation is the constant of Stefan-Boltzmann. The material constant $\epsilon$ is called the emission coefficient or emissivity. It is equal to unity for black bodies and between 0 and 1 otherwise - Table 4. By symmetry, the problem is independent of $\theta$.

### 6.3 Pseudo-Unsteady approach

Our procedure in this case is analogous to the one described in Section 2; we consider the problem
\[
\frac{\partial T}{\partial t} = \gamma \nabla^2 T - \frac{\partial T}{\partial z}; \quad (r, z) \in \Omega
\]
\[
T(r, 0, t) = T_m; \quad 0 \leq r \leq \frac{a}{2}
\]
\[
\frac{\partial T}{\partial z}(r, L, t) = 0; \quad 0 < r < \frac{a}{2}
\]
\[
\frac{\partial T}{\partial z}(0, z, t) = 0; \quad 0 < z < L,
\]
\[
\frac{\partial T}{\partial z}\left(\frac{a}{2}, z, t\right) = \varphi(T^4 - T_0^4); \quad 0 < z < L
\]
\[
T(r, z, 0) = T^0(r, z)
\]

where the initial condition \(T^0\) is an initial guess for the exact solution. By applying the method of lines we obtain the ODE system

\[
\frac{dT}{dt} = \tilde{A}T + \tilde{f},
\]

where \(\tilde{f}\) is a non-linear vector function of \(T\) which arises from the contributions from the boundary conditions.

We split the matrix \(\tilde{A}\) into a matrix \(A_r\) and a matrix \(A_z\), derived from discretising respectively the \(r\) and the \(z\) terms in the first equation of (34), so that \(\tilde{A} = A_r + A_z\). The scheme we obtain reads

\[
(I - \frac{1}{2} \Delta t A_r)T^{(l)} = (I + \frac{1}{2} \Delta t A_z)T^{(n)} + \frac{1}{2} \Delta t \tilde{f}; \quad (35)
\]
\[
(I - \frac{1}{2} \Delta t A_z)T^{(n+1)} = (I + \frac{1}{2} \Delta t A_r)T^{(l)} + \frac{1}{2} \Delta t \tilde{f}. \quad (36)
\]

7 Numerical Solution

Like before, we will take (14) and (16) into account. The boundary conditions can be prescribed by

\[
\begin{align*}
    z = 0 & \quad (k = 1); \quad T_{j,1} = T_m; \\
    z = L & \quad (k = N + 1); \quad T_{j,N+2} = T_{j,N}; \\
    r = \frac{a}{2} & \quad (j = M + 1); \quad T_{M+2,k} = T_{M,k} + 2\varphi \Delta r(T^4_{M+1,k} - T^4_0).
\end{align*}
\]

By first taking the \(r\)-derivative implicitly as in (35) and central differences in the \(z\)-direction, we obtain

\[
\begin{aligned}
T^{(l)}_{j,k} & = T^{(n)}_{j,k} - \frac{1}{2} \Delta t \frac{\Delta r^2}{\Delta z^2} [\gamma \alpha_{j} T_{j-1,k} - 2\gamma T_{j,k} + \beta_{j} \gamma T_{j+1,k}]^{(l)} \\
& = T^{(n)}_{j,k} + \frac{1}{2} \Delta t \frac{\Delta z}{\Delta x^2} [(\gamma + \frac{\Delta z}{2})T_{j,k-1} - 2\gamma T_{j,k} + (\gamma - \frac{\Delta z}{2})T_{j,k+1}]^{(n)}, \quad (38)
\end{aligned}
\]
for \( j = 2, \ldots, M + 1 \) and \( k = 2, \ldots, N + 1 \), whilst for \( j = 1 \) and \( k = 2, \ldots, N + 1 \) we have

\[
T^{(l)}_{1,k} - \frac{1}{2} \frac{\Delta t}{\Delta r^2} [-4\gamma T_{1,k} + 4\gamma T_{2,k}]^{(l)} = T^{(n)}_{1,k} + \frac{1}{2} \frac{\Delta t}{\Delta z^2} ((\gamma + \frac{\Delta z}{2}) T_{1,k-1} - 2\gamma T_{1,k} + (\gamma - \frac{\Delta z}{2}) T_{1,k+1})^{(n)}. \tag{39}
\]

Like before, instead of working the the vector of unknowns

\[
[T^{(l)}_{1,2}, \ldots, T^{(l)}_{M+1,2}, T^{(l)}_{1,3}, \ldots, T^{(l)}_{M+1,3}] \cdots [T^{(l)}_{1,N+1}, \ldots, T^{(l)}_{M,N+1}]^T,
\]

we will solve this system line by line by working with the nonlinear system

\[
(I - \frac{1}{2} \frac{\Delta t}{\Delta r^2} A) \begin{bmatrix}
T_{1,k} \\
T_{2,k} \\
\vdots \\
T_{M+1,k}
\end{bmatrix}^{(l)} + N \begin{bmatrix}
T_{1,k} \\
T_{2,k} \\
\vdots \\
T_{M+1,k}
\end{bmatrix}^{(l)} = b_j^{(n)}, \tag{41}
\]

for each \( k = 2, \ldots, N + 1 \). Here \( A \) is the square matrix

\[
A = \begin{bmatrix}
-4\gamma & 4\gamma & \gamma \beta_2 \\
\gamma \alpha_2 & -2\gamma & \gamma \beta_2 \\
\vdots & \ddots & \ddots \\
\gamma \alpha_{M-1} & -2\gamma & \gamma \beta_{M-1} \\
\gamma \alpha_M & -2\gamma
\end{bmatrix},
\]

\( N \) is the nonlinear equation (which arises from the boundary condition at \( r = \frac{\pi}{2} \)) given by

\[
N \begin{bmatrix}
T_{1,k} \\
T_{2,k} \\
\vdots \\
T_{M+1,k}
\end{bmatrix}^{(l)} = -2\gamma \varphi \frac{\Delta t}{\Delta r} \beta_j \begin{bmatrix}
0 \\
0 \\
\vdots \\
T^4_{M+1,k} - T^4_0
\end{bmatrix}^{(l)}
\]

and \( b_j^{(n)} \) is a vector which depends on \( k \) and \( n \):
\[ \mathbf{b}_k^{(n)} = \begin{bmatrix} T_{1,k} \\ T_{2,k} \\ \vdots \\ T_{M+1,k} \end{bmatrix}^{(n)} + \frac{1}{2 \Delta z^2} \left( \begin{array}{c} T_{1,k-1} \\ T_{2,k-1} \\ \vdots \\ T_{M+1,k-1} \end{array} \right) - 2\gamma \mathbf{T}^{(n)}_{M+1,k} \]

\[ + (\gamma - \frac{\Delta z}{2}) \begin{bmatrix} T_{1,k+1} \\ T_{2,k+1} \\ \vdots \\ T_{M+1,k+1} \end{bmatrix}^{(n)} \]

This will allow us to perform the first half step of each iteration and compute, for each \( k = 2, \ldots, N \), the vector

\[ \begin{bmatrix} T_{1,k} \\ T_{2,k} \\ \vdots \\ T_{M+1,k} \end{bmatrix}^{(l)} \]

The nonlinear system we obtain can be solved by using the well-known Newton’s Method.

If we now take the \( z \)-derivative implicitly, as in (36), we can write

\[ T_{j,k}^{(n+1)} - \frac{1}{2 \Delta z^2} [(\gamma + \frac{\Delta z}{2})T_{j,k-1} - 2\gamma T_{j,k} + (\gamma - \frac{\Delta z}{2})T_{j,k+1}]^{(n+1)} \]

\[ = T_{j,k}^{(l)} + \frac{1}{2 \Delta z^2} [\gamma \alpha_j T_{j-1,k} - 2\gamma T_{j,k} + \gamma \beta_j T_{j+1,k}]^{(l)}, \quad (42) \]

for \( j = 2, \ldots, M + 1 \) and \( k = 2, \ldots, N + 1 \). Proceeding like before we get, for the same values of \( k \) and for \( j = 1 \),

\[ T_{1,k}^{(n+1)} - \frac{1}{2 \Delta z^2} [(\gamma + \frac{\Delta z}{2})T_{1,k-1} - 2\gamma T_{1,k} + (\gamma - \frac{\Delta z}{2})T_{1,k+1}]^{(n+1)} \]

\[ = T_{1,k}^{(l)} + \frac{1}{2 \Delta z^2} [-4\gamma T_{1,k} + 4\gamma T_{2,k}]^{(l)}. \quad (43) \]

Thus, for \( j = 1, \ldots, M + 1 \), we will have to solve the linear system
\[
(I - \frac{1}{2} \frac{\Delta t}{\Delta z^2} C) \begin{bmatrix}
T_{j,2} \\
T_{j,3} \\
\vdots \\
T_{j,N}
\end{bmatrix}^{(n+1)} = d_j^{(l)} + \tilde{g}_k,
\] (44)

where \( C \) is the matrix given by

\[
C = \begin{bmatrix}
-2\gamma & \gamma - \frac{\Delta z}{2} \\
\gamma + \frac{\Delta z}{2} & -2\gamma & \gamma - \frac{\Delta z}{2} \\
\vdots & \ddots & \ddots \\
\gamma + \frac{\Delta z}{2} & -2\gamma & \gamma - \frac{\Delta z}{2} & 2\gamma & -2\gamma
\end{bmatrix}
\]

and \( d_j^{(l)} \) can be written

\[
d_j^{(l)} = \begin{bmatrix}
T_{j,2} \\
T_{j,3} \\
\vdots \\
T_{j,N+1}
\end{bmatrix}^{(l)} + \frac{1}{2} \frac{\Delta t}{\Delta r^2} \left( \gamma \alpha_j \begin{bmatrix}
T_{j-1,2} \\
T_{j-1,3} \\
\vdots \\
T_{j-1,N+1}
\end{bmatrix} - 2\gamma \begin{bmatrix}
T_{j,2} \\
T_{j,3} \\
\vdots \\
T_{j,N+1}
\end{bmatrix} + \gamma \beta_j \begin{bmatrix}
T_{j+1,2} \\
T_{j+1,3} \\
\vdots \\
T_{j+1,N+1}
\end{bmatrix} \right),
\]

for \( j = 2, \ldots, M+1 \). With \( j = 1 \) we have

\[
d_1^{(l)} = \begin{bmatrix}
T_{1,2} \\
T_{1,3} \\
\vdots \\
T_{1,N+1}
\end{bmatrix}^{(l)} + \frac{1}{2} \frac{\Delta t}{\Delta r^2} \left( -4\gamma \begin{bmatrix}
T_{1,2} \\
T_{1,3} \\
\vdots \\
T_{1,N+1}
\end{bmatrix} + 4\gamma \begin{bmatrix}
T_{2,2} \\
T_{2,3} \\
\vdots \\
T_{2,N+1}
\end{bmatrix} \right).
\]

Finally, \( \tilde{g}_j \) is given by

\[
\tilde{g}_j = \frac{1}{2} \frac{\Delta t}{\Delta r^2} (\gamma + \frac{\Delta z}{2}) \begin{bmatrix}
T_m \\
0 \\
\vdots \\
0
\end{bmatrix}.
\]
8 Numerical results

As in Section 5, we also need to prescribe an initial solution in order to be able to use the procedure described in the previous sections. Here we take the function given by

\[ T^0 = \frac{T_m}{[1 + \frac{3}{5}(\frac{z}{a})(5\epsilon)^{\frac{1}{2}}]^{\frac{1}{4}}} \]

as the initial guess function - Figure 8.

![Figure 8: Initial guess function $T^0$.](image)

The results presented in this section were obtained when taking $L = 0.5$ and a tolerance of $10^{-3}$ in the Newton Method. With these values, $M = N = 20$ and $\Delta r = 10^{-6}$, we calculate the numerical solution represented in Figure 9.

![Figure 9: Numerical Solution.](image)
Finally, in Table 5, we present, for different values of \( M \) and \( N \), the values of \( \Delta t \) we had to use in order to attain the stoppage condition (31), as well as the maximum norm of the difference between the numerical solution \( T_\infty \) obtained for each case and the initial guess function \( T^0 \). From the analysis of the values presented in the Table it seems that as we take a finer grid the quantity \( \|T_\infty - T^0\|_\infty \) tends to stop growing.

\[\begin{array}{|c|c|c|c|}
\hline
  & M = N = 10 & M = N = 20 & M = N = 30 & M = N = 40 \\
\hline
\Delta t & 10^{-5} & 10^{-6} & 10^{-6} & 10^{-6} \\
\hline
\|T_\infty - T^0\|_\infty & 345.8482 & 376.6578 & 383.9995 & 387.1731 \\
\hline
\end{array}\]

Table 5: Numerical results.

Remark:
If we had taken the non-linear term \( N \) appearing in (41) as an explicit term, by writing

\[
(I - \frac{1}{2 \Delta r^2} A) \begin{bmatrix} T_{1,k} \\ T_{2,k} \\ \vdots \\ T_{M+1,k} \end{bmatrix}^{(l)} = -N \begin{bmatrix} T_{1,k} \\ T_{2,k} \\ \vdots \\ T_{M+1,k} \end{bmatrix}^{(n)} + b_j^{(n)}, \quad (45)
\]

instead of (41), then in the end we would still obtain similar numerical results.

\[\begin{array}{|c|c|c|c|}
\hline
  & M = N = 10 & M = N = 20 & M = N = 30 & M = N = 40 \\
\hline
\|T_{\infty}^1 - T_{\infty}^2\|_\infty & 0.4294 & 0.3854 & 0.3078 & 0.2449 \\
\hline
\end{array}\]

Table 6: Numerical results.

Indeed, in Table 6 we show (for different values of \( M \) and \( N \)) the maximum norm of the difference between the numerical solutions \( T_{\infty}^1 \) and \( T_{\infty}^2 \), obtained respectively by using (41) and (45). Notice that solving the non-linear system (41) is harder than solving (45), which means in the latter case we get virtually the same results but saving in computational time.
References
