NTNU

Norwegian University of
Science and Technology

# A Simulated Annealing Approach to Electrical Resistivity Tomography

**Håvard Eidal Wiken**

MSc in Physics
Submission date: June 2018
Supervisor: Alex Hansen, IFY

Norwegian University of Science and Technology
Department of Physics

**Abstract**

In this study, a new approach to Electrical Resistivity Tomography (ERT) data inversion was investigated. ERT is a geophysical method for imaging the subsurface. An electrical current is injected into the surface, and the electrical properties of the subsurface is determined from measurements of the voltage difference. The objective of these measurements are to estimate the unknown resistivity distribution of the subsurface from information gathered on the surface. The most common way to approach this problem is by the local optimization method, least squares. Due to the ill-posedness of inverse problems, least squares method require a prior information of the subsurface. This may not be readily available. A prior information is needed to form an approximated initial model, and employ appropriate regularizations. In this study, we use Simulated Annealing, to estimate the resistivity distribution of a resistor network. Simulated Annealing is a stochastic optimization method for approximating global optimum. This method does not require a prior information or regularizations, and can approximate solutions in a large search spaces. An algorithm was developed and implemented from scratch for this study. We showed that the resistivity distribution of a network, can be approximated using Simulated Annealing. However, problem of local indeterminable resistivity is observed for challenging resistor distributions. Time consumption is one of the main obstacle for Simulated Annealing to be a competitive optimization method. Practice, limitations and improvements of the method is also discussed in this thesis.

**Sammendrag**

I denne studien undersøkes en ny tilnærming til inversjon av Electrical Resistivity Tomography (ERT) data. ERT er en geofysisk metode for avbildning av bakken. Elektrisk strøm sendes in i overflaten, og de elektriske egenskapene til bakken blir bestemt ved målinger av spenningsforskjellen. Målet med disse målingene er å estimere den ukjente resistivitetsfordelingen i bakken fra informasjon samlet på overflaten. Den vanligste måten å tilnærme seg dette problemt på er ved hjelp av den lokale optimiseringsmetoden, least squares. Invers problemer er illposed, og behøver forhåndsinformasjon om bakken. Dette er ikke nødvendigvis lett tilgjengelig. Forhåndsinformasjonen er nødvendig for å lage en tilnærmet initiel model, og for å tilrettelegge for riktige matematiske reguleringer. I denne studien har vi brukt Simulated Annealing for å estimere resistivitetsfordelingen i et resistornettverk. Simulated Annealing er en stokastisk optimaliseringsmetode for å approksimere et globalt optimum. Metoden behøver ingen forhåndsinformasjon eller matematiske reguleringer, og kan approksimere løsninger i et stort søkerom. Algoritmen ble utviklet og implementert fra bunnen av i denne studien. Vi viste at resistivitetsfordelingen i et nettverk kan bli tilnærmet med Simulated Annealing. Det er dog et problem med lokal ubestemmelig resistivitet for utfordrene resistorfordelinger. Tidsbruken er en av hovedutfordringene for at Simulated Annealing kan være en konkurransedyktig optimaliseringsmetode. Bruk, begresninger og forbedringer av metoden er også diskutert i oppgaven.

## Preface

The following thesis is the finishing part of my Master's degree in physics, and concludes my studies at the Norwegian University of Science and Technology, NTNU. It has been five challenging but educational years, having taken on the discipline I deem one of the most difficult and complex. Understanding the world through physics is not something that have always been innate to me, but the curiosity and desire to find out how it all fit together and solve puzzles brought me on this journey.

This thesis have been conducted under the supervision of Prof. Alex Hansen at the Department of Physics, and director of PoreLab, Centre of Excellence. I would like to thank Alex for the opportunity to work on this challenging and exiting project, and for the guidance, advises and support during the project. Attempting to challenge decades of research and knowledge, to come up with new solutions, is no easy task. But for every new path taken, new opportunities arise. The product of this thesis may only be regarded as a proof of concept, and even though a lot of further research is needed, the opportunities are numerous. The Wright brothers did not make the Boeing, but they showed the world it was possible to make an airplane!

I would like to express my deepest gratitude to my collaborator in this project, M.Sc candidate Frode Thorsen Børseth. His deep insight and perseverance truly pushed this project forward to what it became, so again, thank you!

I would like to give a special thanks to my family for the continuous support and motivation. Lastly, I would like to thank Live Rud-Johansen for the support and encouragement, and also for the proofreading of my thesis, I truly appreciate you!

# Contents

# List of Figures

# 1 Introduction

Electrical resistivity is one of the most sensitive indicators of change in nature. The properties of matter are fundamentally related to how electricity interact with it. The ability of electricity for non-intrusive investigation, makes resistivity an important physical quantity for numerous technical applications. Resistivity surveys are utilized for internal mapping, from finding cracks in a concrete slab [1] to imaging of the human body [2], and determining whole subsurface structures [3]. All of which are vital applications in modern society. *Electrical Resistivity Tomography* (ERT) is a geophysical method used to image the subsurface texture. By sending electricity down through the ground, and measure an apparent resistivity, it is possible to estimate the resistivity distribution of the subsurface. The resistivity is connected to various geophysical quantities of the ground, such as material content, density, fluid composition, porosity, or water saturation of rocks [3]. ERT's ability to gain perception of the subsurface, makes it an important application for geological, environmental and archaeological surveys[4, 5, 6]. ERT is a non-linear inverse problem, where a finite number of measurements at the surface, are used to identify the physical properties of the medium. Inverse problems are the opposite of forward problems, where researchers attempt to find exact solutions to describe physical properties, such as propagation of sound, heat and current conduction, and resistivity. The inverse problem of electrical conductivity of the subsurface was first attempted by Tikhonov in the 1940s, who at first believed it to be impossible. From a mathematical view, *ill-posed* inverse problems were at that time believed to be unsolvable. However, after seeing that it was possible to find approximated solutions, and correctly differentiate subsurface structures, he later formulated the theory of ill-posed inverse problems [7].

The first applicable solution for use with computers was worked out by Loke, and his method is the one widely used today [3]. His method is based on measuring the apparent resistivity on the ground surface, and compute a resistivity map by using smoothness-constrained least squares optimization method [8]. This method has given ERT global recognition, and it is used in a broad spectrum of geological monitoring, exploration, and research. Least squares is a method for finding local minima, and requires the initial model to be in the vicinity of the solution. This requires a prior knowledge of the problem at hand, and corresponding constrains and regularizations. In addition, local optimization algorithms, such as least squares, are sensitive to measurement noise, error, and other offsets to the gathered data. Lastly,

the inherent ill-posedness of data inversion can easily trap local optimization methods in sub-optimal solution, and produce wildly inaccurate models. Electrical resistivity mapping has been stuck in its local sphere of success since its rise to fame in the late 1990s, and it is due for new approaches.

Simulated Annealing (SA) is an optimization algorithm devised for finding the global optimum, among many local and sub-optimal optimums [9]. Its intelligent heuristics makes it unyielding to greedy solutions, resistant to data errors, and amused by the innate ill-posedness of inverse problems. In this study, it is attempted to approach apparent resistivity calculations, using simulated annealing, to find approximated resistivity distribution of a network.

This thesis is organized in the following manner: In Section 2 we outline the theory behind electrical resistivity survey, how to model an electrical conduction system, and how to approach the calculations of such a system. In Section 3 we provide the details on algorithms and implementations. Results are presented in Section 4, and discussed in Section 5. Finally, Section 6 will contain a conclusion for this thesis, and make a few suggestions for further work.

# 2 Theory

The first section will outline the fundamental physics behind electrical resistivity surveys, in accordance with how Loke describes it in his *Tutorial: 2-D and 3-D electrical imaging surveys* [3]. For the second section, we will briefly talk about inverse problems. In the third section we will develop a model for calculating the resistivity distribution, based on the basic electrical resistivity theory. Lastly we will describe the method for building such a model.

## 2.1 Introduction to Electrical Resistivity Surveys

Electrical resistance is a well known and studied physical quantity of many media, and makes a good predictor of what medium we are dealing with, and/or its properties. The basic idea behind electrical resistivity survey is that we can inject a known current into a medium from one point, and measure the difference in the electrical potential at another point. From this the resistance can be calculated. If we do this for several points, we get several resistances, which can be compared to each other to form a resistivity map. Resistivity surveys are based on Ohm's law, which provide us the equation of current flow in a medium

$$\vec{\mathbf{J}} = \sigma\vec{\mathbf{E}} \tag{1}$$

where $\sigma$ is the conductivity of a medium, $\vec{\mathbf{J}}$ is the current density, and $\vec{\mathbf{E}}$ is the electrical field intensity. In geophysical surveys, it is the mediums resistivity $\rho$, the inverse of conductivity $\rho = 1/\sigma$, which is most commonly used, and in this study we will also follow this convention.
The relationship between the electrical potential, and the field intensity is:

$$\vec{\mathbf{E}} = -\nabla\Phi \tag{2}$$

and by combing (1), and (2), gives us:

$$\vec{\mathbf{J}} = -\sigma\nabla\Phi. \tag{3}$$

For simplicity, we consider the current from a point source. For an elemental volume $\Delta V$, surrounding the current source $I$, placed in an arbitrarily coordinate system at ( $x_s, y_s, z_s$), the relationship between the current density, and the current is given by [10]:

$$\nabla \cdot \vec{\mathbf{J}} = \frac{I}{\Delta V}\delta(x - x_x)\delta(y - y_s)\delta(z - z_s) \tag{4}$$

where $\delta$ is the dirac delta function. By substituting $\vec{\mathbf{J}}$ from (3), Eq. (4) may be rewritten as

$$\nabla \cdot [\sigma(x,y,z)\nabla\Phi(x,y,z)] = \frac{I}{\Delta V}\delta(x-x_s)\delta(y-y_s)\delta(z-z_s) \tag{5}$$

which is the basic equation for calculating the potential distribution in the ground due to a point current source. This is the forward modelling problem, and gives us the potential which would have been observed over a given subsurface structure. Analytical methods for finding the exact resistivity distribution exists for homogeneous and symmetrical shapes. For inhomogeneous and disordered systems, numerical approaches are required, and we will derive some expressions and techniques for numerically estimating such systems.

Let us first consider the simplest of cases, a single current source on the top layer of a homogeneous half-space subsurface. The current will in this case flow radially away from the source, and the potential will depend inversely on distance from the current source. The potential can then be given by

$$\Phi = \frac{\rho I}{2\pi r} \tag{6}$$

where $r$ is the distance from an arbitrarily point in the medium to the current source.

In practice, the point current sources will be electrodes, and at least two electrodes, a positive and a negative, is needed. The potential in a medium from a pair of electrodes is given by

$$\Phi = \frac{\rho I}{2\pi}\left(\frac{1}{r_{C1}} - \frac{1}{r_{C2}}\right) \tag{7}$$

where $r_{C1}$ and $r_{C2}$ are distances from the first and second electrode.

Loke further describes that the potential difference is measured for electrical resistivity surveys, where a typical arrangement are four electrodes, and the potential difference is then given by

$$\Delta\Phi = \frac{\rho I}{2\pi}\left(\frac{1}{r_{C1P1}} - \frac{1}{r_{C2P1}} - \frac{1}{r_{C1P2}} + \frac{1}{r_{C2P2}}\right) \tag{8}$$

which would give the potential measured over a homogeneous half-space of a four electrode array.

Real surveys are conducted on inhomogeneous mediums, where the resistivity distribution is in three dimensions. By injecting current into the ground through two electrodes ( C1 and C2), and measuring the resulting voltage

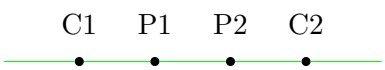



**Figure 1:** Typical electrode arrangement to measure the subsurface resistivity. C1 and C2 are the positively and negatively current sources, and P1 and P2 are measurement nodes.

difference at two other electrodes (P1 and P2), an observed, or *apparent* resistivity $\rho_a$ can be calculated

$$\rho_a = k\frac{\Delta\Phi}{I} \tag{9}$$

where

$$k = \frac{2\pi}{\left(\frac{1}{r_{C1P1}} - \frac{1}{r_{C2P1}} - \frac{1}{r_{C1P2}} + \frac{1}{r_{C2P2}}\right)}$$

$k$ is a geometrical factor that comes from the arrangement of the four electrodes, as exemplified in Figure 1. Resistivity measurements normally provides a resistance value, $R = \Delta\Phi/I$, and the *apparent* resistivity is in practice calculated by

$$\rho_a = kR. \tag{10}$$

The resistance value is not the true resistivity of the medium, but the resulting measured resistivity which a homogeneous medium of the same electrode arrangement would have given. The relationship between the apparent resistivity and the true resistivity, is given by solving the *inverse problem*. The challenge of electrical resistivity tomography is then how to appropriately process the given apparent resistivity values to approximate the resistivity distribution of the subsurface.

## 2.2 Inverse and Forward Problems

Fundamental physic equations help us calculate the effect a physical field, process or phenomena may have upon a medium. The equations works as a model framework, which can predict the outcome. Eq.(5) is an example of this, where a current will act upon the medium, and introduce an electrical field, of which we can predict the field strength. This is a forward, or direct problem. A set of model parameters may be used to predict the observation, and describe the relationship between the model and the observed outcome. Mathematically a forward problem can be formulated

$$\mathbf{M} \longrightarrow \mathbf{d} = \mathbf{g}(\mathbf{M}) \tag{11}$$

where $\mathbf{M}$ is the model parameters, $\mathbf{d}$ is the data or the observable parameters, and $\mathbf{g}$ and is the forward operator that describe the relationship between the model parameters and the data. Most forward problem are *well-posed* problems, which have the following properties:

- a solution exists in the model space.
- the solution is unique.
- the solution's behavior changes continuously with the initial conditions.

The opposite of a forward problem, is an inverse problem. For an inverse problem the outcome is used to calculate the cause, and can help us figure out e.g. the properties of a medium. This can again help us make prediction of sources (e.g. heat, waves, current, potential differences), material, location, structure, defects, and so on. For a linear inverse problem, the formula from the forward problem, rewritten as

$$\mathbf{d} = \mathbf{g}\mathbf{M} \tag{12}$$

can be used to invert the operator $\mathbf{g}$ to find the model parameters $\mathbf{M}$

$$\mathbf{M} = \mathbf{d}\mathbf{g}^{-1}. \tag{13}$$

The operator $\mathbf{g}$ can rarely easily be inverted to fit together with the data $\mathbf{d}$ to give the model parameters $\mathbf{M}$. This is due to the fact that most inverse problems are non-linear, and inherently *ill-posed*, meaning at least one of the three conditions of well-posedness are not met. Inverse problems are therefore subject to the following:

- a solution does not exists everywhere in the model space, or may only be approached approximately.
- a solution is non-unique, meaning there can be many models that fit the data.
- a solution's behavior is unstable ( i.e. poor initial conditions, arbitrarily small changes or measurement errors can produce large errors in the solutions).

For such problems, we have to search through the model space, and attempt to find model(s) $\mathbf{M}$ with the smallest possible discrepancy for the data $\mathbf{d}$. Numerical methods of optimization may be used for this, and we will get back to this in Section 3.
In electrical resistivity studies we seek to find a model that provides a response similar to the actual measured values. The inverse problem of this study is to find the resistivity distribution for the subsurface, that will minimize the difference between the calculated and measured apparent resistivity values. An initial model is proposed, and modified in iterations by comparing the difference, to achieve the smallest possible discrepancy $g$. Lokes method is based on solving a variation of Gauss-Newton least-square optimization with regularizations. The basic least-square equation is the following,

$$\mathbf{J^T J \Delta q}_i = \mathbf{J^T g} \tag{14}$$

where $\mathbf{g}$ is the discrepancy vector, $\Delta\mathbf{q}$ is the model parameter change vector and $\mathbf{J}$ and $\mathbf{J}^T$ is the Jacobian matrix and its transpose, respectively. The equation for practical use is subject to Levenberg–Marquardt modification, smoothness restrains and reweighting, and looks like this,

$$(\mathbf{J^T J} + \lambda \mathbf{F_R})\mathbf{\Delta q}_i = \mathbf{J^T R_d g} - \lambda \mathbf{F_R q}_i \tag{15}$$

where $\lambda$ is a dampning factor, $\mathbf{F_R}$ is the smoothness restrains, and $R_d$ weighting matrix. This equation is the general method for geophysical inversion, and can be even further modified to include known information about the subsurface. For further details and explanation on Lokes methods and technical solutions, the author refers to his tutorial[3].
Least-square is a method for finding a local minima, and having a proper initial model is crucial for success. Such initial models require a prior knowledge of the subsurface, and may not be trivial to find. We will in this thesis therefore attempt a new approach to solve the inverse problem for electrical resistivity distribution using simulated annealing. For this purpose we need to make a model that simulates the electrical conduction in an inhomogeneous medium.

## 2.3 Electrical Conduction Modelling in Media

To make an image of the subsurface from resistivity, we need a way of modelling how the electricity conduct itself in a disordered media. In this section

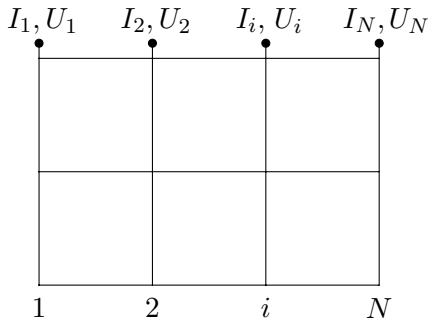



**Figure 2:** Simplified electrical network with $N$ top nodes. A current is inject into node $i$, and extracted at node $j$ until all $i$,$j$ node pairs are measured and have an apparent resistivity between them.

a description of how the setup to a model of electrical conduction in the subsurface is provided.

The principle of this study is to estimate the resistivity distribution of internal structures, based on measurements of the electrical potential at the boundary. For this purpose we make a simplified model of the subsurface as a resistor network (Figure 2), to see if its possible to estimate the value of each resistor in the network.
First, let us state a few assumptions for the model:

- The half-space above the top layer have infinite resistance, and will not conduct a current.
- All the injected current will be absorbed by a current sink on the same boundary surface.
- The injection and sink current sources coincide with the measurement nodes, so that the potential difference is between the injection and sink current sources.

Discretizing the subsurface into a electrical network of square lattices, gives a good conceptual model for this study. As a side note, studies have also shown that continuous electrical conduction in a media, can be approach with discretization models as a resistor network. Rigorous mathematical justification for modelling electrical conduction in media as a resistor network have been conducted by Borcea et al. [12].

For an electrical network, Ohm's laws (1) can be simplified and applied to calculate the resistance,

$$R = U/I \tag{16}$$

where $R$ is the resistance, $U$ is the electrical potential, and $I$ is the current. To make predictions of the resistivity distribution, based on the measurements at the top layers, is no easy task. For this, we have to invert the system, build it from the bottom and work our way back to the top, resistor by resistor. This way, we attempt to recreate the probable pathing of the electricity. For this purpose we introduce two of Kirchhoff's laws. Kirchhoffs's current law state that at a junction in an electrical network, the total current flowing into the junction, have to be the same flowing out, which is equivalent to

$$\sum_{k=1}^{n} I_k = 0 \tag{17}$$

as current can be expressed as positive or negative depending on if it flows towards, or against a junction. Here $n$ is the number of branches in the junction.
Kirchhoff's voltage law says that the electrical potential difference for a closed network is zero, and can be expressed as:

$$\sum_{k=1}^{n} U_k = 0 \tag{18}$$

where $n$ is the number of voltages measured. Together the two laws form Kirchhoff's circuit laws, which can be combined and used to calculate electrical networks by a system of linear equations.
In a large electrical network system, the equation for every junction can be generalize to

$$\sum_{j \neq k}^{n} G_{jk}(U_k - U_j) = 0 \tag{19}$$

where $G_{jk} = G_{kj}$ is the conductance between junction $k$ and $j$, and $U_k$ is the voltage at junction $k$. When we apply a current $I$ at a top junction, say $k$, the expression becomes

$$I_k = G_{kk}U_k - \sum_{j \neq k}^{n} G_{jk}u_j. \tag{20}$$

We can combine the above junction equations for all junctions, and express them in matrix form

$$\begin{bmatrix} G_{11} & G_{12} & \dots & G_{1n} \\ G_{21} & G_{22} & \dots & G_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ G_{n1} & G_{n2} & \dots & G_{nn} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_n \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix},$$

or basically

$$\mathbf{GU} = \mathbf{I}. \tag{21}$$

To find the resistivity distribution, we want to calculate the $\mathbf{G}$ matrix, in the form of resistor values $R = 1/G$. For this, we introduce the transfer-matrix approach to random resistor network.

## 2.4 Random Resistor Network and Transfer-Matrix Method

A random resistor network, is a model network for studying conduction in a disordered system. It consists of random-valued conduction elements connected through connection points or nodes, usually in the form of a lattice pattern. The network can take on several geometric forms depending on the problem at hand, Figure 3 shows some typical geometries. Random resistor networks have been used extensively in the study of disordered systems, and have been proven to be quite fruitful for studying conduction properties of electric current, heat and fluid in all sorts of media [11, 15, 16].

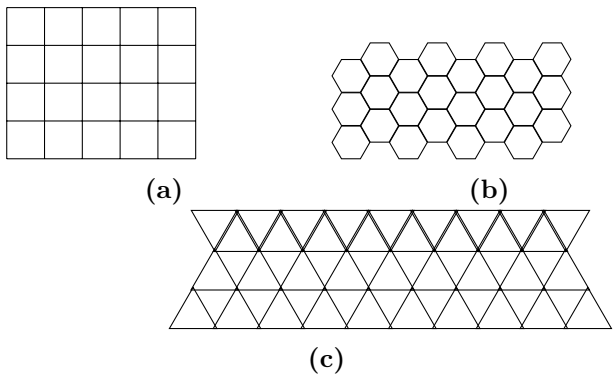



**Figure 3:** Various resistor networks based on square lattice (a), hexagonal lattice (b) and triangular lattice (c).

A simple resistor network would be one consisting of square lattices (such as Figure 3a), where a node in each corner of the square, is connected by

four resistors, making up the "walls" of the square, with the exception of the top, bottom and edges. In this study, the square lattice pattern will be utilized.

Transfer-matrix is an acknowledged method in statistical mechanics known for solving, among many other, the famous two-dimensional Ising model [17]. The method defines a matrix corresponding to a model, so that the physical properties of a system can be extracted from the matrix values. For this study, the conduction properties of an electrical network are examined, and a transfer-matrix is constructed to reflect the properties of a resistor network. The transfer-matrix approach is performed in accordance with Derrida et al., and any resistor distribution can be used for the random resistor network [11].
We consider the square lattice random network, where a current $I_i$ can be injected in each top node $i$, with an associated potential $U_i$. The potential is dependent on the current $I_i$, and are related through a $N \times N$ transfer-matrix $\mathbf{A}$:

$$I_i = \sum_{j=1}^{N} (\mathbf{A})_{ij}\ U_j. \tag{22}$$

We can see that the elements of $\mathbf{A}$ are related to $\mathbf{G}$ in Eq. (21).

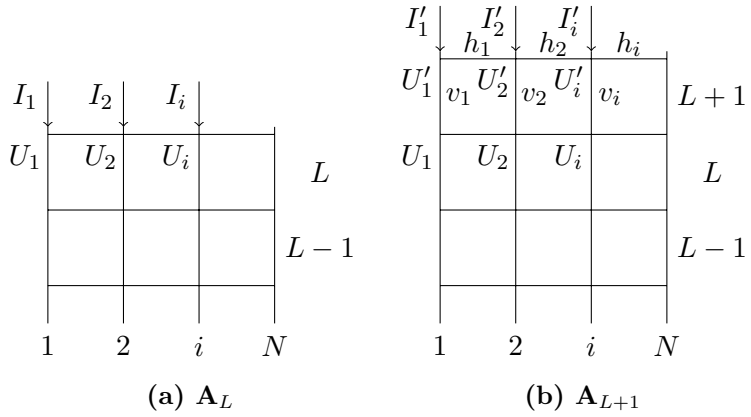
Starting with a single row of resistors connecting the top nodes, the resistivity between each node can simply be found by measuring the potential difference, and calculating the resistance by Ohm' law, Eq.(16). The resistivity values will then be stored in the matrix $\mathbf{A}$ as an equivalent to the apparent resistivity.
This network can then be built, strip by strip, by transforming the matrix $\mathbf{A}_L$ into a new matrix $\mathbf{A}_{L+1}$ after adding horizontal and vertical resistors (Figure 4). If we first assume only addition of vertical resistors $v_i$, and fix the potential $U_i'$ at nodes $i$ in layer $L+1$ by external sources, we will get:

$$U_i = U_i' - v_i I_i. \tag{23}$$

Thus, the current $I_i$ and the potentials $U_i'$ are related through a matrix $\mathbf{B}_{L+1}$:

$$\mathbf{I} = \mathbf{B}_{L+1}\mathbf{U}'. \tag{24}$$

This can be expressed in matrix form as




**(a)** $\mathbf{A}_L$ **(b)** $\mathbf{A}_{L+1}$

**Figure 4:** Recursive construction of transfer-matrix $\mathbf{A}$ 4a, adding layers of horizontal $h_i$ and vertical $v_i$ resistors 4b.

$$\mathbf{U} = \mathbf{U}' - \mathbf{V}\mathbf{I} \tag{25}$$

where the matrix $\mathbf{V}$ is the diagonal, thus

$$V_{ij} = v_i \delta_{ij}. \tag{26}$$

We can then add together Eq. (22) and (25), which gives

$$\mathbf{U} = (1 + \mathbf{V}\mathbf{A}_L)^{-1}\mathbf{U}'. \tag{27}$$

Hence, the matrix $\mathbf{B}_{L+1}$ is given by

$$\mathbf{B}_{L+1} = \mathbf{A}_L(1 + \mathbf{V}\mathbf{A}_L)^{-1}. \tag{28}$$

Now we can add the horizontal resistors $h_i$. Again we fix the voltages $U'_i$ at every node $i$ of layer $L+1$, and get a current $j_i$ in the horizontal resistors $h_i$

$$j_i = [U'_{i+1} - U'_i]/h_i. \tag{29}$$

Therefore, the current $I'_i$ in the resistor connected to the node $i$ of layer $L+1$ is

$$\begin{aligned} I'_i &= I_i + j_{i-1} - j_i \\ &= I_i + [1/h_i + 1/h_{i-1}]U'_i - [1/h_i]U'_{i+1} - [1/h_{i-1}]U'_{i-1} \end{aligned} \tag{30}$$

which also have a matrix form

$$\mathbf{I}' = \mathbf{I} + \mathbf{H}\mathbf{U}' \tag{31}$$

where $\mathbf{H}$ is the matrix which represent the effect of the horizontal resistors at layer $L+1$:

$$H_{ij} = [1/h_i + 1/h_{i-1}]\delta_{ij} - [1/h_i]\delta_{j,i+1} - [1/v_{i-1}]\delta_{j,i-1}. \tag{32}$$

Finally, this leads to a recursive formulation for building a random resistor network, strip by strip:

$$\mathbf{A}_{L+1} = \mathbf{H} + \mathbf{A}_L(1 + \mathbf{V}\mathbf{A}_L)^{-1}. \tag{33}$$

The transfer-matrix method can be expressed in a more numerically friendly maneuver, where each resistor is added one by one. This method will be outlined in the next section.

# 3 Numerical Methods

Estimating the parameters of a model for an inverse problem is not a trivial pursuit, and thorough exploration of the model space has to be performed. Luckily, powerful algorithms for searching, and finding optimal solutions have been developed. First we will show how to implement the construction of the random resistor network and the transfer-matrix, then we will describe the optimization algorithm, and how it is implemented.

## 3.1 Transfer-Matrix Implementation

Constructing a random resistor network, and updating the transfer-matrix accordingly is the first step to able to perform the optimization algorithms. We use the theory from Section 2.4, and Derrida et al. [14] to outline a numerically implementation.

The size of the transfer-matrix $\mathbf{A}$ is predetermined from the number $N$ of top nodes we choose to use (Figure 5). We can think of $N$ as the width of the system, and $L$ as the number of vertical resistor layers. Therefore, the depth will be $L+1$, since the last layer is of horizontal resistors. Together this form a system of $N \times (L+1)$ nodes, or lattice points. A resistor is placed between two nodes, so that the total number of resistors in the network becomes $(L+1)(N-1) + NL$. First, a single resistor is added between the first two

nodes $i = 1$ and its neighbour $j = 2$, and $\mathbf{A}_{11} = I_i/U_j = \mathbf{A}_{22} = I_j/U_i$, and the transfer-matrix will look like this:

$$\mathbf{A} = \begin{bmatrix} (I_1/U_2) & 0 & \dots & 0_{1n} \\ 0 & (I_2/U_1) & \dots & 0_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{n1} & 0_{n2} & \dots & 0_{nn} \end{bmatrix}$$

Adding the remaining resistors of the first layer, contributes to the matrix $\mathbf{A}$ by the following relation:

$$\mathrm{A}'_{ij} = \mathrm{A}_{ij} + \frac{(\delta_{\alpha j} - \delta_{\beta j})(\delta_{\alpha i} - \delta_{\beta i})}{r} \tag{34}$$

where we also introduced the indices $\alpha$ and $\beta$ to keep track of the node sites, from the matrix-elements $i, j$ in $\mathbf{A}$.

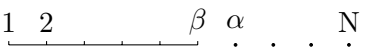
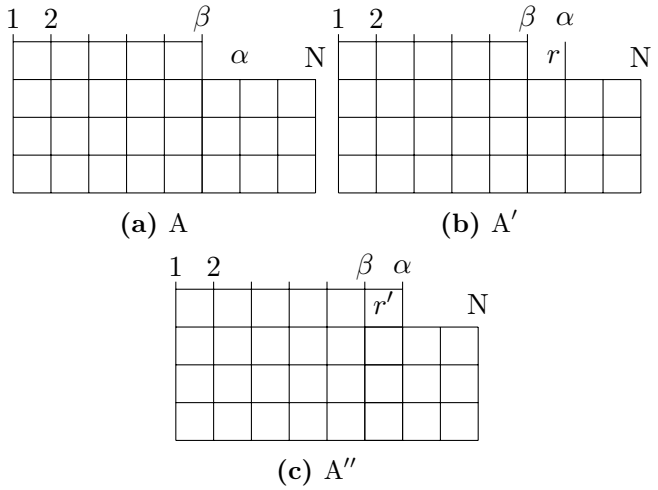



**Figure 5:** Construction of the first horizontal layer of the random resistor network.

After constructing the first layer of the resistor network, we can add to the network, resistor by resistor. The only thing we need to do is transform the




**(a)** A **(b)** A′

**(c)** A″

**Figure 6:** Recursive construction of a network, adding resistor by resistor from a single layer network. For every vertical resistor added, the matrix $A$ is modified into A′, and for every horizontal resistor added, A′ is modified into A″.

**A** matrix for every resistor that is added (Figure 6).
In this study, a two-dimensional system have been considered, but the following description can also be generalized for three-dimensions [11].
For a two dimensional system, there are only horizontal and vertical resistors can be added. Adding a vertical resistor $r$ on site $\alpha$, produces a new matrix $\mathbf{A'}$ (Figure 6b. The new matrix $\mathbf{A'}$ is related to $\mathbf{A}$ through

$$\mathrm{A}'_{ij} = \mathrm{A}_{ij} - \frac{\mathrm{A}_{i\alpha}\mathrm{A}_{\alpha j} r}{1 + \mathrm{A}_{\alpha\alpha} r}. \tag{35}$$

Similarly, we can do this for a horizontal resistor $r'$ between the sites $\alpha$ and $\beta$ (Figure 6c). The matrix $\mathbf{A'}$ changes into a new matrix $\mathbf{A''}$ given by

$$\mathrm{A}''_{ij} = \mathrm{A}'_{ij} + \frac{(\delta_{\alpha j} - \delta_{\beta j})(\delta_{\alpha i} - \delta_{\beta i})}{r'} \tag{36}$$

where $\delta_{ij}$ is the Kronecker delta.
The following pseudocode shows how to implement the construction of the transfer-matrix:

**Algorithm 1** Transfer-matrix approach to random resistor network

```
Initialize transfer-matrix A
Initialize matrix of horizontal H and vertical V resistor values
for i = 0 → (N − 1) do
  Calculate transfer-matrix for first row of horizontal resistor
end for
for l = 0 → L do
  Calculate transfer-matrix for every layer of vertical resistor
  for n → N do
    Calculate for every horizontal resistor layer
  end for
end for
return A
```

## 3.2 Simulated Annealing

Simulated annealing (SA) is an optimization algorithm, designed to find a global optimum, among many local optimums. An introduction to it can be found in e.g the book *Numerical recipes* [18]. The method's name stems from the thermodynamic analogy of cooling and annealing metals for specific properties (e.g. strength, hardness, ductility). At high temperatures, the molecules in a liquid metal moves freely with respect to each other - if the liquid is cooled slowly, thermal mobility is gradually lost, and the molecules will orderly arrange themselves to form a pure crystal. This pure crystal will be at the lowest possible energy state. Most optimization algorithms always change towards a state of lower energy, or go downhill, which often is the analogy of minimization. If a liquid metal is cooled too quickly, molecules will be squished into sub-optimal configurations, and the metal will end up with a somewhat higher energy state. This energy state will then be a so-called local minimum, and resulted from only going downhill.
From statistical mechanics, we have Boltzmann probability distribution, which has its own minimization procedure integrated in the formula,

$$\text{Prob}(E) \sim \exp(-E/k_B T) \tag{37}$$

where $k_B$ is Boltzmann's constant. For simulated annealing, the $k_B$ constant is in practice omitted as it is set to 1.
The principle from Boltzmann distribution is that a system in thermal equilibrium at temperature $T$ has its own energy probabilistically distributed among all different energy states $E$. So even for low temperatures the system has a chance of being in a high energy state. If we study a system and consider a transition from configuration 1 with energy $E_1$ to configuration 2 with energy $E_2$, we can substitute $E$ in Boltzmann distribution (37), with the energy difference between $E_1$ and $E_2$, giving

$$P(E) = \exp[-(E_2 - E_1)/T]. \tag{38}$$

For $E_2 > E_1$ we will have a probability between 0 and 1. Hence a SA system will sometimes have the opportunity to go uphill, and possibly jerk itself out of local minima valleys. If $E_2 < E_1$ the probability will exceed unity, and will in that case be set to $P = 1$. Thus the system will always pick the new configuration.

The general recipe for SA goes as following:

1. Make a description for the possible *system configurations*.

2. Construct a generator of random changes in the configuration, which is presented as options for the system to evaluate.

3. An *objective evaluation function* $E$ (analogy of energy), which compare current configuration with the one presented in the previous point. The goal is to minimize this function.

4. A control parameter $T$ (analogy of temperature), and an *annealing schedule* to manage after *how many* iterations the temperature should change, and by *how much*.

SA is a complex, but powerful algorithm, consisting of several algorithmic parts. For point 2 and 3 in the general recipe, an algorithm called Metropolis-Hasting is the functional method for generating and evaluating the systems configurations, and discussed in the next subsection. The temperature control parameter $T$ from point 4, is discussed in Section 3.2.2.

### 3.2.1 Metropolis-Hasting Algorithm

The Metropolis-Hasting algorithm is a subclass of Markov Chain Monte Carlo method (MCMC). MCMC is a class of random sampling methods, designed to estimate model parameters and their degree of uncertainty. The purpose of the Metropolis-Hasting algorithm is to generate configurations from a probability density $\pi(\mathbf{x})$, and evaluate the configurations based on the previous state. There are two important aspects to MCMC; *ergodic property* and *detailed balance*. Ergodic property in the means of Markov chains, is that there is a positive probability to go from any state $i$, to any other state $j$, in just one step. Detailed balance state that the probability of transition from a state $i$ to state $j$ is equal to transition from $j$ to $i$, given by the detailed balance equation:

$$\pi(\mathbf{x}_1)P(\mathbf{x}_2 \mid \mathbf{x}_1) = \pi(\mathbf{x}_2)P(\mathbf{x}_1 \mid \mathbf{x}_2) \tag{39}$$

where $\pi(\mathbf{x}_1)$ and $\pi(\mathbf{x}_2)$ are non-normalized probability density for state 1 and 2, respectively. $P(\mathbf{x}_1 \mid \mathbf{x}_2)$ is the transition probability for transition to state 2, given the system is in state 1, and vice versa in $P(\mathbf{x}_2 \mid \mathbf{x}_1)$.
The possibility of sometimes taking an uphill step is known as Metropolis algorithm, and was generalized by Hasting to the so-called Metropolis-Hasting algorithm. The derivation of the algorithm starts with the detailed balance equation (39), rewritten as:

$$\frac{P(\mathbf{x}_2 \mid \mathbf{x}_1)}{P(\mathbf{x}_1 \mid \mathbf{x}_2)} = \frac{\pi(\mathbf{x}_2)}{\pi(\mathbf{x}_1)}. \tag{40}$$

Given a proposal distribution $q(\mathbf{x}_2 \mid \mathbf{x}_1)$ for proposing state 2, given the system is in state 1, and an acceptance distribution $\alpha(\mathbf{x}_2, \mathbf{x}_1)$ for accepting state 2, the transition probability is

$$P(\mathbf{x}_2 \mid \mathbf{x}_1) = q(\mathbf{x}_2 \mid \mathbf{x}_1)\alpha(\mathbf{x}_1, \mathbf{x}_2). \tag{41}$$

Inserting this relation into (40) gives us

$$\frac{\alpha(\mathbf{x}_1, \mathbf{x}_2)}{\alpha(\mathbf{x}_2, \mathbf{x}_1)} = \frac{\pi(\mathbf{x}_2)q(\mathbf{x}_1 \mid \mathbf{x}_2)}{\pi(\mathbf{x}_1)q(\mathbf{x}_2 \mid \mathbf{x}_1)}. \tag{42}$$

For a new configuration, where the energy is lower than the previous configuration, the condition from simulated annealing state that the acceptance probability go to a unity of 1. Therefore Eq. (42), turn to

$$\alpha(\mathbf{x}_1, \mathbf{x}_2) = \min\left(1, \frac{\pi(\mathbf{x}_2)q(\mathbf{x}_1 \mid \mathbf{x}_2)}{\pi(\mathbf{x}_1)q(\mathbf{x}_2 \mid \mathbf{x}_1)}\right). \tag{43}$$

For proposal distributions which only depend on the difference $\mid \mathbf{x}_1 - \mathbf{x}_2 \mid$ between energy state 1 and energy state 2, the ratio $q(\mathbf{x}_1 \mid \mathbf{x}_2)/q(\mathbf{x}_2 \mid \mathbf{x}_1)$ is just 1. The probability density $\pi(\mathbf{x})$ for a system that follows the Boltzmann probability distribution (37), becomes $\pi(\mathbf{x}) = P(E) = \exp(-E/kT)$, and we finally arrive at the acceptance probability:

$$\alpha(\mathbf{x}_1, \mathbf{x}_2) = \min\left(1, \exp[-(E_2 - E_1)/kT]\right) \tag{44}$$

since $\exp(-E_2)/\exp(-E_1) = \exp[-(E_2 - E_1)$.

The energy $E$ is the objective function that is used to compare states. In this study, we are building a resistor network, with no real energy connected to it. The approach to evaluate the resistor networks are by comparing the matrix norm of the transfer-matrices of the measured system, and the annealing system,

$$E = \parallel \mathbf{A}_m - \mathbf{A}_c\parallel^2 \tag{45}$$

where $\mathbf{A}_m$ and $\mathbf{A}_c$ are the transfer-matrices of the measured and the annealing system, respectively.

The implementation for the Metropolis-Hasting algorithm is as following:

1. Initialize with a random starting state $x_0$ in the model space at time $t = 0$.
2. Generate a random candidate state $x_c$.
3. Calculate the acceptance probability $\alpha(x_0 \mid x_c)$ by Eq.(44).
4. Generate a uniform random number $r \in [0, 1]$.
   i If $r < \alpha(x_0 \mid x_c)$, accept the new state, and set $x_{t+1} = x_c$.
   ii If $r > \alpha(x_1 \mid x_c)$, reject the new state, and set $x_{t+1} = x_1$.
5. Increment $t = t + 1$.

### 3.2.2 Temperature

The temperature $T$ in Boltzmann distribution correspond to a control parameter in simulated annealing, sometimes referred to as a *computational temperature*. At high $T$, the system performs a coarse search of the model space to find good minima. If we look at Eq.(38), we can see that by lowering $T$, the negative exponential will increase, and thereby reduce the probability for uphill transition. This is a sort of fine tuning of the search space, where searches will be narrowed down to the neighbourhood of an already determined minimum, and attempt to capture the configuration in the lowest possible energy state. For simulated annealing, an *annealing schedule* is introduced to gradually lower the temperature $T$. The annealing schedule is critical for the efficiency of simulated annealing. The annealing schedule include *when* and *how much* the temperature is lowered. The *when* is in this study coined the *annealing time step* $k$, and is related to the iterative time step $t$. Annealing time step is usually defined by lowering the temperature after a certain amount of either new configuration attempts or approvals. A stopping criterion is also commonly based on the annealing time step too. If a high number of rejections (low approval-to-attempt ratio) occurs, the optimization procedure is concluded. Ideally the system should have time to reach a near-equilibrium state for every new configuration, where the energy average have minimal fluctuations, before every temperature cooling. If $T$ is lowered too fast, its a high possibility of getting stuck in local minima valleys. For the opposite situation, where $T$ is lowered too slow, the algorithm doesn't converge, and may never find global minimum. Finding a usable annealing schedule varies with every problem, and is about trial and error. Although some general schedules have been proposed in the literature for *how much* the temperature should be lowered [9, **?**]:

- Exponential schedule: $T_k = a^k T_0$, where $a$ is a cooling constant usually between 0.85 and 0.995, $k$ is the annealing time step, and $T_0$ is the initial temperature.
- Linear schedule: $T_k = T_0 - \eta k$, where $\eta$ is a decay parameter.

Exponential and linear are the original schedules from Kirkpatrick's paper [9], and are the easiest to implement.
Lastly, an initial temperature $T_0$ also needs to be considered, as this might influence the result and the computation time. If the initial temperature is too high, almost all configuration changes will be accepted, and the system will be kept rambling in high energy states for much of the computing time. On the other hand, if the starting temperature is too low, the system might get stuck on a path towards a local minimum, similarly to the result of reducing the temperature too rapidly.
The initial temperature should give the system a fair chance of accepting worse solutions (go uphill), so it doesn't easily get trapped in local minima valleys. Some suggestions on how to approach the initial temperature have been described in Kirkpatrick et al. paper on SA [9]:

- Initial temperature is suggested to be larger or equal to the maximal energy difference between any neighbouring configurations, $T_0 = \Delta E_{max}$, for a normal run.
- Initial temperature should give an acceptance probability $\alpha_0$ for the first iteration sequence before temperature change, where $\alpha_0$ may be any number between 0 and 1, but it is recommended to be close to 1.

As for the annealing schedule, initial temperature is again mostly about trial and error, and customization of the parameter have to be done according to the results of trial runs.

### 3.2.3 Implementation and Pseudocode

In this study, the system configuration is the resistor elements in the network. Every resistor, or its value, is stored in a list. This list is used to to pick the values, that goes into making the transfer-matrix. The transfer-matrix is equivalent to the apparent resistivity measurements, and used to compare the annealing configuration to the true measured configuration. New configurations are generated by switching out a randomly chosen resistor, and replace it with a randomly generated value. When a resistor

is switched out, the transfer-matrix have to be recalculated for the whole network.
The objective evaluation function is made by comparing the norm of the measured transfer-matrix to the annealing transfer-matrix $\| \mathbf{A}_m - \mathbf{A}_c \|^2$.
The pseudocode representation of SA:

**Algorithm 2** Simulated annealing

Initialize time $t = 0$
Initialize starting temperature, $T_k = T_0$
Initialize system configuration, $x_t = x_0$, starting energy $E_t = E_0$
**repeat**
  Generate a random candidate state $x_c$, and calculate energy $E_c = \| \mathbf{A}_m - \mathbf{A}_c \|^2$
  Calculate acceptance probability $\alpha = \exp(-\frac{(E_c - E_t)}{T})$
  Generate random number $r$ uniformly in (0,1)
  **if** $\alpha > r$ **then**
    accept new configuration, $x_{t+1} = x_c$ with $E_{t+1} = E_c$
  **else**
    reject new configuration, and keep the previous configuration $x_{t+1} = x_0$ with $E_{t+1} = E_t$
  **end if**
  Decrease temperature according to annealing schedule, $T_{k+1} = T_k - \Delta T$
  Increase time $t = t + 1$
**until** $T \approx 0$
**return** $x_t$

## 3.3 Simulation Procedure

The objective of this study, is to demonstrate that a resistor distribution of an electrical resistor network may be approximated from the measured apparent resistivity of the total network. For this purpose, a few different artificial resistor mappings have been composed by the author, to produce transfer-matrices of a supposedly measured systems. These are used to compare the annealing system, to the supposedly measured system, to see if it can approach the true resistor distribution.
A uniform random number generator was used for generation of random numbers. Simulations performed for the result section was produced using seed(12345). Several seeds was tested for every procedure, to avoid lucky or unlucky simulations, and to see that the simulations performed within the same result range.
A matrix was made from the list of resistors to make an image. For $L = 3$ the images have 7 layers, and for $L = 6$, the images have 13 layers. The reason for this is due to how the images are built in a matrix format. The first row, are the resistor values of the first horizontal row, and the second row, is the first layer of vertical resistors and so on. Therefore, each row is one layer of either horizontal or vertical resistors. For $L = 3$, three layers of vertical resistors, we get $2L + 1 = 7$ layers in the image, and for $L = 6$, we get 13 layers in the image.
For $N$ top nodes, we get $N - 1$ horizontal resistors for each layer, and $N$ vertical resistors. Due to this, the $N \times L$ matrix is missing the $N$'th horizontal resistor for every layer. The $N$'th pixel of the first, and every other row, will by default be zero, (as the matrix is initialized as a zero matrix), and therefore be black. This is just to complete the image, and does not effect the simulations or the results.
Python was the working programming language for development of the algorithms. The program was later transferred to Cython, an C-extension for Python, to speed up the simulations (see A.1). A note should be made to remind the reader that Pythons ranging starts at 0, and the indices of the images will therefore also start at 0.

The simulations executed in this study, was performed with resistors valued between 0.5 and 1.5. The initial configuration was always randomized, as can be seen in Figure 7 and 8.
A network consisting of $N = 12$ top nodes (width), and $L = 3$ layers (depth) as in Figure 7 was used for most simulations. At this lattice size, an E symbol (for electricity), was made with resistors of value 1.5 while all the

other resistors were set to 0.5. For this network, different simulations for run times $t$, initial temperatures $T_0$, annealing time step $k$ and temperature lowering were performed.

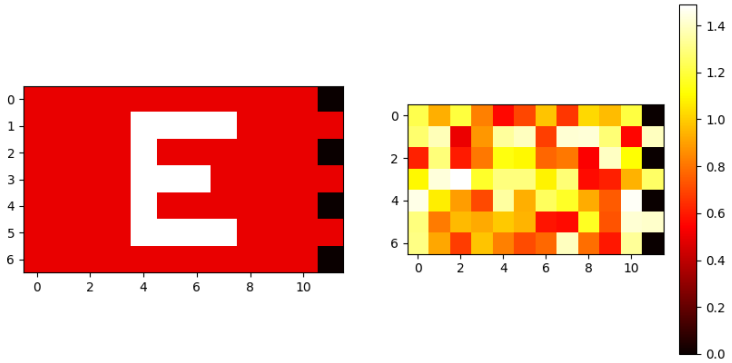



**Figure 7:** Left panel shows the artificial resistor distribution of a $12 \times 3$ network. The red pixels illustrate 0.5 resistors. An arbitrarily shape, in this case an E, is formed from 1.5 resistors. Right figure is a random starting configuration for the annealing system. The scale on the far right, shows the color mapping of the resistance values. The black dots on the right edge of each image, are zero resistances. These are to fill in the matrix used to make the image.

Simulations were also performed with lattice size $N = 25$ and $L = 6$ (Figure 8), where a checkered pattern consisting of 0.5 and 1.5 resistors, were made to increase the difficulty. Lastly, for a real challenge, a grading pattern were made for $N = 25, 26, 27, 28$ and $L = 6$ (Figure 9), with resistor ranging from 0.5 to 1.5, with a 0.1 interval.

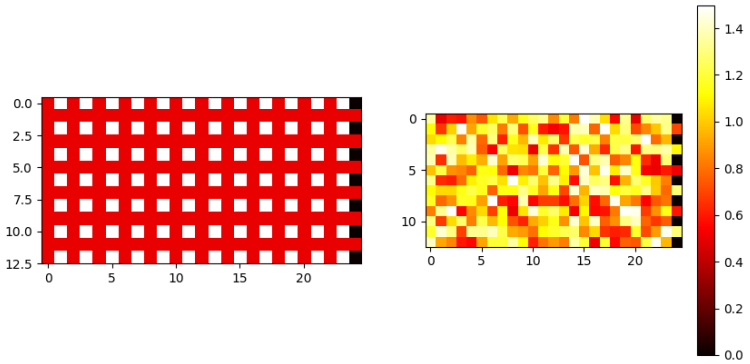



**Figure 8:** Left panel is the resistor configuration of an artificial $25 \times 6$ checkered pattern. Red pixels are 0.5 resistors, and white pixels are 1.5 resistors. Right figure is the random starting configuration for the annealing system. Scale on the far right shows the color mapping of the resistance values.

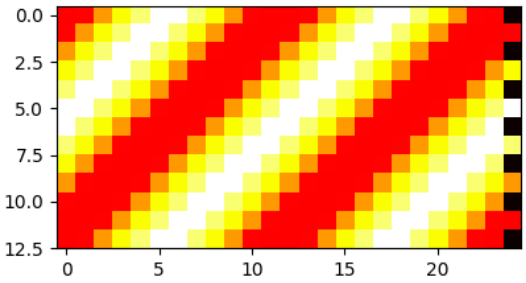



**Figure 9:** Grading pattern for $25 \times 6$ network, with resistors ranging from 1.5 (white pixels) to 0.5 (red pixels) at a 0.1 interval.

# 4 Results

In this section we present results of the simulations performed using the algorithms from Section 3. The $12 \times 3$ system is considered in the first Section 4.1, where we compare the simulation parameters. The results from running optimizations on a $25 \times 6$ checkered pattern is shown in Section 4.2. Optimization of a few graded patterns is presented in Section 4.3. Lastly, we show a summarizing figure of the three main networks of this study in Section 4.4.

## 4.1 Parameters of Simulated Annealing

Simulations for illustration of SA parameters were performed with a system of $N = 12$ top nodes, and $L = 3$ layers. The $12 \times 3$ network was able to perform about 9000 iterations per second within the Cython framework. In pure Python, the same network had about 2700 iterations per second. A pure C code was also tested, and was able to perform up to 20000 iterations per second.

Figure 10, shows how the simulated annealing system develops when compared to the true resistor distribution in Figure 7, the E-symbol. The initial temperature is $T_0 = 1$, annealing time step $k = 100$ and temperature lowering $T_k = 0.95^k T_0$ for these results. The parameters of this run was set to show how some of the first trials went. The parameters are not optimal, but we can see some interesting features anyway. Firstly, we can see that the simulation system is able to shape out the top layers, and hint out that there is something in the middle of the system after about $t = 10000$ iterations. From $t = 10000$ to $t = 20000$, we can actually see the simulation evening out the surroundings, and shaping out the contours of an object. From $t = 20000$ to $t = 1000000$, 980000 iterations later, there are barely any improvement.

The effect initial temperature $T_0$ has on the simulation results is illustrated in Figure 11. All parameters are constant, except for the initial temperatures. Simulations were run with $T_0 = 0.1, 1, 10, 20, 50$ and 100, to illustrate the difference between too low, good, and too high initial temperatures. We can see that the simulation is able to approximate an object in the middle for all of the different initial temperatures, but the quality of the simulations appear dependent on $T_0$.

Annealing time step $k$ is illustrated in Figure 12. The annealing time steps used are $k = 50, 100, 200, 300, 400$ and 500, with all other parameters constant. We can again see that the simulations are able to approximate the

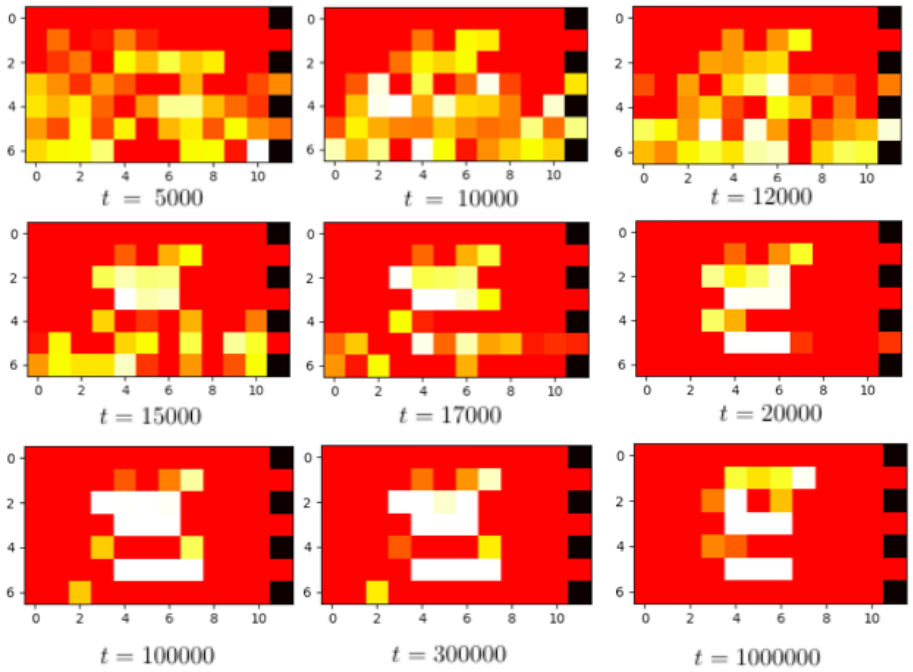



**Figure 10:** Development of the simulated annealing system through different run times $t$ as it attempts to approach the measured system with an E symbol in the middle. The other parameteres were constant at $T_0 = 1$, $k = 100$ and $T_k = 0.95^k T_0$.

object in the middle, but in some variable degree. The annealing time step $k$ appear to have an impact on the optimization procedure.
The second annealing schedule parameter, temperature lowering, is investigated in Figure 13. In this figure we have also included images from different run times $t$ to illustrate speed vs. accuracy. The temperature lowerings, $T_k = a^k T_0$, illustrated are $a = 0.85, 0.90, 0.95$ and $0.99$, at time $t = 100000, 200000$ and $500000$. All other parameters are constant. We can see that all simulations are able to approximate an object in the middle, but the slowest cooling $a = 0.99$ need more time to do so. The quickest cooling is able to find a good approximation after the least amount of time. However, the slowest one also seems to provide the best optimization.

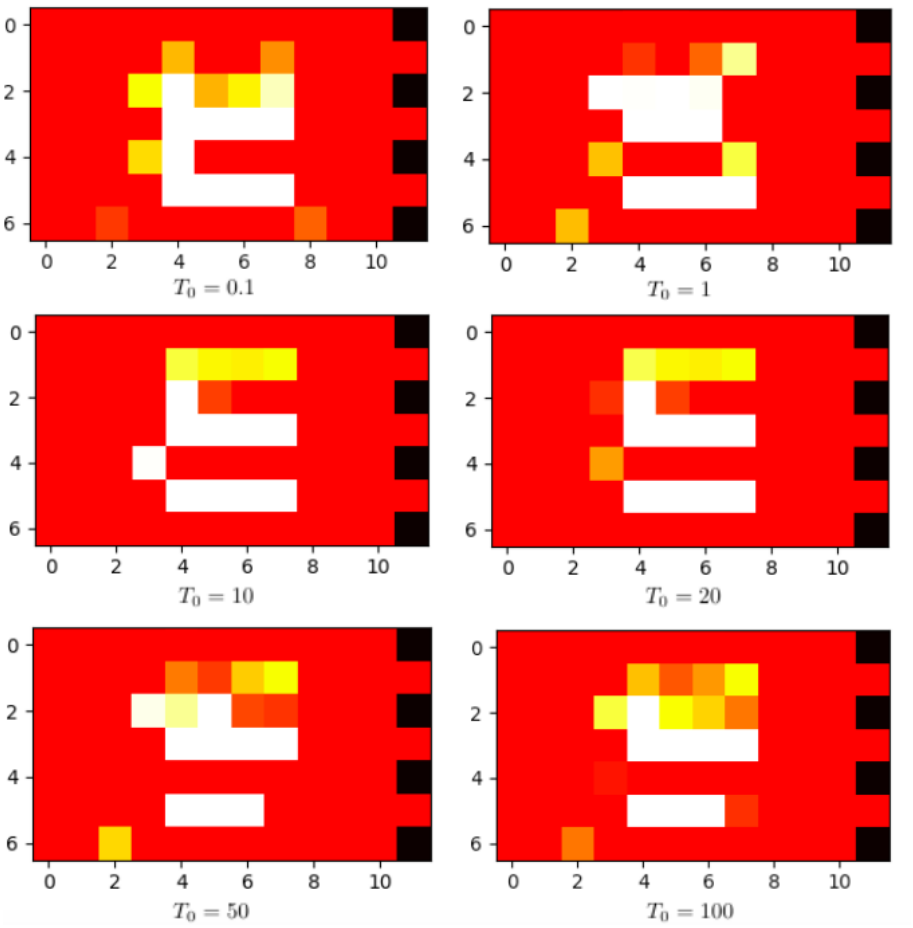



**Figure 11:** Simulations with different initial temperature $T_0$. Run time $t = 100000$, annealing step = 100 and temperature lowering $T_k = T_0\ 0.95^k$ were constant.

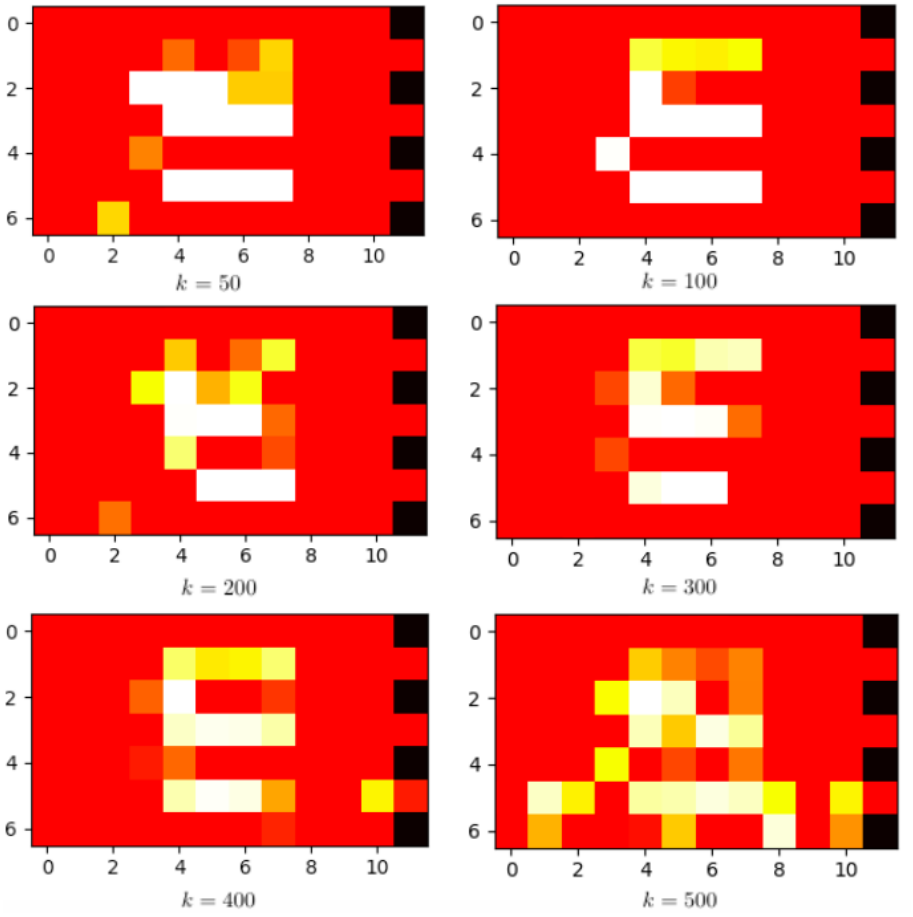



**Figure 12:** Simulations with different annealing time step $k$, $T_0 = 10$, run time $t = 100000$, and temperature lowering $T_k = T_0\ 0.95^k$.

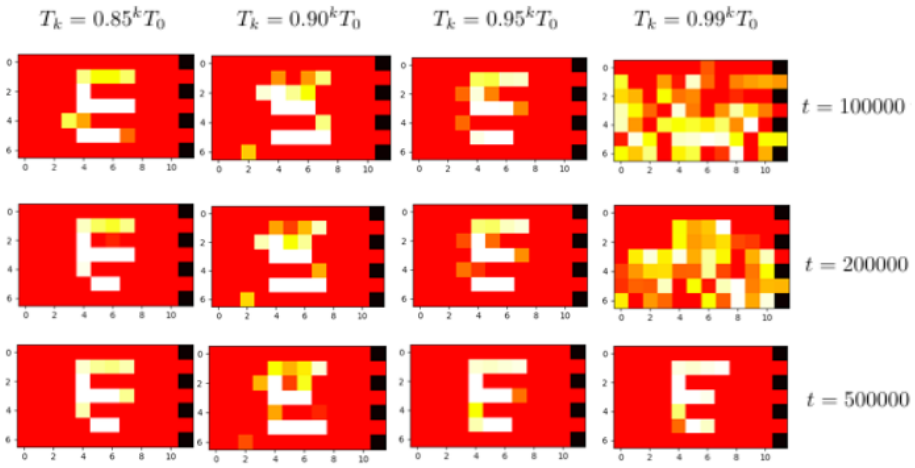



**Figure 13:** Simulations with different temperature lowering, different run times, annealing time step $k = 300$, initial temperature $T_0 = 10$. The rows are different run times $t$, and the columns are for different temperature lowering $T_k = a^k T_0$.

## 4.2 Checkered Pattern

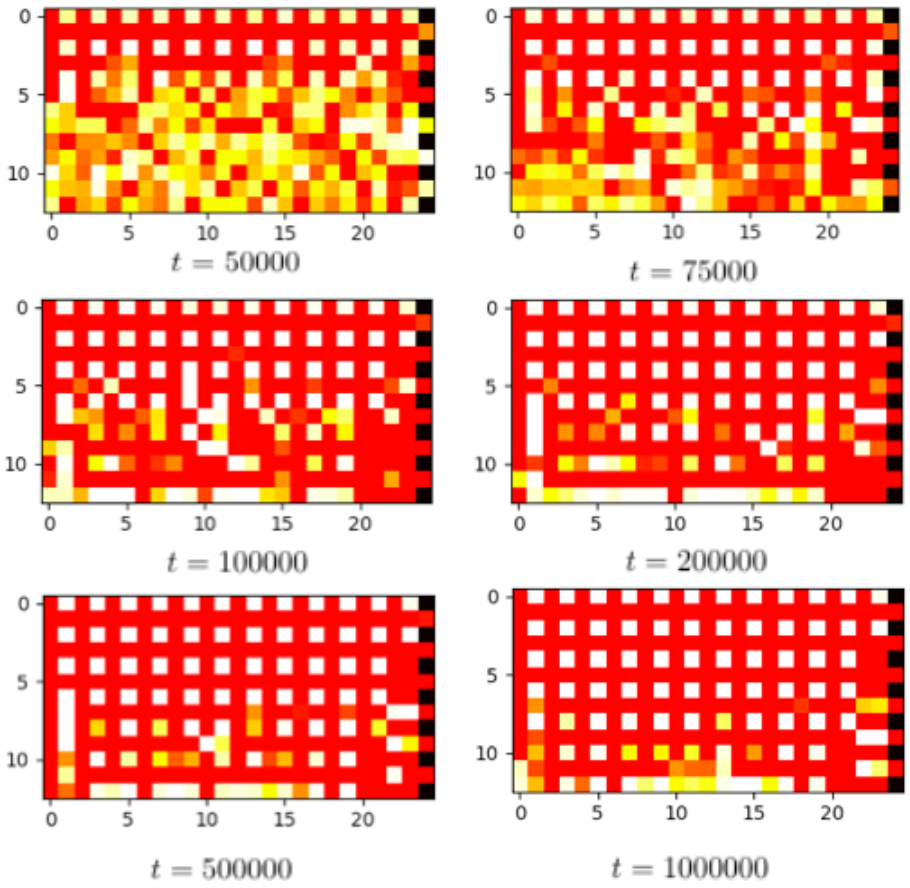



**Figure 14:** Simulation of a checkered pattern at different times $t$. Initial temperature $T_0 = 20$, annealing time step $k = 500$, and temperature lowering $T_k = 0.90^k T_0$ were constant.

The size of the network was extended, to see if simulated annealing was able to approximate solutions for larger networks and to compare time consumption. In addition, a checkered pattern was constructed at this size to provide the system with an additional challenge. The $25 \times 6$ network did about 3000 iterations per second within the Cython framework. A $60 \times 12$ network was tested for iteration speed only, and performed about 250 iterations per second.
Trials with different run times $t$, initial temperatures and annealing schedules were attempted, before producing the representative simulation in Fig-

ure 14. This means that the simulations does not necessarily show the best or the worst results, but it does show a good simulation of the checkered pattern. Parameters of the resulting run, was $T_0 = 20$, $k = 500$, and $T_k = 0.90^k T_0$ We can see in this figure that the annealing system quite quickly was able to approximate the measured system at the first few layers. At $t = 100000$ half the pattern was recognized. From this, it took longer and longer for the remaining network to be approximated. The image was close to complete at $t = 1000000$, but the system is not able to differentiate some of the few remaining resistors at the lower edges and bottom. Simulation attempts of another few million iterations showed almost no improvement, and the system can be said to be settled in an approximated optimum.

## 4.3 Graded Pattern

Lastly a graded pattern was attempted for network sizes of $25 \times 6$, $26 \times 6$, $27 \times 6$ and $28 \times 6$. The parameters was set as following: $T_0 = 50$, $k = 1000$, $T_k = 0.97^k T_0$, and run time $t = 1000000$. We can see in Figure 15 that the top two layers are comparable to the measured system, but after that the grading pattern melts away. The interesting thing is that we can see that where the grading vanishes, the contour of the measured system continues. The image quality can be said to be pretty bad and from the first glance does not look like the pattern of the measured system. However, some resemblance to the global patterns are made.

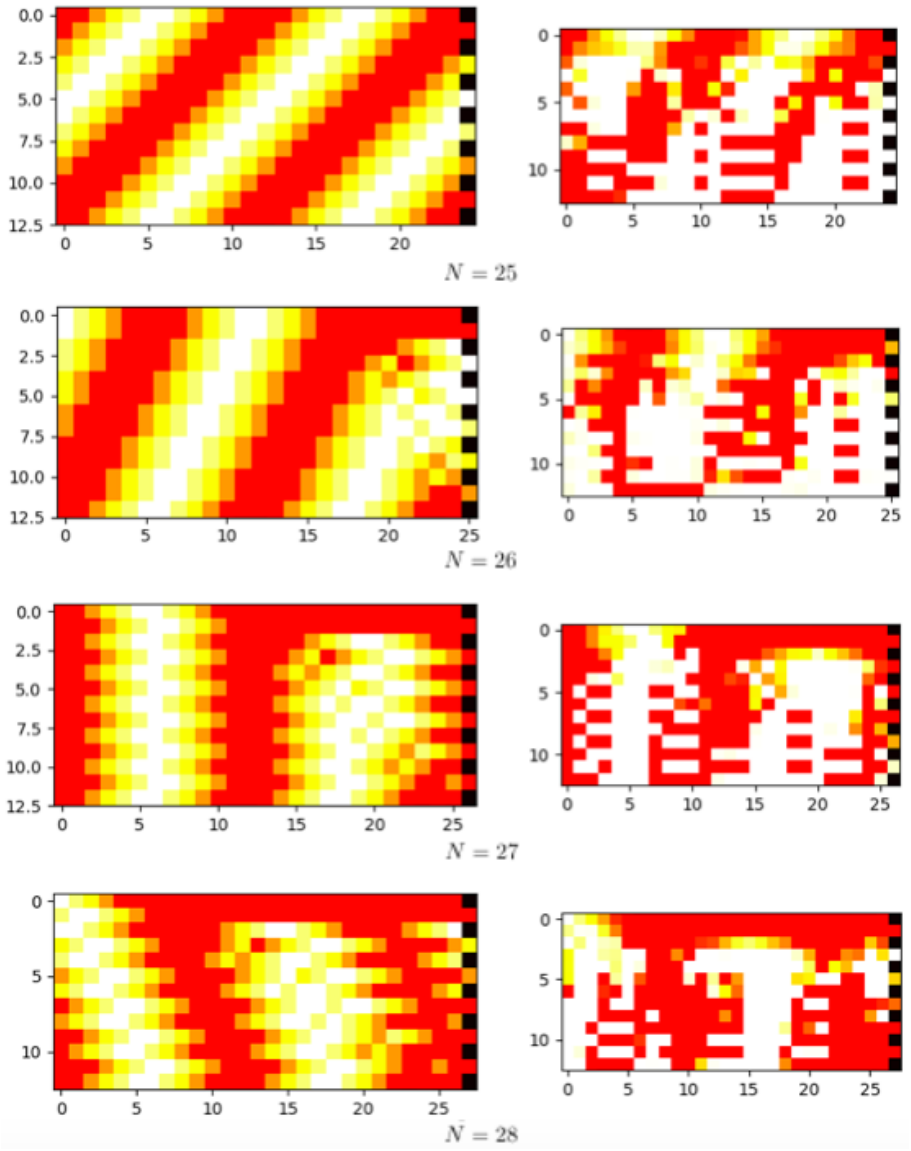



**Figure 15:** Simulations of a graded pattern for different sizes $N$, initial temperature $T_0 = 50$, annealing time step $k = 1000$, and temperature lowering $T_k = T_0\ 0.97^k$ and run time $t = 1000000$. Measured system on the left, and annealing system on the right.

## 4.4 Synopsis of the Modelling Optimizations

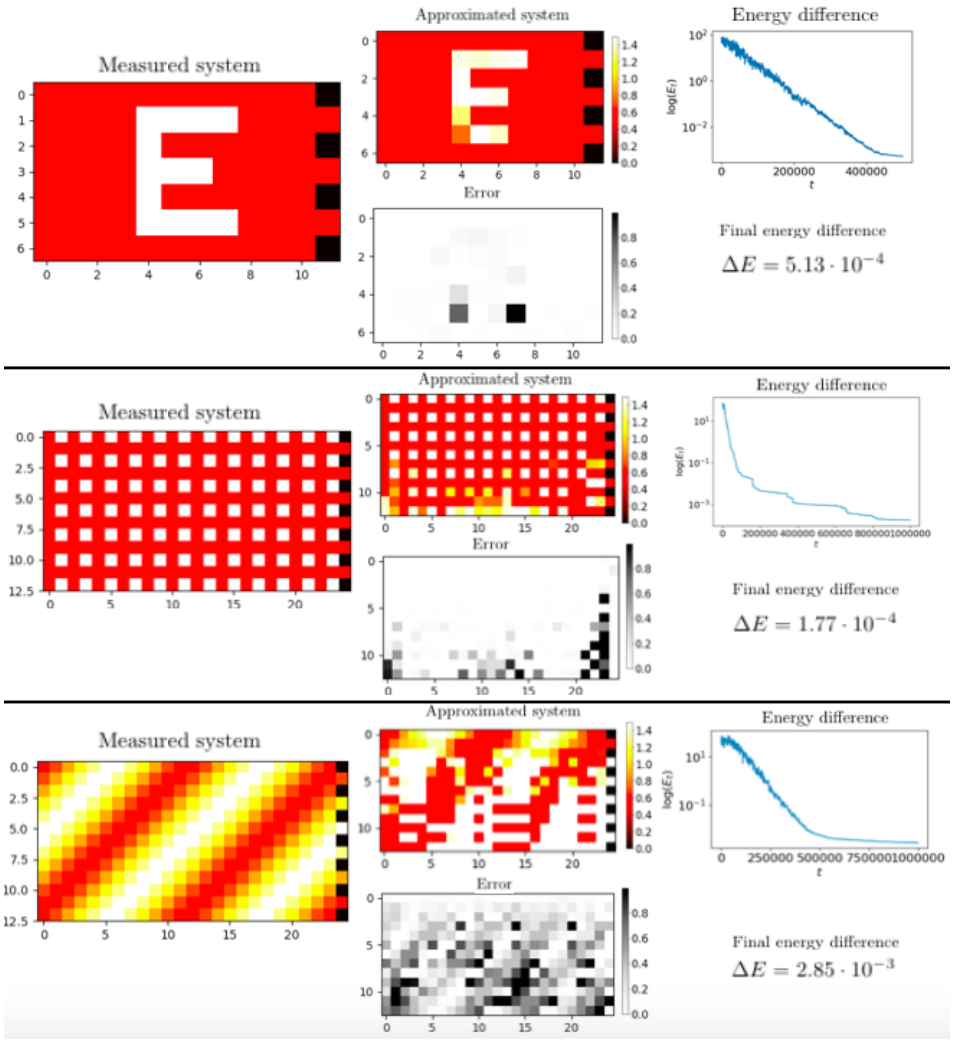



**Figure 16:** Comparison of simulations from the three different main network sizes. For each measured system we have: The approximated annealing system on the top left, with error below it. On the top right, the energy difference during the simulated annealing process, and below it, the final difference between the measured system, and the annealed system.

The results shown in Figure 16 are to summarize the optimization procedure performed on three of the main networks: E-symbol, checkered pattern and graded pattern. In this figure the measured system is shown together with a

final approximation, final error, energy difference throughout the run time, and the final energy difference. The parameters of the E-symbol was set as following: $T_0 = 10$, $k = 300$, $t = 500000$, $T_k = 0.99^k T_0$. Parameters of the checkered pattern was as following: $T_0 = 20$, $k = 500$, $t = 1000000$, $T_k = 0.90^k T_0$. The parameters of the graded pattern was as following: $T_0 = 50$, $k = 1000$, $t = 1000000$, $T_k = T_0\ 0.97^k$. We can see that there is a difference in how the systems are processed. The E-symbol energy difference goes down quite steadily, until it almost levels out around $t = 450000$. The checkered pattern had a faster annealing schedule and we can see the energy difference rapidly declining the first 100000. After this it goes into a slow descend, with sudden drops, until it levels out after about 800000 iterations. The graded pattern have a steady energy difference descend until 450000 iterations, where it rapidly almost levels out for the next 500000 iterations.

# 5 Discussion

In this section we will discuss the results presented in Section 4, and the implications of these results. We will also discuss the algorithm used in this study, how to improve it, and comparing it to the established method for finding a solution in electrical resistivity tomography.

## 5.1 Discussion of the Results

### 5.1.1 Parameters of Simulated Annealing

Optimization algorithms are known to be drawn to the first optimum it encounters. SA is acknowledged for its capability to not fall for greedy solution. Though, this does not mean it cannot be trapped by pseudo solutions. The parameters are critical to its efficiency to approximate the global optimum. Figure 10 illustrate a simulation run, with only loose consideration of the simulation parameters. We can see that the system is quite quickly able to find many of the correct resistors in the first few layers, and hint to an object in the middle. But from $t = 20000$ to $t = 100000$ there is barely any improvement. This perfectly illustrate that the system have got stuck in a local sub-optimal solution, and can hardly improve from there. This is possibly due to too low initial temperature, or too rapid annealing schedule.

The importance of initial temperature $T_0$ is illustrated in Figure 11. We can see that the figure is able to approximate an object in the middle for all the different initial temperatures, but the quality of the image is clearly dependent on $T_0$. For $T_0 = 0.1$ and $T_0 = 1$, the lower image qualities, suggests that the initial temperatures were too low, and the simulations got stuck in a local minima. For $T_0 = 10$ and $T_0 = 20$ the images are very close to the measured system, which suggest these initial temperatures were quite good. For $T_0 = 50$, and $T_0 = 100$ we can see that the images are again of lower quality. This is because the simulations have spent much of their time searching in the higher states, and haven't had time to properly settle for a minimization path. Given more run time $t$, they should be able to produce images of the same quality as $T_0 = 10$ and $T_0 = 20$. As the images for $T_0 = 10$ and $T_0 = 20$, became quite good, this also suggests that the annealing schedule was within a good range.

Figure 12 illustrates the impact of the annealing time step $k$. The annealing time steps $k = 50$ and $k = 200$, show that an object is found, but the

quality is low. This suggests that the systems cooled too rapidly, and got stuck in local minima. Annealing time step $k = 100$ shows an image of very high quality, but attempts with other seeds showed this to be an artifact. For $k = 500$, the system had not enough run time $t = 100000$ to settle in the object, and produced an image of low quality. Given more run time it would produced an image of higher quality, but at the expense of increased computational time. The best attempts were performed with $k = 300$ and $k = 400$, and a run time $t$ of 100000, which took about 40 seconds. This suggests that these are good annealing time step for the $12 \times 3$ network together with temperature lowering $T_k = T_0 0.95^k$. In this study, the annealing time step $k$ was defined as how many new configurations were suggested before changing the temperature. Another approach commonly utilized is to define $k$ as how many new configurations are accepted before changing temperature. Counting the rejected attempts is also a method for defining $k$, but is usually used as a stopping criterion (together with a temperature threshold) [19].

Figure 13 shows the development of the annealing system with different temperature lowerings $T_k = T_0 a^k$, at run time $t = 100000, 200000$ and 500000. We can see that the faster temperature lowerings, were able to find approximate solutions quite quickly, but can't improve much further after a certain number of iterations. For smaller and less complicated systems, a quicker annealing schedule can be almost as good as a slower one. For finding the best solution, longer run time $t$, and a slower temperature lowering is needed. This we can see from the difference between $a = 0.85$ and $a = 99$ in Figure 13. Several cooling schedules are mentioned in the literature and a few have their own name, such as Fast simulated annealing [20] and very fast simulated re-annealing [21].
We can from the figures in Section 4.1 see that the parameters of SA have an important impact on the optimization procedure. SA is not a straight forward algorithm, it is in fact a *metaheuristic*, and requires tweaking of the parameters.

### 5.1.2 Checkered Pattern

A larger network $25 \times 6$, with a checkered pattern was attempted to increase the difficulty of approximating the resistivity distribution. The larger system took a longer time to run, about 350 seconds, and required more time steps $t$ to find good approximations. Additionally, larger systems require larger annealing time step $k$ to successfully bring the system to an energy

equilibrium for each temperature lowering. Thus, contributing to a lengthened run time $t$. For this reason, just one of the simulations are shown in the results. This is nevertheless a representable optimization of this pattern. We can see that the pattern is quite recognizable, but some local resistivity is not quite mapped out. This will be discussed further in Section 5.2.2.

### 5.1.3 Graded Pattern

The graded pattern was believed to be very difficult for the simulations to distinguish each resistor element, and therefore produce inaccurate images. We can see in Figure 15 that this was somewhat correct. The top two layers are comparable to the measured system, but after that the grading pattern melts away. The interesting thing is that we can see that where the grading vanishes, the contour of the measured system continues. The image quality can be said to be pretty bad and from the first glance does not resemble the pattern of the measured system at all. However, from the point of recognizing structures in a disordered system ( such as a lump of metal in the soil), one could argue that the simulation is able to approximate and produce indications of an object. The issue of indeterminable resistivity is further discussed in Section 5.2.2.

### 5.1.4 Synopsis of the Modelling Optimizations

The final figure 16 shows some features of the simulation process and results. The E-symbol has a steady decline in energy difference, and is able to lock onto something that leads to a near-optimal approximation. A few resistors are not fully exposed, but the final energy difference and error mapping substantiates the claim of a good approximation. The checkered pattern seems to have a too rapid annealing schedule. The approximation is nevertheless able to lock on to a good approximation, and reveals most of the resistivity distribution. This pattern may not have been as challenging to distinguish as first believed by the author. The lower edges and bottom is again difficult to correctly differentiate, and will be further discussed in Section 5.2.2. The graded pattern gave the annealing system a real challenge. Local resistivity is poorly distinguished, however, it was able to produce a global mapping of the underlying resistivity pattern. This again is a discussion of indeterminability of local resistivity in Section 5.2.2. The annealing schedule seems to be too rapid, and levels out after only half of the run time. Slower annealing schedule, and increased run time might improve the approximation.

## 5.2 Further Discussion

### 5.2.1 Least Squares vs. Simulated Annealing

Least squares is a method for modelling data by minimizing the sum of squares of the difference between observed and estimated values. Newton-Gauss algorithm is a type of least squares, on which Lokes method is based on. For highly non-linear problems, this method requires constrains and regularizations. This can lead to posterior distributions that does not bring out the true underlying model, and important details can be lost. Arbitrary and subjective regulariszation may give solutions which are biased, and parts of the true model can be neglected or omitted. For larger macroscopic surveys of the subsurface this might not be a big problem, however for discovery or assessment of smaller details in e.g. the ground, finding such details would be an important quality of the method. Additionally, due to the ill-posedness of inversion problems, a vast amount of local optima appear as viable solution to local optimization methods, such as least squares. This requires a good initial model solution of the problem, which require a prior and/or numerous trial runs, for it to be able to approximate the global optimum. Least squares is a simpler and faster algorithm, and consequently led the practice of inverting geophysical field data to this method. From a time where the computer power was only a fraction of what it is today, running a heavy heuristic like SA on a desktop computer would have been redundant. Least squares therefore has decades of research, experimentation and optimization to back it up. A direct and fair comparison between these two methods are hence not easy to perform.
SA searches for the global optimum, and is found able to lock down on good approximated solutions to inverse problems, regardless of initialization. This can be seen on all the figures in the result section, since all initial configurations were random. Drawbacks of SA are its general and computationally heavy heuristic. The generality makes it non-trivial to implement, and customizations have to be considered for most new problems. The time consumption can be tremendous for large and complex problems. Given these two arguments together, puts SA on the lower end of the algorithm choice shelf, despite its powerful and intelligent framework.

### 5.2.2 Local Indeterminable Resistivity

Finding the values of resistors, only from measurements at the top layers, naturally have some obstacles. The point of doing several measurements at the top, is to be able to compare the probable pathing of current from

one node to another in the network, and hence estimate the resistivity. For each layer, the impact from local resistor on the apparent resistivity measurements are reduced, and finding the true resistivity becomes increasingly difficult. This is especially prominent on the lower edges, corners and bottom (see Figure 14 and 15). When approximating the resistivity, only by external measurements, the values of some resistors are easy to mix up, again especially in the lower layers. If we e.g. have two resistors in the bottom corner, one with a higher value, and the other one with a lower value. The approximation can just as likely average the high and low value of those two resistors, and thus the resistors effectively appear to be equal. Same goes for two equal resistors, that can appear as one high and the other low. This makes some of the resistors effectively indeterminable, and further from the top, the more likely this is to occur. We could observe the indeterminacy readily in the graded pattern in Figure 15, where differentiating the local resistor values quickly became difficult. However, the global shape of a pattern was still possible to distinguish.
In this study only a maximum of 6 layers were attempted to be estimated, and the lower levels still have enough impact on the apparent resistivity to be determinable. For larger systems, the difficulty of determining the resistivity of the lower levels, will likely increase to the impossible.

### 5.2.3 Algorithmic Procedure

This study was about proof of concept; the resistor values used was between 0.5 and 1.5, and for most approximations, either 0.5 or 1.5. This difference between smallest and largest is quite small, and when comparing the configurations as a whole, the differences becomes slight. In addition, the random number generator chooses values between 0.5 and 1.5, which in practice is an infinite amount of numbers, and makes it impossible to hit the absolute correct value. On the other hand, the small difference between the resistors, made it faster and easier than if the difference was several thousands. In Figure 15, we could see that the small contrast between the resistor elements, made it hard to distinguish single resistor values from each other.
There are three factors which make this algorithm inefficient:

- Switching out one resistor at a time.
- Random number generator can choose an infinite amount of values between the highest and lowest resistor value.
- Reconstruction of the transfer-matrix for every iteration.

The annealing system was run for up to a million iterations to approximate the checkered system. The system was $25 \times 6$, consisting of a total of 294 resistors. New candidate configurations are generated by switching out one resistor, with a random valued resistor. When combined with the random number generator which can choose any value between 0.5 and 1.5, we can see that the approximation is deemed to take some time. A more efficient candidate generator, with a finite number of random resistor values to choose from between the highest and lowest values could potentially increase the performance of the algorithm.
For every new resistor switched out to generate a new candidate configuration, the transfer-matrix has to be rebuilt from the first to the last resistor in the network. This is a algorithmic process that contains several loops, and a double for-loop, which have the iterate over the whole $N$ width and $L$ depth of the network. This creates a dense matrix, which is computationally heavy to work with. This has to be done for every iteration of the annealing process, and is without doubt the most time consuming part of the algorithm. Improving the processing of the transfer-matrix would be highly beneficial for the efficiency of the algorithm. When only one resistor is switched out in the network, only the proximal resistivity is affected, and thereby only part of the transfer-matrix. Instead of rebuilding the transfer-matrix, one should rebuild only the affected parts of it. Other improvements to the transfer-matrix could be to modify or decompose it to more sparse and manageable matrices.
The transfer-matrix is in this study used to store the model parameters of the configuration states. The objective evaluation function $E$ uses the transfer-matrix to compare the annealing configuration to the measured system. There could be other and better methods for storing and/or comparing the configuration states.
Simulated tempering is a modified version of SA, which consider model copies at different temperatures, and picks the most viable. It is shown to increase the accuracy of the annealing algorithm [23].

### 5.2.4 Time Consumption

Simulated annealing is infamous for its extensive time consumption. The time consumption can be a good reason for deselecting simulated annealing, despite showing promising results. In this study, the time spent approximating a solution quickly became a problem. The algorithm for producing the transfer-matrix, requires it to "build" the resistor network from the bottom and up, for every new configuration candidate within the simulated

annealing algorithm. The construction of the transfer-matrix consists of two for-loops, one of which is a double for-loops to span the $N$ width, and the $L$ layers of the network. Therefore, for larger models the computational time increased significantly. A few basic optimizations were performed to increase the iteration process, such as switching out a few for-loops, and using Cython as a C-extension to the Python code. Performance of the Cython program was about 3.5 times faster. A pure C code was also tested for the purpose of speed comparison of constructing the transfer-matrix, and gave another 2.2 times speed up. A network of $N > 100$ is desired for practical applicability, but this would take days to approximate even in pure C language framework. A note on the performance should be stated: the simulations were performed on a laptop with 1,4 GHz Intel Core i5 processor, and performance could greatly benefit from more dedicated processing power (stationary computer, computer cluster, cloud computing, supercomputer).

Parallelization of the annealing process has shown to efficiently decrease the time consumption, but due to the fact that simulated annealing is a naturally sequential algorithm makes it non-trivial to implement [22].
Genetic algorithm (GA) is a global optimization method, where a pool of configurations, rather than just one are evaluated. New candidate configurations are not only generated by making new configurations (as in SA), but also by combining configurations from the pool. Evaluation of the configurations are done in the similar manner as in simulated annealing. Accepted candidates go into the configuration pool, and excess configurations are discarded from the pool in an accept/reject-probability manner. SA-GA hybrid has shown to be more effective in identifying global optimum than SA and GA alone, and additionally, easy to ameliorate through parallelization [24].

# 6 Conclusion

The purpose of this thesis was to investigate if it is possible to approximate the resistor distribution of a resistor network, given an apparent resistivity measured at the top layer of the system. For this purpose, we built a random resistor network, used transfer-matrix method to store and compare the apparent resistivities, and simulated annealing to approximate the measured system. Results from this study show that the annealed system was able to approximate the measured system with resistors valued between 0.5 and 1.5 up to the size of $N = 25$ top nodes and $L = 6$ layers. However, some problems with indeterminable resistors were observed. Especially patterns where resistors are gradually changing values made it challenging to estimate local resistivity, but a global distinction of the pattern was still visible. What have been provided in this thesis may be seen as a proof of concept for approaching resistivity distribution calculations, using simulated annealing. An obstacle for investigation of bigger systems were that the annealing system is incredibly time consuming. Improvement of the algorithm should be considered to achieve a competitive method for solving non-linear inverse problems.

## 6.1 Future Research

The results presented in Section 4 show promising outlook for utilizing SA as optimization method for processing data of highly non-linear inverse problems. However, the algorithm developed in this study was mainly constructed for conceptual proof. Improvement of the algorithm for faster and more accurate computations should be considered for further research. Several optimizations and performance improvements was discussed in Section 5. These could be subject for further work, and include:

- Increase the efficiency of transfer-matrix construction, or modified transfer-matrix.
- How the new configuration candidates are generated.
- Modified SA methods, e.g. simulated tempering, SA-GA hybrid.
- Parallelization.

The simulations were performed as a conceptual proof. This raised an important next question: Is this method applicable to real conduction systems? To answer this, further research and experimentation is needed.

Electrical networks have numerous physical analogies. A few selected analogous systems are provided in the Appendix A.2. An interesting question is whether it could be possible to transfer this method, and perform simulations on analogous systems.

Investigation of these subjects could possibly lead to innovative methods for mapping of internal structures.

# References

[1] Lataste, J. F., et al. Electrical resistivity measurement applied to cracking assessment on reinforced concrete structures in civil engineering. NDT and E International 36.6 (2003): 383-394.

[2] Borcea, Liliana. Electrical impedance tomography. Inverse problems 18.6 (2002): R99.

[3] M. H. Loke, Tutorial : 2-D and 3-D electrical imaging surveys, course note, University of Alberta, (2004)

[4] A. Samouëlian, I. Cousin, A. Tabbagh, A. Bruand, G. Richard, Electrical resistivity survey in soil science: a review. Soil and Tillage research, 83(2), 173-193. (2005)

[5] J. E. Chambers, O. Kuras, P. I. Meldrum, R. D. Ogilvy, J. Hollands, Electrical resistivity tomography applied to geologic, hydrogeologic, and engineering investigations at a former waste-disposal site. Geophysics, 71(6), B231-B239. (2006)

[6] M. Noel, B. Xu, Archaeological investigation by electrical resistivity tomography: a preliminary study. Geophysical Journal International, 107(1), 95-102. (1991)

[7] M. S. Zhdanov, Inverse Theory and Applications in Geophysics, (2015)

[8] M. H. Loke, R. D. Barker, Rapid least-squares inversion of apparent resistivity pseudosections by a quasi-Newton method. Geophysical prospecting, 44(1), 131-152, (1996)

[9] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, Optimization by Simulated Annealing, (1983), Science, Vol. 220, No. 4598, pp. 671–680.

[10] A. Dey, H.F Morrison, Resistivity modelling for arbitary shaped two-dimensional structures, Geophysical Prospecting 27(1):106 - 136, (1979)

[11] B. Derrida, J Vannimenu, A transfer-matrix approach to random resistor networks, J. Phys. A: Math. Gen. 15 L557-L564, (1982)

[12] L. Borcea, V. Druskin, F. Guevara Vasquez, A.V. Mamonov, Resistor network approaches to electrical impedance tomography, ArXiv e-prints 1107.0343, (2011)

[13] D. J. Griffiths, Introduction to electrodynamics, third edition, (1999)

[14] B. Derrida et al., A Transfer Matrix Program to Calculate the Conductivity of Random Resistor Networks, Journal of Statistical Physics, Vol. 36, Nos. 1/2, (1984)

[15] D. Gerenrot,L. Berlyand, J. Phillips, Random network model for heat transfer in high contrast composite materials. IEEE transactions on advanced packaging, 26(4), 410-416.(2003)

[16] Z. Wu et al., Current flow in random resistor networks: the role of percolation in weak and strong disorder, Phys Rev E Stat Nonlin Soft Matter Phys. 2005 Apr;71(4 Pt 2):045101. Epub 2005 Apr 7.

[17] L. Onsager, Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition, Phys. Rev. 65, 117 – Published 1 February (1944)

[18] W. H. Press, S.A Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes* 3rd Edition: The Art of Scientific Computing, third edition, Cambridge University Press, ISBN 978-0-521-88068-8, (2007)

[19] H. Orsila, E. Salminen, T. D. Hämäläinen, Best Practices for Simulated Annealing in Multiprocessor Task Distribution Problems, Department of Computer Systems, Tampere University of Technology https://pdfs.semanticscholar.org/fcd2/ec53ef145d031cccd0d3afeedddc73f7b4e3.pdf (Visited 12.06.2018)

[20] H. Szu, R. Hartley, Fast simulated annealing. Physics letters A, 122(3-4), 157-162,(1987)

[21] L. Ingber, Very fast simulated re-annealing. Mathematical and computer modelling, 12(8), 967-973, (1989)

[22] Z. Wang, Y. Zhao, Y Liu, C Lv, Speculative parallel simulated annealing algorithm based on Apache Spark, https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4429, (2018)

[23] E. Marinari, G. Parisi, Simulated Tempering: A New Monte Carlo Scheme, EPL (Europhysics Letters), Volume 19, Number 6, (1992)

[24] D.Chen, C. Lee, C. Park, P. Mendes, Parallelizing simulated annealing algorithms based on high-performance computer, J Glob Optim 39:261–289, (2007)

# A Appendix

## A.1 Cython Code for the Numerical Methods

```
import random
import copy
import matplotlib.pyplot as plt
import numpy as np

cimport numpy as np
cimport cython

DTYPE = np.float64
ctypedef np.float64_t DTYPE_t


random.seed(12345)

cdef int N = 25  # number of top nodes, a
resistor is placed between two nodes
cdef int L = 6   # number of vertical resistor layers
cdef float low = 0.5 # lowest resistor value
cdef float high = 1.5 # highest resistor value

cdef int l


A_matrix = np.zeros((N,N))

#Construction of a measured transfer matrix A
def build(np.ndarray[DTYPE_t, ndim=2] matrix):
    cdef int i,l,n,j

    cdef np.ndarray[DTYPE_t] R_start_h =
    np.ones((N - 1) * L + (N - 1))*low
    cdef np.ndarray[DTYPE_t] R_start_v = np.ones(N*L)*low

    #To make checkered pattern
    for i in range(1,(N - 1) * L + (N - 1),2):
        R_start_h[i] = high
```

```
    cdef np.ndarray[DTYPE_t, ndim=2] A_matrix = np.zeros((N,N))
    cdef np.ndarray[DTYPE_t, ndim=2] A = np.zeros((N,N))

    for i in range(N - 1):
        A_matrix[i,i] += 1 / R_start_h[i]
        A_matrix[i,i + 1] += -1 / R_start_h[i]
        A_matrix[i + 1,i] += -1 / R_start_h[i]
        A_matrix[i + 1,i + 1] += 1 / R_start_h[i]

    for l in range(L):
        for n in range(N):
            A = copy.copy(A_matrix)
            for i in range(N):
                for j in range(N):
                    A_matrix[i,j] = A[i,j] -
                    ( (A[i,n] * A[n,j] *
                    R_start_v[l*N+n]) /
                    (1 + A[n,n]*R_start_v[l*N+n]))

        for i in range(N-1):
            A_matrix[i,i] += 1/R_start_h[l*(N-1)+i+(N-1)]
            A_matrix[i,i+1] += -1/R_start_h[l*(N-1)+i+(N-1)]
            A_matrix[i+1,i] += -1/R_start_h[l*(N-1)+i+(N-1)]
            A_matrix[i+1,i+1] += 1/R_start_h[l*(N-1)+i+(N-1)]
    return A_matrix, R_start_h, R_start_v

A_matrix, R_start_h, R_start_v = build(A_matrix)

start_resistors = np.zeros((2*L+1, N))

for l in range(N*L):
    start_resistors[2*L-1-2*(l/N % N),l%N] = R_start_v[l]
for l in range((N-1)*L+(N-1)):
    start_resistors[2*L-2*(l/(N-1)),l%(N-1)] = R_start_h[l]


@cython.boundscheck(False)
@cython.wraparound(False)
```

```
#Simulated annealing algorithm
def SA(np.ndarray[DTYPE_t, ndim=2] matrix):
    cdef int t = 0
    cdef double E0 = (N+L)**2
    cdef double T = 20
    cdef int update = 0

    cdef np.ndarray[DTYPE_t] R_trial_h =
    np.random.uniform(low,high,(N - 1) * L + (N- 1))
    cdef np.ndarray[DTYPE_t] R_trial_v =
    np.random.uniform(low,high,(N*L))
    cdef np.ndarray[DTYPE_t] R_best_h = copy.copy(R_trial_h)
    cdef np.ndarray[DTYPE_t] R_best_v = copy.copy(R_trial_v)


    cdef double random_resistor_value, E1,p,alpha,probability_number
    cdef int random_position
    cdef int a,l,n,j
    cdef np.ndarray[DTYPE_t, ndim=2] A_matrix_trial = np.zeros((N,N))
    cdef np.ndarray[DTYPE_t, ndim=2] A = np.zeros((N,N))
    cdef np.ndarray[DTYPE_t, ndim=2] U_matrix = np.zeros((N,N))

    while t < 500000:
        random_resistor_value = random.uniform(low, high)
        random_position = random.randrange(0,(2*N-1)*L+(N-1))
        if random_position >= ((N-1)*L+(N-1)):
            R_trial_v[random_position-((N-1)*L+(N-1))]
            = random_resistor_value
        else:
            R_trial_h[random_position] = random_resistor_value

        A_matrix_trial = np.zeros((N,N))

        for a in range(N-1):
            A_matrix_trial[a,a] += 1/R_trial_h[a]
            A_matrix_trial[a,a+1] += -1/R_trial_h[a]
            A_matrix_trial[a+1,a] += -1/R_trial_h[a]
            A_matrix_trial[a+1,a+1] += 1/R_trial_h[a]

        for l in range(L):
```

```
            for n in range(N):
                V_matrix[n,n] = 1 / R_trial_v[l*N+n]
            A_matrix_trial = A_matrix_trial.dot(np.linalg.solve
            (V_matrix + A_matrix_trial, V_matrix))

            for a in range(N-1):
                A_matrix_trial[a,a] += 1/R_trial_h[l*(N-1)+a+(N-1)]
                A_matrix_trial[a,a+1] += -1/R_trial_h[l*(N-1)+a+(N-1)]
                A_matrix_trial[a+1,a] += -1/R_trial_h[l*(N-1)+a+(N-1)]
                A_matrix_trial[a+1,a+1] += 1/R_trial_h[l*(N-1)+a+(N-1)]

        E1 = np.linalg.norm(A_matrix-A_matrix_trial)**2
        p = np.exp(-(E1-E0)/T)
        alpha = min(1,p)
        probability_number = random.random()

        if alpha > probability_number:
            E0 = E1
            R_best_h = copy.copy(R_trial_h)
            R_best_v = copy.copy(R_trial_v)
            update += 1
        else:
            E1 = E0
            R_trial_h = copy.copy(R_best_h)
            R_trial_v = copy.copy(R_best_v)

        if update != 0 and t % 300 == 0:
            T = T*0.95
        t += 1
    return A_matrix_trial, R_best_h,R_best_v,T

A_matrix_trial, R_best_h,R_best_v, T = SA(A_matrix)

#Make an image matrix
resistors = np.zeros((2*L+1, N))

for l in range(N*L):
    resistors[2*L-1-2*(l/N % N),l%N] = R_best_v[l]
for l in range((N-1)*L+(N-1)):
    resistors[2*L-2*(l/(N-1)),l%(N-1)] = R_best_h[l]
```

```
fig, (ax0,ax1) = plt.subplots (figsize = (10,5), ncols=2)
im1 = ax0.imshow(start_resistors, cmap="hot", interpolation="nearest")
im2 = ax1.imshow(resistors, cmap="hot", interpolation="nearest")
plt.colorbar(im2)
plt.show()
```

## A.2 Analogies to Electrical System

A few selected analogies to electrical systems, with corresponding parameters:

**Table 1:** Corresponding parameters to an electrical system.

| Electrical | | Fluid | | Thermal | | Mechanical | |
|---|---|---|---|---|---|---|---|
| Voltage | $V$ | Pressure | $P$ | Temperature | $T$ | Velocity | $u$ |
| Conductance | $G$ | Mobility | $M$ | Conductance | $\kappa$ | Stiffness | $k$ |
| Current | $I$ | Volume flux | $Q$ | Thermal power | $\dot{Q}$ | Force | $F$ |