POWER GRID CORRECTION USING SENSITIVITY ANALYSIS UNDER AN RC MODEL

by

Pamela Al Haddad

A thesis submitted in conformity with the requirements for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

Copyright © 2011 by Pamela Al Haddad
Abstract

Power Grid Correction Using Sensitivity Analysis Under an RC Model

Pamela Al Haddad

Master of Applied Science

Graduate Department of Electrical and Computer Engineering

University of Toronto

2011

Verifying the power grid requires one to check if the voltage drops on all the nodes of the grid do not exceed a certain threshold. The goal of this thesis is to correct the grid in case some voltage drops violate the threshold condition, by making minor changes to the original design. Previous work has been done in [1] on the DC model of the grid and this thesis deals with the $RC$ model. We first propose an accurate approach to correct the grid which turns out to be too slow. We therefore propose another approach, much more suitable for large grids and which can be summarized as follows. The voltage drop is estimated as a function of the metal widths on the grid. A non-linear optimization problem is then formulated and the required metal line width changes that reduce the voltage drops by a sufficient amount are determined. There is a tradeoff between accuracy and speed of the algorithm. However the results show that the gain in speed achieved by the second method, outweighs greatly the loss in accuracy. We therefore recommend it for correcting large grids.
Acknowledgements

First, I would like to thank my advisor, Professor Farid N. Najm for being a great mentor and providing an endless support on different levels not only research. Without his guidance time and supervision, none of this work would have been achieved. I am lucky and proud to be his student.

I would also like to thank professors Andreas Veneris, Jason Anderson and Costas Sarris who spent time reviewing this thesis.

I extend my gratitude to my fellow graduate students Ankit Goyal, Mehmet Avci, Nahi Abdul Ghani and Sari Onaissi. Having them in my group helped me in successfully completing my thesis. A special thanks to Meric Aydonat. She has been a great friend and colleague and made my first year of MASc smooth and enjoyable.

Women grad students, Myrto Pappadoulo, Iona Burcea, Danyo Wang, Maryam Alvandi and Natalie Enright Jerger. Our get togethers were always a pleasant escape from work.

A special thank you to all the nice people I met in Toronto. You made my stay unforgettable.

Finally my greatest thanks are to my parents Nawal Hokayem and Gen. Dr. Mansour Al Haddad, and my siblings Christelle and Michel. Without their continuous support and encouragement I would not have been here.
Contents

1 Introduction ........................................... 1
   1.1 Motivation and Objective ................................. 1
   1.2 Thesis Organization ....................................... 3

2 Literature Review ....................................... 4
   2.1 Chowdhury and Breuer ................................... 4
      2.1.1 The objective Function ............................... 6
      2.1.2 The constraints ....................................... 6
      2.1.3 Survey of Applicable General Optimization Methods .. 10
   2.2 Jakushokas and Friedman ................................. 10
   2.3 Dutta and Sadowska ....................................... 11
   2.4 Tan, Shi and Lee ......................................... 11
      2.4.1 Problem formulation ................................. 12
      2.4.2 The constraints ....................................... 13
      2.4.3 General Optimization Procedure ........................ 14
   2.5 Conclusion .............................................. 15

3 Power Grid Verification and Linear Programming Review .......... 16
   3.1 Power Grid Model ....................................... 16
   3.2 Current Constraints ..................................... 17
   3.3 Transient Robustness .................................... 18
4 Grid Correction

4.1 A First Approach: $V_\infty$ as an Objective Function

4.1.1 First Time Step

4.1.2 Steady State Voltage Drop Estimation

4.1.3 Correction of the Offending Node

4.1.4 Top Level Algorithm

4.1.5 Experimental Results

4.2 A Second Approach: $V_a$ as a Proxy Objective Function

4.2.1 Nonlinear Optimization Problem

4.2.2 Top level algorithm

4.2.3 Experimental Results

5 Conclusion

5.1 Future Work

Appendices

A Line Search Methods

A.1 Standard Search Directions

A.2 Step Length

A.2.1 Cubic Interpolation

Bibliography
List of Tables

4.1 Required width changes for the case of 10 parameters (Method 1 applied) 34
4.2 Required width changes for the case of 10 parameters (Method 2 applied) 34
4.3 Maximum steady state voltage drops on the grids before and after correction and the run-times of the overall algorithm (Method 1 applied) . . . 34
4.4 Maximum steady state voltage drops on the grids before and after correction and the run-times of the overall algorithm (Method 2 applied) . . . 34
4.5 Required width changes for the case of 10 parameters (Method 2 applied) 40
4.6 Required width changes for the case of 10 parameters after rounding (Method 2 applied) .................................................. 41
4.7 Maximum steady state voltage drops on the grids before and after correction and the run-times of Algorithms 1 and 2 (Method 2 applied) . . . . 41
List of Figures

2.1 A graph topology for a power grid [3] ................................. 5

4.1 Run-Time of the overall algorithm as a function of the number of nodes (Method 1) .......................................................... 35

4.2 Number Of Simplex algorithm runs needed to correct the grid for method 1 and method 2 ....................................................... 35

4.3 Variation of the worst case first time step voltage drop at node $k$ and its threshold with respect to the number of Simplex solutions in a 598 node $RC$ grid ......................................................... 39

4.4 Correlation plot of the steady state voltage drops before and after correction for a 1594 node grid ............................................. 40

4.5 Steady state voltage drop histograms before and after correction for a 70421 node grid. All drops have been reduced below the threshold of 100mV ................................................................. 43
Chapter 1

Introduction

1.1 Motivation and Objective

With technology scaling, the supply and threshold voltages are decreasing. Hence, voltage drops on the power grid become more significant and can result in longer circuit delays, leading to soft errors. Checking the integrity of the voltage on the power grid has become crucial in reliable chip design.

Power grid verification via traditional circuit simulation, requires full knowledge of the current waveforms drawn by every circuit block attached to the grid. Once these waveforms are known, the grid is simulated and the voltage drop at every node is determined. This voltage drop is then compared to a threshold to check if any node on the grid is unsafe. Verifying the grid using this approach requires the simulation of a comprehensive set of currents and full knowledge of the circuit. The latter is problematic if one would like to do the verification early in the design flow, before all the circuit details are available, which is typically the case.

A new approach was proposed in [8] to deal with these problems. It is based on the current constraints concept to capture the uncertainty about the circuit details and circuit behavior. These current constraints can be obtained from simulations of the
circuit or from knowledge of the overall power dissipation of the circuit blocks. They are a set of upper bounds on the currents drawn by the underlying circuit. A linear program (LP) is then formulated using these constraints, to check if the voltage drop at any of the nodes exceeds a certain threshold under all possible current waveforms that satisfy the constraints. In case none of the nodes violates its voltage drop requirement, the grid is said to be robust. Unlike the simulation based approach, the current constraints approach allows the verification to be done early in the design process when grid modifications can be most easily incorporated.

Once the grid verification is completed, we may have some nodes violating the voltage threshold. It then becomes useful to be able to make some minor changes on the grid so that it becomes safe without having to redo the design from scratch. A method has been presented in [1] to correct an $R$ model of the grid. This work presents two methods to correct an $RC$ model of the grid. In the first method, we formulate a nonlinear optimization problem with the steady state voltage drop (which will be defined later in the thesis) as the objective function to be minimized. However that algorithm is slow, which makes it unsuitable for large grids. We therefore devise a second method, a bit less accurate but with a significant improvement in speed. It can be summarized as follows. Rather than working on reducing the steady state voltage drops below the threshold in a direct way, we work with the first time step voltage drops (defined later in the thesis). This is possible and in fact much easier because of the relation between the upper bound on the peak steady state voltage drop and the first time step voltage drop. As we will see, reducing the first time step voltage drop also reduces the steady state voltage drop. The correction of the grid will be achieved by doing minor changes to the widths of some metal branches.

The second method presented here uses linear programming concepts to express the first time step voltage drops as a function of the width parameters. It then uses a nonlinear optimization method to find the width changes that cause a sufficient reduction
to the first time step voltage drop. The steady state voltage drops are consequently reduced. A preliminary version of this work will appear in [6].

1.2 Thesis Organization

This thesis is organized as follows. Chapter 2 briefly reviews the existing literature dealing with optimal widths determination in a power grid. Chapter 3 defines the grid model and the constraint based approach used in our grid verification. It also explains the concept of transient robustness and introduces some basic linear programming concepts that are useful in the grid correction algorithm. Two approaches to correcting the grid are presented in chapter 4 along with their experimental results. Chapter 5 gives some concluding remarks and some possible extensions to the current work. Finally, the appendix describes a few mathematical concepts used in the thesis.
Chapter 2

Literature Review

A VLSI chip is composed of logic modules that have an internal power and ground distribution system. This distribution system can be problematic because current flow can cause metal migration and undesired voltage drops. Hence, in order to lower the current densities and voltage drops, the power and ground routes are made much wider than the signal routes.

In the rest of the thesis, the topology of the power grid is assumed to be known and the problem boils down to assigning the widths to each branch of the power grid so that it is robust without having an unnecessary increase in the area. Interested readers are referred to [11] and [17] for methods on constructing these topologies.

In this chapter, we will review some previous work done in [3], [9], [19] and [16] to determine the optimal widths of the power grid wires. In most cases, the widths are found by solving a nonlinear optimization problem, where the total route area is the objective function to be minimized subject to certain technological constraints.

2.1 Chowdhury and Breuer

This section reviews the method devised by Chowdhury and Breuer in [3] to determine the optimal widths of the power grid wires.
A graph topology for a power distribution system is depicted in Fig. 2.1. Each edge (labeled by the boldface integers) of this graph is a continuous constant width segment and at any time, the current in this segment is the same at any section of the segment.

The nodes of the graph (labeled by lower case integers) can be the power pad, the power pin on a module, or the intersection of two or more edges. Two node types are distinguished: 1) the leaf nodes which represent the power pins on the modules and 2) the nonleaf nodes which represent the rest of the nodes. The voltage drop $v_i$ across edge $i$ is expressed by $v_i(t) = r_i I_i(t) = \rho I_i(t) l_i / w_i$ with $r_i$, $I_i$, $w_i$ and $l_i$ representing respectively the resistance, the current flow, width and length associated with edge $i$, which lies on a layer of sheet resistivity $\rho$. The current accounts for static and transient current through

---

**Figure 2.1:** A graph topology for a power grid [3].
edge $i$ and is therefore time dependent.

### 2.1.1 The objective Function

The objective function [3] is the sum of the area of the individual routes ($i.e.$ edges) which can be expressed as:

$$ A = \sum_{i=1}^{n} l_i w_i = \sum_{i=1}^{n} \rho l_i^2 / r_i $$  \hspace{1cm} (2.1)

where $n$ is the total number of edges. For the graph shown in Fig. 2.1:

$$ A = \sum_{i=1}^{13} \rho l_i^2 / r_i $$  \hspace{1cm} (2.2)

### 2.1.2 The constraints

There are many constraints that Chowdhury and Breuer considered in [3]. The *voltage drop constraints* ensure correct logic operation and switching speed. *Metal migration constraints* are related to the expected life of the chip. *Minimum widths of wires* constraints are imposed by chip fabrication technologies; in some cases the width has to be a multiple of some constant $\lambda$. Other constraints called *coupling constraints* are due to some regularity considerations. The remaining constraints are imposed by Kirchoff’s current and voltage laws.

**Voltage drop constraints** An nmos transistor is turned ON if the gate to source voltage $v_{gs}$ is greater than the voltage threshold $v_{th}$. Consider the case where the output of an ON transistor is connected to the gate of another transistor. In case of a considerable voltage drop in the ground line between these 2 transistors, then $v_{gs}$ of the second transistor might exceed the threshold value and thus the device might be turned ON by mistake.

Therefore, the voltage drop on the ground lines between communicating transistors needs to be restricted. Similarly, we need to restrict the voltage drop on the power lines
between pairs of communicating devices so that proper logic 1 voltage level is maintained. These constraints are called *horizontal voltage drop constraints*. Chowdhury and Breuer dealt with another type of voltage drop constraints, called *vertical constraints* which restrict the voltage drop between the power(ground) pads and power(ground) pins on the modules. In fact, this drop, when above a certain value, reduces the current through the active devices and therefore reduces their switching speed. These *vertical constraints* are nonlinear and can be expressed as:

$$\sum_{i \in p_j} c_i I_i r_i \leq v_j, \quad j = 1, 2, \ldots, m \quad (2.3)$$

where $p_j$ is a path from a power pad to a power pin $j$ on a module, $v_j$ is the given maximum voltage between these two points and $c_i$ is 1 if the direction of current $I_i$ is the same as that in which the path $p_j$ is traversed and $-1$ otherwise. Considering the graph of Fig. 2.1, the vertical constraints can be expressed as:

$$I_9 r_9 + I_8 r_8 + I_1 r_1 - v_1 \leq 0 \quad (2.4)$$
$$I_{13} r_{13} + I_7 r_7 + I_5 r_5 + I_2 r_2 - v_2 \leq 0 \quad (2.5)$$
$$I_{13} r_{13} + I_7 r_7 + I_5 r_5 + I_3 r_3 - v_3 \leq 0 \quad (2.6)$$
$$I_{11} r_{11} + I_4 r_4 - v_4 \leq 0 \quad (2.7)$$

**Metal Migration Constraints** This is an aging effect that is observed in the wires that carry currents in the same direction which leads to the migration of metal over an extended period of time. This is dependent on the current density through the wire cross section and therefore the migration can be reduced by reducing the current density. In fact, the current density in a metal route is an important factor in determining its mean time before failure (MTBF) because MTBF is inversely proportional to some power $p$ of current density. Moreover, high temperature worsens the problem. The MTBF must be larger than the average life of a chip. Therefore, there is an upper bound on the allowable current density in the power and ground routes, and hence a lower bound.
on their widths. The metal migration constraints can be expressed as $I_i/w_i \leq \sigma$ or $|I_i|r_i \leq \rho_l \sigma$, for $1 \leq i \leq n$, with $\sigma$ representing the maximum allowable current density for a fixed thickness of the metal layer. To make these constraints differentiable, they are replaced by a pair of linear constraints:

$$I_i r_i \leq \rho_l \sigma \quad \text{and} \quad -I_i r_i \leq \rho_l \sigma,$$

for $1 \leq i \leq n$ (2.8)

**Minimum Width Constraint** These are lower bounds imposed on the width of power routes because of technological limitations such as minimum feature size. They are linear and can be expressed as:

$$w_i = \rho l_i/r_i \geq W, \quad \text{for} \ 1 \leq i \leq n$$

(2.9)

or

$$r_i \leq \rho l_i/W$$

(2.10)

with the constant $W$ being the minimum width of a metal wire.

**Coupling constraints** Some edges might be required to have the same width, such as the edges along the boundary of a module and the edges constituting a ring connecting power pads. These boundaries are linear and can be formulated as $w_i = w_j$ or

$$\frac{r_i}{l_i} - \frac{r_j}{l_j} = 0$$

(2.11)

for some pairs $(i, j)$ such that $1 \leq i \leq j \leq n$. Considering the graph of Fig. 2.1, the coupling constraints are:

$$w_9 = w_{10} = w_{11} = w_{12} = w_{13}$$

(2.12)

$$w_6 = w_8$$

(2.13)

$$w_7 = w_5$$

(2.14)
**Equipotential Constraints** These constraints ensure that there is no voltage drop between power pads. They are of the form

\[ \sum_{i \in p_{k,j}} d_i I_i r_i = 0 \] (2.15)

Where \( p_{k,j} \) represents any path from power pad \( k \) to power pad \( j \), and \( d_i = -1 \) in case the assigned current direction for \( I_i \) is opposite to the direction in which the path \( p_{k,j} \) is traversed and \( +1 \) otherwise. Denoting the number of pads by \( n_p \), then the number of these constraints is \( n_p - 1 \). For the graph of Fig. 2.1, we have two pads and therefore one equipotential constraint for the graph, which can be expressed as:

\[ I_9 r_9 - I_{10} r_{10} = 0 \] (2.16)

**Loop Constraints** These constraints are due to Kirchoff’s voltage laws for the independent loops in the graph. To come up with these constraints, we need to traverse each independent loop either counterclockwise or clockwise. If we denote the independent loops by \( L_j \), then we can express the nonlinear constraints as:

\[ \sum_{i \in L_j} e_i I_i r_i = 0, \quad j = 1, ..., q \] (2.17)

With \( e_i \) being \( \pm 1 \), depending if the assigned current direction for the current in edge \( i \) is the same or opposite to the direction to traverse the loop. Considering the graph in Fig. 2.1, the loop constraints are:

\[ I_9 r_9 - I_{10} r_{10} + I_{11} r_{11} - I_{12} r_{12} - I_{13} r_{13} = 0 \] (2.18)

\[ I_9 r_9 + I_8 r_8 - I_6 r_6 - I_7 r_7 - I_{13} r_{13} = 0 \] (2.19)

**Node Constraints** These are the linear constraints that correspond to Kirchhoff’s current law which determines the relationship among currents in edges connected to a common node. They can be expressed as

\[ \sum_{i \in N_j} f_i I_i = 0 \] (2.20)
With $N_j$ being the set of indices of edges connected to node $j$, $f_i$ being $\pm 1$, depending if the current direction for edge $i$ is toward node $j$ or not. For each non-pad node, there is one such constraint. For the pad nodes we do not have node constraints since we consider the pad nodes as current sources and therefore Kirchoff’s current laws at these nodes are not needed. Consider the graph of Fig. 2.1, the node constraints are:

\[
\begin{align*}
I_9 + I_{10} - I_8 &= 0 \quad (2.21) \\
I_8 + I_6 - I_1 &= 0 \quad (2.22) \\
I_7 - I_6 - I_5 &= 0 \quad (2.23) \\
I_{13} - I_{12} - I_7 &= 0 \quad (2.24) \\
I_{12} + I_{11} - I_4 &= 0 \quad (2.25) \\
I_5 - I_3 - I_2 &= 0 \quad (2.26)
\end{align*}
\]

2.1.3 Survey of Applicable General Optimization Methods

In the optimization formulated above, we can clearly see that both the objective function and many of the constraints are nonlinear. We can classify the optimization methods to solve this problem into three categories which are: the methods of feasible direction, the penalty function methods and the augmented Lagrangian methods. For further explanation, the reader is referred to [3]. Chowdhury and Breuer used the augmented Lagrangian method.

2.2 Jakushokas and Friedman

In [16], the authors determined a closed form solution for the wire width $w$ that results in the minimum voltage drop over a single layer of a p/g network. The voltage to be minimized is expressed as follows:

\[
V_{\text{drop}}(w) = i(\tau)R_{\text{eff}}(w) + \frac{di(\tau)}{d\tau}L_{\text{eff}}(w) \quad (2.27)
\]
where, $R_{eff}$ is the effective resistance of the network, $i(\tau)$ is the time dependent current, and $L_{eff}$ is the effective inductance. The resulting optimal width is represented by the following closed form formula:

$$w_{opt} = \sqrt{\frac{2\pi \rho s i(\tau)}{\mu_0 t (3/2 + \ln(\frac{2}{\pi})) \frac{ds}{d\tau}}} \tag{2.28}$$

where, $\rho$ is the metal resistivity, $s$ is the line to line spacing $t$ is the line thickness and $\mu_0$ is the permeability of a vacuum. Note that the optimal wire width is time dependent, meaning that it is determined at a specific time or time interval (average effect).

### 2.3 Dutta and Sadowska

In [9], the authors devised a method to determine wire segment widths. Several assumptions are made such as “the blocks comprising the design have already been placed, the currents drawn by blocks are known and the topology of the p/g network has been determined”. They formulated an optimization problem and used only resistance values as variables. All the constraints (electromigration, voltage drops and minimum and maximum widths) are expressed in terms of node voltages and branch currents which are obtained by solving an electrical network explicitly. The problem formulated is nonlinear and they used the feasible direction method to solve it. At each iteration, node voltages and branch currents as well as their gradients should be found. It is important to mention that the authors assume that currents are known at some discrete points in time and therefore the solution found satisfies the voltage drop constraints at these discrete points in time.

### 2.4 Tan, Shi and Lee

The authors of [19], build on the work done by Chowdhury in [2] and devise a method for determining the optimal widths of the power and ground routes. Recall that optimal
widths refer to a minimum amount of chip area used for wiring power/ground networks while “avoiding reliability failures due to electromigration and excessive \( IR \) drops” [19]. This method also assumes that the topology of p/g networks is fixed and the optimal widths of the wires are to be determined. Chowdhury has formulated in [2] a convex nonlinear optimization problem to find the optimal widths of wires. However, such a problem is hard to solve for very large grids. The novelty in [19] is in showing that this nonlinear problem can be transformed into a sequence of linear programs. Therefore the solution is obtained faster. We have to keep in mind that in [3], [2] and [19], the current drawn by each module is “assumed to be known” [19]. In the rest of this section we will briefly review the problem formulation, the constraints and the optimization procedure used in [19].

### 2.4.1 Problem formulation

The power grid network is represented by a graph \( G = (N, B) \) of \( n \) nodes \( N = \{1, ..., n\} \) and \( b \) branches \( B = \{1, ..., b\} \). Each branch \( i \) in \( B \) connects a pair of nodes \( i_1 \) and \( i_2 \) such that the positive direction of current is from \( i_1 \) to \( i_2 \). For each branch \( i \), the resistance \( r_i \) can be written as

\[
r_i = \rho \frac{l_i}{w_i}
\]  

(2.29)

where \( \rho \) is the sheet resistivity, \( l_i \) and \( w_i \) are respectively length and width of branch \( i \).

The objective function to be minimized is the p/g routing area which can be expressed as:

\[
f(V, I) = \sum_{i \in B} l_i w_i = \sum_{i \in B} \frac{\rho I_i l_i^2}{V_{i1} - V_{i2}}
\]  

(2.30)

The authors of [19], chose to solve for the branch current \( I_i \) and nodal voltages \( V_{i1} \) and \( V_{i2} \) instead of taking \( w_i \) as variables.
2.4.2 The constraints

In this section, we will briefly review the constraints used in [19].

**The voltage IR drop constraints**  The voltage values at the nodes connected to p/g pads should be very close to the values at the pads to ensure reliable logic operation:

\[ V_j \geq V_{j \text{,min}} \]  \hspace{1cm} (2.31)

in case \( j \) is connected to a power pad and,

\[ V_j \leq V_{j \text{,max}} \]  \hspace{1cm} (2.32)

in case \( j \) is connected to a ground pad. The right hand sides of these inequalities are some given constants.

**The minimum width constraints**  These constraints are due to technological limitations. Each p/g segment must have its width greater than the given minimum width that is allowed in the layer where it lies. Therefore,

\[ w_i = \rho \frac{l_i I_i}{V_{i1} - V_{i2}} \geq w_{i \text{,min}}, \]  \hspace{1cm} (2.33)

with \( w_{i \text{,min}} \) being some given constants.

**The current density constraints (electromigration)**  To reduce the electromigration effect, a constraint is imposed on the current density of the p/g segment. For each branch \( i \), this constraint can be formulated as:

\[ |I_i| \leq w_i \sigma \]  \hspace{1cm} (2.34)

where \( \sigma \) is a fixed thickness of a layer.
Kirchoff’s current law (KCL)  For each node $j$, the following KCL constraint is available:

$$\sum_{i \in B(j)} s_i l_i = 0,$$

(2.35)

where $B(j)$ is the set of indices of branches connected to node $j$ ($j = 1, ..., n$) and $s_i$ is 1 or $-1$ depending on whether the current direction for branch $i$ is toward node $j$ or not.

The constrained nonlinear optimization problem that was defined above (i.e. with (2.30) as the objective function to be minimized subject to the constraints (2.31)-(2.35)) will be referred to as problem $P$.

2.4.3 General Optimization Procedure

In [2], Chowdhury reduced the complexity of solving $P$ by proposing a relaxed optimization procedure. He first started with an initial feasible solution of the problem and then solved the optimization problem by assuming that the branch currents are fixed. The objective function is hence only dependent on the voltage and can be expressed as:

$$f(V) = \sum_{i \in B} \frac{\alpha_i}{V_{i1} - V_{i2}}$$

(2.36)

with $\alpha_i = \rho I_i l_i^2$. The authors of [19] restricted the changes of the voltages at the nodes so that their current directions remain the same during the optimization i.e.:

$$\frac{V_{i1} - V_{i2}}{I_i} \geq 0$$

(2.37)

Therefore, the minimum width constraints can be rewritten as:

$$\frac{V_{i1} - V_{i2}}{I_i} \leq \frac{\rho l_i}{w_{i,min}}$$

(2.38)

Hence, a first step is to solve the simplified optimization problem denoted by $P1$ i.e. find the nodal voltages that minimize (2.36) subject to the constraints (2.31)-(2.35). A second step is to solve the simplified optimization problem that results from assuming
that all nodal voltages are fixed \( i.e. \) its objective function can be expressed as:

\[
f(I) = \sum_{i \in B} \beta_i I_i
\]  

(2.39)

with \( \beta_i = \frac{\rho l_i^2}{V_i - V_i^2} \). In [19], the current directions were restricted:

\[
\frac{I_i}{V_i - V_i^2} \geq 0
\]  

(2.40)

The idea is to find the currents that minimize \( f(I) \) subject to (2.31)-(2.35). This problem is denoted by \( P2 \). Therefore solving the general problem \( P \) involves starting with an initial feasible solution and then iteratively solve \( P1 \) and \( P2 \). Solving \( P2 \) is easy since it is just a linear program. However solving \( P1 \) is expensive. Chowdhury converted \( P1 \) to an unconstrained convex optimization problem and solved it using the conjugate gradient method. The novelty presented in [19] is solving \( P1 \) using a sequence of linear optimization problems instead of using the conjugate gradient method. The basic idea is to linearize the nonlinear objective function (2.36).

The advantages presented by this method is that it runs faster than the conjugate gradient method and finds solutions that use much less chip area than the conjugate gradient method.

### 2.5 Conclusion

In this chapter, we reviewed previous work done to determine the optimal widths of power grid wires. In the literature we surveyed, it is assumed that the average current drawn by each module is known and is modeled as an independent current source [18]. However this might lead to an overestimation in the increase in width needed. Our approach, that will be presented shortly, takes into consideration the uncertainties in the currents drawn by the underlying circuit.
Chapter 3

Power Grid Verification and Linear Programming Review

3.1 Power Grid Model

We give a brief review of the $RC$ model of the power grid, where each branch is represented by a resistor and where there exists a capacitor from every node to ground. In addition, some nodes have current sources (to ground) that represent the current drawn by the underlying circuit, and some grid nodes have voltage sources (to ground) that represent the external voltage supply. Let the power grid consist of $n + p$ nodes, where nodes $1, \ldots, n$ have no voltage sources attached, and nodes $(n + 1), \ldots, (n + p)$ are the nodes where the $p$ voltage sources are connected. Let $c_k$ be the capacitance from every node $k$ to ground. Let $i_k(t)$ be the current source connected to node $k$, where the direction of current is from the node to ground. We assume that $i_k(t) \geq 0$ and that $i_k(t)$ is defined for every node so that the nodes which have no current source attached have $i_k(t) = 0$, $\forall t$. Let $i(t)$ be the vector of all current sources $i_k(t)$, and $u(t)$ be the vector of all node voltages. If we apply Modified Nodal Analysis (MNA) to the grid [12], we can
write:

\[ Gu(t) + C\dot{u}(t) = -i(t) + GV_{dd} \]  

(3.1)

where \( G \) is the \( n \times n \) conductance matrix of the grid, \( C \) is the \( n \times n \) diagonal capacitance matrix, and \( V_{dd} \) is a constant vector each entry of which is equal to the supply voltage value. Let \( v(t) = V_{dd} - u(t) \) be the vector of voltage drops. Then, (3.1) can be written as:

\[ Gv(t) + C\dot{v}(t) = i(t) \]  

(3.2)

### 3.2 Current Constraints

As already stated, we deal with circuit current uncertainties by using the current constraints approach proposed in [8], which we now review briefly. We distinguish two types of current constraints, *local constraints* and *global constraints*.

*Local constraints* are upper bounds on individual current sources. For example, one may specify that the peak value of the current \( i_k \) at node \( k \) is less than a certain bound, \( i_{L,k} \). We obtain this value from prior simulations of the block, from power budgets, or from engineering judgment, based on the area of the cell or block. We assume that every current source tied to the grid has an upper bound associated with it, so that if a node does not have a current source attached, the upper bound for that current is 0. We can express these constraints as:

\[ 0 \leq i(t) \leq i_L, \quad \forall t \geq 0 \]  

(3.3)

*Global constraints* are upper bounds on the sums of certain subsets of current sources. For example, if the total power consumption of a certain functional block is known, then an upper bound can be specified on the sum of currents drawn by all its internal sub-blocks or cells.
Chapter 3. Power Grid Verification and Linear Programming Review

Assuming we have a total number of $m$ global constraints, then we can express them in matrix form as:

$$0 \leq S_i(t) \leq i_G, \quad \forall t \geq 0$$  \hspace{1cm} (3.4)

where $S$ is an $m \times n$ matrix that contains only 0’s and 1’s, which indicate if that node is included in the constraint or not, and $i_G$ is the vector of the upper bound values. The local and global constraints can be combined into a single inequality as follows:

$$0 \leq Ui(t) \leq i_m, \quad \forall t \geq 0$$  \hspace{1cm} (3.5)

where $U$ is an $(n+m) \times n$ matrix whose first $n$ rows form an identity matrix corresponding to the local constraints, and whose remaining $m$ rows form the $S$ matrix, and where $i_m$ is a $(n + m) \times 1$ vector which is the combination of the vectors $i_L$ and $i_G$.

### 3.3 Transient Robustness

Applying a time-discretization to (3.2) leads to:

$$Gv(t) + C\frac{v(t) - v(t - \Delta t)}{\Delta t} = i(t)$$  \hspace{1cm} (3.6)

As in previous work on power grid verification [5], the problem is formulated by assuming the grid had zero current stimulus for all time $t \leq 0$ and then monitoring the solution of the grid under all possible feasible currents as $t \to \infty$. With zero initial currents, $v(0) = 0$, and at the first time step, $t = \Delta t$, (3.6) leads to:

$$Av(\Delta t) = i(\Delta t)$$  \hspace{1cm} (3.7)

$$v(\Delta t) = A^{-1}i(\Delta t)$$  \hspace{1cm} (3.8)

where $A = G + \frac{C}{\Delta t}$ can be shown to be a symmetric positive definite $M$-matrix [13], for which $A^{-1} \geq 0$ so that $v(\Delta t) \geq 0$. In general, at $t = p\Delta t$ we have [5]:

$$v(p\Delta t) = \sum_{j=0}^{p-1} \left( A^{-1} \frac{C}{\Delta t} \right)^j A^{-1}i((p - j)\Delta t)$$  \hspace{1cm} (3.9)
In the rest of the thesis, we use the term *steady state voltage drop* to refer to the value of the worst-case voltage drop when \( p \) tends to infinity. We are now ready to define transient robustness. A grid is said to be robust if, for every node \( k \), the steady state voltage drop at \( k \) over all possible currents satisfying (3.5) is less than a given threshold \( v_{th} \). Let \( v_{ub}(p\Delta t) \) be the vector of upper bounds on the peak voltage drop at time point \( p\Delta t \), defined in [7]. In particular, we will denote \( v_{ub}(\Delta t) \) by \( V_a \). From [7], we have that, as \( p \) tends to infinity, \( v_{ub}(p\Delta t) \) converges to:

\[
V_\infty = \left( I + G^{-1} \frac{C}{\Delta t} \right) V_a
\]

for all \( \Delta t \geq 0 \). Hence, in this thesis we consider that a grid is robust if the following inequality is satisfied:

\[
V_\infty = \left( I + G^{-1} \frac{C}{\Delta t} \right) V_a \leq V_{th}
\]

where \( V_{th} \) is a vector whose every entry is equal to \( v_{th} \). Using [8], the \( k^{th} \) entry \( V_{ak} \) of \( V_a \) can be found as the solution of the linear program (LP):

\[
\text{maximize } e_k^T v(\Delta t)
\]

subject to

\[
0 \leq UA v(\Delta t) \leq i_m
\]

where \( e_k \) is an \( n \times 1 \) vector of all 0’s except that its \( k^{th} \) entry is 1, and the constraints are obtained using (3.5) and (3.7).

### 3.4 Linear Programming

We solve the LP problem given in (3.12)-(3.13) using a similar approach to the one given in [1], which we now briefly review. The inequality constraints in (3.13) are converted to equality constraints by introducing slacks:

\[
x = \begin{bmatrix} v(\Delta t) \\ s \end{bmatrix}, \quad D = \begin{bmatrix} UA & I \\ -UA & I \end{bmatrix}, \quad b = \begin{bmatrix} i_m \\ 0 \end{bmatrix}, \quad c = \begin{bmatrix} e_k \\ 0 \end{bmatrix}
\]
where $s \geq 0$ is a $(2n + 2m) \times 1$ vector of slack variables, $I$ is the identity matrix of size $2n + 2m$, $b$ is a vector of size $2n + 2m$, and $c$ is a vector of size $3n + 2m$. The problem can be written in standard LP form as:

\[
\text{maximize } v_k(\Delta t) = c^T x \tag{3.15}
\]

\[
\text{such that } Dx = b \tag{3.16}
\]

\[
x \geq 0 \tag{3.17}
\]

The set of all feasible solutions $\mathcal{X}$ \cite{10} is defined by:

\[
\mathcal{X} = \{ x \mid Dx = b, \, x \geq 0 \} \tag{3.18}
\]

$D$ has a rank of at least $2n + 2m$ because it includes an identity matrix of size $2n + 2m$. Hence, there are at least $2n + 2m$ linearly independent columns $\{d_{j_1}, d_{j_2}, \ldots, d_{j_{2n+2m}}\}$ of $D$, called the basis. The corresponding variables $\{x_{j_1}, x_{j_2}, \ldots, x_{j_{2n+2m}}\}$ are called the basic variables of the LP. Given a basis, we denote the index set of these variables by $B = \{j_1, j_2, \ldots, j_{2n+2m}\}$. We also denote the index set of the remaining variables by $R$.

Assuming that the columns forming the basis are moved to the first $2n + 2m$ columns of $D$ by permutation, we get:

\[
D = \begin{bmatrix} B & R \end{bmatrix}, \quad x = \begin{bmatrix} x_B \\ x_R \end{bmatrix}, \quad c = \begin{bmatrix} c_B \\ c_R \end{bmatrix} \tag{3.19}
\]

Using (3.19), rewrite (3.15) and (3.16) as:

\[
v_k(\Delta t) = c_B^T x_B + c_R^T x_R \tag{3.20}
\]

\[
B x_B + R x_R = b \tag{3.21}
\]

Because $B$ has full rank, then $B^{-1}$ exists, and we have:

\[
x_B = B^{-1}(b - R x_R) \tag{3.22}
\]

A basic feasible solution is a feasible solution for which

\[
x_B = B^{-1}b, \quad x_R = 0 \tag{3.23}
\]
Chapter 3. Power Grid Verification and Linear Programming Review

If the LP has a feasible solution, then it also has a basic feasible solution that gives the same objective function value $v_k(\Delta t)$. The reader is referred to [10] for a detailed proof. So, if a problem has an optimal solution then it also has an optimal basic feasible solution and it is enough to deal with the basic feasible solutions only.

A feasible basis $\mathcal{B}$ is optimal if:

$$g = c^T_R - c^T_B R^{-1} R \leq 0$$  \hspace{1cm} (3.24)

The Simplex Method [4] uses this criterion to find the optimal solution of the LP. When the Simplex Method terminates, we must have, not only (3.24), but also:

$$y^T = c^T_B R^{-1} R \geq 0$$  \hspace{1cm} (3.25)

because, for our problem, $c_R = 0$ [1].
Chapter 4

Grid Correction

If the grid is found to be unsafe, our optimal goal would be to do minor changes to the design so that it becomes safe. Let $r$ be the vector of parameters that can be modified, typically the widths of metal branches in the grid. In our case, we consider that the grid may span several metal layers, each of which may be composed of several regions, and each region has metal lines of uniform width value $r_i$. Given this, then $G(r)$ is linear in $r$ because a change in widths on any branch would have a direct proportional effect on the conductances of that branch. The diagonal capacitance matrix $C$ also depends on $r$. Each entry $C_{ii}(r_i)$ is the sum of capacitance due to the grid metal branches and other fixed capacitance due to the circuit MOSFETS, and other sources, which we denote by $C_{ii}(0)$. Using the simple parallel plate capacitance model, $C_{ii}(r_i)$ is linear in $r_i$ and we can write:

$$C_{ii}(r_i) = \frac{\epsilon_{ox} l r_i}{d} + C_{ii}(0)$$

(4.1)

where $\epsilon_{ox}$ is the permittivity of the metal-oxide, $l$ is the length of the branch and $d$ is the thickness of the dielectric. As well, $D$, $x$ and $v_k$ are functions of $r$. Our LP problem
formulated in (3.15)-(3.17), can hence be rewritten as:

\[
\begin{align*}
\text{maximize} & \quad v_k(\Delta t, r) = c^T x(r) \\
\text{subject to} & \quad D(r)x(r) = b \\
& \quad x(r) \geq 0 
\end{align*}
\]

Solving this LP corresponds to finding the \( k^{th} \) entry \( V_{ak}(r) \) of the vector \( V_a(r) \) defined earlier. To determine the effect of changing \( r \) on the worst-case steady state voltage drop, the brute-force approach would be to solve the LP (4.2)-(4.4) at every value of \( r \), for all values of \( k \), which gives \( V_a(r) \), multiply \( V_a(r) \) by \( \left( I + G^{-1}(r) \frac{C(r)}{\Delta t} \right) \) to get \( V_\infty(r) \), and then check if \( V_\infty(r) \leq V_{th} \). However this is too expensive. Instead we propose two more efficient approaches. While both approaches solve a nonlinear optimization problem to find the appropriate widths changes, their objective functions differ as we will see below.

### 4.1 A First Approach: \( V_\infty \) as an Objective Function

If we are able to reduce \( V_\infty(r) \) below the threshold, then the grid becomes robust. At a node \( k \) that violates the threshold condition, we formulate the following nonlinear optimization problem:

\[
\begin{align*}
\text{minimize} & \quad V_\infty_k(r) = e_k \left( I + G^{-1}(r) \frac{C(r)}{\Delta t} \right) V_a(r) 
\end{align*}
\]

Method 1 uses \( V_\infty_k \) as its objective function. Before any further explanations on how that method works, let us rewrite (3.10) as:

\[
V_\infty(r) = M(r)V_a(r)
\]

where,

\[
M(r) = \left( I + G^{-1}(r) \frac{C(r)}{\Delta t} \right)
\]

Thus every entry \( k \) of \( V_\infty(r) \) is related to \( V_a(r) \) and \( M(r) \). The first, second and third partial derivatives of \( V_\infty(r) \) with respect to \( r \) (denoted by \( V_\infty^{(i)}(r) \)) where \( i \) refers to the
Chapter 4. Grid Correction

partial derivatives’ order) are given by the following equations:

\[ V^{(1)}_{\infty}(r) = M^{(1)}(r)V_a(r) + M(r)V^{(1)}_a(r) \]  
(4.8)

\[ V^{(2)}_{\infty}(r) = M^{(2)}(r)V_a(r) + 2M^{(1)}(r)V^{(1)}_a(r) + M(r)V^{(2)}_a(r) \]  
(4.9)

\[ V^{(3)}_{\infty}(r) = M^{(3)}(r)V_a(r) + 3M^{(2)}(r)V^{(1)}_a(r) + 3M^{(1)}(r)V^{(2)}_a(r) + M(r)V^{(3)}_a(r) \]  
(4.10)

In the subsections that follow, we give an estimation of the \(i^{th}\) derivative of \(V^{(i)}_a(r)\) with respect to \(r\) denoted by \(V^{(i)}_a(r)\) in a neighborhood around an initial vector \(r_0\) and a closed form expression for \(M^{(i)}(r)\) (i.e. the \(i^{th}\) derivative of \(M\) with respect to \(r\)). We then derive the Taylor expansion of \(V^{(i)}_{\infty}(r)\). (Note that for the Taylor expansion around \(r_0\), we just need the expressions for \(V^{(i)}_a(r_0)\) and \(M^{(i)}(r_0)\). However the estimation of \(V^{(i)}_a(r)\) in a neighborhood around \(r_0\) will be useful later in the chapter).

### 4.1.1 First Time Step

An expensive brute-force approach to evaluate \(V^{(i)}_a(r)\), would be to solve the LP (4.2)-(4.4) at every given value of \(r\). Instead, we borrow an approach for doing this from [1], which turns out to require much fewer solutions of the LP. To briefly summarize this approach, suppose we have solved the LP at some nominal point \(r_0\). It can be shown that, in a neighborhood around this value of \(r_0\), defined by:

\[ y(r) \geq 0 \quad \text{and} \quad x_B(r) \geq 0 \]  
(4.11)

the solution of the LP (4.2)-(4.4), i.e. \(V^{(i)}_a(r)\), can be obtained by solving the linear system:

\[ v_k(\Delta t, r) = c_B^T x_B(r) \]  
(4.12)

\[ B(r)x_B(r) = b \]  
(4.13)

and we can deduce from that \(V^{(i)}_a(r)\) which is obtained by solving:

\[ v_k^{(i)}(\Delta t, r) = c_B^T x_B^{(i)}(r) = c_B^{T}(-1)^{(i)}i!(B^{-1}\frac{dB}{dr})^i x_B(r) \]  
(4.14)
where \( x_B(r) \) is given by (4.13). This neighborhood is referred to as the safety region and the points along its boundary as the breakpoints. This approach is efficient because solving the linear system is much faster than solving the LP. In fact, we will see below that it’s possible to estimate \( V_{ak}(r) \) inside the safety region using an even faster approach based on Taylor series expansion, and this works very well in practice as we will observe later on.

In order to identify the safety region and the breakpoints, we start by finding the multi-variable Taylor series expansions of \( x_B \) and \( y \) around the initial operating point \( r_0 \). Because the conductance and capacitance matrices are linear in \( r \), then \( A \) is linear in \( r \) and hence \( B \) and \( R \) are also linear in \( r \) and the second derivatives of \( A, B, \) and \( R \) with respect to \( r \) are zero. Furthermore, to simplify the notation, we will drop the arguments \( r \) or \( r_0 \) in connection with matrices like \( A, B, \) and \( R \).

**First Time Step Voltage Drop Estimation**

As in [1], the multi-variable Taylor series expansion for \( x_B(r) \) around \( r_0 \) can be written using multi-index notation [15] as:

\[
x_B(r) = x_B(r_0) + \sum_{|\alpha|=1}^{\infty} \frac{\partial^\alpha x_B(r_0)}{\alpha!}(r - r_0)\alpha
\]

where,

\[
\partial^\alpha x_B(r) = \sum_{|\beta|=1} -i_\beta B^{-1} \partial^\beta B \partial^{\alpha-\beta} x_B(r)
\]

and the scalar constant \( i_\beta \) is the value of the nonzero index \( \beta \) in \( \alpha \). For example, let us take \( \alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_p\} \). For \( \beta = \{1, 0, \ldots, 0\} \), \( i_\beta \) is \( \alpha_1 \), for \( \beta = \{0, 1, \ldots, 0\} \), \( i_\beta \) is \( \alpha_2 \), and so on.

Using (4.12), we can express the worst-case first time step voltage drop in the safety region as:

\[
v_k(\Delta t, r) = v_k(\Delta t, r_0) + c_B^T \sum_{|\alpha|=1}^{N} \frac{\partial^\alpha x_B(r_0)}{\alpha!}(r - r_0)\alpha
\]
and using (4.14), we can express the derivative of order \(i\) of the worst-case first time step voltage drop in the safety region as:

\[
v_k^{(i)}(\Delta t, r) = v_k^{(i)}(\Delta t, r_0) + c_B^T(-1)^{i+1}B^{-1}\frac{dB}{dr}i!\sum_{|\alpha|=1}^N \frac{\partial^\alpha x_B(r_0)}{\alpha!}(r-r_0)^\alpha \quad (4.18)
\]

### Safety Region Estimation

As in [1], the multi-variable Taylor series expansion for \(y(r) = R^T\pi(r)\) (we define \(\pi(r) = B^{-T}(r)c_B\) in (3.25)) around the initial point \(r_0\) can be written as:

\[
y(r) = y(r_0) + \sum_{|\alpha|=1}^\infty \frac{\partial^\alpha y(r_0)}{\alpha!}(r-r_0)^\alpha \quad (4.19)
\]

where \(\partial^\alpha y(r_0)\) is given by:

\[
\partial^\alpha y(r) = \left( \sum_{|\beta|=1} i_\beta \partial^\beta R^T \partial^{\alpha-\beta}\pi(r) \right) + R^T \partial^\alpha \pi(r) \quad (4.20)
\]

and where the previous definition of \(i_\beta\) is valid, and \(\partial^\alpha \pi(r)\) is given by:

\[
\partial^\alpha \pi(r) = \sum_{|\beta|=1} -i_\beta B^{-T} \partial^\beta B^T \partial^{\alpha-\beta}\pi(r) \quad (4.21)
\]

which has the same form as (4.16). As a result, the safety region (corresponding to estimating \(V_k^{(i)}(r)\)) is defined by the values of \(r\) for which all the elements of the vectors defined in (4.15) and (4.19) remain nonnegative. In practice, these expressions can be truncated up to an order \(N\). At the breakpoints, at least one entry, \(j\), of one of these vectors must satisfy:

\[
x_{Bj}(r_0) + \sum_{|\alpha|=1}^N \frac{\partial^\alpha x_{Bj}(r_0)}{\alpha!}(r-r_0)^\alpha = 0 \quad (4.22)
\]

\[
y_j(r_0) + \sum_{|\alpha|=1}^N \frac{\partial^\alpha y_j(r_0)}{\alpha!}(r-r_0)^\alpha = 0 \quad (4.23)
\]

These are \(N^{th}\) order polynomials in \(r\). For example, if we take \(N = 3\) and, say, there are two parameters under consideration, i.e. \(r = (r_1, r_2)\) and \(r_0 = (r_{10}, r_{20})\), then (4.22)
(4.23) would have the form:

\[
\begin{align*}
& m_{111}j(r_1 - r_{10})^3 + m_{112}j(r_1 - r_{10})^2(r_2 - r_{20}) \\
& + m_{122}j(r_1 - r_{10})(r_2 - r_{20})^2 + m_{222}j(r_2 - r_{20})^3 \\
& + m_{11j}(r_1 - r_{10})^2 + m_{12j}(r_1 - r_{10})(r_2 - r_{20}) \\
& + m_{22j}(r_2 - r_{20})^2 \\
& + m_{1j}(r_1 - r_{10}) + m_{2j}(r_2 - r_{20}) \\
& + m_{0j} \\
& = 0
\end{align*}
\]  

(4.24)

where the \( m \)'s are parameters that depend on the partial derivatives of \( x_Bj(r) \) and \( y_j(r) \).

The breakpoint in a given direction is given by the intersection of these equations and the direction vector (which will be explained below). For example, if the direction vector is \( r = (u_1t, u_2t) \) where \( u_1^2 + u_2^2 = 1 \) and \( t \) is a scalar variable, then (4.24) would be a third order polynomial in \( t \):

\[
\begin{align*}
n_{3j}t^3 + n_{2j}t^2 + n_{1j}t + n_{0j} &= 0
\end{align*}
\]  

(4.25)

where the \( n \)'s are parameters that depend on the \( m \)'s in (4.24). We can easily solve these equations for all entries in \( x_B \) and \( y \) to find the smallest \( t \) corresponding to the breakpoint.

### 4.1.2 Steady State Voltage Drop Estimation

We will start by considering the estimation of the steady state voltage drop for a single parameter case \( i.e. \) when \( r \) is scalar. We then generalize to the multiparameter case. Suppose we have already solved (4.6) at some initial value \( r_0 \) \( i.e. \) this entails solving (4.2)-(4.4) at each value of \( k \) to obtain \( V_a(r_0) \) and then multiplying by \( M(r_0) \). Consider a node \( k \) that is not safe. The Taylor expansion of \( V_{\infty k}(r) \) in a neighborhood
around \( r_0 \), can be expressed as:

\[
V_{\infty k}(r) = \sum_{i=0}^{\infty} \frac{V_{\infty k}(r_0)}{i!} (r - r_0)^i
\]  

(4.26)

From (4.6), we can notice that the partial derivatives of \( V_{\infty k} \) with respect to \( r \) depend on both the partial derivatives of \( M \) and the partial derivatives of the first time step voltage drop vector \( V_a \) with respect to \( r \). The latter were estimated in the previous subsection (in fact \( v_k^{(i)} \) is the \( k \)th entry of \( V_a^{(i)} \)). We will now focus on finding the partial derivatives of \( M(r) \). Deriving both sides of (4.7) we get:

\[
\frac{\partial M}{\partial r} = \frac{\partial (I + G^{-1} \frac{C}{\Delta t})}{\partial r}
\]  

(4.27)

\[
= \frac{\partial G^{-1}}{\partial r} \frac{C}{\Delta t} + G^{-1} \frac{\partial C}{\partial r}
\]  

(4.28)

\[
= \left( -G^{-1} \frac{\partial G}{\partial r} G^{-1} \right) \frac{C}{\Delta t} + G^{-1} \frac{\partial C}{\partial r} \frac{K}{\Delta t}
\]  

(4.29)

where \( K \) is an \( n \times n \) constant matrix because \( C \) is linear in \( r \), as shown in (4.1). Taking the derivative once again leads to:

\[
\frac{\partial^2 M}{\partial r^2} = 2G^{-1} \left( \frac{\partial G}{\partial r} G^{-1} \right) \frac{C}{\Delta t} - 2G^{-1} \left( \frac{\partial G}{\partial r} G^{-1} \right) \frac{K}{\Delta t}
\]  

(4.30)

In general, \( M^{(n)}(r) \) (i.e. the derivative of \( M \) with respect to \( r \) of order \( n \)) has the form:

\[
M^{(n)}(r) = (-1)^n n! G^{-1} \left( \frac{\partial G}{\partial r} G^{-1} \right)^n \frac{C}{\Delta t} + (-1)^{n-1} n! G^{-1} \left( \frac{\partial G}{\partial r} G^{-1} \right)^{n-1} \frac{K}{\Delta t}
\]  

(4.31)

which we will now prove using induction.

**Base case:** If we set \( n = 1 \) in (4.31) we get:

\[
M^{(1)} = (-1)^1 1! G^{-1} \left( \frac{\partial G}{\partial r} G^{-1} \right)^1 \frac{C}{\Delta t} + (-1)^0 1! G^{-1} \left( \frac{\partial G}{\partial r} G^{-1} \right)^0 \frac{K}{\Delta t}
\]  

(4.32)

\[
= -G^{-1} \left( \frac{\partial G}{\partial r} G^{-1} \right)^1 \frac{C}{\Delta t} + G^{-1} \frac{K}{\Delta t}
\]  

(4.33)

\[
= -(G^{-1} \frac{\partial G}{\partial r} G^{-1}) \frac{C}{\Delta t} + G^{-1} \frac{K}{\Delta t}
\]  

(4.34)

which agrees with (4.29). Hence the base case is true.
Induction step: Let us assume that (4.31) is true for \( n \) and we will prove that it is true for \( n + 1 \). So, we want to prove that:

\[
M^{(n+1)} = (-1)^{n+1}(n+1)!G^{-1}(\frac{\partial G}{\partial r} G^{-1})^{n+1} \frac{C}{\Delta t} + (-1)^n(n+1)!G^{-1}(\frac{\partial G}{\partial r} G^{-1})^n \frac{K}{\Delta t} \tag{4.35}
\]

To do so, take the derivative of both sides of (4.31):

\[
M^{(n+1)} = ((-1)^n n!G^{-1}(\frac{\partial G}{\partial r} G^{-1})^n \frac{C}{\Delta t})' + ((-1)^{n-1} n!G^{-1}(\frac{\partial G}{\partial r} G^{-1})^{n-1} \frac{K}{\Delta t})' + (-1)^n(n-1)!G^{-1} \frac{(\frac{\partial G}{\partial r} G^{-1})^n}{\Delta t} \frac{K}{\Delta t} + (-1)^{n-1} n!G^{-1} \frac{(\frac{\partial G}{\partial r} G^{-1})^{n-1}}{\Delta t} \frac{K}{\Delta t}
\]

Expanding leads to:

\[
M^{(n+1)} = (-1)^n n!((-G^{-1} \frac{\partial G}{\partial r} G^{-1})^n + G^{-1} n! \frac{(\frac{\partial G}{\partial r} G^{-1})^n}{\Delta t} \frac{K}{\Delta t}) + (-1)^{n-1} n!(-G^{-1} \frac{\partial G}{\partial r} G^{-1}(\frac{\partial G}{\partial r} G^{-1})^{n-1} \frac{K}{\Delta t}) + G^{-1} \frac{\partial (G^{-1} \frac{\partial G}{\partial r} G^{-1})^{n-1}}{\Delta t} \frac{K}{\Delta t} + (-1)^{n-1} n!G^{-1} \frac{(\frac{\partial G}{\partial r} G^{-1})^{n-2}}{\Delta t} \frac{K}{\Delta t}
\]

\[
M^{(n+1)} = (-1)^n n!((-G^{-1} \frac{\partial G}{\partial r} G^{-1})^n + n G^{-1} \frac{(\frac{\partial G}{\partial r} G^{-1})^n}{\Delta t} \frac{K}{\Delta t}) + G^{-1} \frac{\partial (G^{-1} \frac{\partial G}{\partial r} G^{-1})^{n-1}}{\Delta t} \frac{K}{\Delta t} + G^{-1} \frac{(\frac{\partial G}{\partial r} G^{-1})^{n-2}}{\Delta t} \frac{K}{\Delta t}
\]
Performing more mathematical manipulations, we get:

\[
M^{(n+1)} = (-1)^n n!((-G^{-1}(\frac{\partial G}{\partial r})^{-1})^{n+1} + nG^{-1}(\frac{\partial G}{\partial r}(-G^{-1}\frac{\partial G}{\partial r}G^{-1}))(-G^{-1}\frac{\partial G}{\partial r}G^{-1})^{n-1}) \frac{C}{\Delta t} \\
+ G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n} \frac{K}{\Delta t} + (-1)^{n-1} n!(-G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n} \frac{K}{\Delta t} \\
(n-1)G^{-1}\frac{\partial G}{\partial r}G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n-2} \frac{K}{\Delta t}
\]

\[
= (-1)^n n!(-G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n+1} \frac{C}{\Delta t} - nG^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n+1} \frac{C}{\Delta t} \\
+ G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n} \frac{K}{\Delta t} + (-1)^{n-1} n!(-G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n} \frac{K}{\Delta t} \\
-(n-1)G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n} \frac{K}{\Delta t}
\]

\[
= (-1)^n n!(-(n+1)G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n+1} \frac{C}{\Delta t} + G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n} \frac{K}{\Delta t} \\
+(-1)^{n-1} n!(-G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n} \frac{K}{\Delta t} \\
= (-1)^{n+1}(n+1)!G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n+1} \frac{C}{\Delta t} + (-1)^n n!G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n} \frac{K}{\Delta t} \\
+(-1)^n n!nG^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n} \frac{K}{\Delta t} \\
= (-1)^{n+1}(n+1)!G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n+1} \frac{C}{\Delta t} \\
+(-1)^n n!G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n} \frac{K}{\Delta t}(1 + n) \\
= (-1)^{n+1}(n+1)!G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n+1} \frac{C}{\Delta t} \\
+(-1)^n(n+1)!G^{-1}(\frac{\partial G}{\partial r}G^{-1})^{n} \frac{K}{\Delta t}
\]

which agrees with (4.35). So, by induction, we have verified that the equation (4.31) is correct.

In the multiparameter case, i.e. when \( r \) is a vector, the multi-variable Taylor expansion of \( V_{\infty_k} \) around \( r_0 \) can be expressed using the multi-variable index notation [15] as:

\[
V_{\infty_k}(r) = V_{\infty_k}(r_0) + \sum_{|\alpha|=1}^{\infty} \frac{\partial^n V_{\infty_k}(r_0)}{\alpha!} (r - r_0)^\alpha
\]
4.1.3 Correction of the Offending Node

Suppose that, at node \( k \), the worst case voltage drop, i.e. \( V_{\infty k} \), is larger than the threshold (we call this node the offending node). The purpose of Algorithm 1 is to find the sufficient widths changes that will make that node safe. It does that by solving the following nonlinear optimization problem:

\[
\text{minimize } V_{\infty k}(r) = e_k \left( I + G^{-1}(r) \frac{C(r)}{\Delta t} \right) V_a(r)
\]

There are several nonlinear optimization methods [14] that can be used to solve (4.37). We used the steepest descent method with cubic interpolation for determining the step length at an initial value of \( r \) (the reader is referred to the appendices section for further explanation of this method). In the remainder of this section we will describe how this algorithm works. As a first step, at line 3, we solve the LP in (4.2)-(4.4) for every value of \( k \) (i.e. we solve \( n \) LP’s) using the Simplex method and find each time the corresponding optimal basis which will be used to find \( v_k(r) \), \( v_k^{(1)}(r) \), \( v_k^{(2)}(r) \) and \( v_k^{(3)}(r) \). The safety region is then updated using these bases. Hence, after line 3 is executed, we will have \( v_k(r) \), \( v_k^{(1)}(r) \), \( v_k^{(2)}(r) \) and \( v_k^{(3)}(r) \) for all the values of \( k \) (\( 1 \leq k \leq n \)). Using these partial derivatives (i.e. the partial derivatives of the worst case first time step voltage drop), the Taylor expansions of \( V_{\infty k}(r) \) and \( \nabla V_{\infty k}(r) \) (i.e. the gradient of \( V_{\infty k}(r) \)) are determined. Recall that the gradient of a scalar function \( g(y_1, y_2, ..., y_n) \) is denoted by \( \nabla g \). It is the vector whose components are partial derivatives of \( g \): \( \nabla g = (\frac{\partial g}{\partial y_1}, ..., \frac{\partial g}{\partial y_n}) \).

The nonlinear optimization algorithm is then used. The direction is found using the steepest descent line search method. The maximum step length that is going to be used in our steepest descent algorithm is determined by finding the breakpoints in that direction. Having the maximum step length, we compute an appropriate step length using cubic interpolation. Using the step length and direction, we then update the value of the width \( r \) and recompute \( V_{\infty k}(r) \) using its Taylor expansion. We also recompute \( \nabla V_{\infty k}(r) \) using its Taylor expansion to find the new search direction. If the maximum step length is
taken, then we re-solve the \( n \) LP’s and find the new optimal bases and safety region.

**Algorithm 1** node_correction (Method 1 applied)

1: \( r_{req} = r_0 \)

2: \textbf{while} (\( V_{\infty k}(r) \geq v_{th} \)) \textbf{do}

3: \hspace{1em} For every value of \( k \) (1 ≤ \( k \) ≤ \( n \)) solve the LP given in (4.2)-(4.4) using the Simplex method at \( r = r_{req} \) and get the optimal basis. Store the values of \( v_k(r_{req}), v_k^{(1)}(r_{req}), v_k^{(2)}(r_{req}) \) and \( v_k^{(3)}(r_{req}) \) (obtained using (4.14)) and update the safety region

4: \hspace{1em} Find \( V_{\infty k}(r_{req}), V_{\infty k}^{(1)}(r_{req}), V_{\infty k}^{(2)}(r_{req}) \) and \( V_{\infty k}^{(3)}(r_{req}) \) (using the partial derivatives of \( V_a(r_{req}) \) with respect to \( r \) i.e. the \( v_k(r_{req})'s, v_k^{(1)}(r_{req})'s, v_k^{(2)}(r_{req})'s \) and \( v_k^{(3)}(r_{req})'s \) that were found in step 3 and plugging in (4.8)-(4.10))

5: \hspace{1em} Find the Taylor expansions for \( V_{\infty k}(r) \) and \( \nabla V_{\infty k}(r) \)

6: \hspace{1em} \textit{max\_step\_taken} = FALSE.

7: \hspace{1em} \textbf{while} (\textit{max\_step\_taken} is FALSE) \textbf{do}

8: \hspace{2em} \textit{p} = \text{FIND\_SEARCH\_DIRECTION}(r_{req}),

9: \hspace{2em} \lambda_{max} = \text{FIND\_MAX\_STEP\_LENGTH}(p, r_{req}),

10: \hspace{2em} \lambda = \text{FIND\_STEP\_LENGTH}(p, r_{req}, \lambda_{max}),

11: \hspace{2em} r_{req} = r_{req} + \lambda p,

12: \hspace{2em} Using Taylor expansion with \( r = r_{req} \), find \( V_{\infty k}(r_{req}) \).

13: \hspace{2em} \textbf{if}(\lambda = \lambda_{max}) \textbf{then} \textit{max\_step\_taken} = TRUE.

14: \hspace{1em} \textbf{end while}

15: \hspace{1em} \textbf{end while}

16: return \( r_{req} \)

**4.1.4 Top Level Algorithm**

Recall that a grid is called robust if the largest worst case voltage drop among all nodes is less than the given threshold. While Algorithm 1 deals with a single node, Algorithm 2 uses Algorithm 1 as a subroutine to correct the entire grid. It starts by identifying the node with the largest voltage drop \( V_{\infty k} \). If it is less than the threshold, then the grid is safe and the
algorithm exits. If not, it calls Algorithm 1 that returns the required width changes so that the offending node becomes safe. It then checks if any other nodes are still unsafe. If that is the case, the steps above are repeated until the grid is deemed robust. In all our test cases, once the offending node was safe, the entire grid became robust. Hence Algorithm 2 was called once.

Algorithm 2 Top Level Algorithm (Method 1 applied)

1: grid_unsafe = TRUE

2: \( r_f = r_0 \).

3: while (grid_unsafe) do

4: Verify all the nodes at \( r = r_f \).

5: If \( (V_\infty \leq V_{th}) \) then, grid_unsafe = FALSE.

6: else pick the node \( k \) such that \( V_{\infty k} \) is the largest entry in \( V_\infty \).

7: Call Algorithm 1 to find the required change in parameter, \( r_{req} \) to reduce \( V_{\infty k} \) below \( v_{th} \).

8: \( r_f = r_{req} \)

9: end while

4.1.5 Experimental Results

We tested this approach on a number of small grids of different sizes. In our tests, the number of parameters taken was equal to 10. Table 4.1 shows the percentage width increases returned by the algorithm for each of the 10 parameters to fix the grid. Most of these changes are very small and thus the area increase is small as well. Table 4.3 shows the time needed to correct all the offending nodes. A graphical representation is given in Fig. 4.1. We can see that, with larger grids, the total run time of the algorithm increases dramatically. A reason for this is the number of linear programs solved which increases significantly with larger grids as illustrated in Fig. 4.2. In fact, when the algorithm hits a breakpoint, it has to re-solve \( n \) LP’s using the Simplex method. For large values of \( n \) this is very time consuming. We therefore propose a second approach that is less accurate (but conservative) but much faster (as shown in tables 4.2 and 4.4) and certainly more useful for larger grids. In the next section, we explain in detail
how this method works and give some experimental results.

Table 4.1: Required width changes for the case of 10 parameters (Method 1 applied)

<table>
<thead>
<tr>
<th>Power Grid</th>
<th>Percentage increase in each of the 10 parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>( r_1 )</td>
</tr>
<tr>
<td>182</td>
<td>6.63</td>
</tr>
<tr>
<td>598</td>
<td>6.60</td>
</tr>
<tr>
<td>764</td>
<td>6.95</td>
</tr>
<tr>
<td>1594</td>
<td>5.10</td>
</tr>
</tbody>
</table>

Table 4.2: Required width changes for the case of 10 parameters (Method 2 applied)

<table>
<thead>
<tr>
<th>Power Grid</th>
<th>Percentage increase in each of the 10 parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>( r_1 )</td>
</tr>
<tr>
<td>182</td>
<td>11.86</td>
</tr>
<tr>
<td>598</td>
<td>10.74</td>
</tr>
<tr>
<td>764</td>
<td>11.91</td>
</tr>
<tr>
<td>1594</td>
<td>11.12</td>
</tr>
</tbody>
</table>

Table 4.3: Maximum steady state voltage drops on the grids before and after correction and the run-times of the overall algorithm (Method 1 applied)

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>maximum voltage drop before correction(mV)</th>
<th>maximum voltage drop after correction(mV)</th>
<th>Number of Simplex solutions</th>
<th>Individual node correction time</th>
<th>Total runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>182</td>
<td>110.70</td>
<td>99.25</td>
<td>2548</td>
<td>6.31min</td>
<td>6.40min</td>
</tr>
<tr>
<td>598</td>
<td>102.77</td>
<td>92.36</td>
<td>8372</td>
<td>1.31hrs</td>
<td>1.32hrs</td>
</tr>
<tr>
<td>764</td>
<td>106.04</td>
<td>95.08</td>
<td>11460</td>
<td>2.55hrs</td>
<td>2.57hrs</td>
</tr>
<tr>
<td>1594</td>
<td>110.74</td>
<td>99.48</td>
<td>25504</td>
<td>14.86hrs</td>
<td>14.89hrs</td>
</tr>
</tbody>
</table>

Table 4.4: Maximum steady state voltage drops on the grids before and after correction and the run-times of the overall algorithm (Method 2 applied)

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>maximum voltage drop before correction(mV)</th>
<th>maximum voltage drop after correction(mV)</th>
<th>Number of Simplex solutions</th>
<th>Individual node correction time</th>
<th>Total runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>182</td>
<td>110.70</td>
<td>96.09</td>
<td>5</td>
<td>0.74sec.</td>
<td>5.82sec.</td>
</tr>
<tr>
<td>598</td>
<td>102.77</td>
<td>91.72</td>
<td>9</td>
<td>20.68sec.</td>
<td>35.93sec.</td>
</tr>
<tr>
<td>764</td>
<td>106.04</td>
<td>92.03</td>
<td>3</td>
<td>24.14sec.</td>
<td>45.66sec.</td>
</tr>
<tr>
<td>1594</td>
<td>110.74</td>
<td>96.30</td>
<td>35</td>
<td>2.13min.</td>
<td>3.13min.</td>
</tr>
</tbody>
</table>
Figure 4.1: Run-Time of the overall algorithm as a function of the number of nodes (Method 1).

Figure 4.2: Number Of Simplex algorithm runs needed to correct the grid for method 1 and method 2.
4.2 A Second Approach: \( V_a \) as a Proxy Objective Function

Our second method is inspired by the following claim:

**Claim 1.** Let \( V_{a_{th}} = (I + G^{-1} \frac{C}{\Delta t})^{-1} V_{th} \). If \( V_a \leq V_{a_{th}} \), then \( V_\infty \leq V_{th} \) and the grid is robust. The converse is however not true.

**Proof.** From (3.3), we know that the source current vector is element-wise positive. Because \( A \) is an \( M \) matrix, then \( A^{-1} \geq 0 \), and hence from (3.8), \( v(\Delta t) \geq 0 \) for all \( \Delta t \geq 0 \). As we mentioned earlier, \( V_a = v_{ub}(\Delta t) \) is the vector of upper bounds on the peak voltage drop at time point \( \Delta t \). Hence, \( V_a \geq v(\Delta t) \geq 0 \) and both sides of the inequality \( V_a \leq V_{a_{th}} \) are therefore positive, and multiplying them by the positive matrix \((I + G^{-1} \frac{C}{\Delta t})\) results in \( V_\infty \leq V_{th} \). To prove that the converse of the statement is not true, we will consider the following counter example

where: \( \frac{C}{\Delta t} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \), \( G = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix} \), and \( V_{th} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} \). Then \( A^{-1}G = \begin{bmatrix} 7 & -1 \\ -1 & 8 \end{bmatrix} \)

and \( V_{a_{th}} = A^{-1}GV_{th} = \frac{1}{\Pi} \begin{bmatrix} 0.6 \\ 0.7 \end{bmatrix} \). Now let \( V_a = \begin{bmatrix} 0.001 \\ 0.07 \end{bmatrix} \), then \( V_a \not\leq V_{a_{th}} \). However,

\[
(I + G^{-1} \frac{C}{\Delta t})V_a = G^{-1}AV_a = \begin{bmatrix} 0.078 \\ 0.491 \end{bmatrix} = \begin{bmatrix} 0.0156 \\ 0.0982 \end{bmatrix} \]

and thus \((I + G^{-1} \frac{C}{\Delta t})V_a \leq V_{th}\). \(\square\)

Hence, if we are able to reduce \( V_a(r) \) below \( V_{a_{th}(r)} \), then we are guaranteed that \( V_\infty(r) \) is less than \( V_{th} \) and hence that the grid is robust. However this might lead to an overestimation of the width changes needed to fix the grid, because the condition is only sufficient but not necessary, and thus even if \( V_a \) is not less than \( V_{a_{th}} \), the grid may already be safe. What we propose, therefore, is to use nonlinear optimization to reduce \( V_{a_k}(r) \) as an objective function, but also to occasionally compute \( V_\infty(r) \) and check if \( V_\infty(r) \leq V_{th} \). We already described the computation of \( V_{a_k}(r) \) in the subsection 4.1.1. In the next subsection, we will describe the nonlinear optimization loop around it.
4.2.1 Nonlinear Optimization Problem

Inside a safety region, and in order to reduce $V_{a_k}(r)$, we make use of the nonlinear optimization problem:

$$\begin{align*}
\text{minimize} \quad & v_k(\Delta t, r) = c_B^T x_B(r) \\
\text{such that} \quad & x_B(r) \geq 0 \\
& y(r) \geq 0
\end{align*} \tag{4.38}$$

where $x_B(r)$ and $y(r)$ are given by (4.15) and (4.19). A number of nonlinear optimization algorithms are available to solve such problems [14]. We use the steepest descent line search method, with cubic interpolation, which leads to our node correction algorithm summarized in Algorithm 3. In the remainder of this section, we will describe how this algorithm works.

As a first step, at line 3, we find the optimal basis using the solution of the nominal linear program at a given $r$. We then compute the Taylor expressions for the first time step voltage drop and the safety region, using this basis. The search direction is then found using the steepest descent line search method. The maximum step length that can be taken is calculated by finding the breakpoint in that direction. Having the maximum step length, we then compute an appropriate step length that reduces the objective function. Step length computation is done using the cubic interpolation method [14]. Because the first time step threshold vector $V_{\text{th}}$ is not a constant and varies with $r$, then that threshold vector is evaluated after every step. The cost involved in the computation is due to the $LU$ factorizations. However, in all test cases, we observed that no more than 35 $LU$ factorizations are needed and the cost was only moderate. Using the step length and the direction, the parameter values are updated and the new $V_{a_k}(r)$ is computed. If the maximum step is taken (a breakpoint is reached), the $LP$ is re-solved to get the optimal basis in the new region.

While doing all the above steps, we track the difference between $V_{a_k}(r)$ and $V_{\text{th}}(r)$. When this difference decreases by 10%, we check if $V_{\infty}(r) \leq V_{\text{th}}$ based on (3.10). Finding $V_{\infty}(r)$ effectively checks the voltage drop at all the grid nodes, not just at node $k$. If this check succeeds, then the grid is robust and the algorithm terminates, returning with the appropriate flag setting to the Top Level Algorithm (Algorithm 4) which causes that algorithm to perform
Algorithm 3 node_correction (Method 2 applied)

1: $r_{req} = r_0$

2: while ($V_{a_k} > V_{ath_k}$ and not $inf\_ty\_check$) do

3: Solve the LP given in (4.2)-(4.4) using the Simplex method at $r = r_{req}$ and get the optimal basis.

4: Find the Taylor expansion for $v_k(\Delta t, r)$ using (4.17).

5: Find the safety region expressions using (4.15) and (4.19)

6: $max\_step\_taken = FALSE$.

7: while ($max\_step\_taken$ is $FALSE$) do

8: $p = FIND\_SEARCH\_DIRECTION(r_{req})$,

9: $\lambda_{max} = FIND\_MAX\_STEP\_LENGTH(p, r_{req})$,

10: $\lambda = FIND\_STEP\_LENGTH(p, r_{req}, \lambda_{max})$,

11: $r_{req} = r_{req} + \lambda p$,

12: recompute $V_{ath}(r_{req})$,

13: Using Taylor expansion with $r = r_{req}$, find $V_{a_k}(r_{req}) = v_k(\Delta t, r_{req})$.

14: if($\lambda = \lambda_{max}$) then $max\_step\_taken = TRUE$.

15: end while

16: If($0 \leq (V_{a_k} - V_{ath_k}) \leq 0.9 \times (old\_V_{a_k} - old\_V_{ath_k})$)

17: do check the voltage drops at infinity.

18: if ($V_\infty < V_{th}$) then $not\_inf\_ty\_check = FALSE$

19: end while

20: return $r_{req}, not\_inf\_ty\_check$
Figure 4.3: Variation of the worst case first time step voltage drop at node $k$ and its threshold with respect to the number of Simplex solutions in a 598 node RC grid.

no additional work and exit. Note that it is important to keep the number of $V_\infty$ checks somewhat small, because each check is as expensive as a full grid verification, which can be costly.

However, it is possible for the sufficient condition in the claim of section 4.2 to be met before the $V_\infty$ check is observed. In this case, we know that node $k$ is safe, but we don’t know about the safety of other nodes. So, the algorithm exits and returns to the Top Level Algorithm with a different setting of the flag, which then checks if all nodes have indeed been corrected.

### 4.2.2 Top level algorithm

Our overall approach, which uses the node correction as a subroutine, is given in Algorithm 4 and works as follows. It starts with a grid verification for the given $r$, and finds $V_\infty(r)$. If $V_\infty(r) \leq V_{th}$, then the grid is safe. Otherwise, it finds $V_{ath}(r)$ and identifies the node $k$ such that $V_{ak} - V_{ath,k}$ is the largest. The required parameter values that result in $V_{ak}$ less than $V_{ath,k}$ or $V_\infty$ less than $V_{th}$ are found using Algorithm 3. In the first case, the grid is re-verified.
Table 4.5: Required width changes for the case of 10 parameters (Method 2 applied)

<table>
<thead>
<tr>
<th>Power Grid</th>
<th>Percentage increase in each of the 10 parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>r1</td>
</tr>
<tr>
<td>1594</td>
<td>11.48</td>
</tr>
<tr>
<td>3427</td>
<td>11.12</td>
</tr>
<tr>
<td>13241</td>
<td>11.5</td>
</tr>
<tr>
<td>23378</td>
<td>11.5</td>
</tr>
<tr>
<td>70421</td>
<td>15.5</td>
</tr>
<tr>
<td>103927</td>
<td>14.99</td>
</tr>
</tbody>
</table>

using the new parameter values to check if any other nodes exceed the threshold and the above procedure is repeated until \( V_\infty \leq V_{th} \). In the second case, the verification was already done in Algorithm 3 and we therefore exit the Top Level Algorithm. In all our test cases, Algorithm 3 was called only once and we did not need to re-verify all the nodes in the Top Level Algorithm which was hence executed only once.

![Figure 4.4: Correlation plot of the steady state voltage drops before and after correction for a 1594 node grid.](image)
Algorithm 4 Top Level Algorithm (Method 2 applied)

1: grid_unsafe = TRUE, not_infty_check = TRUE.
2: r_f = r_0.
3: while (grid_unsafe = TRUE and not_infty_check = TRUE) do
   4:   Verify all the nodes at r = r_f. If (V_\infty \leq V_{th}) then, grid_unsafe = FALSE.
   5:   else find V_{a_\text{th}}(r_f) = (I + G^{-1}(r_f)C(r_f)\Delta t)^{-1}V_{th}.
   6:   Pick the node k such that the difference V_{a_k} - V_{a_\text{th}_k} is the largest.
   7:   Call Algorithm 3 to find the required change in parameter r_{req} to either reduce V_{a_k} below V_{\text{a_\text{th}_k}} or reduce V_{\infty} below V_{th}.
   8:   r_f = r_{req}
9: end while

Table 4.6: Required width changes for the case of 10 parameters after rounding (Method 2 applied)

<table>
<thead>
<tr>
<th>Power Grid</th>
<th>Percentage increase in each of the 10 parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>r1</td>
</tr>
<tr>
<td>1594</td>
<td>10</td>
</tr>
<tr>
<td>23378</td>
<td>15</td>
</tr>
<tr>
<td>70421</td>
<td>15</td>
</tr>
<tr>
<td>103927</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 4.7: Maximum steady state voltage drops on the grids before and after correction and the run-times of Algorithms 1 and 2 (Method 2 applied)

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>maximum voltage drop before correction(mV)</th>
<th>maximum voltage drop after correction(mV)</th>
<th>Number of Simplex solutions</th>
<th>Individual node correction time</th>
<th>Total runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1594</td>
<td>110.75</td>
<td>98.20</td>
<td>35</td>
<td>2.21min</td>
<td>3.17min</td>
</tr>
<tr>
<td>3427</td>
<td>105.78</td>
<td>94.29</td>
<td>28</td>
<td>4.9min</td>
<td>7.34min</td>
</tr>
<tr>
<td>13241</td>
<td>108.90</td>
<td>97.71</td>
<td>24</td>
<td>21.66min</td>
<td>34.2min</td>
</tr>
<tr>
<td>23378</td>
<td>108.89</td>
<td>97.70</td>
<td>23</td>
<td>41.71min</td>
<td>1.1hrs</td>
</tr>
<tr>
<td>70421</td>
<td>104.15</td>
<td>91.45</td>
<td>31</td>
<td>4.9hrs</td>
<td>6.8hrs</td>
</tr>
<tr>
<td>103927</td>
<td>102.19</td>
<td>90.32</td>
<td>31</td>
<td>9.06hrs</td>
<td>12.55hrs</td>
</tr>
</tbody>
</table>
4.2.3 Experimental Results

We implemented the grid correction algorithm in C++. Our algorithm was tested on a set of grids generated according to user specifications of metal layers (M1-M9), pitch and width per layer, current source distribution and grid dimensions. The computations were performed on a 64-bit Linux machine with 24 GB memory. Some of the width changes returned by the algorithm are very small, such as 1% increase or less on many parameters, and the largest widths changes are less than 16%. In our tests, we varied the number of parameters between 10 and 20. Table 4.5 shows the percentage width increase for each of the 10 parameters used so that the voltage drops on all the nodes at steady state are below the threshold. The percentages increase are continuous values; however it makes more sense to increase the widths by some discrete and significant percentages and discard the very small changes. We did that by adding a rounding step to the algorithm, which returned the changes needed to the nearest 5% or 10% setting. This rounding step was done once $V_{a_k} - V_{a_{th_k}}$ drops by 10% and before doing the infinity check step. In the tests that we did, the very first time we performed rounding, the grid was found to be safe. Table 4.6 shows the widths changes needed after rounding. As one can notice, only 4 out of 10 parameters need to be changed. Table 4.7 gives the time needed to correct a single node and the total CPU time of the algorithm. The latter includes full grid verification done before the correction algorithm to pick the node whose first time step voltage drop peak is farthest away from its corresponding threshold, and the time needed to perform the steady state check. Therefore, at least 2 full grid verifications are required. In all the tests that we did, we observed that no more than 2 full grid verifications were needed. The reason why the run time of the algorithm is larger than that of the correction of a single node is because of the full grid verifications, which are highly parallelizable operations. Hence, the total run time can be reduced with more efficient techniques of full grid verification.

While testing our algorithm on different grids, we observed many interesting results. As the algorithm progresses, $V_{a_{th_k}}$ is increasing and $V_{a_k}$ is decreasing as shown in Fig. 4.3. Hence there will be an eventual intersection between the two curves. However, this intersection usually happened after a large number of Simplex solutions because 1) the increase and the decrease in $V_{a_{th_k}}$ and $V_{a_k}$, respectively, slow down as the algorithm progresses, 2) for most of the grids
we tested, the gap between $V_{athk}$ and $V_{ak}$ was very large (in fact in most cases $\frac{V_{ak}}{V_{athk}} \leq 2.6\%$). The need to check the voltage drops at steady state after a certain number of Simplex solutions is therefore justified. This number is determined by tracking the difference between $V_{athk}$ and $V_{ak}$. The first time we do the infinity check is when the value of that difference has dropped by 10% from its initial value. Subsequent steady state checks are done when the difference drops
by 10% from the last steady state check. In all our tests, the 10% decrease was sufficient and the grid was fixed the first time we did an infinity check. Hence, only two full grid verifications were needed before the grid was deemed to be safe. The first was needed before the correction algorithm to select the node to be corrected, and the second was needed when we did an infinity check while in the correction algorithm. The intersection point (corresponding to $V_{ak} = V_{a_{thk}}$) was never reached and the correction happened much earlier than that. Fig. 4.3 illustrates how our algorithm avoids correction overestimation on a 598 node $RC$ grid. If no infinity checks were done and we waited for the intersection between $V_{ak}$ and $V_{a_{thk}}$ to happen, then we would end up with 55 Simplex solutions and therefore an overestimated correction. In fact, the grid was safe after 8 Simplex solutions only and therefore for smaller changes in $r$. Our algorithm successfully captured that by performing an infinity check.

In Fig. 4.5, the histograms of the voltage drops before and after correction for a 70421 node grid, are given. We only show the voltage drops larger than 60 mV for better visibility. We see from the plots that the algorithm successfully reduced all the voltage drops exceeding the threshold of 100mV. The correlation plot in Fig. 4.4, also shows that all the voltage drops on a 1594 grid were reduced under the threshold of 100mV.
Chapter 5

Conclusion

With the rising demand of low voltage designs, supply voltage integrity verification has become a crucial step in reliable high-speed chip design. In this thesis, we studied two approaches that determine the necessary changes in the wires widths needed to make the grid safe. While both approaches use linear programming concepts and nonlinear optimization algorithms, they differ in the objective function to be minimized. For small grids, minimizing the steady state voltage drop is a good strategy because it is very accurate (i.e. almost without overestimation). For large grids however, this approach is very time consuming because of the number of LP's solved. We therefore proposed a second approach that includes more overestimation than the first one but significantly much faster. It corrects the grid with minimal changes, when some nodes do not satisfy the threshold. We extended the work done in [1] to the case of an RC model of the grid. Instead of reducing the steady state voltage drops directly, we work on reducing the first time step voltage drops, which do have an impact at infinity. We use linear programming concepts and the current constraints-based verification approach and we formulate a nonlinear optimization problem to find the required width changes needed. Our second method showed good results: A grid of 70421 nodes was corrected in a total time of 6.8 hours and the changes needed in the widths of metal branches on some levels of the grid were less than 16%.
5.1 Future Work

The correction of an $RC$ model of the grid was done by changing the widths of metal branches. However, other parameters can be considered such as the pitch between metal lines. In that case the grid topology is changed. Insertion of decaps at the right places could also be considered to correct an unsafe $RC$ model of the grid.

Moreover, the method we proposed in this thesis, deals with an $RC$ version of the grid. It does not take into account the effect of inductance. A more general method, would consider an $RLC$ model of the power grid. In that case, dynamic $L \frac{di(t)}{dt}$ noise should be taken into account and insertion of decaps might be indispensable to achieve a robust grid. The optimization objective function would be not only the area of the wires but also that of the decaps.

Finally, we observed that the sensitivities of $V_{\infty}$ parallel the sensitivities of $V_a$ to the width parameters. Future work could be to establish a mathematical relationship between both sensitivities. Such a relationship is of great importance because it increases the accuracy of our second method. In fact, instead of performing an infinity check when $V_a - V_{ath}$ decreases by a certain percentage, we would reduce $V_a$ by a sufficient amount that guarantees the reduction of the steady state voltage drop below the threshold. That sufficient reduction amount is determined from the mathematical relationship already established between both sensitivities.
Appendices
Appendix A

Line Search Methods

Line search methods are a class of methods for solving unconstrained optimization problems [14] of the form:

\[ \text{minimize} \quad f(x) \]

where \( f : \mathbb{R}^n \to \mathbb{R} \) is a continuously differentiable function. In general a line search method starts with an initial guess \( x_0 \) and, at every step \( k \), it chooses a descent direction \( p_k \) and a step length \( \lambda_k \) i.e. how far we should move along that direction and computes the next iterate \( x_{k+1} = x_k + \lambda_k p_k \). As we mentioned, \( p_k \) should be a descent direction to guarantee that \( f \) decreases along the search direction at least initially, i.e. we should have

\[ p_k^T \nabla f_k \leq 0 \]

(A.2)

In general, \( p_k \) is represented as:

\[ p_k = -B_k^{-1} \nabla f_k \]

(A.3)

where \( B_k \) is a nonsingular symmetric matrix.

A.1 Standard Search Directions

There are several search directions that are commonly used. In this section, we will briefly review some of them. For a more detailed discussion, the reader is referred to [14].
The Steepest descent direction is the most straightforward choice of the search direction. In that case, $B_k$ is simply the identity matrix $I$ and thus $p_k = -\nabla f_k$.

The Newton direction is obtained by setting $B_k$ to be the exact Hessian of the function to be minimized, i.e. $B_k = \nabla^2 f(x_k)$.

The Quasi-Newton Direction is a modified version of the Newton direction, by taking $B_k$ to be an approximation to the Hessian of $f$ updated at every iteration by means of a low-rank formula.

A.2 Step Length

There is a tradeoff when it comes to finding the appropriate step length. In fact, we would like to pick a step length that induces a significant reduction in the function $f$, but at the same time, we do not want to waste a lot of time on choosing an appropriate step length [14]. Ideally, one would like to choose the step length in a way that $\lambda$ is the global minimizer of the univariate function $\phi(\lambda)$ expressed as:

$$\phi(\lambda) = f(x_k + \lambda p_k), \quad \lambda \geq 0 \quad (A.4)$$

However, this is too expensive. It is therefore more practical to adopt an inexact line search method to choose the step length. In general, line search algorithms try a sequence of possible values for $\lambda$ and stop to choose one of them when certain conditions are met. There are two stages in a line search algorithm. A bracketing phase that determines an appropriate interval containing desirable step lengths and an interpolation phase that finds a good step length within this interval.

A.2.1 Cubic Interpolation

The purpose of cubic interpolation is to find an appropriate step length by interpolation of known function and derivatives values of the function $\phi$ [14]. The goal is to obtain a value of
the step length $\lambda$ that satisfies the following condition (called the *sufficient decrease condition*):

$$
\phi(\lambda_k) \leq \phi(0) + c\lambda_k \phi'(0) \tag{A.5}
$$

without it being *too small*. In that equation, $c$ is a small constant value, say, $c = 10^{-4}$ and thus the *sufficient decrease condition* requires a little more decrease in $f$. (Notice that $c\lambda_k \phi'(0)$ is negative because $c \geq 0$, $\lambda_k \geq 0$ and $\phi'(0) \leq 0$. In fact the latter inequality can be verified because from (A.4) we have that $\phi'(0) = p_k^T \nabla f_k$ and from (A.2) we know that $p_k^T \nabla f_k \leq 0$). At each iteration, a new value of $\lambda$ is generated such that $\lambda_i$ is not too much smaller than $\lambda_{i-1}$. Suppose we are given an initial value of the step length $\lambda_0$, then in case we have the following inequality satisfied

$$
\phi(\lambda_0) \leq \phi(0) + c\lambda_0 \phi'(0) \tag{A.6}
$$

the search terminates because this step length satisfies the condition. Else, the interval $[0, \lambda_0]$ contains acceptable step lengths. We interpolate $\phi_0$, $\phi'_0$ and $\phi(\lambda_0)$ to obtain $\phi_q(\lambda)$ that quadratically approximates $\phi(\lambda)$. $\phi_q(\lambda)$ is expressed as:

$$
\phi_q(\lambda) = \left( \frac{\phi(\lambda_0) - \phi(0) - \lambda_0 \phi'(0)}{\lambda_0^2} \right) \lambda^2 + \phi'(0) \lambda + \phi(0) \tag{A.7}
$$

The next term in the sequence, i.e. the new trial value $\lambda_1$, is the minimizer of this quadratic:

$$
\lambda_1 = \frac{-\phi'(0) \lambda_0^2}{2[\phi(\lambda_0) - \phi(0) - \phi'(0) \lambda_0]} \tag{A.8}
$$

In case we satisfy the sufficient decrease condition at the value $\lambda_1$ then the search is terminated. Else, a cubic function interpolating $\phi_0$, $\phi'_0$, $\phi(\lambda_0)$ and $\phi(\lambda_1)$ is constructed:

$$
\phi_c(\lambda) = a\lambda^3 + b\lambda^2 + \lambda\phi'(0) + \phi(0) \tag{A.9}
$$

where

$$
\begin{bmatrix}
  a \\
  b
\end{bmatrix} = \frac{1}{\lambda_0^2 \lambda_1^2 (\lambda_1 - \lambda_0)} \begin{bmatrix}
  \lambda_0^2 & -\lambda_1^2 \\
  -\lambda_0^3 & \lambda_1^3
\end{bmatrix} \begin{bmatrix}
  \phi(\lambda_1) - \phi(0) - \phi'(0) \lambda_1 \\
  \phi(\lambda_0) - \phi(0) - \phi'(0) \lambda_0
\end{bmatrix} \tag{A.10}
$$

The minimizer $\lambda_2$ of $\phi_c$ is in the interval $[0, \lambda_1]$. This can be seen by differentiating $\phi_c(x)$. $\lambda_2$ is expressed as:

$$
\lambda_2 = \frac{-b + \sqrt{b^2 - 3a\phi'(0)}}{3a} \tag{A.11}
$$
If needed, the above process is repeated using a cubic interpolant of $\phi(0)$, $\phi'(0)$ and the two most recent values of $\phi$ until we find a value of $\lambda$ that satisfies the sufficient decrease condition. In case $\lambda_i$ is very close to or much smaller than its predecessor $\lambda_{i-1}$, $\lambda_i$ is reset to $\lambda_{i-1}/2$. This ensures a reasonable progress at every iteration and a final value of $\lambda$ that is not too small.
Bibliography


