EFFICIENT ITERATIVE METHODS FOR ILL-CONDITIONED LINEAR AND NONLINEAR NETWORK PROBLEMS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Victoria E. Howle

January 2001

EFFICIENT ITERATIVE METHODS FOR ILL-CONDITIONED LINEAR AND NONLINEAR NETWORK PROBLEMS

Victoria E. Howle, Ph.D.

Cornell University 2001

We propose two methods in this thesis: an iterative method for solving a complex-symmetric linear system arising in electric power networks, and a method for numerical solution of the swing equations. The swing equations are a system of differential-algebraic equations (DAE) used to determine transient stability of electrical power networks. Our iterative method extends Gremban, Miller, and Zagha's [11] support-tree preconditioner to handle complex weights and vastly different admittances. The underlying iteration is a modification to BiCG-STAB to enhance accuracy. Our method for numerical solution of the swing equations is based on a standard linear multi-step method for solving DAEs combined with the linear solver described above. In the case of "islanding," our method represents a significant improvement over methods currently used. Since the linear solver does not apply directly to the transient stability equations, we include analysis to show that the DAE solver's inner loop can be reduced to a problem amenable to the technique proposed in the first part of this thesis. Computational results for both methods are described.

BIOGRAPHICAL SKETCH

In the not too distant past, Victoria E. Howle was born in Harrogate, England to Billy and Sue Howle. Shortly thereafter, Billy, Sue, and brother Jeff all returned to Maryland with their new family member. There Vicki grew up, eventually attending Arundel High School, where she enjoyed various classes, playing field hockey and lacrosse, and playing piano for theater events. In her Junior year, Vicki also began dating Kevin Long, whom she married a few years later.

In 1984, she graduated high school and started college at the University of Maryland, later transferring to Rutgers University when Kevin began graduate school at Princeton. In 1988, she graduated from Rutgers with a BA in English Literature and soon began a career as a technical writer. This career spanned about five years and several towns and companies, the highlight in both categories being V.I. Corporation in Northampton, MA. Although she learned many things and met some wonderful people in her technical writing positions, eventually Vicki began wishing for a more technical career. After much thought, and one extremely uninspiring job, Vicki decided to return to school to study mathematics. She attended the State University of New York at Buffalo, which she lived near at the time. There she took mathematics classes part time for two years and applied to graduate school.

In 1995, Vicki moved to Ithaca, NY (Kevin followed a little later) and began graduate school at Cornell University in the Center for Applied Mathematics. About five years later, graduate school comes to a close. Although there are many things that she enjoyed about graduate school, and especially about Ithaca, it is time to finish up. She is moving on to California, the land of sunshine and many, many jobs, leaving behind a great department, beloved friends, and this thesis, but taking many wonderful memories with her.

To Mom for first sparking my mathematical interest by teaching me about Pi when I was in fourth grade and you were taking real estate courses. I already liked math, but that was the first time I loved it.

To Dad for teaching me my first iterative method—using Newton's method to find the square root of a number on our little four-function calculator.

To both of you for your past and continuing love, encouragement, and support.

ACKNOWLEDGMENTS

Many people have helped me along the way to getting to graduate school and completing this thesis. I would first like to thank my advisor, Steve Vavasis, without whom this thesis would never have happened. Steve's insight, support, patience, enthusiasm, and encouragement were all invaluable and greatly appreciated. He has been tremendously helpful in all aspects of my graduate life, and it has been a joy to work with him. Tom Coleman, Lars Wahlbin, and Steve Strogatz were of great help in many aspects of my graduate career, not the least of which was serving on my special committee and providing feedback and advice along the way. Nick Trefethen was the one who first sparked my interest in numerical linear algebra, which ultimately led to my choosing to focus my research interests on numerical analysis. His inspiring and enthusiastic teaching and his efforts and insights during our research project secured my interest in the field.

I spent several summers during graduate school working at Sandia National Laboratories in Livermore, CA. Many thanks to Patty Hough and the rest of the CSMR department. They provided much appreciated mentoring and experience and are a wonderful group of people to work with.

During my time as an undergraduate at SUNY Buffalo, John Ringland and Roger Waxler provided encouragement and mentoring, in addition to their lively and inspiring teaching. I also had the good fortune as an undergratuate to participate in the Mills Summer Mathematics Institute. Ami Radunskaya and Erica Flapan of Pomona College were a great inspiration and helped to solidify my determination to go to graduate school.

Many people have made my graduate years memorable and fun. Dolores Pendell buffered me and the rest of CAM from most of Cornell's bureaucracy. My thanks to her for that and for always having a smile and friendly words when I needed them. It would have been hard to survive my first years in graduate school without Stephanie Mason. We pulled many an all-nighter our first year, working and talking and laughing our way through the night and unwinding with Star Trek episodes when we needed a break. Kathryn Nyman helped me keep my sanity with her friendship and musical diversions. Who could ever forget her polka rendition of our alma mater? Dawn Chamberlain has become a good friend, always willing to talk and listen or just to hang out and watch a movie. In addition to the gift of her friendship, Dawn also introduced me to the book groups BibliaFiles and Books & Cooks, both of which helped me, and continue to help me, remember than there is much more to life than mathematics. Thanks also for many fun conversations and general hanging out to Aaron Deever, Joel Ariaratnam, Guðbjörn Jónsson, Debra Goldberg, Ed Mosteig, Peter Calabrese, Suzanne Shontz, and all the rest of my my fellow CAMsters. You have been a wonderful group of people to work and play with.

There are many people who I met and grew close to in Ithaca as a result of

spending much of time free time at St. Paul's UMC. St. Paul's felt like home from the first day I walked in the door, and some of the people I met there will be part of my family for the rest of my life. Thanks especially to the members of the RCMA and its spin-offs and to my small group. Rebecca Dolch and Carrie Call have forever raised the standards for what I expect of a pastor.

I don't have words adequate to express my gratitude for the friendship of Jacqueline Bonner. My world would be so much less bright without her presence. I am grateful for the time we have spent together, for all of the precious memories of working and playing and talking together. Although there are many miles between us at the moment, she is and always will be close to me.

As has been true for many years now and through many changes in my life, Kevin Long's love, encouragement, and belief in me have given me the courage to try, and stick with, graduate school. He has done more than his share of many things so I'd have time to work on my thesis; he's been there to discuss mathematics, politics, feminism, and religion, and anything else that came to mind; and most importantly, he has been there with loving words and hugs and music. I am grateful for his continuing love, which I have had the pleasure of receiving and returning for more than half of my life now.

Finally, there is the inevitable need for funding. This work has been supported in part by NSF grant CCR-9619489. Research was supported in part by NSF through grant DMS-9505155 and ONR through grant N00014-96-1-0050.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

We propose in this thesis an iterative method for solving a complex-symmetric linear system arising in electric power networks and a method for numerical solution of the swing equations, which are a system of differential-algebraic equations (DAE) used to determine transient stability of electrical power networks. In this introduction, we briefly describe the linear system and the related transient stability problem.

1.1 The Linear System

Consider an alternating-current (AC) electrical network composed of generators and loads joined by a graph. Each node in the graph has a voltage, which is a complex number. The magnitude of the complex number is the magnitude of the voltage, and the arg is the phase difference of the voltage with respect to some reference phase. Similarly, currents in the system are also complex numbers associated with graph edges. In steady-state, the generators can be modeled as voltage sources with a fixed voltage. The loads can be modeled as devices with fixed impedance. Impedance is a complex number with a positive real part.

For a network with m edges and n nodes, if we are given the voltages of the generators and the impedances of the loads, then the problem of recovering the voltages at all nodes reduces to solving a linear equation of the form:

$$A^T D^{-1} A\mathbf{v} = A^T D^{-1}\mathbf{b} \tag{1.1}$$

in which A is an $m \times n$ real matrix, D is an $m \times m$ complex diagonal matrix whose diagonal entries have positive real parts, $\mathbf{b}$ is a complex m-vector, and $\mathbf{v}$ is the n-vector of unknowns.

There are two main issues we address in our iterative method for solving this system: obtaining a good preconditioner, and applying the preconditioner accurately. The second issue is important since even if we have a good preconditioner M for $K = A^T D^{-1} A$, the system matrix K, and therefore the preconditioner M^{-1}, can be arbitrarily ill-conditioned in the presence of a fault that divides the network into disjoint components. Therefore, even though $M^{-1}K$ is well-conditioned, the product $M^{-1}(K\mathbf{v})$ may still be computed inaccurately.

In Chapter 2, we address the first issue through extensions to support tree preconditioners and the second through a splitting technique.

1.2 The Swing Equations

Transient stability analysis determines whether a system locks back onto a steady state solution after a large disturbance (fault) occurs and is cleared. The transient stability problem can be stated as follows. Assume that we have an electrical power network, consisting of generators and loads, operating in equilibrium. Consider the following fault scenario. At some point in time, a fault occurs in the system. Relays detect the fault and cause circuit breakers to trip, resulting in a nearly open circuit. The circuit breakers remain open until the fault has cleared, then they reclose, re-establishing the original circuit. The fault sequence leads to a transient in the system. If the transient dies out, and the system locks back onto an equilibrium condition, the system is said to be transient stable [1] for the given fault sequence. Note that this fault scenario is one of many possible scenarios.

We are interested in particular in solving this problem in the catastrophic case of system "islanding." Islanding is when one or more parts of the network become effectively disconnected from the rest of the network. We describe islanding in more detail in Chapter 3.

To determine whether a network is transient stable for a given fault scenario, we need to solve a system of differential-algebraic equations, called the swing equations, which model the behavior of the generators in the network. They consist of a system of differential equations for each generator in the network coupled with the algebraic equations (2.2). The components of (2.2) are as described in (2.3) except that the generators are no longer constant voltage sources, since we are no longer considering the steady-state case. Standard methods for numerically integrating DAEs include a Newton-like inner iteration, which requires solving a linear system involving the Jacobian matrix of the DAEs. The Jacobian matrix of the swing equation can be arbitrarily ill-conditioned in the case of islanding, making the solution of the DAE inaccurate.

The method we propose in Chapter 3 can accurately integrate the swing equations during the fault (i.e., while the circuit breakers are still open), even in the case of islanding.

Our method for integrating the swing equations combines the new method of Chapter 2 for stably solving the extremely ill-conditioned linear systems arising in faulty AC networks with standard methods for solving differential algebraic equations (DAE). The method described in Chapter 2 does not directly apply to transient stability equations, however. In Chapter 3 we include analysis to show that the DAE solver's inner loop can be reduced to a problem amenable to the technique of Chapter 2.

Chapter 2

An Iterative Method for Solving Complex-Symmetric Systems Arising in Power Modeling*

2.1 AC power networks

Consider the linear system

$$A^T D^{-1} A\mathbf{v} = A^T D^{-1}\mathbf{b} \tag{2.1}$$

in which A is an $m \times n$ real matrix, D is an $m \times m$ complex diagonal matrix whose diagonal entries have positive real parts, $\mathbf{b}$ is a complex m-vector, and $\mathbf{v}$ is the n-vector of unknowns.

*Portions of this chapter are based on "An Iterative Method for Solving Complex-Symmetric Systems Arising in Electrical Power Modeling," [15] submitted to SIAM J. Matrix Anal. Appl..

Equation (2.1) arises in the analysis of an alternating-current (AC) electrical network composed of generators and loads joined by a graph. Each node in the graph has a voltage, which is a complex number. The magnitude of the complex number is the magnitude of the voltage, and the arg is the phase difference of the voltage with respect to some reference phase.

Similarly, currents in the system are also complex numbers associated with graph edges. The generators can be modeled as voltage sources with a fixed voltage. The loads can be modeled as devices with fixed impedance. Impedance is a complex number with a positive real part.

If one is given the voltages of the generators and the impedances of the loads, then the problem of recovering the voltages at all nodes reduces to solving linear equations of the form (2.1). In this case, A is the node-arc incidence matrix of the network. A *node-arc incidence matrix* (NAI) of a directed graph has one row for every edge of a graph and one column for every node. In each row, all entries are zeros except for exactly one '1' and one '-1' per row, which correspond to the endpoints of the graph edges.

The diagonal matrix D stores impedances of the loads, $\mathbf{b}$ holds the generator voltages, and $\mathbf{v}$ is the vector of node voltages.

The linear system (2.1) is obtained from Ohm's law and Kirchhoff's law (current balance):

$$\begin{aligned} D\mathbf{i} + A\mathbf{v} &= \mathbf{b} \\ A^T\mathbf{i} &= \mathbf{0}. \end{aligned} \tag{2.2}$$

If we multiply Ohm's law by $A^T D^{-1}$ and apply current balance, we obtain the linear system (2.1).

As and example of the various components in equation (2.1), consider the following simple network.

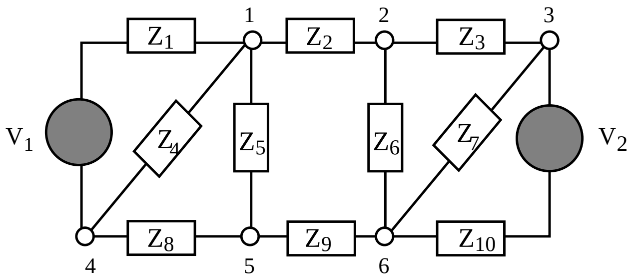

Figure 2.1: A simple AC network with two generators. Each edge j has a given constant impedance Z_j.

For this network,

$$
\begin{aligned}
A &= \begin{bmatrix}
1 & & & -1 & & \\
1 & -1 & & & & \\
 & 1 & -1 & & & \\
1 & & & -1 & & \\
1 & & & & -1 & \\
 & 1 & & & & -1 \\
 & & 1 & & & -1 \\
 & & & 1 & -1 & \\
 & & & & 1 & -1 \\
 & & 1 & & & -1
\end{bmatrix}, \\
D &= \operatorname{diag}(\begin{bmatrix} Z_1 & Z_2 & Z_3 & Z_4 & Z_5 & Z_6 & Z_7 & Z_8 & Z_9 & Z_{10} \end{bmatrix}), \\
\mathbf{b} &= \begin{bmatrix} V_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & V_2. \end{bmatrix}^T
\end{aligned}
\tag{2.3}
$$

When the network is faulty, e.g., near open-circuit exists in transmission lines, D is ill-conditioned. See Bergen [1] for more information about modeling AC networks. Modeling faulty networks is important in practice since load-regulating devices must be designed to function properly even if part of the network fails.

We assume throughout this thesis that the gap between the magnitudes of high impedance wires and low impedance wires is large, and that removal of the high impedance wires disconnects the graph. In this case, the matrix $K = A^T D^{-1} A$ can be arbitrarily ill-conditioned.

We make two main contributions in this chapter. The first contribution is an extension of Gremban's support tree preconditioner to cover complex weights (i.e., AC networks) and widely-varying edge weights (i.e., faults). Once we have a good preconditioner M, in the presence of a fault that disconnects the graph, K and therefore M^{-1} can be extremely ill-conditioned separately. Even though the product $M^{-1}K$ is well-conditioned, $M^{-1}(K\mathbf{v})$ may be computed inaccurately. Our second contribution is a technique that computes $M^{-1}(K\mathbf{v})$ accurately by splitting $K\mathbf{v}$ into its components in the range and null space of the functioning edges. For our algorithm to work efficiently, we need an efficient projection into the range and null spaces of the functioning edges.

A more general approach to achieving high accuracy in this kind of layered system was proposed by Bobrovnikova and Vavasis [3]. The method of [3] does not assume that there is an efficient projection into the range and null space of the functioning edges. But that method appears to be very difficult to precondition.

A direct algorithm known as Complete Orthogonal Decomposition was proposed by Hough and Vavasis [13]. This method applies to the weighted least squares problem associated with faulted DC power networks. However, the method relies on the system being real and positive definite. There is no simple extension to the complex-symmetric case.

Other previous related work includes another combinatorial preconditioner for weighted node-arc adjacency matrices by Guo and Skeel [12], previous versions of support tree preconditioners by Vaidya [19] and Bern, et al. [2], and work by Vuik, Segal, and Meijerink [20] on a related mathematical problem arising in diffusion modeling using an explicit eigenvector projection. The problem analyzed by Vuik et al. involves a real, symmetric, positive definite matrix that is highly ill-conditioned due to a large contrast in permeability coefficients in the system being modeled. The method proposed by Vuik et al. relies on a good choice of projection vector, which involves knowing properties of the eigenvectors.

2.2 Support Trees

Our system (2.1) is singular because the nodes are ungrounded. We address this detail by projecting vectors onto the range space of $A^T D^{-1} A$. This projection is ignored for the remainder of the thesis. After grounding the system, it is still ill-conditioned for two reasons.

The first source of ill-conditioning is inherent in NAI matrices. For example, if A describes an $n \times n$ grid-graph, $\kappa(A^T A) = O(n^2)$. This ill-conditioning has been addressed by *support-tree* preconditioners developed by Gremban, Miller, and Zagha [11]. We also use support-tree preconditioners in our method.

Gremban et al. form a support tree preconditioner as follows. First, divide the nodes of the network graph into some number of approximately equal-sized components. Then recursively subdivide the components, etc., until all of the individual nodes have been separated. Note that the method does not depend on the number of components in each subdivision. In our computational examples at the end of this chapter, we recursively subdivide into quarters. Next, build a tree based on this partitioning. The root of the tree is in correspondence with the entire original graph. The children of the root are in correspondence with the components of the graph obtained from the first partition, and so on down to the leaves of the tree, which are in correspondence with the individual nodes of the original graph. Next assign weights to the edges of this tree based on the edge weights in the original graph. Let G be the original network graph and let ST be the support tree. Let v be a support tree node corresponding to subgraph V of G, and let e be the support tree edge from v to its parent. Assign weight to e equal to the sum of the conductances (i.e., reciprocal resistances) of edges in G connecting V to $G - V$.

Definition 1 *We define the* **weighted Laplacian matrix** $L(G)$ *of an* n*-node graph* G *as the* $n \times n$ *matrix whose* j*th diagonal entry corresponds to the sum of weights of the edges incident on the* j*th node of the graph. The* (i, j) *entry of* $L(G)$ *is equal to the negative of the weight of the edge in* G *connecting nodes* i *and* j*.*

Note that the system matrix K is the weighted Laplacian of the input graph G. Let T be the weighted Laplacian matrix of the new network ST. If n is the number of original circuit nodes, and t is number of nonsingleton subgraphs during partitioning, then $n + t$ is the number of support tree nodes. The matrix T is an $(n + t) \times (n + t)$ very sparse matrix.

Let M be the Schur complement of T obtained by eliminating the internal tree nodes. Gremban, Miller, and Zagha showed that M is good preconditioner for $A^T D^{-1} A$ in the real case (DC) with uniform edge weights (no faults). In particular, they showed that for equally-weighted instances of (2.1) (i.e., $D = I$), the condition number of grid-graphs is reduced from $O(n^2)$ to $O(n \log n)$. Although M is dense, linear systems of the form $M\mathbf{v} = \mathbf{r}$ can nonetheless be solved in linear time using Cholesky factorization on the larger sparse matrix T. Note that T has a perfect elimination order since it is the weighted Laplacian matrix of a tree and we can eliminate from the leaves to the root. They show that solving

$$\begin{pmatrix} T_1 & T_2 \\ T_2^T & T_3 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{r} \\ \mathbf{0} \end{pmatrix} \tag{2.4}$$

and letting $\mathbf{v} = \mathbf{v}_1$ is equivalent to solving $M\mathbf{v} = \mathbf{r}$ where $M = T_1 - T_2 T_3^{-1} T_2^T$ is the support tree preconditioner. Thus, the preconditioner is efficient in practice.

To extend the idea of support tree preconditioners to the case of an AC network with faults, we make a change to how we build the support tree as follows. First, assume that removal of the faulted edges disconnects the graph into at least two components. (If this assumption does not hold, then our method does not constitute an improvement over previous work [10].) We require the top level of the support tree to be composed entirely of faulted edges, and children of the root should be connected components of the network after faulty edges are deleted. We include all of the faulted edges in the first separator. This in turn puts all of the weights corresponding to faulted edges in the top level of the support tree. We build the rest of the support tree as before. We show below that with this change, the support tree preconditioners are good preconditioners for $A^TD^{-1}A$. In applying the preconditioner M in the AC case, the matrices $K = A^TD^{-1}A$ and M are complex-symmetric and hence the Cholesky factorization technique does not apply directly. We show that the LU factorization can still be stably performed without pivoting (and thus without fill-in). Thus, as before, we can solve systems of the form $M\mathbf{v} = \mathbf{r}$ by solving the larger sparse system involving T.

Theorem 1 *T has an LU factorization, and the elements of L are bounded, i.e., the computation is stable without pivoting.*

PROOF. The diagonal elements of T are exactly the negative sums of the off-diagonal elements. In addition, because we have ordered the tree nodes from the leaves to the root, T has perfect elimination order. The elements of L are thus only 0's, 1's, and -1's. The L matrix has 1's on the diagonal, -1's in the elements of T below the diagonal that are nonzero, and 0's elsewhere. ∎

In addition to the ill-conditioning inherent in NAI matrices, there is ill-conditioning in our application of (2.1) caused by widely varying weights in a faulted system. Let ρ_1 be the absolute value of a typical admittance in functioning wires, and let ρ_2 be the absolute value of a typical admittance in faulty wires. Then $\rho_1 \gg \rho_2$, and original linear system can be decomposed as $K = A^TD^{-1}A = K_1 + K_2$, where $K_1 = A_1^TD_1^{-1}A_1$ and $K_2 = A_2^TD_2^{-1}A_2$. Here subscripts $1, 2$ denote the partition into functioning and faulty wires respectively, hence $\|D_1\| \approx \rho_1$ and $\|D_2\| \approx \rho_2$.

Gremban et al. analyze only the case of nearly equal weights. We extend their analysis of condition numbers of the preconditioned system first to graphs having edge weights with widely varying magnitudes, assuming the change in forming the support tree that we discussed above, then to graphs with complex edge weights. We assume for simplicity in the following theorems that the faults are nearly open circuits, i.e., low admittance wires.

In the rest of this section, we extend Gremban's analysis as follows. Since electrical power networks are laid out geographically, grid-graphs are a reasonable first approximation to consider. Therefore, we first extend Gremban's analysis to DC networks with faults by proving that in the DC grid-graph case with faults along the median edges, the condition number of the preconditioned system is $O(n \log n)$, where n is the number of nodes in the system. The median edges are the edges running through the middle of the grid graph, horizontally and vertically. That is, removal of the median edges would divide the grid graph into four approximately equal sized components. Next, since in general electrical power networks are not actually laid out on grid-graphs and faults are not confined to being along the median edges, we show results for general graphs. Little is known about the behavior of support tree preconditioners on general graphs. Although we do not have an upper bound on the condition number of the preconditioned system on a general graph, we can show that for general DC networks with arbitrarily located faults, there is a bound on the condition number of the preconditioned system that is independent of the relative magnitude of the faults. Finally, we extend our results for grid-graphs and general graphs to the AC case. Given certain assumptions about the impedance values in the network, we show that we can bound the condition number in the AC faulted case based on the condition number of the related DC network gotten by taking the real part of the impedance values.

This series of theorems then shows that we can extend Gremban's support tree preconditioner, with certain changes, to be a good preconditioner in the case of AC networks with faults.

We first introduce some notation that will be useful in the proofs that follow.

Definition 2 *We refer to the* **condition number** *of the preconditioned system* $M^{-1}K$ *as* $\kappa(M,K)$*, where*

$$\kappa(M,K) = \max \frac{|\mathbf{x}^* M \mathbf{x}|}{|\mathbf{x}^* K \mathbf{x}|} \cdot \max \frac{|\mathbf{x}^* K \mathbf{x}|}{|\mathbf{x}^* M \mathbf{x}|}, \tag{2.5}$$

where the maxima are taken over nonzero vectors $\mathbf{x}$ *in the range space of* K *(which is equal to the range space of* M*). In the real case, the absolute value symbols are not needed.*

Note that this definition is equivalent to the usual eigenvalue definition of condition number in the real case.

Definition 3 *[10] For two real, positive semidefinite matrices* A *and* B*, the* **support** *of* B *for* A*,* $\sigma(A,B)$*, is defined to be the greatest lower bound over all* τ *such that* $\tau B - A$ *is positive semidefinite.*

We define the congestion as follows.

Definition 4 *[10] Let* A *and* B *be graphs. Let* p_0 *be an injective mapping from nodes of* B *into nodes of* A*. Suppose there exists a mapping* p *from edges in* B *to paths in* A *such that if* $e = (a,b)$ *is an edge in* B*, then the endpoints of* $p(e)$

are $(p_0(a), p_0(b))$. *For an edge* f *in* A, *let* $d_1, \ldots, d_k$ *be the edges in* B *such that* $f \in p(d_i)$, *that is,* d_i *are the* B *edges whose embedding path in* A *includes edge* f. *The* **congestion** γ *of edge* f *is*

$$\gamma = \frac{\sum_{i=1}^{k} weight(d_i)}{weight(f)}. \tag{2.6}$$

The **congestion** *of the mapping* $\gamma(A, B)$ *is the maximum congestion over all edges in* A.

Definition 5 *[10] Given an embedding of a graph* B *into a graph* A *as in Definition 4, the* **dilation** *of the embedding is defined to be the length of the longest path in* A *onto which an edge of* B *is mapped.*

We next prove a series of lemmas that we will use to show that in the DC (real) case, the upper bound on the condition number of the preconditioned system does not depend on the values of the faulted edges, and that for $n \times n$ (real) grid graphs, the condition number is $O(n \log n)$.

Lemma 1 *The support* $\sigma(A, B)$ *of a matrix* B *for a matrix* A *is less than or equal to the maximum congestion over edges in* B *times the maximum dilation over paths in* B.

Proof. This lemma is proved by Gremban[10] in the DC, unfaulted case. No change is necessary for the faulted DC case. ∎

Lemma 2 *Let* $K = A^T D^{-1} A$ *correspond to a DC (real weights) network. Let* T *and* M *be as defined above. Let* $\tilde{K}$ *be* K *extended with zeros to the same size as* T, *i.e.,*

$$\tilde{K} = \begin{bmatrix} K & 0 \\ 0 & 0 \end{bmatrix}. \tag{2.7}$$

Then $\kappa(M, K) \leq \sigma(\tilde{K}, T)\sigma(M, K)$.

PROOF. The addition of non-uniform edges does not change this proof. See Gremban [10] for the proof. ∎

Lemma 3 *Let* $K = A^T D^{-1} A$ *correspond to a DC (real weights) network on an* $n \times n$ *grid graph containing wires of two magnitudes, high admittance (functioning) wires and low admittance (nearly open circuit) wires, where the faults are along the median edges of the grid. Let* M *be the support tree preconditioner as defined above. Then* $\sigma(M, K) = O(n)$.

PROOF. In this proof, we further assume that n is a power of two. This assumption is made to simplify notation, but the theorem is true for general n. We are considering how well the system matrix K supports the preconditioner M. We assume the following standard mapping of the edges of M onto paths in the grid graph corresponding to K. Let e be an edge in M. The nodes of M correspond to the nodes in the original network K. If edge $e \in M$ connects nodes v_M and w_M in M, we map edge e to the path in K obtained by starting at the node $v_K \in K$ that corresponds to v_M and moving vertically until we reach the same row as node w_K (the node in K corresponding to node v_M). We then complete the path by moving horizontally until we reach node w_K.

We prove this lemma by invoking Lemma 1. We bound the congestion and dilation in separate arguments.

We first bound the dilation. The dilation is defined to be the longest path in K onto which an edge of the preconditioner M is embedded. The longest possible path given our embedding is from one corner of the grid to the diagonally opposite corner. The length of this path, and therefore the maximum dilation, is $O(n)$.

Next we bound the congestion. Let p be the mapping defined above between edges in M and paths in K. Let e be an edge in K, and let $d_1, \ldots, d_k$ be the edges in M such that $e \in p(d_i)$, that is, the d_i are the edges of M whose embedding path in K includes edge e. Recall that the congestion γ_e of edge e is defined to be

$$\gamma_e = \frac{\sum_{i=1}^{k} \mathrm{w}eight(d_i)}{\mathrm{w}eight(e)}. \tag{2.8}$$

Assume that the weights of functioning edges are a constant $O(1)$. (Later we will pick $1/4$.) Let ϵ denote the order of magnitude of weights on faulted edges, and assume $\epsilon \ll 1$.

To determine a bound on the congestion, we need to bound the sum in the numerator. Since M is the weighted Laplacian matrix associated with the graph obtained by reducing the internal nodes in the support tree, and element $M(j,k)$ of M, where $j \neq k$, is the conductance of the edge connecting nodes j and k in the graph. So if we can bound the elements of M, we bound the edge weights in the graph associated with M.

Let $\mathbf{v}$ be a vector of voltages at the nodes in M. Then $M\mathbf{v} = \mathbf{i}$, where $\mathbf{i}$ is a vector of the net current at each node in M. If $\mathbf{e}_j$ is the jth column of the n by

n identity matrix, then $M\mathbf{e}_j$ gives the jth column of M, and if $M\mathbf{e}_j = \mathbf{i}$, the kth element of $\mathbf{i}$ is the (k, j) element of M. Therefore, if we set the voltage of node j in M to one and the voltage of all of the other nodes to zero, i.e., make $\mathbf{v} = \mathbf{e}_j$, and solve for the current at the node k, this value is the same as the (k, j) element of M (and the same as the (j, k) element since M is symmetric). Bounding this for general j and k nodes bounds the entries of M and therefore the weights of the d_i edges in M.

We perform the conductance experiment just described to bound the numerator in the formula for the congestion of edge e. The matrix M is obtained from T by Gaussian elimination on the internal nodes of the support tree. Gaussian elimination corresponds to combining conductors in series, so the current flow through the two systems is identical. Therefore we can perform the experiment on the support tree and get the same result if we leave the internal node voltages floating. In particular, we set the voltage at leaf node j in the support tree to one, the voltage at leaf node k to zero, the voltages of the remaining leaf nodes to zero, and leave the internal node voltages floating, but require current balance at those nodes.

The conductance experiment we described above can be written as

$$T \begin{bmatrix} \mathbf{y} \\ \mathbf{e}_j \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{i} \end{bmatrix}, \tag{2.9}$$

where the unknowns are internal node voltages $\mathbf{y}$ and the vector $\mathbf{i}$ of current flowing out of each leaf node, and where

$$T = \begin{bmatrix} X & R \\ R^T & C \end{bmatrix}. \tag{2.10}$$

To find the conductance of an edge in M connecting nodes j and k, we look at the kth element in $\mathbf{i}$.

There is exactly one path between nodes j and k in the support tree. To solve for the kth element in $\mathbf{i}$, we reduce all nodes other than those on the path between j and k up to where they connect to the (j, k) path, performing the relevant parallel and series reductions as we go. For example, if we have a sixteen-node grid graph, Figure 2.2 shows the reduction of all non-path nodes up to the (j, k) path.

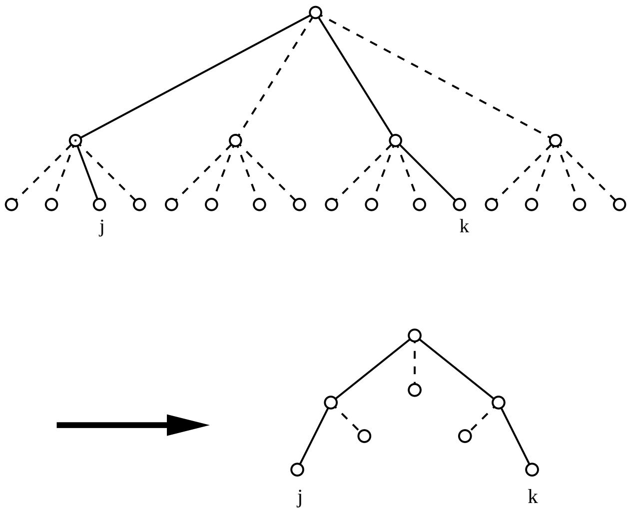

Figure 2.2: The solid lines in the first tree show the path between nodes j and k. The dashed lines are the edges that we reduce up to the (j, k) path. The second tree shows the result of reducing all other nodes up to the path between j and k. The (j, k) path is again shown with solid lines, which the reduced edges shown as dashed lines.

There are two cases we need to deal with in our conductance experiment: either the path between nodes j and k goes through the root node, or it does not.

First consider the case where the path goes through the root node. To reduce all of the nodes up to the (j, k) path, we perform a series of serial reductions, and a series of parallel reductions. Assuming a power-of-two grid (i.e., $n = 2^h$ nodes on a side for some h), we have the length of an edge of one separated block at level i is $e_i = n/2^i$, where level $i = 0$ corresponds to the 0th cut (the original graph), level $i = h$ corresponds to the final cut that completely separates the nodes of the original matrix. The size of the boundary of a separated block in the original grid graph is $\beta_i \leq 4e_i = 4n/2^i$, and the number of nodes per block is $n_i = e_i^2 = (n/2^i)^2$. Assuming, without loss of generality, that the original functioning edges have resistance (impedance) of 4 and the faulted edges have a resistance of $4/\epsilon$, an edge in the support tree connecting a child node w of node u at level $(i-1)$ has a resistance (impedance) of $1/2^{h-i-1}$.

As in Gremban's analysis, if we reduce nodes up to a child w of a node u at level $(i-1)$ the reduced system is equivalent to a node connected to u with an edge resistance less than or equal to $1/2^{i-1}$.

Therefore, the reduced system will look like the system in Figure 2.3 for a power-of-two grid. In this figure, v_0 represents the jth node of the tree (whose voltage has been set to one), v_0' represents the kth node of the tree (whose voltage has been set to zero), and $\hat{v}_l$ represents the root of the tree.

Once we have reduced the system to the (j, k) path, we solve for the net current at the kth node, i.e., at v_0' in Figure 2.3. This current will equal $M(j, k)$.

Solving current balance equations at each floating node, we get the recurrence relations

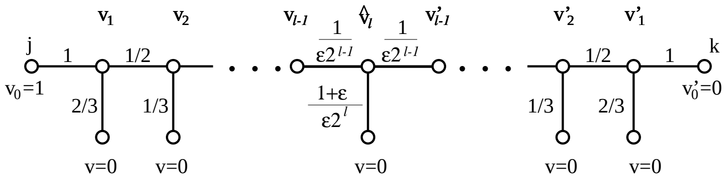

Figure 2.3: Support Tree reduced to the (j, k) path. The node $\hat{v}_l'$ represents the root of the tree, v_0 represents the j nodes, and v_0' represents the k node.

$$
\begin{aligned}
4v_{i+2} - 9v_{i+1} + 2v_i &= 0, \\
4v'_{i+2} - 9v'_{i+1} + 2v'_i &= 0.
\end{aligned}
\tag{2.11}
$$

We assume $v_i = \alpha^i$ and solve for the roots of $4\alpha^2 - 9\alpha + 2 = 0$ giving $\alpha = 1/4, 2$. We can now write the v_i and v_i' equations in terms of α:

$$
\begin{aligned}
v_i &= A(1/4)^i + B(2)^i \\
v_i' &= A'(1/4)^i + B'(2)^i
\end{aligned}
\tag{2.12}
$$

for some A, B, A', and B'. Using (2.12) at $i = 0$ and the fact that we have set $v_0 = v_j = 1$, we get $A = 1 - B$. Similarly, using the v_i' equation in (2.12) for $i = 0$ and the fact that we have set $v_0' = v_k = 0$ we get $A' = -B'$. Therefore, we have the equations

$$
\begin{aligned}
v_i &= (1/4)^i + (2^i - (1/4)^i)B \\
v_i' &= (2^i - (1/4)^i)B'
\end{aligned}
\tag{2.13}
$$

Using current balance at $\hat{v}_l$, we get the following equation for $\hat{v}_l$ in terms of v_{l-1} and

v'_{l-1}:

$$-(\epsilon 2^l + (\frac{\epsilon}{1+\epsilon})2^l)\hat{v}_l + \epsilon 2^{l-1}v_{l-1} + \epsilon 2^{l-1}v'_{l-1} = 0 \tag{2.14}$$

Similarly, we can use current balance at v_{l-1}, v'_{l-1}, v_{l-2}, and v'_{l-2} to get the following equations:

$$\begin{aligned}
-(2^{l-2} + 32^{l-3} + \epsilon 2^{l-1})v_{l-1} + 2^{l-2}v_{l-2} + \epsilon 2^{l-1}\hat{v}_l &= 0 \qquad (2.15)\\
-(2^{l-2} + 32^{l-3} + \epsilon 2^{l-1})v'_{l-1} + 2^{l-2}v'_{l-2} + \epsilon 2^{l-1}\hat{v}_l &= 0\\
2^{l-3}v_{l-3} - (2^{l-2} + 2^{l-3} + 32^{l-4})v_{l-2} + 2^{l-2}v_{l-1} &= 0\\
2^{l-3}v'_{l-3} - (2^{l-2} + 2^{l-3} + 32^{l-4})v'_{l-2} + 2^{l-2}v'_{l-1} &= 0
\end{aligned}$$

Putting (2.14) and (2.15) together along with (2.13) applied at v_{l-1}, v_{l-2}, v'_{l-1}, and v'_{l-2}, we can solve this system of equations for B'. We solve this system for B' (using Mathematica), getting the result

$$B' = \frac{7\epsilon 2^{-3l}}{72} + O(\epsilon 2^{-6l}) + O(\epsilon^2) \tag{2.16}$$

Using this value for B', we can solve for the current at the kth node.

$$\begin{aligned}
\mathbf{i}_k &= 1(\mathbf{v}'_1 - \mathbf{v}'_0)\\
&= \mathbf{v}'_1\\
&= (2 - 1/4)B'\\
&= \frac{49\epsilon 2^{-3l}}{288} + O(\epsilon 2^{-6l}) + O(\epsilon^2), \qquad (2.17)
\end{aligned}$$

which bounds the element $M(j,k)$ in the case where the (j,k) path goes through the root of the support tree.

We now have a bound of $(49\epsilon 2^{-3l})/288 + O(\epsilon 2^{-6l}) + O(\epsilon^2)$ for edge weights in M of edges whose corresponding path in the support tree passes through the root of the tree. To bound the congestion in this case, we also need a lower bound on the weight of $e \in K$. The (j,k) path passing through the root of the support tree means that e may be a faulted edge in K. The maximum congestion will occur when this is the case, i.e., when the weight of e is $O(\epsilon)$. Therefore the maximum congestion in this case is

$$\begin{aligned}
\gamma &= \frac{\sum_{i=1}^{k} \frac{49\epsilon 2^{-3l}}{288} + O(\epsilon 2^{-6l}) + O(\epsilon^2)}{O(\epsilon)} && (2.18)\\
&= \frac{O(2^{3l}\frac{49}{288}\epsilon 2^{-3l})}{O(\epsilon)} && (2.19)\\
&= O(1). && (2.20)
\end{aligned}$$

Next, we consider the case where the (j,k) path does not go through the root node of the tree. In this case, after reducing the tree as before to the (j,k) path, we have a completely unfaulted path between nodes j and k in the support tree, with an extra path off of the root $\hat{v}_l$ containing all of the faulted edges. This extra path will only reduce the current making it from node j to node k. Therefore, we can get an upper bound on the current at node k by removing this branch. Since we are assuming that the functioning wires have conductance of $O(1)$, the current at node k will be $O(1)$. As in Lemma 3, the edge of K supported by this path cannot be a faulted edge. Therefore, the congestion in this case is also $O(1)$.

Therefore, regardless of whether the (j,k) path passes through the faulted edges of the support tree, the congestion γ_e of edge e is $O(1)$, and thus the maximum congestion over all edges $e \in K$ is $O(1)$.

Since the support $\sigma(M, K)$ is less than or equal to the product of the congestion and the dilation, and we already showed that the dilation is $O(n)$, we have that the support of K for M, $\sigma(M, K)$, is $O(1) \cdot O(n) = O(n)$. ∎

Lemma 4 *Let $K = A^T D^{-1} A$ correspond to a DC (real weights) network on an $n \times n$ grid graph containing wires of two magnitudes, high admittance (functioning) wires and low admittance (nearly open circuit) wires, where the faults are along the median edges of the grid. Let T be as defined above and let $\tilde{K}$ be as defined in Lemma 2. Then $\sigma(\tilde{K}, T) = O(\log n)$.*

PROOF. In this case, we are considering how well the support tree matrix T supports the extended system matrix K. The only edges of $\tilde{K}$ are the edges of K, therefore we only need to map edges of K into paths of T. We assume the following standard mapping of edges of K onto paths in T. Let e be an edge in K connecting nodes v_K and w_K. The leaves of the support tree correspond exactly to the nodes of the original network. Since v_K and w_K are nodes in the original network, they correspond to two leaf nodes in the support tree. We map edge e to the unique path between the corresponding leaves v_T and w_T in the support tree.

As before, the support of matrix T for $\tilde{K}$ is less than or equal to the product of the maximum congestion over edges in T and the maximum dilation over paths in T, i.e., $\sigma(\tilde{K}, T) \leq \gamma(\tilde{K}, T)\rho(\tilde{K}, T)$. We bound these two quantities separately.

The maximum dilation comes from the edge in K that must be mapped through a path that goes through the root of T. The length of this path is $2 \log_2 n + 2$.

Next we find the maximum congestion. Let p be the mapping described above from edges in K to paths in T, let e be an edge in T, and let $d_1, \ldots, d_k$ be the edges in K such that $e \in p(d_i)$ for some i. The congestion is defined, as before as

$$\frac{\sum_{i=1}^{k} weight(d_i)}{weight(e)}. \tag{2.21}$$

Assume that edge $e \in T$ connects nodes v_j and v_k where v_j is a child node of v_k. Then we have defined the edge weights of T such that the weight of edge e equals the sum of the edges in K that connect the nodes associated with the leaves of the tree rooted at v_j to the rest of the graph. This sum is exactly $\sum_{i=1}^{k} weight(d_i)$. Therefore, the congestion over any edge in T is one.

Thus, the support $\sigma(\tilde{K}, T)$ is less than or equal to $1 \times O(\log n) = O(\log n)$. ∎

Using Lemmas 1 through 4, we can show that in the case of an $n \times n$ grid graph with faults along the median edges of the grid, the condition number of the preconditioned system is $O(n \log n)$. We summarize this result in the following theorem.

Theorem 2 *Let $K = A^T DA$ correspond to a DC (real weights) network on an $n \times n$ grid graph containing wires of two magnitudes, high admittance (functioning) wires and low admittance (nearly open circuit) wires, where the faults are along the median edges of the grid. Let the preconditioner M be as above. Then $\kappa(M, K) = O(n \log n)$.*

PROOF. Lemmas 1 through 4 taken together prove this case. ∎

In the non-grid case, we show that the condition number of the preconditioned system has a bound that is independent of the magnitude of the fault.

Theorem 3 *Let $K = A^T DA$ correspond to a connected DC (real weights) network containing wires of two magnitudes, high admittance (functioning) wires and low admittance (nearly open circuit) wires. Let the preconditioner M be as above. Assume that the admittance of a typical functioning edge is ρ_1, and the admittance of a typical faulted edge is ρ_2. Then $\kappa(M, K)$ has an upper bound that is independent of ρ_2/ρ_1.*

PROOF. From Lemma 2, we have $\kappa(M, K) \leq \sigma(\tilde{K}, T)\sigma(M, K)$, where K, $\tilde{K}$, T, and M are as before. We have also shown in Lemma 1 that for matrices A and B, $\sigma(A, B)$ is less than or equal to the product of the maximum congestion over edges in B and the maximum dilation over paths in B. We therefore prove this theorem by showing separately that the upper bounds on $\sigma(\tilde{K}, T)$ and $\sigma(M, K)$ are independent of ρ_2/ρ_1. We do this by considering the congestion and dilation of the respective embeddings. In particular, we show that both the congestion and dilation are independent of ρ_2/ρ_1 for the case of K supporting M and for the case of T supporting $\tilde{K}$.

We start with the case of K supporting M, $\sigma(M, K)$. Fix a mapping from edges in M to paths in K. Such a mapping exists by the assumption that the network is connected. We further require that if there exists a path from a to b in K using unfaulted edges, then such a path must be selected for the embedding. The dilation is given by the longest path in K onto which an edge of M is embedded. For any mapping we choose, this path can be as long as the longest path in the K graph, but this length does not depend on ρ_2/ρ_1.

For the congestion, we follow a proof similar to that for the grid-graph case. As in that case, we need an upper bound on the elements of M, which will give us a bound on the edge weights of the graph associated with M. Choose an edge $e \in K$. There are two cases to consider; either e is a faulted edge or it is not a faulted edge. If e is not a faulted edge, then weight$(e) = \Omega(\rho_1)$. Let $d_1, \ldots, d_k$ be the M edges whose K-paths include edge e. We claim that these edges all have weight of $O(\rho_1)$. We show this by performing the same conductance experiment and branch/path reduction as in Lemma 3, reducing the support tree to the (j, k) path that connects the leaf nodes of T that correspond to the nodes in K connected by edge e. We can get an upper bound on the current at node k by removing the faulted branch of the reduced system. We can do this because the extra edges associated with the faulted parts of the system will only make the current that gets to the kth node less. The current at the k-th node is therefore $O(\rho_1)$ giving us that $M(j, k) = O(\rho_1)$. As before, the current at node k equals the value of $M(j, k)$, so the M edges whose K-paths include edge e have weights of $O(\rho_1)$. Thus the congestion in this case is $O(\rho_1)/\Omega(\rho_1) = O(1)$ and is independent of ρ_2/ρ_1.

If e is a faulted edge, then weight$(e) = \Omega(\rho_2)$. We again let $d_1, \ldots, d_k$ be the M edges whose K-paths include edge e. We claim that in this case these edges have weights of $O(\rho_2)$. We again show this be performing the same conductance experiment and branch/path reduction as in Lemma 3. By construction, any K-path using e must be a path between different K_1-components of K. Therefore, the (j, k) path in T must go through the root of the tree. Contracting as before to the (j, k) path, we can see that the current at node k must be $O(\rho_2)$. Therefore $M(j, k) = O(\rho_2)$, and the M edges whose K-paths include edge e have weights of

$O(\rho_2)$. Thus the congestion in this case is $O(\rho_2)/\Omega(\rho_2) = O(1)$ and is independent of ρ_2/ρ_1.

Since we have now shown that the dilation and congestion are separately independent of ρ_2/ρ_1, their product, which is an upper bound on the support of K for M, $\sigma(K, M)$, is also independent of ρ_2/ρ_1.

Next we consider the support of T for $\tilde{K}$, $\sigma(\tilde{K}, T)$. We map the edges of K (which are the same as the edges of $\tilde{K}$) onto paths in T using the same mapping as in Lemma 3. As in the previous case, the dilation is given by the longest path in T onto which an edge of K is mapped. This path can be as long as the longest path in T, but is nevertheless independent of ρ_2/ρ_1. For the congestion, let e be an edge in T and let $d_1, \ldots, d_k$ be the edges in K whose embedding path in T includes edge e. If e is a faulted edge, that is $weight(e) = O(\rho_2)$, then the d_i are faulted edges of K since only faulted edges of K would be mapped to paths that pass through the faulted edges of T. Therefore, $\sum_{i=1}^{k} weight(d_i) = O(k\rho_2)$. Since the weight of e is also $O(\rho_2)$ and the congestion is defined as the previous sum divided by the weight of e, the congestion is independent of ρ_2/ρ_1. If e is not a faulted edge, then some of the d_i may be faulted edges and some may not. In this case $\sum_{i=1}^{k} weight(d_i) = O(\rho_1 + \rho_2) = O(\rho_1)$. Since e is not a faulted edge, its weight is $O(\rho_1)$, and the congestion is again independent of ρ_2/ρ_1. Since the congestion and dilation separately are independent of ρ_2/ρ_1, so is their product, which is an upper bound on the support of T for $\tilde{K}$, $\sigma(\tilde{K}, T)$.

We have thus shown that the upper bounds on $\sigma(K, M)$ and $\sigma(\tilde{K}, T)$ are each independent of ρ_2/ρ_1. Therefore their product, which is an upper bound on the condition number of the preconditioned system, $\kappa(M, K)$, is also independent of ρ_2/ρ_1. ∎

Finally, we extend the analysis to the AC network case.

Definition 6 *Let B be a matrix of the form $B = A^T D^{-1} A$, where A is real with full column rank and D is a diagonal matrix. We say that B is* **cone positive definite** *if the diagonal elements of D lie in a pointed cone in the complex plane. In other words, if d_j is a diagonal element of D, then $d_j = x_j + iy_j$ where $x_j > 0$ and $|y_j| \leq \mu x_j$ for some positive cone constant μ.*

We assume that the original system $K = A^T D^{-1} A$ is cone positive definite. Note that satisfying this assumption simply requires that all of the impedances have some resistive component. In this case, we can bound the condition number of the AC case by the condition number of the real part of the system and a function of the cone constant. Strictly speaking, our matrices are not cone positive definite because of the 1-dimensional nullspace arising because the nodes are ungrounded. Again, this detail is not significant since our system matrix and our preconditioner have the same nullspace.

Lemma 5 *If the original system matrix K is cone positive definite, so is the preconditioner M.*

Proof. The edge weights of the support tree are by construction sums of edge weights of the original graph. Therefore the weighted Laplacian matrix T of the support tree is cone positive definite with the same cone constant as K. The preconditioner M is the Schur complement of T obtained by eliminating the internal tree nodes. Gaussian elimination performed on the internal nodes of the support tree is equivalent to performing series circuit reduction on those nodes [10]. If we reduce a node with edges connecting two nodes in series with admittances c_1 and c_2, the resulting edge has admittance $c_{new} = c_1c_2/(c_1 + c_2)$. Thus, if c_1 and c_2 are in the pointed cone with cone constant μ, so is c_{new} since

$$c_{new} = \frac{c_1c_2}{c_1 + c_2} = \frac{1}{\frac{1}{c_1} + \frac{1}{c_2}}, \tag{2.22}$$

and pointed cones are closed under addition and under taking reciprocals. Therefore, after eliminating all of the internal nodes of the support tree, we have a graph whose edge weights lie in the original pointed cone. The preconditioner M is the weighted Laplacian matrix of this graph, therefore M is cone positive definite with the same cone constant μ as the original system matrix K. ∎

Theorem 4 *In the AC case with impedances lying in a pointed cone in the complex plane,* $\kappa(M, K) \leq (1 + \mu^2)\kappa(Re(M), Re(K))$, *where* μ *is the cone constant.*

Proof. Assume that $K = A^T D^{-1} A$ as before and that $D^{-1} = E + iF$. Since the preconditioner M is also a weighted Laplacian matrix, we can write $M = R^T \Delta R$ where R is a node arc incidence matrix and Δ is a diagonal matrix. As with the components of K, R is a real matrix made up of ones, negative ones, and zeros, and Δ is a diagonal matrix with complex diagonal entries lying in the same cone as the diagonal entries of D. Let $\Delta = B + iC$.

We first establish bounds on $|\mathbf{x}^* A^T D^{-1} A\mathbf{x}|$ in terms of the real part of D^{-1} and the cone constant μ. Let $\mathbf{w} = A\mathbf{x}$. Let d_j represent the diagonal elements of D^{-1}, e_j represent the diagonal elements of E, and f_j represent the diagonal elements of F. Then we have

$$\begin{aligned} |\mathbf{x}^* A^T D^{-1} A\mathbf{x}| &= |\mathbf{w}^* D^{-1} \mathbf{w}| \\ &= \left| \sum_j |w_j|^2 \cdot d_j \right| \\ &\leq \sum_j \left(|w_j|^2 \cdot |d_j| \right) \\ &\leq \sum_j \left(|w_j|^2 \sqrt{1+\mu^2} \cdot e_j \right) \\ &= |\mathbf{w}^* E \mathbf{w}| \sqrt{1+\mu^2} \\ &= |\mathbf{x}^* A^T E A\mathbf{x}| \sqrt{1+\mu^2}. \end{aligned} \tag{2.23}$$

Next we bound $|\mathbf{x}^* A^T D^{-1} A\mathbf{x}|$ from below. In this case we have

$$\begin{aligned} |\mathbf{x}^* A^T D^{-1} A\mathbf{x}| &= |\mathbf{w}^* D^{-1} \mathbf{w}| \\ &= |\mathbf{w}^* (E + iF) \mathbf{w}| \\ &= \left| \sum |w_j|^2 e_j + i \sum |w_j|^2 f_j \right| \\ &\geq \left| \sum |w_j|^2 e_j \right| \\ &= |\mathbf{x}^* A^T E A\mathbf{x}|. \end{aligned} \tag{2.24}$$

The preconditioner M is also a weighted Laplacian matrix of the form $M = R^T \Delta R$ where R is a node arc incidence matrix and Δ is a diagonal matrix. In addition, by Lemma 5, M is cone positive definite with the same cone constant μ as K. Therefore, the same bounds apply to M. Namely,

$$|\mathbf{x}^* R^T \Delta R\mathbf{x}| \leq |\mathbf{x}^* R^T C R\mathbf{x}| \sqrt{1+\mu^2}, \tag{2.25}$$

and

$$|\mathbf{x}^* R^T \Delta R \mathbf{x}| \geq |\mathbf{x}^* R^T C R \mathbf{x}|. \tag{2.26}$$

Using these bounds, we can establish an upper bound on the condition number of the preconditioned AC network in terms of the condition number of the real part of the network.

$$\begin{aligned}
\kappa(M, K) &= \max \frac{|\mathbf{x}^* M \mathbf{x}|}{|\mathbf{x}^* K \mathbf{x}|} \cdot \max \frac{|\mathbf{x}^* K \mathbf{x}|}{|\mathbf{x}^* M \mathbf{x}|} \\
&\leq \max \frac{\sqrt{1+\mu^2}|\mathbf{x}^* R^T C R \mathbf{x}|}{|\mathbf{x}^* A^T E A \mathbf{x}|} \cdot \max \frac{\sqrt{1+\mu^2}|\mathbf{x}^* A^T E A \mathbf{x}|}{|\mathbf{x}^* R^T C R \mathbf{x}|} \\
&= (1+\mu^2) \max \frac{|\mathbf{x}^* R^T C R \mathbf{x}|}{|\mathbf{x}^* A^T E A \mathbf{x}|} \cdot \max \frac{|\mathbf{x}^* A^T E A \mathbf{x}|}{|\mathbf{x}^* R^T C R \mathbf{x}|} \\
&= (1+\mu^2) \kappa(Re(M), Re(K)).
\end{aligned} \tag{2.27}$$

∎

This analysis seemingly implies that we may as well precondition the AC network based only on its DC components. But our experiments indicate that keeping the imaginary part in the preconditioner gives better results. Therefore, there may exist a stronger analysis of the complex case.

2.3 Splitting

Since we are assuming that the gap between the magnitudes of high impedance wires and low impedance wires is large, and that removal of the high impedance wires disconnects the graph, the matrix $K = A^T D^{-1} A$ can be arbitrarily ill-conditioned. Even though the preconditioner effectively reduces condition number, it does not lead to accurate solution by itself under these conditions. Again letting M be the preconditioner for K, the difficulty is that although the product $M^{-1}K$ is well-conditioned, nonetheless the resulting vector $M^{-1}(K\mathbf{v})$ may be computed inaccurately because M^{-1} and K are very ill-conditioned separately.

We solve this problem by splitting K into its "large space," i.e., the range of K_1 (where K_1 denotes, as before, the weighted Laplacian of the functioning wires) and its "small space," which is null(K_1). These spaces ("large" versus "small") are reversed for M^{-1}. We can easily identify these spaces from the graph and then split $\mathbf{r} = (K_1+K_2)\mathbf{v}$ (in $O(n)$ time) into $\mathbf{r} = \mathbf{r}_R+\mathbf{r}_N$. We then compute $\mathbf{y} = M^{-1}(\mathbf{r}_R+\mathbf{r}_N)$ term by term, changing interim quantities that would be zero in exact arithmetic to zero, then recombining.

In particular, our algorithm is as follows. Each time we form $M^{-1}(K\mathbf{v}) = M^{-1}(K_1 + K_2)\mathbf{v}$, we first split $(K_1 + K_2)\mathbf{v}$ into its components in range(K_1) and null(K_1). To do this, we form an orthogonal projector PP^T onto the null space of K_1. The projector PP^T is formed based on the graph structure of the network, and can be applied in $O(n)$ time as follows. Removing the high impedance (faulty) wires disconnects the graph into separate components of functioning wires, which we refer to as K_1-components. A vector in range(K_1) is defined by the property that its entries over each K_1-component sum to zero. On the other hand, any vector in null(K_1) has the defining property that entries corresponding to nodes within an K_1-component of the graph have the same value. Thus, we build an orthogonal projector onto null(K_1) using a connected component search. Consider forming the product $\mathbf{r} = K\mathbf{v}$. We can split $\mathbf{r}$ according to: $\mathbf{r}_N = PP^TK\mathbf{v}$ and $\mathbf{r}_R = (I - PP^T)K\mathbf{v}$. We compute $\mathbf{r}_N$ as $PP^TK_2\mathbf{v}$ since the other term drops out. Similarly, we compute $\mathbf{r}_R$ as $K_1\mathbf{v} + K_2\mathbf{v} - PP^TK_2\mathbf{v}$. We then invoke our preconditioning algorithm with this split $\mathbf{r}$.

In invoking our preconditioning algorithm, as described in the section on preconditioning, instead of solving $M\mathbf{y} = \mathbf{r}$, we solve the equivalent (but sparse) problem

$$LU \begin{bmatrix} \mathbf{y} \\ \mathbf{c} \end{bmatrix} = G\mathbf{r}, \tag{2.28}$$

where LU is the (sparse) LU factorization of T, and G is the matrix that extends a vector with zeros to be the size $T\mathbf{w}$, i.e.,

$$G\mathbf{r} = \begin{bmatrix} \mathbf{r} \\ \mathbf{0} \end{bmatrix}. \tag{2.29}$$

We next forward solve for $\mathbf{u}_R = L^{-1}G\mathbf{r}_R$ and $\mathbf{u}_N = L^{-1}G\mathbf{r}_N$. Let S be the set of row numbers of nodes whose paths to the root contain only high-impedance edges. (Because of our definitions, S is exactly the root and children of the root.) Let J be the matrix that zeros out the components in S (i.e., J is a diagonal matrix with zeros in the diagonal positions corresponding to elements of S, and ones elsewhere on the main diagonal). We claim below that in exact arithmetic $\mathbf{u}_R = J\mathbf{u}_R$. Next we do the two backward solves, $\mathbf{w}_N = U^{-1}\mathbf{u}_N$ and $\mathbf{w}_R = U^{-1}J\mathbf{u}_R$. Note the important step in our algorithm of explicitly applying J to $\mathbf{u}_R$ before performing the back substitution.

Finally, we extract and return the part of $\mathbf{w}_R + \mathbf{w}_N$ that corresponds to the $\mathbf{y}$ in $M\mathbf{y} = \mathbf{v}$, i.e., $\mathbf{y} = G^T(\mathbf{w}_R + \mathbf{w}_N)$.

Lemma 6 *Let LU be the LU factorization of T and let Δ be a diagonal matrix consisting of the main diagonal elements of U. Let $\tilde{U} = \Delta^{-1}U$. Applying L^{-1} sums the entries from the leaves to the root of the support tree. The matrices $\tilde{U}$ and L have the properties that $\|\tilde{U}^{-1}\| \leq c_n$, $\|\tilde{U}\| \leq d_n$, $\|L^{-1}\| \leq k_n$, and $\|L\| \leq f_n$, where c_n, d_n, f_n, and k_n are constants that depend only on n.*

PROOF. These properties can all be proven combinatorially. We omit the proof. ∎

Lemma 7 *In exact arithmetic, $J\mathbf{u}_R = \mathbf{u}_R$.*

PROOF. Let n_i be a node in the support tree, and let N_i be the set of leaves rooted at n_i. Recall that the leaves of the support tree correspond directly to the nodes of the original network. The weight of the parent edge of n_i is constructed (as in Gremban) to be the sum of the edges connecting the original graph nodes N_i to the rest of the graph. This sum is known as the *frontier* of N_i. Since in our algorithm we force all high impedance edges to be in the top level graph separator, the nodes in the support tree indexed by elements of S are those whose corresponding nodes in the original graph are connected to the rest of the graph by only high impedance edges. This implies that leaves rooted at such a node make up one or more K_1-components.

Since applying L^{-1} sums the entries from the leaves to the root of the support tree, if $\mathbf{u}_R = L^{-1}G\mathbf{r}_R$ for example, then $u_R(i)$ equals the sum of the entries in $\mathbf{r}_R$ associated with leaves rooted at n_i. As mentioned above, entries in $\mathbf{r}_R$ corresponding to an K_1-component sum to zero since $\mathbf{r}_R \in \text{range}(K_1)$. Therefore, if $i \in S$, then $u_R(i) = 0$ in exact arithmetic. ∎

Let the constants α and β be defined as

$$\begin{aligned} \alpha &= \max\left(\frac{\max_i(D_1^{-1}(i,i))}{\min_i(D_1^{-1}(i,i))}, \frac{\max_i(D_2^{-1}(i,i))}{\min_i(D_2^{-1}(i,i))}\right) \\ \beta &= \max_i(D_1^{-1}(i,i)), \end{aligned} \tag{2.30}$$

where the $D_1^{-1}(i,i)$ are the weights of the low impedance edges, and the $D_2^{-1}(i,i)$ are the weights of the high impedance edges.

Lemma 8 *$\|\Delta^{-1}J\| \leq c_n \cdot \alpha/\beta$, where c_n is a constant depending only on n.*

PROOF. The elements of Δ are either sums of low impedance weights, sums of high impedance weights, or a mixture of low and high impedance weights. The elements of Δ consisting of the sums of high impedance weights correspond to the row numbers of nodes whose paths to the root contain only high impedance edges, i.e., elements whose associated entries in $\mathbf{r}_R$ correspond to an K_1-component. As shown in Lemma 7, these are the same elements of J that are set to zero. Therefore the product $\Delta^{-1}J$ is diagonal with the only nonzero elements being sums of low impedance weights. Therefore $\|\Delta^{-1}J\|$ is equal to the reciprocal of a sum of low impedance (high admittance) weights. The number of weights in the sum is bounded by the number of nodes in the support tree. Therefore, $\|\Delta^{-1}J\| \leq c_n \cdot 1/\min_i(D_1^{-1}(i,i)) \leq c_n \cdot \alpha/\beta$.

∎

Theorem 5 *Using this algorithm, the computed $M^{-1}(K\mathbf{v})$ has the form $(M^{-1}K + E)\mathbf{v}$, where E satisfies $\|E\| \leq \epsilon_{mach} c_n c_m \alpha + O(\epsilon_{mach}{}^2)$, and ϵ_{mach} is machine epsilon.*

Remark 1 *In contrast, if we compute $M^{-1}K\mathbf{v}$ in the naive manner, the computed result has the form $(M^{-1}K + E)\mathbf{v}$, where $\|E\| \leq \|K\| \cdot \|M^{-1}\| \cdot \epsilon_{mach}$, which could be very large.*

PROOF. As before, let $K = K_1 + K_2$, let T be the extended support-tree preconditioner, and $T = LU$ be its LU factorization. Recall that we can apply the preconditioner M^{-1}, where M is the Schur complement of T, by applying T^{-1} to the same vector extended with zeros. As before, let G be a matrix that extends $\mathbf{v}$ with zeros to be the size appropriate for multiplication by T.

The first step in computing $\mathbf{y} = M^{-1}(K\mathbf{v})$ is to apply $K = K_1 + K_2$ to $\mathbf{v}$ and split the result into its components in null(K_1) and range(K_1) as described above. The computed $\mathbf{r}_N$ has the form $\hat{\mathbf{r}}_N = (PP^TK_2 + E^0)\mathbf{v}$, where $\|E^0\| \leq \|K_2\| \cdot \epsilon_{mach} \cdot c_n$, and c_n is a small constant that depends only on n. This follows from the standard properties of matrix-vector multiplication and the fact that $\|PP^T\| = 1$. Note that we will reuse the symbol c_n in this proof to mean a constant depending only n whose value may change from statement to statement.

The range space component of $\mathbf{r}$ is then given by $\mathbf{r}_R = K_1\mathbf{v} + K_2\mathbf{v} - PP^TK_2\mathbf{v} = (I - PP^T)(K_1 + K_2)\mathbf{v}$, and the computed $\mathbf{r}_R$ has the form $\hat{\mathbf{r}}_R = [(I - PP^T)(K_1 + K_2) + E']\mathbf{v}$, where $\|E'\| \leq \|K_1\| \cdot \epsilon_{mach} \cdot c_n$. We are assuming here that $\|K_2\| \ll \|K_1\|$.

We next extend $\mathbf{r}_N$ and $\mathbf{r}_R$ with zeros (i.e., apply G), and perform the two forward solves $\mathbf{u}_R = L^{-1}G\mathbf{r}_R$ and $\mathbf{u}_N = L^{-1}G\mathbf{r}_N$. Note that $\mathbf{u}_R$ and $\mathbf{u}_N$ are no longer necessarily in the range(K_1) and null(K_1). The computed results again have the form $\hat{\mathbf{u}}_N = (L^{-1}G + E'')\hat{\mathbf{r}}_N$ and $\hat{\mathbf{u}}_R = (L^{-1}G + E''')\hat{\mathbf{r}}_R$, where $\|E''\| \leq \|L^{-1}\| \cdot \epsilon_{mach} \cdot c_n$ and $\|E'''\| \leq \|L^{-1}\| \cdot \epsilon_{mach} \cdot c_n$.

Next, let J be the matrix that sets to zero the elements of $\hat{\mathbf{u}}_R$ that would have been zero in exact arithmetic. (This step is explained in Lemma 7.)

Now we do the two back solves, $\mathbf{w}_N = U^{-1}\mathbf{u}_N$ and $\mathbf{w}_R = U^{-1}J\mathbf{u}_R$. The computed

$\mathbf{w}_R$ has the form $(U+F)\hat{\mathbf{w}}_R = J\hat{\mathbf{u}}_R$, where $|F| \leq |U| \cdot \epsilon_{mach} \cdot c_n$ entrywise. Rearranging we get $\hat{\mathbf{w}}_R = (U+F)^{-1}J\hat{\mathbf{u}}_R$. Substituting the Taylor series approximation for $(U+F)^{-1}$ and dropping the high-order terms, we have $\hat{\mathbf{w}}_R = (U^{-1}+U^{-1}FU^{-1})J\hat{\mathbf{u}}_R$.

Recall that $U = \Delta\tilde{U}$, where Δ is a diagonal matrix and $\tilde{U}$ is well-conditioned. Since $|F| \leq |U| \cdot \epsilon_{mach} \cdot c_n$ entrywise, F has the same structure, i.e., $F = \Delta\tilde{F}$, where Δ is the same diagonal matrix and $|\tilde{F}| \leq |\tilde{U}| \cdot \epsilon_{mach} \cdot c_n$ entrywise. We can now write the computed $\mathbf{w}_R$ as $\hat{\mathbf{w}}_R = (\tilde{U}^{-1} + \tilde{U}^{-1}\tilde{F}\tilde{U}^{-1})\Delta^{-1}J\hat{\mathbf{u}}_R$.

Putting all of the steps together, we have that

$$\begin{aligned} \hat{\mathbf{w}}_R &= (\tilde{U}^{-1} + \tilde{U}^{-1}\tilde{F}\tilde{U}^{-1})(\Delta^{-1}J)(L^{-1}G + E''') \\ &\quad [(I - PP^T)(K_1 + K_2) + E']\mathbf{v} \\ &= [(U^{-1}JL^{-1}G(I - PP^T)(K_1 + K_2)) + E_R]\mathbf{v}, \end{aligned} \tag{2.31}$$

where

$$\begin{aligned} E_R = &\tilde{U}^{-1}\tilde{F}\tilde{U}^{-1}\Delta^{-1}JL^{-1}G(I - PP^T)(K_1 + K_2) \\ &+ \tilde{U}^{-1}\Delta^{-1}JL^{-1}GE' \\ &+ \tilde{U}^{-1}\Delta^{-1}JE'''(I - PP^T)(K_1 + K_2) \\ &+ O({\epsilon_{mach}}^2). \end{aligned} \tag{2.32}$$

Thus we have

$$
\begin{aligned}
\|E_R\| &\leq \|\tilde{U}^{-1}\tilde{F}\tilde{U}^{-1}\Delta^{-1}JL^{-1}G(I-PP^T)(K_1+K_2)\| \\
&\quad + \|\tilde{U}^{-1}\Delta^{-1}JL^{-1}GE'\| \\
&\quad + \|\tilde{U}^{-1}\Delta^{-1}JE'''(I-PP^T)(K_1+K_2)\| \\
&\quad + O({\epsilon_{mach}}^2) \\
&\leq \|\tilde{U}^{-1}\tilde{F}\tilde{U}^{-1}\| \cdot \|\Delta^{-1}J\| \cdot \|L^{-1}G(I-PP^T)(K_1+K_2)\| \\
&\quad + \|\tilde{U}^{-1}\| \cdot \|\Delta^{-1}J\| \cdot \|L^{-1}G\| \cdot \|E'\| \\
&\quad + \|\tilde{U}^{-1}\| \cdot \|\Delta^{-1}J\| \cdot \|E'''\| \cdot \|(I-PP^T)(K_1+K_2)\| \\
&\quad + O({\epsilon_{mach}}^2) \\
&\leq \epsilon_{mach} \cdot c_n \cdot \alpha/\beta \cdot \|K_1+K_2\| + O({\epsilon_{mach}}^2),
\end{aligned}
\tag{2.33}
$$

where α and β are as described for Lemma 8.

Similarly for $\mathbf{w}_N$ we have

$$
\begin{aligned}
\hat{\mathbf{w}}_N &= (U^{-1}+U^{-1}FU^{-1})(L^{-1}G+E'')\hat{\mathbf{r}}_N \\
&= (U^{-1}+U^{-1}FU^{-1})(L^{-1}G+E'')(PP^TK_2+E^0)\mathbf{v} \\
&= (\tilde{U}^{-1}+\tilde{U}^{-1}\tilde{F}\tilde{U}^{-1})\Delta^{-1}(L^{-1}G+E'')(PP^TK_2+E^0)\mathbf{v} \\
&= [(U^{-1}L^{-1}GPP^TK_2)+E_N]\mathbf{v},
\end{aligned}
\tag{2.34}
$$

where

$$
\begin{aligned}
E_N = \tilde{U}^{-1}&\Delta^{-1}E''PP^TK_2 \\
&+ \tilde{U}^{-1}\Delta^{-1}L^{-1}GE^0 \\
&+ \tilde{U}^{-1}\tilde{F}\tilde{U}^{-1}\Delta^{-1}L^{-1}GPP^TK_2 \\
&+ O({\epsilon_{mach}}^2).
\end{aligned}
\tag{2.35}
$$

So we have

$$\begin{aligned}\|E_N\| \leq & \|\tilde{U}^{-1}\| \cdot \|\Delta^{-1} E'' P P^T K_2\| \\ & + \|\tilde{U}^{-1}\| \cdot \|\Delta^{-1} L^{-1} G E^0\| \\ & + \|\tilde{U}^{-1}\| \cdot \|\tilde{F}\| \cdot \|\tilde{U}^{-1}\| \cdot \|\Delta^{-1} L^{-1} G P P^T K_2\| \\ & + O({\epsilon_{mach}}^2).\end{aligned} \tag{2.36}$$

We have shown in Lemma 6 that $\|\tilde{U}^{-1}\| \leq c_n$ and $\|L^{-1}\| \leq c_n$. As above, $\|E''\| \leq \epsilon_{mach} \cdot c_n \|L^{-1}\|$ and $\|E^0\| \leq \epsilon_{mach} \cdot c_n \cdot \|K_2\|$. Finally, we are assuming that the high impedance weights are significantly larger than the low impedance weights, i.e., that $\|K_2\| \leq \rho \|K_1\|$ for a small constant ρ.

Since Δ is a diagonal matrix with sums of high impedance and low impedance weights on its diagonal, $\|\Delta^{-1}\| \leq 1/\rho \cdot \max_i (1/D_1^{-1}(i,i))$, where $D_1^{-1}(i,i)$ are the weights of the low impedance edges. Therefore, $\|\Delta^{-1}\| \leq \alpha/(\rho\beta)$, where α and β are as described for Lemma 8.

Therefore we can bound the error term $\|E\|$ as

$$\begin{aligned}\|E_N\| & \leq c_n \frac{1}{\rho} \frac{\alpha}{\beta} \epsilon_{mach} \rho \|K_1\| + O({\epsilon_{mach}}^2) \\ & \leq \epsilon_{mach} c_n \frac{\alpha}{\beta} \|K_1\| + O({\epsilon_{mach}}^2).\end{aligned} \tag{2.37}$$

Therefore the total computed $\mathbf{w}$ has the form

$$\begin{aligned}\hat{\mathbf{w}} & = \hat{\mathbf{w}}_R + \hat{\mathbf{w}}_N \\ & = [(U^{-1} J L^{-1} G (I - P P^T)(K_1 + K_2)) + E_R]\mathbf{v} \\ & \quad + [(U^{-1} L^{-1} G P P^T (K_1 + K_2)) + E_N]\mathbf{v} \\ & = (M^{-1} K + E)\mathbf{v},\end{aligned} \tag{2.38}$$

where

$$\begin{aligned} \|E\| &= \|E_R + E_N\| \\ &\leq \epsilon_{mach} c_n \frac{\alpha}{\beta} \|K\| + O({\epsilon_{mach}}^2). \end{aligned} \tag{2.39}$$

Finally, since $\beta = \max_i(D_1^{-1}(i,i))$ and the functioning admittances are assumed to be larger than the faulty admittances, β is the maximum of all of the admittances. In particular, $\|D^{-1}\|/\beta = 1$ therefore $\|K\|/\beta = c_m$, where c_m is a constant depending on the number of edges in the graph. Therefore we have

$$\|E\| \leq \epsilon_{mach} c_n c_m \alpha + O({\epsilon_{mach}}^2). \tag{2.40}$$

∎

2.4 Computational experiments

We have tested this algorithm on two qualitatively different graphs: grid graphs and sample power network graphs from the *MATPOWER* [22] power flow simulation package.

In our analysis, we assume that all of the high-impedance (non-functioning) edges are included in the top level of the graph separator. For these examples we defined the high-impedance edges to be exactly those edges cut in the top level graph partition. All remaining edges were taken to be low-impedance (functioning) edges. The low-impedance weights were chosen uniformly at random from a disk in **C** centered at 2 of radius 1. The high-impedance (low admittance) weights were chosen uniformly at random from a disk centered at $2 \cdot 10^{-10}$ of radius $1 \cdot 10^{-10}$. We tried several size problems of this kind, with similar results.

All experiments were conducted in using sparse matrix operations. The extended preconditioner T was LU-factored without pivoting (and, as mentioned earlier, with no fill-in). The graph partitioning was done using the MATLAB Mesh Partitioning Toolbox [21].

Exact solutions were computed by first forming a random solution vector $\mathbf{x}$, then obtaining $\mathbf{b}$ for the problem $K\mathbf{x} = \mathbf{b}$ by multiplying $K\mathbf{x}$.

Figure 2.4 shows the results from a 16,384-node grid graph and compares unpreconditioned BiCG (without splitting) and preconditioned BiCG with splitting. (Note that CG and LSQR are not applicable since (2.1) is not Hermitian. We have also had success with QMR [6] and GMRES not reported here.)

We see from the figure that our preconditioned BiCG with splitting is far superior in terms of reducing the error. For BiCG without preconditioning or splitting, the error was hardly reduced at all from the initial guess.

It is important to note that we are plotting error $||\mathbf{x}^{(j)} - \mathbf{x}||$ rather than residual $||K\mathbf{x}^{(j)} - b||$. The residual for such an ill-conditioned problem would not give a valid picture of convergence.

Figure 2.5 shows the results from a 300-node network example from the *MATPOWER* [22] power flow simulation package. As in the previous example, our preconditioned BiCG with splitting is far superior in terms of reducing the error. For BiCG without preconditioning or splitting, the error was hardly reduced at all from the initial guess.

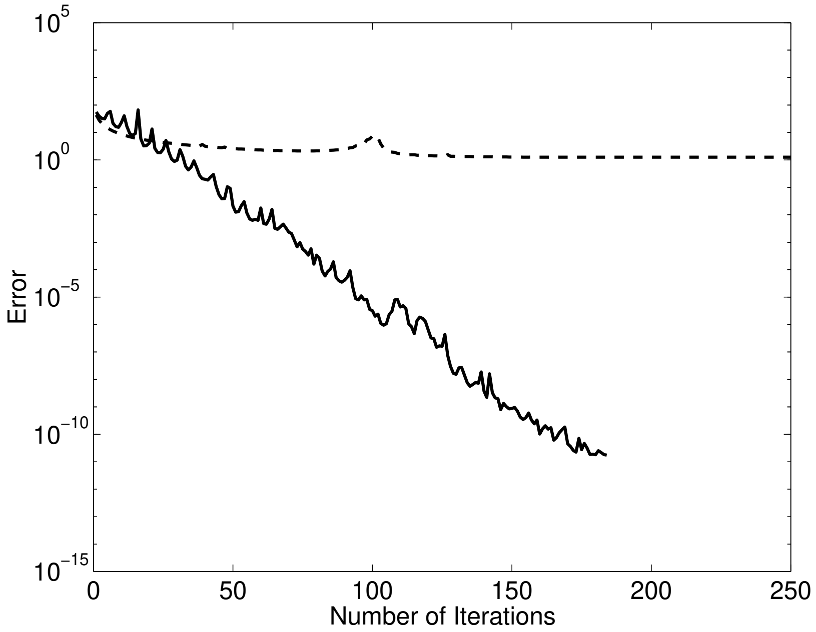

Figure 2.4: Two iterative methods applied to the same 16,384-node AC network problem on a grid graph. The dashed line is unpreconditioned BiCG without splitting; the solid line is our preconditioned BiCG with splitting.

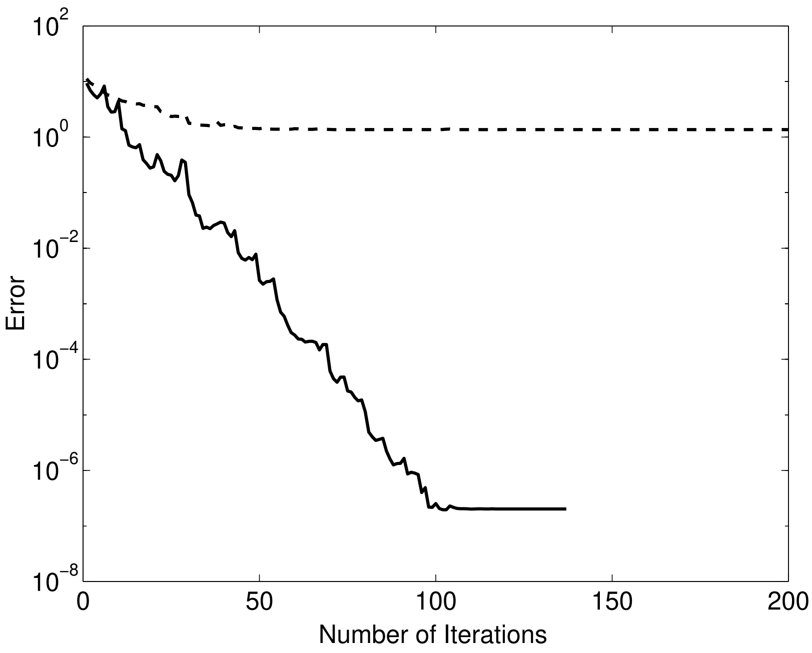

Figure 2.5: Two iterative methods applied to the same 984-node AC network problem on a graph from *MATPOWER* [22]. The dashed line is unpreconditioned BiCG without splitting; the solid line is our preconditioned BiCG with splitting.

Next we compare our algorithm to BiCG with an Incomplete LU (ILU) preconditioner. In Figure 2.6, we compare BiCG with an ILU preconditioner (with no fill) to our algorithm on a 16,384-node AC network with no faults. In this example, our algorithm and BiCG with ILU perform similarly. In Figure 2.7, however, we make the same comparison on a system with faults. Again we see that in a faulted system, our algorithm is far superior in terms of reducing the error.

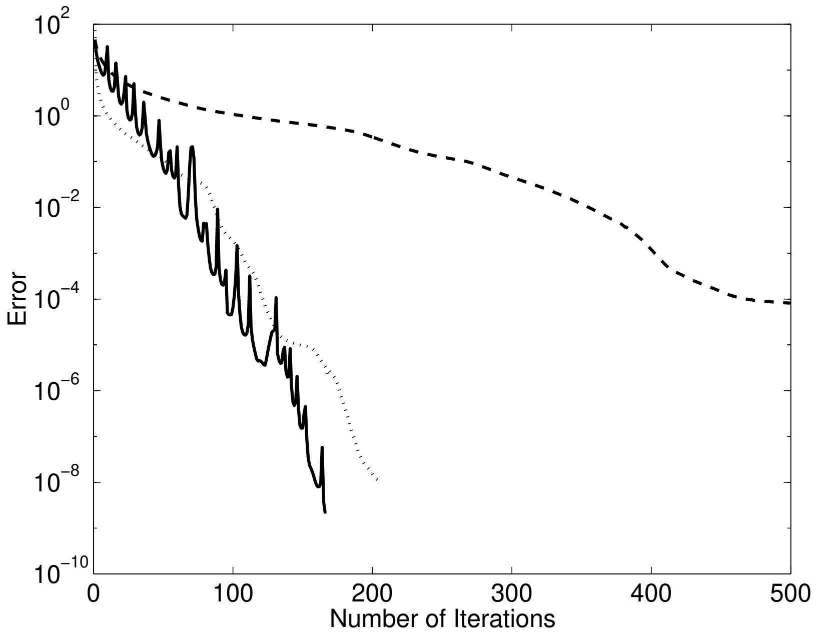

Figure 2.6: Our algorithm, BiCG, and BiCG with ILU preconditioning applied to the same unfaulted 16,384-node AC network problem on a grid graph. The dashed line is unpreconditioned BiCG without splitting; the dotted line is BiCG with an ILU preconditioner (no fill), and the solid line is our preconditioned BiCG with splitting.

The convergence rate of BiCG is not known to be related to the condition number

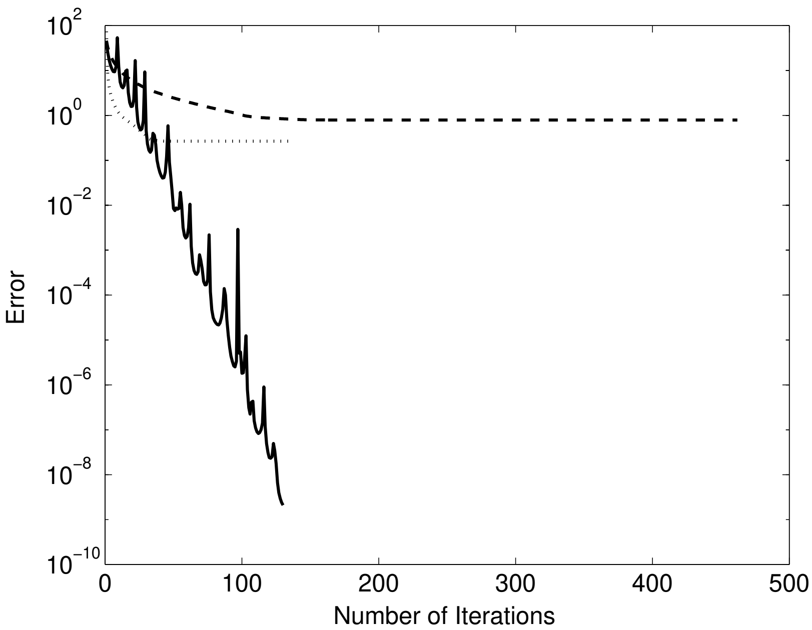

Figure 2.7: Our algorithm, BiCG, and BiCG with ILU preconditioning applied to the same faulted 16,384-node AC network problem on a grid graph. The dashed line is unpreconditioned BiCG without splitting; the dotted line is BiCG with an ILU preconditioner (no fill), and the solid line is our preconditioned BiCG with splitting.

in Definition 2 (or indeed, to any condition number). Nonetheless, our experience indicates that these two are related. Note that the condition number in Definition 2 is related to the condition number of $(M^*M)^{-1}(K^*K)$, which is the condition number that would apply if we were using CGNR.

2.5 Conclusions

We have presented an iterative method for solving the complex-symmetric linear system arising in electrical power networks. We extend Gremban, Miller, and Zagha's [11] support tree preconditioner to handle the case of faulted AC networks, i.e., complex weights and vastly different admittances. In addition to these extensions, we present a splitting technique that allows us to apply the preconditioner accurately even when the system matrix, and therefore the preconditioner, is arbitrarily ill-conditioned. Our computational results show that this iterative method works well in practice in reducing the error.

Note that these results can also apply to the sandstone and shale problem of Vuik [20] and may also apply to interior point methods if we have nullspace information.

Chapter 3

Accurate Integration of the Transient Stability Equations*

3.1 Introduction

We present in this chapter an accurate method for efficiently solving the swing equations. Our method is based in part on the iterative method [15] presented in the previous chapter. The swing equations are used to determine the transient stability of electrical power networks. Transient stability analysis determines whether a system locks back onto a steady state solution after a large disturbance (fault) occurs and is cleared. The method we present in this chapter combines the iterative method of Chapter 2 with standard methods for solving differential algebraic equations (DAE). The iterative method presented in Chapter 2 does not directly apply

*Portions of this chapter are based on "Accurate Integration of the Swing Equations," [14] in preparation.

to transient stability equations, however. A contribution of this thesis is analysis to show that the DAE solver's inner loop can be reduced to a problem amenable to the technique developed in Chapter 2.

The transient stability problem can be stated as follows. Assume that we have an electrical power network, consisting of generators and loads, operating in equilibrium. Consider the following fault scenario. At some point in time, a fault occurs in the system. Relays detect the fault and cause circuit breakers to trip, resulting in a nearly open circuit. The circuit breakers remain open until the fault has cleared, then they reclose, re-establishing the original circuit. The fault sequence leads to a transient in the system. If the transient dies out, and the system locks back onto an equilibrium condition, the system is said to be transient stable [1] for the given fault sequence. Note that this fault scenario is one of many possible scenarios.

We are interested in particular in solving this problem in the catastrophic case of system "islanding." Islanding is when one or more parts of the network become effectively disconnected from the rest of the network. One way in which islanding can occur is from circuit breakers tripping in response to a fault. When a circuit breaker trips it may completely eliminate an edge or only partially eliminate the edge. In this thesis we are concerned with the case where the open circuit breakers partially disconnect part of the network, that is, when the circuit breakers lead to parts of the network being connected only by extremely high impedance edges. In this situation, the linear system that must be solved during a transient stability analysis can be arbitrarily ill-conditioned. In the original network, all parts of the system are connected and there is one reference node (ground) against which all of the other node voltages are measured. During islanding, however, the nearly disconnected components are mostly isolated from the reference node, which is the source of the ill-conditioning.

The behavior of the generators is modeled with the swing equations, which are a system of nonlinear differential algebraic equations (DAE). One can answer the transient stability question by solving the swing equations. We describe the swing equations in the next section.

There are several methods commonly used for transient stability analysis, including numerical integration of the swing equations and direct/Lyapunov methods. Numerical integration methods and direct methods differ in how they determine whether the post-fault system returns to an equilibrium solution. Both classes of methods, however, rely on numerically integrating the swing equations during the fault (while the circuit breakers are open) to get initial conditions for solution of the post-fault system.

Standard methods for numerically integrating DAEs include a Newton-like inner iteration, which requires solving a linear system involving the Jacobian matrix of the DAEs. The Jacobian matrix of the swing equation can be arbitrarily ill-conditioned in the case of islanding, making the solution of the DAE inaccurate. Inaccurate solution of the swing equations during the fault then leads to an inaccurate initial condition for the post-fault system, and therefore to a possibly incorrect determination of transient stability.

The method we propose in this chapter can accurately integrate the swing equations during the fault (i.e., while the circuit breakers are still open), even in the case of islanding. We then have good initial conditions for solving the post-fault system and determining the transient stability of the system for a given fault scenario.

In Section 3.2 we describe the physical system we are considering and the swing equations in more detail. In Section 3.3 we describe our method for solving the swing equations in all three fault stages, but focusing on the second (fault-on) stage. In Section 3.4 we present computational experiments showing that our method achieves higher accuracy than current numerical integration methods for the transient stability problem.

3.2 The Swing Equations

Assume that we are given an AC electrical power network consisting of AC synchronous generators and constant impedance loads. A *synchronous generator* is one that operates at constant speed and constant frequency in steady state. A typical generator has a rotating part called the *rotor* and a stationary part called the *stator*. The rotor, which contains a DC-excited winding (magnet), rotates within the stator winding, leading to the generation of alternating current according to Faraday's law of electromagnetic induction.

Figure 3.1 shows a cut-away schematic of a generator. Most real generators are three-phase generators, which have three sets of windings on the stator. For simplicity, we consider a single-phase generator in this thesis.

The motion of the rotor is influenced by mechanical torque from an external prime mover such as steam, hydro power, or a gas turbine, and by an electromag-

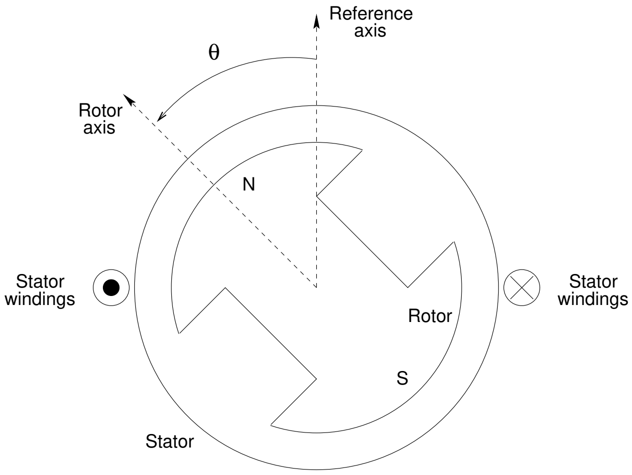

Figure 3.1: Schematic of a single-phase generator. See, for example, Figure E8.1 in [1].

netic torque caused by the interaction between the generated current and the rotor magnet. Note that in engineering, it is conventional to multiply these torques by the constant rotation frequency ω_0 to obtain powers. Therefore, we will refer to the input (mechanical) power of the generator (from the prime mover) and the output (electrical) power provided to the system.

Typically, the rotor rotates at 3600 revolutions per minute to generate a frequency of $\omega_0 = 60\text{Hz}$. In steady state, each generator's rotor angle θ_k increases uniformly with time, i.e., $\theta_k = 2\pi\omega_0 t + \delta_k$, where δ_k is a constant relative offset from some reference angle (for each generator and as measured from a synchronous rotating reference frame).

The state of the generators is determined through the relative offset angles, δ_k for $k = 1, \ldots, g$, where g is the number of generators. The behavior of the δ_k is described by a set of differential algebraic equations (DAE) called the *swing equations*. For each of the g generators in the network, the relative offset angle of the generator δ_k obeys the differential equation

$$\ddot{\delta}_k = -d_k\dot{\delta}_k + P_{in_k} - P_{out_k}(\delta_k, i_k(\boldsymbol{\delta})), \tag{3.1}$$

where d_k is a damping constant, P_{in_k} represents the input power from the prime mover, and P_{out_k} is the power output by generator k. The output power P_{out_k} depends on δ_k and the (equilibrium) current i_k on the generator edge. In particular,

$$P_{out_k} = \mathrm{Re}(v_k e^{i\delta_k} \times \overline{i_k}). \tag{3.2}$$

Note that we are representing sinusoidal quantities such as currents and voltages as phasors. For example, the voltage of a generator on edge k is $v_k \cos(\theta_k) = v_k \cos(2\pi\omega t + \delta_k)$. This quantity is represented by the phasor $v_k e^{i\delta_k}$. The validity of phasor notation assumes a constant frequency $\dot{\theta} = 2\pi\omega_0$, an assumption which does not hold for transient stability analysis since δ is not constant in the presence of a fault. Since, however, the offset angular velocity $\dot{\delta}$ is small with respect to ω_0, the phasor representation is still a good approximation to the actual quantities.

In steady-state equilibrium, $\ddot{\delta}$ is zero, so the input power P_{in_k}, output power $-P_{out_k}$, and damping term $-d_k\dot{\delta}_k$ for each generator cancel. Steady state solutions of (3.1) turn out to be solutions where the offset angles δ_k are constants. However, in the presence of a large disturbance (fault), the δ_k can begin “swinging” away from their steady-state constant values. Note that since we are using phasor representation, we only deal with the case where generators begin to swing out of phase with each other. Our method cannot currently deal with the situation where the frequencies of the generators change.

Consider the following three-stage fault sequence as described in [1].

1. System is in pre-fault steady-state condition with all generators operating synchronously, i.e., δ_k is constant for each generator.

2. Fault occurs at time $t = 0$ and circuit breakers instantaneously open. The δ_k move away from their steady-state constant values.

3. Fault is cleared at time $t = T$ and circuit breakers instantaneously reclose.

The transient stability question is whether the system will again lock onto an equilibrium condition after the fault has cleared. The fault presents a shock to the system and causes the offset angles δ_k of the generators to diverge from their equilibrium values over time, i.e., $\theta_k = 2\pi\omega_0 t + \delta_k(t)$. If after the fault has cleared, the offset angles return to a constant (which is not necessarily the same as the pre-fault constant), the system returns to an equilibrium state and is said to be transient stable for the analyzed fault.

Once we are no longer in the steady-state case, the system currents and voltages are no longer constant and the differential equations for the δ_k must be coupled with the algebraic equations for Ohm's law and current balance:

$$\begin{bmatrix} D & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{i} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{gen} \\ \mathbf{0} \end{bmatrix} \tag{3.3}$$

where A is an $m \times n$ real matrix, D is an $m \times m$ complex diagonal matrix whose diagonal entries have positive real parts, $\mathbf{v}_{gen}$ is a complex m-vector, and $[\mathbf{i}; \mathbf{v}]$ is the $(n + m)$-vector of unknown currents (on each edge) and voltages (at each node).

More specifically, A is the node-arc incidence matrix of the network. A *node-arc incidence matrix* (NAI) of a directed graph has one row for every edge of a graph and one column for every node. In each row, all entries are zeros except for exactly one '1' and one '-1' per row, which correspond to the endpoints of the graph edges.

The diagonal matrix D stores the impedances of the loads. The vector $\mathbf{v}_{gen}$ holds the generator voltages and equals $c_k e^{i\delta_k}$ on generator edges and zero on non-generator edges. We are assuming constant-impedance loads in our network model. There are other load models, including other static load models such as constant-current or constant-power as well as dynamic load models. Constant-impedance loads are considered realistic and are frequently used for short time duration calculations, such as for transient stability analysis.

Power networks are usually built with three identical superposed networks, each 120 degrees out of phase; this is called "three-phase" power. Computer modeling of three-phase networks often takes advantage of the symmetry by simulating only a single phase, thus reducing the size of the problem by a factor of three. Unfortunately, in the presence of faults, the symmetry is broken. Nonetheless, computational techniques have been developed to take advantage of symmetry even in the presence of certain kinds of faults in three-phase networks. In this thesis, we do not assume three-phase power, nor do we use symmetry techniques. In other words, to model a three-phase network using the algorithm of this chapter requires explicit representation of all three phases.

We also assume that part of our network is modeled as an infinite bus. An infinite bus represents a portion of the network large enough that is it not affected by the state of the part of the system we are considering, and we represent it as a constant voltage generator. In particular, on the infinite bus edge, $v_{gen_\infty} = c_\infty e^{i\delta_\infty}$ is a nonzero constant.

In Figure 3.2 we show a very simple example of a network. This network has one generator and an infinite bus. Each edge j of the network has a constant impedance load Z_j. In solving (3.3) for this system, we obtain the currents $\mathbf{i}$ on each edge and the voltages $\mathbf{v}$ at each node.

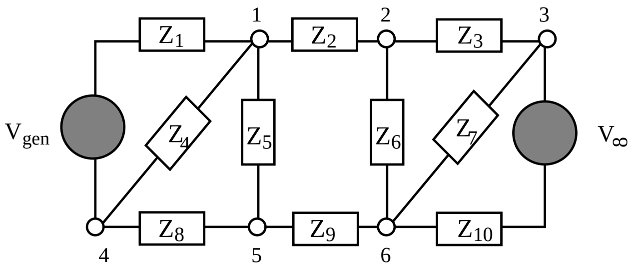

Figure 3.2: A sample network with one generator and another representing an infinite bus. Each edge j has a given impedance Z_j.

In this simple example, the algebraic equations (3.3) have the following components:

$$A = \begin{bmatrix} 1 & & & -1 & & \\ 1 & -1 & & & & \\ & 1 & -1 & & & \\ 1 & & & -1 & & \\ 1 & & & & -1 & \\ & 1 & & & & -1 \\ & & 1 & & & -1 \\ & & & 1 & -1 & \\ & & & & 1 & -1 \\ & & 1 & & & -1 \end{bmatrix}, \tag{3.4}$$

$$D = \mathrm{diag}(\begin{bmatrix} Z_1 & Z_2 & Z_3 & Z_4 & Z_5 & Z_6 & Z_7 & Z_8 & Z_9 & Z_{10} \end{bmatrix}), \tag{3.5}$$

$$V_{gen} = \begin{bmatrix} V_g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & V_\infty. \end{bmatrix}^T \tag{3.6}$$

where again $V_g = c_k e^{i\delta_k}$ and V_∞ is a constant. When the network is faulty, D will be extremely ill-conditioned, possibly making the whole matrix $[D, A; A^T, 0]$ ill-conditioned.

We are particularly interested in the catastrophic case where the fault separates a component of the network from the rest of the network. For example, suppose we had a nearly open-circuit on edges 2 and 9. This fault effectively removes communications between the two halves of the network. This phenomenon is known as islanding and can lead to arbitrarily ill-conditioned systems.

The swing equations (3.1) and (3.3) are a system a DAEs of index one. Given a general DAE,

$$F(t, \mathbf{y}, \mathbf{y}') = 0, \tag{3.7}$$

the *index* of the DAE is the minimum number of times that all or part of (3.7) must be differentiated with respect to t to determine $\mathbf{y}'$ as a continuous function of $\mathbf{y}$ and t [4]. During a fault, the matrix in the algebraic equations (3.3) can be extremely ill-conditioned.

Note that the number of equations equals the number of variables. The variables are:

- On each node: voltage v_i, $i = 1, \ldots, n$,
- On each edge: current i_j, $j = 1, \ldots, m$,
- On each generator edge: δ_k, $k = 1, \ldots, g$.

The currents and voltages are complex quantities; the offset angle is real. Thus we have $m+n$ complex variables and g real variables. Correspondingly, we have g real equations, the differential equations for δ_k (3.1), and $m+n$ complex equations, the system of algebraic equations in (3.3) representing Ohm's law and Kirchhoff's law of current balance.

DAE solvers have difficulty with this mixture of real and complex equations and variables, because it is unclear how to express the Jacobian matrix. Therefore, we rewrote the entire system using real numbers, treating the real and imaginary parts of the complex variables as pairs of real variables. Expressed as a real-only system, the swing equations are

$$\ddot{\delta}_k = -d_k\dot{\delta}_k + P_{in_k} - P_{out_k}(\delta_k, \hat{i}_k(\boldsymbol{\delta})) \tag{3.8}$$

$$\begin{bmatrix} \hat{D} & \hat{A} \\ \hat{A}^T & 0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{i}} \\ \hat{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{v}}_{gen} \\ \mathbf{0} \end{bmatrix}, \tag{3.9}$$

where $\hat{D}$ is a block diagonal matrix of 2×2 blocks. If the jth diagonal element of the complex version of D is $d_j = x_j + iy_j$, then the jth block of $\hat{D}$ is $[x_j, -y_j; y_j, x_j]$.

Similarly, the elements of $\hat{A}$ are 2×2 blocks. In the complex representation, the entries of A were zeros, ones, and negative ones. The entries of $\hat{A}$ are these entries times the 2×2 identity. In other words, $\hat{A}$ is the NAI matrix of the network we get by replacing each original complex edge with two real edges, one representing the real part and one representing the imaginary part of the load impedance. The vectors $\hat{\mathbf{i}}$, $\hat{\mathbf{v}}$, and $\hat{\mathbf{v}}_{gen}$ are similarly real vectors with elements representing the real and imaginary parts of the current on each edge, the voltage at each node, and the generator voltages.

3.3 Our Algorithm

Many commonly used solution methods for index-one DAEs are based on linear multistep methods (LMM) [4]. Our algorithm is based on a standard linear multistep method for solving DAEs. (In particular, we have implemented our method within a version of MATLAB's *ode15s*.)

3.3.1 Linear Multistep DAE Solvers

We consider two related linear multistep methods (LMM) applied to DAEs: a method based on backward differentiation formulas (BDF) and a method based on the numerical differentiation formulas (NDF) developed by Shampine and Reichelt [18].

Given a fully implicit DAE (3.7), the idea behind both methods (and behind all LMM-based DAE solvers) is to replace the derivative $\mathbf{y}'$ in (3.7) by a backward difference. For example, for a first-order BDF method (implicit Euler), the new system is

$$F\left(t, \mathbf{y}_{n+1}, \frac{\mathbf{y}_{n+1} - \mathbf{y}_n}{h_{n+1}}\right) = 0, \tag{3.10}$$

where $h_{n+1} = t_{n+1} - t_n$ [4]. This substitution results in a system of nonlinear equations for the solution $\mathbf{y}_{n+1}$ at the current time, which is typically solved using a variant of Newton's method. This general technique for using numerical ODE methods for solving DAE systems was developed by Gear [8] in 1971.

Both methods under consideration solve the algebraic equation for $\mathbf{y}_{n+1}$ with a simplified Newton iteration. When using BDF of order k, the iteration is started with a predicted value $\mathbf{y}_{n+1}^{(0)}$ that is formed by evaluating the "predictor polynomial" at the next time step t_{n+1}. The predictor polynomial is the degree-k polynomial that interpolates the $\mathbf{y}$ values at the previous $k+1$ time steps [4].

The correction to the current iterate is obtained by solving a linear system involving a scaled iteration matrix of the form

$$M_{iter} = \left(M - \frac{h}{(1-\alpha)\gamma_k} J \right), \tag{3.11}$$

where M is the mass matrix, h is the current step size, α is the NDF coefficient ($\alpha = 0$ for BDF), $\gamma_k = \sum_{j=1}^{k}(1/j)$, and J is an approximation to the Jacobian matrix of the DAE [18].

In such standard DAE solvers as *DASSL* [4] and MATLAB's *ode15s* [18], the linear systems involving the iteration matrix (3.11) are solving using (sparse, if appropriate) LU factorization followed by forward and backward substitution.

3.3.2 Our Modification to the DAE Methods

Our method is based on a modification to the linear solve involving the iteration matrix M_{iter} (3.11).

The use of LU factorization in this linear solve can have several problems. Because of the structure of the iteration matrices (3.11) that arise when applying these LMM-based DAE solvers to the swing equations, the LU factorization can produce significant amounts of fill in the factors. Electrical power networks are large, and the resulting Jacobian matrices (and therefore the iteration matrices) can have dimensions in the tens of thousands. Fill is therefore a significant concern both in computational time and in memory. This problem can be addressed by replacing the LU factorization-based solve with an iterative method such as BiCG-STAB, GMRES, or QMR. The use of iterative methods in this case has been suggested by several authors, including [5], [7], [16], and [17].

These methods, however, do not address the concerns that arise in the presence of faults in the system. In the presence of a fault, M_{iter} may be extremely ill-conditioned, making the solution of the linear system through LU factorization or a standard preconditioned iterative method inaccurate.

The method we adopt is to take advantage at this step of a linear solver developed to solve stably the extremely ill-conditioned matrices that arise in faulty electrical power networks [15]. This method was developed for matrices of the form

$$A^T D^{-1} A\mathbf{v} = A^T D^{-1}\mathbf{b}, \tag{3.12}$$

where again A is a node-arc incidence matrix, D is a diagonal matrix which holds the system impedances (therefore D^{-1} holds the reciprocal impedances, or admittances) and $\mathbf{v}$ and $\mathbf{b}$ are vectors of voltages. In particular, the method described in Chapter 2 is accurate when the system is faulted, even in the extreme case where the faults effectively disconnect a part of the system from the rest of the system ("islanding"). When islanding occurs, the matrix $A^T D^{-1} A$ can be arbitrarily ill-conditioned, but the linear solver we use remains stable and accurate. To apply this linear solver, we first reformulate the iteration matrix M_{iter} into the form expected by the linear solver.

Note that because of the issues we discussed earlier, we have rewritten the system in real-only notation. We similarly updated the solver described in Chapter 2 to solve systems of this form. Note, however, that in this solver we still use a complex preconditioner. We explain in Theorem 3.3.2 why this works. In the rest of this chapter, we are always referring to the real representation of the swing equations and their components.

We can write the swing equations in mass-matrix form as

$$\begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta} \\ \dot{\boldsymbol{\delta}} \\ \hat{\mathbf{i}} \\ \hat{\mathbf{v}} \end{bmatrix}' = \begin{bmatrix} 0 & I & & \\ 0 & -d \cdot I & & \\ & & \hat{D} & \hat{A} \\ & & \hat{A}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta} \\ \dot{\boldsymbol{\delta}} \\ \hat{\mathbf{i}} \\ \hat{\mathbf{v}} \end{bmatrix} \tag{3.13}$$

$$+ \begin{bmatrix} \mathbf{0} \\ \mathbf{P}_{in} - \mathbf{P}_{out}(\boldsymbol{\delta}, \hat{\mathbf{i}}(\boldsymbol{\delta})) \\ -\hat{\mathbf{v}}_{gen} \\ \mathbf{0} \end{bmatrix}, \tag{3.14}$$

where the vectors $\boldsymbol{\delta}$, $\dot{\boldsymbol{\delta}}$, $\hat{\mathbf{i}}$, $\hat{\mathbf{v}}$, $\mathbf{P}_{in}$, $\mathbf{P}_{out}$, and $\hat{\mathbf{v}}_{gen}$, and the matrices $\hat{A}$ and $\hat{D}$ are as described in Section 3.2. We are assuming in (3.13) that the damping constant d is the same for all generators, but it could be a vector of the damping constants for each generator.

Given a system with n edges, m nodes, and g generators, with all of the generator edges numbered first, we can write $\hat{D}$ and $\hat{A}$ as

$$\hat{D} = \begin{bmatrix} \hat{D}_G & 0 \\ 0 & \hat{D}_{NG} \end{bmatrix}, \; \hat{A} = \begin{bmatrix} \hat{A}_G \\ \hat{A}_{NG} \end{bmatrix}, \tag{3.15}$$

where $\hat{D}_G$ is the $2g \times 2g$ block of $\hat{D}$ of impedances on the generator edges, $\hat{D}_{NG}$ is the $2(n-g) \times 2(n-g)$ block of $\hat{D}$ of impedances on non-generator edges. Similarly, $\hat{A}_G$ is the $g \times 2m$ block of $\hat{A}$ corresponding to generator edges, and $\hat{A}_{NG}$ is the $2(n-g) \times 2m$ block of $\hat{A}$ corresponding to non-generator edges.

The Jacobian of the swing equations can then be written as

$$J = \begin{bmatrix} 0 & I & 0 & 0 & 0 \\ M & -d \cdot I & N & 0 & 0 \\ P & 0 & \hat{D}_G & 0 & \hat{A}_G \\ 0 & 0 & 0 & \hat{D}_{NG} & \hat{A}_{NG} \\ 0 & 0 & \hat{A}_G^T & \hat{A}_{NG}^T & 0 \end{bmatrix} \tag{3.16}$$

where I is the $g \times g$ identity; M is the $g \times g$ matrix of derivatives of the generator output powers $P_{out_k}(\delta_k, \hat{\imath}_k(\boldsymbol{\delta}))$ with respect to the offset angles δ_k; d is the damping constant, N is the $g \times 2g$ matrix of derivatives of generator output powers on each generator edge with respect to the currents $\hat{\imath}_k$ on each generator edge; P is the $2g \times g$ matrix of the derivatives of the generator voltages on each generator edge $\hat{v}_{gen_k}$ with respect to the offset angles δ_k, and $\hat{D}_G$, $\hat{D}_{NG}$, $\hat{A}_G$, and $\hat{A}_{NG}$ are as described above.

As explained in more detail in [18], at each step (i) in the simplified Newton iteration, the correction $\Delta^{(i)}$ to the current iterate $\mathbf{y}_{n+1}^{(i)}$ is obtained by solving

$$\begin{bmatrix} I & -\tilde{h}_k I & 0 & 0 & 0 \\ -\tilde{h}_k M & (I - \tilde{h}_k d I) & -\tilde{h}_k N & 0 & 0 \\ -\tilde{h}_k P & 0 & -\tilde{h}_k \hat{D}_G & 0 & -\tilde{h}_k \hat{A}_G \\ 0 & 0 & 0 & -\tilde{h}_k \hat{D}_{NG} & -\tilde{h}_k \hat{A}_{NG} \\ 0 & 0 & -\tilde{h}_k \hat{A}_G^T & -\tilde{h}_k \hat{A}_{NG}^T & 0 \end{bmatrix} \Delta^{(i)} = \mathbf{b}^{(i)}, \tag{3.17}$$

where $\tilde{h}_k = h/(1-\alpha)\gamma_k$. The right hand side $\mathbf{b}$ is given by

$$\mathbf{b}^{(i)} = \tilde{h}_k F(t_{n+1}, \mathbf{y}_{n+1}^{(i)}) - \Psi - (\mathbf{y}_{n+1}^{(i)} - \mathbf{y}_{n+1}^{(0)}), \tag{3.18}$$

where Ψ is a quantity fixed during the computation of $\mathbf{y}_{n+1}$. If we write the right hand side as $\mathbf{b}^{(i)} = [\mathbf{b}_1; \mathbf{b}_2; \mathbf{b}_3; \mathbf{b}_4; \mathbf{b}_5]$, note that $\mathbf{b}_5$ should be identically equal to zero by current balance: $\hat{A}^T\hat{\mathbf{i}} = \mathbf{0}$. In practice, $\mathbf{b}_5$ as calculated by (3.18) is not identically zero. In our method, we set $\mathbf{b}_5$ to zero. In our algorithm, we initialize $\hat{\mathbf{i}}$ so that $\hat{A}^T\hat{\mathbf{i}}$ is zero, and therefore this is preserved by zeroing out the residual of the right hand side of the newton step on each iteration.

Using Gaussian elimination to zero out the $-\tilde{h}_k P$ block of the matrix (that is, the $(3,1)$ block), we have

$$\begin{bmatrix} I & -\tilde{h}_k I & 0 & 0 & 0 \\ -\tilde{h}_k M & (I - \tilde{h}_k dI) & -\tilde{h}_k N & 0 & 0 \\ 0 & 0 & -\tilde{h}_k \hat{D}_G - \tilde{h}_k^2 \Pi_N & 0 & -\tilde{h}_k \hat{A}_G \\ 0 & 0 & 0 & -\tilde{h}_k \hat{D}_{NG} & -\tilde{h}_k \hat{A}_{NG} \\ 0 & 0 & -\tilde{h}_k \hat{A}_G^T & -\tilde{h}_k \hat{A}_{NG}^T & 0 \end{bmatrix} \Delta^{(i)} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \tilde{\mathbf{b}}_3 \\ \mathbf{b}_4 \\ \mathbf{0} \end{bmatrix} \tag{3.19}$$

where

$$\tilde{\mathbf{b}}_3 = \mathbf{b}_3 + \tilde{h}_k \begin{bmatrix} P & 0 \end{bmatrix} \begin{bmatrix} I & -\tilde{h}_k I \\ -\tilde{h}_k M & I + \tilde{h}_k dI \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}, \tag{3.20}$$

and

$$\Pi_N = \begin{bmatrix} P & 0 \end{bmatrix} \begin{bmatrix} I & -\tilde{h}_k I \\ -\tilde{h}_k M & I + \tilde{h}_k dI \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ N \end{bmatrix}. \tag{3.21}$$

We next examine the bottom three equations of (3.19). Dividing through by $-\tilde{h}_k$, they are

$$\begin{bmatrix} \hat{D}_G + \tilde{h}_k \Pi_N & 0 & \hat{A}_G \\ 0 & \hat{D}_{NG} & \hat{A}_{NG} \\ \hat{A}_G^T & \hat{A}_{NG}^T & 0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{i}}_G \\ \hat{\mathbf{i}}_{NG} \\ \hat{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} -\tilde{\mathbf{b}}_3/\tilde{h}_k \\ -\mathbf{b}_4/\tilde{h}_k \\ \mathbf{0} \end{bmatrix}. \tag{3.22}$$

We next rewrite (3.22) in the form of (3.12), which is the form required by the linear solver in Chapter 2. Let

$$C = \begin{bmatrix} \hat{D}_G + \tilde{h}_k \Pi_N & 0 \\ 0 & \hat{D}_{NG} \end{bmatrix} \tag{3.23}$$

be the perturbation to the original $\hat{D}$ matrix.

Multiplying the first two equations of (3.22) by $\hat{A}^T C^{-1}$, (and noting that by the third equation, current balance, $\hat{A}^T \hat{\mathbf{i}} = \mathbf{0}$), we get the following system, which we can solving with our iterative method:

$$\hat{A}^T C^{-1} \hat{A} \hat{\mathbf{v}} = \hat{A}^T C^{-1} \mathbf{b}/\tilde{h}_k, \tag{3.24}$$

where we have recombined the generator and non-generator components of $\hat{A}$ and $\hat{\mathbf{i}}$, and $\mathbf{b} = [\tilde{\mathbf{b}}_3; \mathbf{b}_4]$. The difference now is that the matrix C^{-1} is no longer exactly of the form considered in Chapter 2. We show next that C, and therefore C^{-1}, retains the block diagonal structure required by the linear solver and that the perturbation to $\hat{D}$ is small ($O(\tilde{h}_k^2)$). We will see in Section 3.4 that we still get accurate results using the iterative method developed for solving systems of the form in (3.12).

Claim 1 *The perturbed matrix C* (3.23) *has the same block-diagonal structure as $\hat{D}$.*

PROOF. We prove this claim by considering the structure of the matrices added to $\hat{D}$. We can see from (3.22) that the only change to $\hat{D}$ is the perturbation to the block $\hat{D}_G$ associated with the generator edges. Splitting C into its generator edge and non-generator edge components, the perturbation to $\hat{D}_G$ is given by

$$\mathbf{C}_G = \hat{D}_G + \tilde{h}_k \begin{bmatrix} P & 0 \end{bmatrix} \begin{bmatrix} I & -\tilde{h}_k I \\ -\tilde{h}_k M & I + \tilde{h}_k dI \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ N \end{bmatrix}. \tag{3.25}$$

We next describe the structure of the matrices P, M, and N.

The matrix M contains the derivatives of the (real) output power at each generator with respect to the offset angle δ at that generator. Therefore, M is a diagonal matrix of dimension $g \times g$.

The matrix N contains the derivatives of the (real) generator output powers with respect to the real and imaginary components of the currents. The matrix N has dimension $g \times 2g$ and the following structure:

$$N = \begin{bmatrix} \mathrm{x} & \mathrm{x} & & & & \\ & & \mathrm{x} & \mathrm{x} & & \\ & & & & \ddots & \\ & & & & & \mathrm{x} \quad \mathrm{x} \end{bmatrix}. \tag{3.26}$$

The matrix P contains the derivatives of the real and imaginary components of the generator voltages with respect to the offset angles δ_k on the generator edges.

The matrix P has dimension $2g \times g$ and the following structure:

$$P = \begin{bmatrix} \mathrm{x} & & & \\ \mathrm{x} & & & \\ & \mathrm{x} & & \\ & \mathrm{x} & & \\ & & \ddots & \\ & & & \mathrm{x} \\ & & & \mathrm{x} \end{bmatrix}. \tag{3.27}$$

By considering the nonzero structure of these matrices, we can see that the perturbation to $\hat{D}_G$ has the form

$$\begin{bmatrix} P & 0 \end{bmatrix} \begin{bmatrix} I & -\tilde{h}_k I \\ -\tilde{h}_k M & I + \tilde{h}_k dI \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ N \end{bmatrix} = \begin{bmatrix} \mathrm{x} & \mathrm{x} & & & & & \\ \mathrm{x} & \mathrm{x} & & & & & \\ & & \mathrm{x} & \mathrm{x} & & & \\ & & \mathrm{x} & \mathrm{x} & & & \\ & & & & \ddots & & \\ & & & & & \mathrm{x} & \mathrm{x} \\ & & & & & \mathrm{x} & \mathrm{x} \end{bmatrix}, \tag{3.28}$$

which is the same 2×2 block diagonal structure as in $\hat{D}_G$. Therefore, C has the same block diagonal structure as $\hat{D}$, and since block diagonal structure is preserved under inverses, C^{-1} has the same block structure as $\hat{D}^{-1}$. ∎

Note, however, that the 2×2 blocks of C_G can no longer be viewed as real representations of complex numbers since they do not have the form $[x, -y; y, x]$. We show in the next claim that the perturbation away from this form is small.

Claim 2 *The perturbation to $\hat{D}_G$ from performing Gaussian elimination on* (3.17) *to zero out the $-\tilde{h}_k P$ block is $O(\tilde{h}_k^2)$.*

PROOF. Let $\tilde{M} = [0, I; M, -dI]$. Then we can write

$$\begin{bmatrix} I & -\tilde{h}_k I \\ -\tilde{h}_k M & I + \tilde{h}_k dI \end{bmatrix} = I - \tilde{h}_k \tilde{M}. \tag{3.29}$$

Since $\tilde{M}$ does not depend on $\tilde{h}_k$, $||\tilde{h}_k \tilde{M}|| < 1$ for sufficiently small $\tilde{h}_k$. Thus,

$$(I - \tilde{h}_k \tilde{M})^{-1} \leqslant I + \tilde{h}_k \tilde{M} + \tilde{h}_k^2 \tilde{M}^2 + \cdots . \tag{3.30}$$

Therefore, the perturbation to $\hat{D}_G$ can be bounded as follows:

$$\begin{aligned} C_G &= \hat{D}_G + \tilde{h}_k \tilde{P}(I - \tilde{h}_k \tilde{M})^{-1} \tilde{N} \\ &\leqslant \hat{D}_G + \tilde{h}_k \tilde{P}(I + \tilde{h}_k \tilde{M} + \tilde{h}_k^2 \tilde{M}^2 + \cdots) \tilde{N} \\ &= \hat{D}_G + \tilde{h}_k \tilde{P} \tilde{N} + \tilde{h}_k^2 \tilde{P} \tilde{M} \tilde{N} + O(\tilde{h}_k^3), \end{aligned}$$

where $\tilde{P} = [P, 0]$ and $\tilde{N} = [0; N]$. Because of their structure, we can see that $\tilde{P}\tilde{N} = 0$. We therefore have the following bound on the perturbation to $\hat{D}_G$:

$$C_G = \hat{D}_G + \tilde{h}_k^2 \tilde{P} \tilde{M} \tilde{N} + O(\tilde{h}_k^3). \tag{3.31}$$

∎

In the case of functioning edges or nearly open circuits (high impedance faults) this perturbation is small enough that our iterative method (whose preconditioner assumes an unperturbed $\hat{D}$ matrix) still performs well.

Since the perturbations to $\hat{D}$ are small, and the submatrices $\hat{D}$ and $\hat{A}$ themselves in the Jacobian matrix are constant, we build our preconditioner once assuming complex impedances.

We now try to provide some justification for why our complex-valued preconditioner still works for the real-valued network matrix equation (3.24). First, we introduce some notation. We let $S : \mathbf{R}^{2k} \to \mathbf{C}^k$ be the linear isometry that maps k pairs of real numbers, each pair representing a real and imaginary part, to the corresponding k complex numbers. Note that S is parametrized by k, but we omit the explicit subscript.

For example, suppose D is an $m \times m$ complex diagonal matrix. Then $S^{-1}DS$ is a $2m \times 2m$ real matrix with 2×2 diagonal blocks of the form $[a, b; -b, a]$. The ith diagonal block contains the real and imaginary parts a, b of the ith diagonal entry of D.

The main result of this section is the following theorem. This theorem says that $A^T D^{-1} A$ is a good approximation to $A^T S C^{-1} S^{-1} A$, where C is a close approximation (in a certain precise sense) to the $2m \times 2m$ real matrix $S^{-1}DS$ described in the previous paragraph. Thus, the theorem implies that any preconditioner that works well for $A^T D^{-1} A$ should also work well for the system actually arising in our DAE solver, namely $A^T S C^{-1} S^{-1} A$.

In our algorithm, the node-arc incidence matrix A has rank-deficiency of 1 because we do not explicitly assign a grounded node. Refer to (3.4) in the previous example. In the following theorem, we call upon a result of [3] that applies only to matrices A of full rank. A node-arc matrix for a connected graph can be reduced by deleting the node corresponding to the ground node, leaving a matrix of full rank. In order to be consistent with [3], we assume just for this theorem that our matrix A has full rank, i.e., a node is grounded and deleted. The alternative is to include a projection onto the nullspace in the statement and proof of this theorem, which would needlessly clutter notation.

Theorem 6 *Let $A \in \mathbf{R}^{m\times n}$ be a reduced node-arc incidence matrix and let $D \in \mathbf{C}^{m\times m}$ be a diagonal matrix satisfying the cone condition with cone constant μ. Let $\hat{D} \in \mathbf{R}^{2m\times 2m}$ be defined by $\hat{D} = S^{-1}DS$. Let C be a perturbation to $\hat{D}$ with the following properties. We require that C also be block diagonal. Furthermore, if $[a, b; -b, a]$ is the ith block diagonal entry of $\hat{D}$, then the ith diagonal block of C must have the form $[a + \eta_1, b + \eta_2; -b + \eta_3, a + \eta_4]$ where $|\eta_1|, \ldots, |\eta_4| \leq \delta a$. Here, $\delta > 0$ is chosen sufficiently small so that $(1 + \mu)\delta \leq 1/4$. Then*

$$\|(A^T D^{-1} A)^{-1} A^T S C^{-1} S^{-1} A - I\|_\infty \leq 8(1 + \mu)^3 C_{G,\mu}\delta,$$

where $C_{G,\mu}$ is a constant depending on the underlying graph and on μ, but not on D.

Proof. Let $\mathbf{x} \in \mathbf{C}^n$ be an arbitrary vector with norm 1. Then to prove the theorem requires an upper bound on.

$$((A^T D^{-1} A)^{-1} A^T S C^{-1} S^{-1} A - I)\mathbf{x} = (A^T D^{-1} A)^{-1} A^T S C^{-1} S^{-1} A\mathbf{x} - \mathbf{x}. \quad (3.32)$$

Let $\mathbf{y}$ be the solution to the equation

$$A^T D^{-1} A\mathbf{y} = A^T S C^{-1} S^{-1} A\mathbf{x}. \quad (3.33)$$

Then the right-hand side of (3.32), i.e., the expression we must bound, is $\mathbf{y} - \mathbf{x}$.

Define perturbation matrix $F = \hat{D}(C^{-1} - \hat{D}^{-1})$ This allows us to write C^{-1} as $\hat{D}^{-1}(I + F)$. Let us now analyze F defined in this way. Clearly F is also block diagonal with 2×2 diagonal blocks. Focus on its ith diagonal block, which we denote $F_i = \hat{D}_i(C_i^{-1} - \hat{D}_i^{-1})$. Let $\hat{D}_i$ be written $[a, b; -b, a]$. Then, using the matrix infinity-norm and a standard inequality (Theorem 2.3.4 of [9]) for bounding $\|X^{-1} - Y^{-1}\|$ we obtain

$$\|F_i\| \leq \frac{\|D_i\| \cdot \|D_i^{-1}\|^2 \cdot \|C_i - \hat{D}_i\|}{1 - \|C_i - \hat{D}_i\| \cdot \|D_i^{-1}\|}.$$

By assumption that $|b| \leq \mu a$ we have $\|D_i\| \leq (1+\mu)a$. Similarly, $\|D_i^{-1}\| \leq (1+\mu)/a$. Also, by assumption about the η_i's, $\|C_i - \hat{D}_i\| \leq 2\delta a$. Thus, the numerator in the previous expression is bounded above by $2(1+\mu)^3\delta$. The term $\|C_i - \hat{D}_i\| \cdot \|D_i^{-1}\|$ in the denominator is bounded by $2\delta(1+\mu)$, which is at most $1/2$ by assumption. Thus, the denominator is at least $1/2$, hence

$$\|F_i\| \leq 4(1+\mu)^3\delta.$$

Since this applies to each block, $\|F\| \leq 4(1+\mu)^3\delta$.

Substituting the new formula for C into (3.33) yields

$$A^T S\hat{D}^{-1}S^{-1}A\mathbf{y} = A^T S\hat{D}^{-1}(I + F)S^{-1}A\mathbf{x}.$$

Subtract $A^T S\hat{D}S^{-1}A\mathbf{x}$ from both sides to obtain

$$A^T S\hat{D}^{-1}S^{-1}A(\mathbf{y} - \mathbf{x}) = A^T S\hat{D}^{-1}FS^{-1}A\mathbf{x}$$

i.e.

$$A^T D^{-1}A(\mathbf{y} - \mathbf{x}) = A^T D^{-1}SFS^{-1}A\mathbf{x}.$$

Note that we used the fact that $S\hat{D}^{-1} = D^{-1}S$. Thus,

$$\mathbf{y} - \mathbf{x} = (A^T D^{-1} A)^{-1} A^T D^{-1} S F S^{-1} A\mathbf{x}. \tag{3.34}$$

A theorem proved by Bobrovnikova and Vavasis [3] shows that $\|(A^T D^{-1} A)^{-1} A^T D^{-1}\|$ is bounded by $C_{G,\mu}$ independent of D, where $C_{G,\mu}$ is a constant that depends on G, the graph underlying the matrix A, and on μ, the cone parameter. Thus,

$$\|\mathbf{y} - \mathbf{x}\| \leq C_{G,\mu} \|F\| \cdot \|A\|.$$

To derive this last inequality from (3.34), we used the fact that S and S^{-1} are isometries, and that $\|\mathbf{x}\| \leq 1$. To conclude the proof, observe that $\|A\| \leq 2$, and $\|F\| \leq 4(1+\mu)^3\delta$ as derived above. ∎

Finally, having solved for $\hat{\mathbf{v}}$ as described above, we determine $\hat{\mathbf{i}}$ and $[\Delta_1^{(i)}; \Delta_2^{(i)}]$ as follows:

$$C\hat{\mathbf{i}} = \begin{bmatrix} -\tilde{\mathbf{b}}_3/\tilde{h}_k \\ -\mathbf{b}_4/\tilde{h}_k \end{bmatrix} - \hat{A}\hat{\mathbf{v}} \tag{3.35}$$

$$(I - \tilde{h}_k \tilde{M}) \begin{bmatrix} \Delta_1^{(i)} \\ \Delta_2^{(i)} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} + \tilde{h}_k \begin{bmatrix} \mathbf{0} \\ \tilde{N}\hat{\mathbf{i}}_{NG} \end{bmatrix}. \tag{3.36}$$

3.4 Computational experiments

We have tested this algorithm on two qualitatively different graphs: grid graphs and sample power network graphs from the *MATPOWER* [22] power flow simulation package.

In our analysis, we assume that all of the high-impedance (faulted) edges are included in the top level of the graph separator. For these examples we defined the high-impedance edges to be exactly those edges cut in the top level graph partition. All remaining edges were taken to be low-impedance (functioning) edges. The low-impedance weights were chosen uniformly at random from a disk in $\mathbf{C}$ centered at 2 of radius 1. The high-impedance (low admittance) weights were chosen uniformly at random from a disk centered at $2 \cdot H_{eps}$ of radius $1 \cdot H_{eps}$, where H_{eps} is the magnitude of the fault (admittance). We tried several size problems of this kind, with similar results.

All experiments were conducted in MATLAB using sparse matrix operations. The extended preconditioner T was LU-factored without pivoting (and with no fill-in). The graph partitioning was done using the MATLAB Mesh Partitioning Toolbox [21]. The DAE solver we compare with, and within which we implemented our method, is MATLAB's *ode15s*.

Figure 3.3 shows δ and $\dot{\delta}$ results from a 256-node grid graph with one generator and one infinite bus. Since we do not have an exact solution for comparison, we compare the difference in the results obtained from applying our method within a BDF-based DAE solver versus an NDF-based solver with the difference in results from MATLAB's *ode15s* using BDF and NDF. The figure displays our differences versus MATLAB's differences as we increase the size of the fault. We can see that our method's BDF versus NDF results stay consistent as the fault increases, while MATLAB's results do not.

In addition, the fault of magnitude 10^8 was the largest magnitude fault MAT-

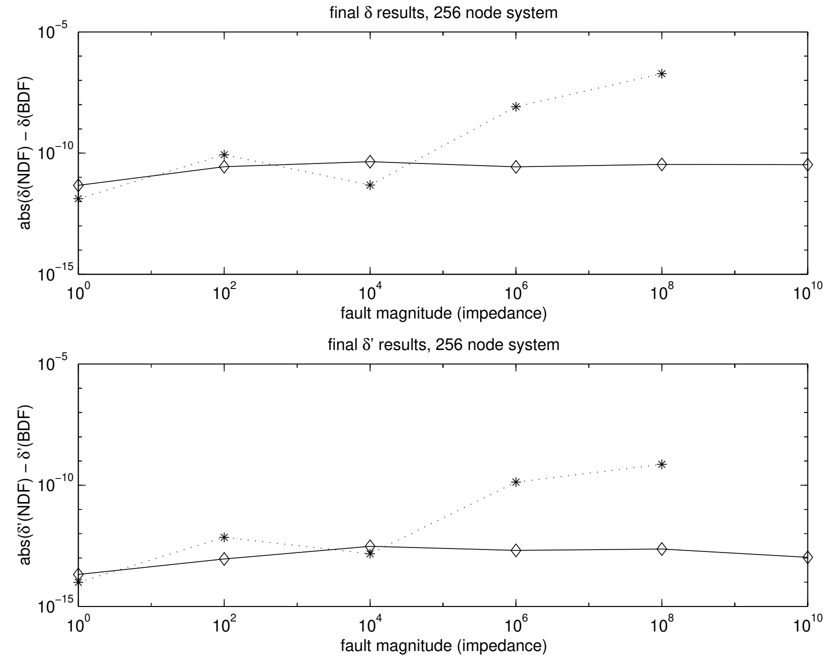

Figure 3.3: Results for 256-node grid-graph system for given tolerances, increasing the fault magnitude (impedance). The top subfigure shows differences in δ from MATLAB's *ode15s* (dotted line) and from our algorithm (solid line). The bottom subfigure shows differences in $\dot{\delta}$ from MATLAB's *ode15s* (dotted line) and from our algorithm (solid line).

LAB's *ode15s* would solve for this system. Our method continues to work for arbitrarily large magnitude faults.

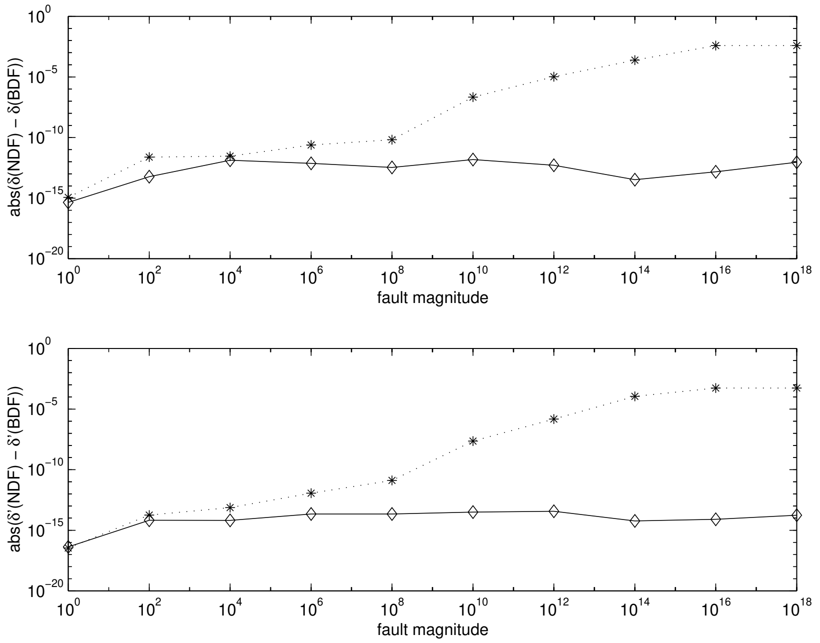

Figure 3.4: Results for 57-node AC network problem on a graph from MATPOWER [22] for given tolerance, increasing the fault magnitude (impedance). The top subfigure shows differences in δ from MATLAB's *ode15s* (dotted line) and from our algorithm (solid line). The bottom subfigure shows differences in $\dot{\delta}$ from MATLAB's *ode15s* (dotted line) and from our algorithm (solid line).

Figure 3.4 show results similar to those in Figure 3.3, but on a 57-node network example from the *MATPOWER* [22] power flow simulation package. As in the previous example, our method's BDF versus NDF results stay consistent as the fault increases, while MATLAB's results do not.

Figure 3.5 show δ and $\dot{\delta}$ results from a 16-node grid graph with one generator and

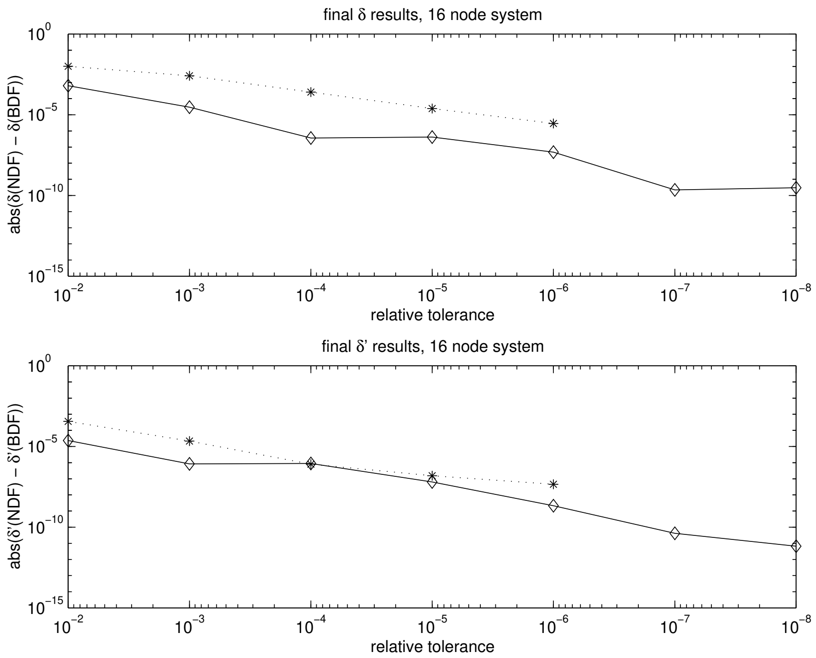

Figure 3.5: Results for 16 node grid-graph system with a fault of size 10^{10} (impedance) and tightening the tolerances. The top subfigure shows δ from MATLAB's *ode15s* (dotted line) and from our algorithm (solid line). The bottom subfigure shows $\dot{\delta}$ from MATLAB's *ode15s* (dotted line) and from our algorithm (solid line). MATLAB's *ode15s* would not solve the system for tolerances tighter than those displayed.

one infinite bus. In this case, we compare the difference between our method within a BDF-based DAE solver and an NDF-based solvers with the difference in results from MATLAB's *ode15s* using BDF and NDF when we hold the fault magnitude constant and tighten the tolerance of the DAE solver. For all of the results using our method, we requested a relative tolerance of 10^{-14} and an absolute tolerance of 10^{-12}. For the MATLAB results, we requested tolerances as tight as possible for the method to solve the problem. For tighter tolerances, MATLAB's *ode15s* simply made no progress; it just continued to decrease the step size without ever progressing forward.

Figure 3.6 shows δ and $\dot{\delta}$ results from a 256-node grid graph with one generator and one infinite bus. Here we compare the results obtained from our method (using the NDF-based DAE solver) and from using MATLAB's solver on the individual islands in the systems. We can see that for small to medium faults, solving the islands separately is very inaccurate. For larger faults, solving the islands separately gives solutions similar to our method.

3.5 Conclusions

We have presented an accurate method for solving the swing equations. The method proposed in this chapter can accurately integrate the swing equations during a fault (i.e., while the circuit breakers are still open), even in the case of islanding. Our method combines the new method of Chapter 2 for stably solving the extremely ill-conditioned linear systems arising in faulty AC networks with standard methods

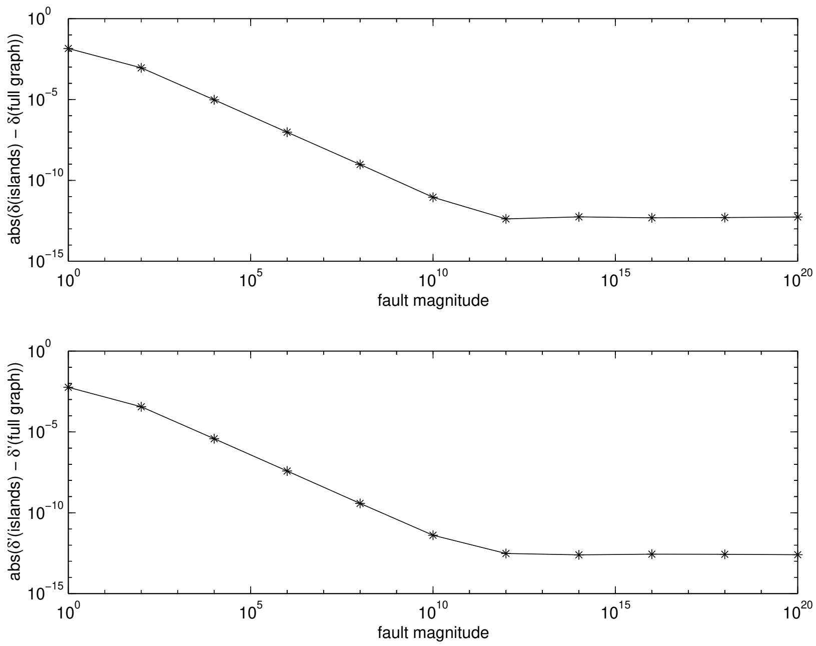

Figure 3.6: Results for 256-node grid-graph system for given tolerances, increasing the fault magnitude (impedance). The top subfigure shows the difference in the final δ computed using our algorithm and by solving the islands separately. The bottom subfigure shows similar differences for $\dot{\delta}$.

for solving differential algebraic equations (DAE). In addition, since the method of Chapter 2 does not directly apply to transient stability equations, we have provided analysis to show that the DAE solver's inner loop can be reduced to a problem amenable to the technique developed in Chapter 2.

We have shown that our method is more accurate than MATLAB's *ode15s* for large faults. In addition, for small to medium faults, it is more accurate than solving the islands separately. Therefore, our method constitutes a good general method applicable to any size fault.

In addition, since we are using a partial fault model, there is no issue of how to reconnect the islands after the fault has cleared. This is a difficulty in solving the islands separately, since do to so we must choose a new reference node for each island.

Our algorithm is more accurate than MATLAB's ode15s even on modest size faults (e.g., impedance magnitude 10^6), and maintains its accuracy for arbitrarily large magnitude faults.

REFERENCES

[1] A. R. Bergen, *Power Systems Analysis*, Prentice Hall, Englewood Cliffs, N.J., 1986.

[2] M. Bern, J. R. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo, *Support-graph preconditioners*, (preprint).

[3] E. Y. Bobrovnikova and S. A. Vavasis, *Accurate solution of weighted least squares by iterative methods*, Tech. Rep. ANL/MCS-P644-0297, Mathematics and Computer Science Division, Argonne National Laboratory, 1997.

[4] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, Philadelphia, 1996.

[5] A. J. Flueck and H.-D. Chiang, *Solving the nonlinear power flow equations with an inexact newton method using GMRES*, IEEE Transactions on Power Systems, 13 (1998).

[6] R. W. Freund and N. M. Nachtigal, *QMR: A quasi-minimal residual algorithm for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[7] F. D. Galiana, H. Javidi, and S. McFee, *On the application of a preconditioned conjugate gradient algorithm to power network analysis*, IEEE Transactions on Power Systems, 9 (1994).

[8] C. W. Gear, *The simultaneous numerical solution of differential-algebraic equations*, IEEE Trans. Ciruit Theory, CT–18 (1971), pp. 89–95.

[9] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 3rd ed., 1996.

[10] K. D. Gremban, *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems*, PhD thesis, Carnegie Mellon University, Oct. 1996.

[11] K. D. Gremban, G. L. Miller, and M. Zagha, *Performance evaluation of a new parallel preconditioner*, Tech. Rep. CMU-CS-94-205, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 1994.

[12] R. Guo and R. D. Skeel, *An algebraic hierarchical basis preconditioner*, Applied Numerical Mathematics, 9 (1992), pp. 21–32.

[13] P. D. Hough and S. A. Vavasis, *Complete orthogonal decomposition for weighted least squares*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 369–392.

[14] V. E. Howle and S. A. Vavasis, *Accurate integration of the swing equations*, in preparation.

[15] ——, *An iterative method for solving complex-symmetric systems arising in electrical power modeling*, submitted to SIAM J. Matrix Anal. Appl., (2000).

[16] M. A. Pai, P. W. Sauer, and A. Y. Kulkarni, *A preconditioned iterative solver for dynamic simulation of power systems*, IEEE, (1995).

[17] A. Semlyen, *Fundamental concepts of a Krylov subspace power flow methodology*, IEEE Transactions on Power Systems, 11 (1996).

[18] L. F. Shampine and M. W. Reichelt, *The Matlab ODE suite*, SIAM J. Sci. Comput., 18 (1997), pp. 1–22.

[19] P. M. Vaidya, *Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners*, Unpublished manuscript. A talk based on the manuscript was presented at the IMA Workshop on Graph Theory and Sparse Matrix Computation, (1991).

[20] C. Vuik, A. Segal, and J. A. Meijerink, *An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients*, J. Comp. Phys., 152 (1999), pp. 385–403.

[21] Xerox Corporation, *Matlab mesh partitioning toolbox*.

[22] R. Zimmerman and D. Gan, *MATPOWER*. http://www.pserc.cornell.edu/powerweb/.