AN ABSTRACT OF THE THESIS OF

Joel W. Kolstad for the degree of Master of Science in Electrical and Computer Engineering presented on June 13, 2007.

Title:

CAM: A New Circuit Augmentation Method for Modeling Interconnects and Passive Components

Abstract approved: ______________________________

Andreas Weisshaar

Contemporary circuit design involves significant computer-based simulation, calling for a balance between circuit model accuracy and simulation run time. Traditionally, circuit modelers concentrated on producing either (1) physically-based models, where each element in the model correlates to a physically meaningful aspect of the device being modeled or (2) mathematically-based "macromodels," designed to be efficient to simulate without ties to the device's physical underpinnings. While each approach has significant value, the strengths of one are often the weaknesses of the other.

This work presents CAM, the Circuit Augmentation Method, that integrates some of the best aspects of each approach. Based on the assumption that high-accuracy, multi-port simulation or measurement data for the device under consideration is available, the methodology augments an existing, user-provided equivalent circuit model that is reasonably accurate over some frequency band with discrepancies elsewhere. Macromodels using rational functions are derived via standard least-squares or vector fittings approaches and are logically "added" to the user model creating a result that is potentially both fast to simulate yet still physically meaningful. An important aspect of CAM is its perturbational nature: It is shown that perturbation of the user model's component values (or similar attributes) is necessary for optimal results.

CAM is a general-purpose technique that is applicable to the often difficult problems of modeling circuits requiring wideband accuracy or those incorporating highly-distributed structures. This utility is demonstrated over several two-port examples, including a broadband oscilloscope probe tip, a spiral inductor on a lossy silicon substrate, and a highly-distributed and lossy test circuit.

CAM: A New Circuit Augmentation Method for Modeling Interconnects and Passive Components

by

Joel W. Kolstad

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 13, 2007
Commencement June 2008

Master of Science thesis of Joel W. Kolstad presented on June 13, 2007.

APPROVED:

Major Professor, representing Electrical and Computer Engineering

---

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

Joel W. Kolstad, Author

ACKNOWLEDGEMENTS

My fiancé Erin deserves more thanks I can possibly give. She gave me the countless nights needed to work on my research and write this thesis, providing support and encouragement when I needed it most; thank you so much for being the perfect woman for me.

My family members – many of them having engineering backgrounds – have always been encouraging. To my grandfather Harry Bischel and family friend Peter Weiler, thank you for demonstrating the joys of electrical engineering long before I had ever heard of Maxwell or Ohm.

The faculty and staff of the Oregon State University electrical engineering department all deserve thanks for providing me with a solid education. My major professor, Andreas Weisshaar and his former departmental co-professor, Raghu Kumar Settaluri are both excellent teachers who gave many an outstanding lecture – thank you. I would also like to thank Karti Mayaram, Gabor Temes, Luca Lucchese, and the late Alan Wallace for presenting excellent courses and engaging in stimulating discussions.

Thank you to Dr. Karti Mayaram, Dr. Thomas Plant, and my GCR Dr. Kagan Tumer for taking the time to review this manuscript, provide suggestions, and for sitting on my graduate committee.

I am grateful to my fellow graduate students for helping with this research and preparing papers and presentations. Christopher Blevins has contributed greatly, and some of the early CAM research was based on the work of Daniel Melendy and Adam Watson. Eric Vernon has also contributed to CAM's code base.

Most of the data files used during CAM's development were provided by John Dunn at Tektronix; a few were provided by National Semiconductor. These companies' financial support of CDADIC are what made this research possible; their generosity is greatly appreciated.

A few final thanks to people who additionally inspired me to pursue engineering so many years ago: My second-grade school bus driver, who taught me that a battery, a light bulb, and some Reynolds Wrap make something akin to a flashlight, and my third-grade teacher who helped me build a crystal radio. While they will surely never read these words, the excitement and sense of discovery they instilled has lasted a lifetime.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF APPENDICES

LIST OF APPENDIX FIGURES

LIST OF APPENDIX TABLES

Chapter 1 – Introduction

1.1 Background and Motivation

For more than four decades there has been a steady progression in the techniques used to design integrated high-speed electronic circuits: What at one time required the manipulation of slide rules, log tables, countless pages of hand calculations and many lab prototypes has changed to be performed on desktop computers using sophisticated circuit simulation software with an expectation of "first pass" success in silicon. Of course, the definition of "high-speed" has changed markedly as well: Over forty years ago in the early 1960s, Tektronix in Beaverton, Oregon produced the first 1GHz bandwidth oscilloscope – the model 519 – spurred on by the U. S. government's desire for high-speed measurements in nuclear detonation experiments. At the time, no circuit simulation software was available – SPICE, the first "killer application" for circuit simulation, wouldn't be readily available for over a decade! The next few decades saw rapid development in automated circuit synthesis techniques, with a realization early-on that accurate component models were needed to produce better and faster designs.

Results were impressive: By the late-1980s 20GHz sampling oscilloscopes made the model 519 looks as obsolete as a Ford model T automobile. The turn of the century saw CPU clock speeds routinely exceeding 1GHz while utilizing over 10 million transistors, and by 2006 18GHz real-time oscilloscopes could be purchased and the fastest sampling heads available exceeded 70GHz bandwidth. Such cutting-edge designs are extraordinarily complex, requiring very accurate models and simulations to complete.

Such impressive technological advances have made for a tighter, more-connected, ever-shrinking world that grows more complex every day. Similarly, in high-speed design ever-increasing clock frequencies and analog bandwidths have made even the simplest of passive component's or interconnect's behavior more complex, while parasitic reactances and losses create significant frequency dependences. Physical structures can no longer be considered entirely "lumped" (electrically small) but are instead best modeled by some uncertain mixture of lumped and distributed effects. Even seemingly trivial items such as vias and bond wires now must use sophisticated models if multi-GHz simulations are to produce accurate results. Circuit modeling has come to the forefront of high-speed design in the 2000s!

Historically, circuit models were created from mathematically ideal circuit components such as resistors, capacitors, inductors, and transmission lines by an expert modeler who would compare the "equivalent circuit model" (**ECM**) to direct measurement data or – more recently – that obtained from a high-precision field solver simulations; such tabulated data are typically provided in the form of frequency-dependent S-, Z- or similar network parameters. This method has several advantages: Firstly, the model can be directly inserted into a SPICE-type simulator, which is ubiquitous in mixed-signal design. Furthermore, the model is typically efficient to simulate, as the modeler will only include elements necessary for a design's intended frequency of operation and accuracy. Finally, when properly constructed, the model is "physically based," in that components in the model correspond to physically identifiable entities in the actual component, such as bond wire partial inductance, package capacitance, etc.; this provides the designer with valuable insight into the object's behavior, and can suggest possible changes in the design to meet specified performance requirements. The primary drawback of this approach is that the time required to develop a high-quality, physically-based model can be substantial, and that expert modelers are a rare commodity.

Developing a fully-automated means of creating mathematical circuit models has been studied for decades with modest success. These approaches are commonly known by a variety of names, including "macromodeling," "black-box" modeling, "behavioral" modeling, and "reduced-order" modeling. As the names suggest, no effort is made to model the underlying physics of the object being modeled, but rather the goal is to create a mathematically accurate and computationally efficient model that will provide accurate responses within a circuit simulator. As such, there is no need to use idealized circuit elements in the model: any mathematical construct supported by the simulator can be used (and many have been tried!). In fact, the most straightforward approach for including frequency-domain data in a time-domain simulator is to first compute the inverse Fourier[1]-transform of the data and then apply convolution at each time step to obtain the desired output [1,2]. Unfortunately, this is computationally expensive as obtaining the output at each time step t requires computing the convolution sum "all the way" back to $t = 0$[2]. Instead then, a more common approach is to fit the data to a set of basis functions with a known, easily-computed time-domain response [3, 4]. Rational functions are commonly used due to their simple time-domain response when the function is expressed as a partial-fraction expansion (i.e., a pole-residue formulation).

The precise choice of basis functions and how their coefficients are fitted to the data is a wide field of study with no single "best" solution; proper application of system modeling includes significant evaluation of solution existence, uniqueness, and accuracy. Popular engineering methods include the "vector fitting" approach [5], state-space approaches [6–8], and relatively "direct" application

[1]Jean Baptiste Joseph Fourier, March 1768-May 1830, was a French mathematician and physicist. After an early life torn between choosing a religious or mathematical pursuit, Fourier's political beliefs led to involvement in the French Revolution (1789-1799). During these years he alternated between teaching, performing military service, being imprisoned, and researching. It was his research into heat propagation whereupon Fourier came up with the ingenious (but highly controversial) insight that arbitrary functions might be described as a summation of sinusoids with various magnitudes and phases. While some of the details were incorrect (e.g., he believed that *all* functions could be described by such summations), he had difficulty convincing Laplace, Lagrange, and other contemporaries that even the fundamentals were plausible. Many decades passed before his genius was fully understood and accepted.

[2]Or at least some point $t-\tau$ where the impulse response $h(t)$ may be assumed to be zero for $t > \tau$. In the general case this point may be difficult to ascertain.

of least-squares fitting methods [9–11]. The choice is influenced by the circuit simulator under consideration: Contemporary SPICE simulators natively accept circuit "elements" that implement prescribed transfer functions in the Laplace domain, i.e., $f(s)$. Of course, Cauer or related synthesis methods [12–14] may be used to decompose most Laplace domain transfer functions into standard circuit elements such as resistors, capacitors, transformers, etc., but this approach is primarily useful for research or teaching purposes, as directly evaluating $f(s)$ will typically be faster. Regardless of the representation chosen, there are a number of constraints that the final model must satisfy including passivity, causality, and stability. Of these, passivity is the most difficult, and methods that enforce passivity typically compromise model accuracy as the trade-off [15–19].

The motivation for this work is to combine the initial "information" and physical intuition of a hand-crafted model while leveraging the inherent accuracy and speed of development of "black box" modeling: we attempt to obtain the "best of both worlds." This new modeling methodology is termed the *Circuit Augmentation Method*, or **CAM**. The method starts with an equivalent circuit model designed to give the user physical insight into the object being modeled. Next, the "golden" data from measurement or an electromagnetic field solver (known generically as "measurement data," abbreviated **MD** or **Meas**) is compared with that of the ECM. The "logical difference" (e.g., subtraction of Y parameters for a model and ECM placed in parallel – this is detailed later) is then fit to a low-order rational function; this result is known as the *augmentation* (abbreviated **Aug**). The *overall* model – subscripted as CAM – consists of the ECM combined with the augmentation and should compare favorably in accuracy with the measurement data.

Implementing this conceptually simple approach exposes various difficulties. For instance, while ECMs are presumed to be accurate over some restricted frequency range (typically low frequencies or some narrow bandpass set of frequencies), the ECM's poles and zeros affect a CAM model (the logical addition of an ECM and an augmentation) over *all* frequencies – the effects falling off as the inverse of the distance between the physical frequency on the $j\omega$ axis and the pole or zero's location in the s-plane. To compensate for these inaccuracies, a significant aspect of CAM is that it is *perturbational.* That is, as part of the "CAM process," parameters of the ECM such as component values, transmission line lengths, etc. are modified to provide the best CAM model possible. CAM uses an iterative process, alternating between modifying ECM attributes and augmenting pole-zero locations to achieve this result.

1.2 Organization of This Work

This thesis presents a complete report on the theoretical background and practical implement aspects of CAM. Research was purposely restricted to passive components to limit the scope of the problem while still addressing a significant industry need in circuit modeling.

Chapter two begins with an expanded discussion of circuit modeling background and the application of SPICE-like circuit simulators, as well as providing the necessary mathematical background related to modeling. Details are provided on rational function approximations, pole-zero modeling, stability, and passivity. Chapter three details the "core" of CAM: Circuit augmentation. Some of the more straightforward topologies used are analyzed with an eye towards utility. Included is discussion of the necessary addition to circuit augmentation to achieve higher accuracy: component value perturbation. A brief background of global optimization and CAM's use of it is provided. Chapter four provides a quick "guided tour" through the MATLAB software developed for CAM, while Chapter five examines various results obtained during the research stages of CAM. Chapter six contains insights on possible future research efforts as well as CAM's overall usability. Finally, a thorough appendix is provided that provides concrete – if not always particularly rigorous – derivations of many of the mathematical methods used internally by CAM. Also provided are additional details on circuit simulator history, circuit analysis, and a few other items that don't dovetail directly with the main work.

One brief word on notation: Except as noted, italicized quantities refer to scalars (e.g., x, $g(z,t)$, etc.) Lower-case letters in bold such as $\mathbf{f}$ or $\mathbf{x}$ refer to vectors, which are generally interpreted as column vectors with transposes added as needed (e.g., $\mathbf{f}^T$) for clarity. Upper-case letters in bold such as $\mathbf{A}$ or $\mathbf{S}$ are matrices.

Chapter 2 – Circuit Modeling

2.1 The Need for Circuit Models

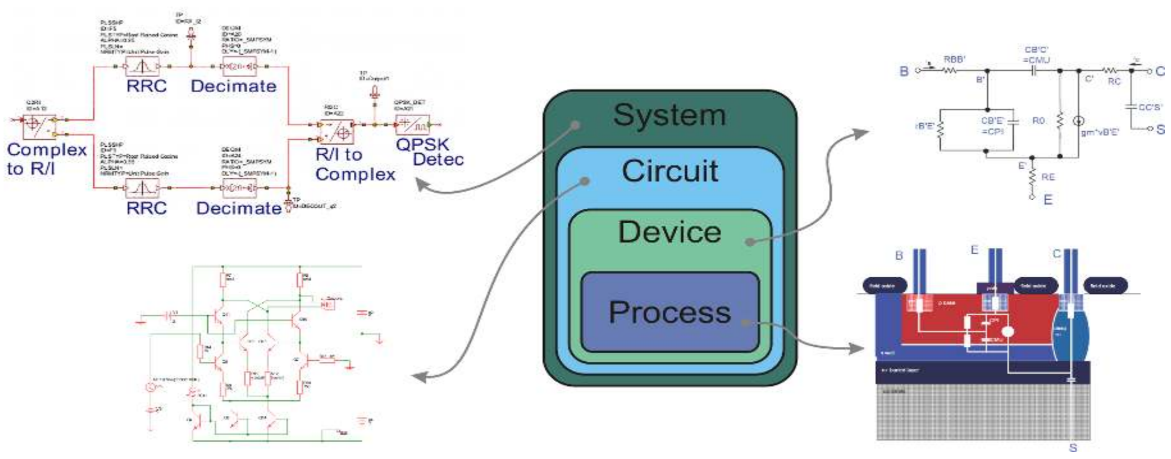


Figure 2.1: The Circuit Modeling Design Hierarchy. Arrows point to model examples.

The job of a design engineer generally boils down to designing one or more levels of the "design hierarchy" shown in Fig. 2.1 in response to the need to build some electronic "widget." For instance, a design engineer (or an entire team of engineers) might be tasked with developing the system-level specifications for a WiMax radio system, while another engineer (or team) might be tasked with designing a mixer circuit for that system, while others work on the transistors to be used in the mixer based on a process that a team of process engineers constantly attempt to improve and shrink.[1]

System level design is often performed based on "high-level" component specifications, such as the 1dB compression point of an amplifier or the IP3 intercept point of a mixer; system level simulators excel at taking such high-level descriptions of components, a specified input source (often a complex waveform, such as a complete WiMax packet), and propagating the various imperfections in the components through the system, at which point one can examine, e.g., power loss, spectral mask margins, etc.

[1] For those curious, the system level model in Fig. 2.1 is that of a QPSK demodulator while the circuit level model is a "Cascomp" amplifier (patented by Tektronix). Both device and process level models are of transistors.

The specified components are themselves typically designed and simulated in a circuit-level simulator, with SPICE being the most well-known. "Generic" versions of SPICE perform DC operating point analysis, small-signal AC simulation, and transient simulation, while contemporary versions have considerably more complex functionality as discussed later. While SPICE is perhaps the most familiar tool to the "traditional" integrated-circuit (IC) designer, historically linear alternating-current (AC) simulators such as those found within Agilent's ADS and Genesys, AWR's Microwave Office, or Ansoft's Designer tool were used by those with a "microwave" or "hard RF" backgrounds. The major difference was that the former generally dealt with "lumped" circuit elements and plotted simple linear, linear-log, or log-log rectangular graphs while the latter seamlessly dealt with multi-port-based data files (e.g., S/Y/Z/ABCD parameters) and could graphically display Smith chart and other specialized plots such as amplifier stability circles. Today, most SPICE-type simulators contain much of this same functionality.

The devices used in circuit simulation are themselves based on various "device" models, such as the well-known Ebers-Moll transistor model based on algebraic expressions, or the table-based IBIS (**I**nput/output **B**uffer **I**nformation **S**pecification). Finally, the devices themselves assume a certain physical model, which is typically the semiconductor substrates used in IC manufacturing, but can also be a higher-level construct such as the fiberglass- and epoxy binder-based materials in a printed circuit board wherein a pad is being modeled. Exact simulation of the process requires working with Maxwell's equations for macroscopic objects such as circuit boards and hole-electron diffusion and energy equations for microscopic objects including transistors, diodes, and so on.

Hierarchical Level	**Common Software Tools**	**Typical Model Implementations**
System	Simulink, SystemVue	Parameterized behavioral models
Circuit	SPICE, ADS, Genesys, MWO	SPICE netlists, Laplace blocks, S parameters
Device	SEDAN, PISCES, MINIMOS, GALENA	Ebers-Moll, Gummel-Poon models, BSIM, IBIS
Process	SUPREM, SAMPLE, SPEEDIE	Diffusion & implantation equations, Maxwell's equations

Table 2.1: Typical software tools and the means by which models are implemented for various levels of the circuit modeling design hierarchy.

A summary of some of the more common software tools and modeling methodologies used while tackling circuit design problems in shown in Table 2.1. As one can surmise, the reason for using different modeling "formats" at different levels of the hierarchy is to provide a balance between the simulator's execution time and the accuracy of results: It doesn't make sense to simulate a simple capacitor using a full-wave electromagnetic field solver when one is simply in need of a 1MHz RF coupling capacitor, just as it doesn't make sense to try to model a 10mm bond wire as a simple L-C pair at 10GHz. Those who insist on doing so anyway may end up with excessively long simulation times, inaccurate results, or prolonged graduation! Each higher level of the design hierarchy is a generalization of the one below, wherein certain assumptions are made about the problem to keep the problem tractable: Maxwell's equations can be derived from quantum electrodynamics given the assumption that macroscopic behavior is being observed, Ohm's and Kirchoff's laws come directly from Maxwell's equations based on the simplifying assumption that behavior is being measured over a distance that is a negligible fraction of a wavelength at the frequencies of interest[2] and thus the elements may be considered "lumped," etc.

As previously discussed, one goal of CAM is to provide an increase in circuit-level – SPICE-like – simulator performance. This challenge may be addressed via improved simulation techniques or improved modeling techniques. The former can become quite complicated, and various approaches are mentioned in the historical overview of circuit simulation in Appendix A. On the other hand, improved modeling techniques – while potentially quite complex in derivation as well – are typically implemented in simulators in one of three ways. These are:

- Table-based Models. While linear RF simulators have been using Touchstone and similar tabulated (table-based) network parameter formats for decades, more recent developments include (Intel's) IBIS and (AWR's) "Xmodels." Such models provide one or more sets of voltages, current, impedances and so on at the device's ports. Besides speeding up simulation, such models provide the means to protect proprietary I/O ring design techniques and avoid revealing various process parameters that may be of interest to competitors (IBIS models in particular tout this ability). Table-based models are straightforward to simulate, with some complexity involved in generating robust numerical interpolation, extrapolation, integration, and differentiation routines. While simulation with table-based models is fast, the obvious drawback is that the table must include data corresponding to the user's expected operation. Even for seemingly "simple" devices such as gigahertz-range RF power transistors, high-quality large-signal models may be difficult to obtain.

- Physical Models. Physical models are any models described by algebraic equations representing the underlying physics of the device. This type of model include most "classical" work such as the Ebers-Moll or Gummel-Poon transistor models or BSIM (**B**erkeley **S**hort-channel **I**GFET **M**odels). Unless a new algebraic model could be "built" as a sub-circuit of existing SPICE components, traditionally the addition of new algebraic models required writing FORTRAN or C code implementing the specified algebraic equations and, often-times, their integrals and/or derivatives; contemporary simulators allow for the straightforward inclusion of new algebraic models with the simulator having the ability to symbolically calculate inte-

[2] An alternative explanation is that the speed of light is infinite and thus *everything* is lumped, although this is arguably a more whimisical than practical interpretation.

grals and derivatives. Physical models have the significant advantage of providing accurate results over a much wider range of operation than table-based or empirical models typically do; the drawback is that simulation time is often the slowest of the three. Even parameterizing physical models may be quite challenging: a BSIM4 transistor model has *over 200* parameters!

- Empirical Models. Empirical models are produced by curve-fitting a set of basis functions to measurement or simulation data. This technique is quite common whenever the underlying physical model is not readily expressible in closed form – the impedance of a microstrip as a function of its width and height above a ground plane is a good example, as are some FET model parameters such as inversion layer width. While typically being fast to simulate and requiring less storage space than table-based models, empirical models have the same drawbacks as table-based models in that they must have included data over the intended operating points. (A classical example of limitations to empirical models is the set of formulas presented in the 1971 edition of the Motorola MECL System Design Handbook for the impedance of a microstrip: By the 1990's, PCB manufacturing technology allowed for narrower microstrips on thinner boards, but using the 1971 equations predicted negative impedances!)

CAM is a hybrid technique, attempting to preserve some of the physical insight provided by a (relatively simple) physical model (the ECM) while providing the speed and accuracy available from an empirical model (the augmentation). The choice of rational functions to represent the augmentation is driven by the ease of integration with a typical SPICE simulator, while providing sufficient accuracy for most circuit-level devices. Simulator analysis is straightforward: For DC operating point calculations the rational function is evaluated with $s = 0$, whereas for small-signal AC simulation it is evaluated with $s = j\omega$. For transient simulation, the partial-fraction expansion of a rational function becomes a simple time-domain expression involving exponentially damped sinusoids (or various degenerations thereof – this is detailed further in Section 2.3), which lends itself to fast simulation. The next section provides more details on the behavior of rational functions as applied to CAM.

As an aside, the preceding discussion has assumed one intends to use a *computer* to simulate their system, circuit, device or process. Obviously this hasn't always been possible: in decades past, one would physically build the item in question, measure the results, and iterate as necessary to obtain a solution; this "cut and try" method was used for decades prior to the widespread availability of PCs. Even today, those armed with experience and intuition can design and debug certain circuits such as 1st- and 2nd-order matching networks more quickly using such methods rather than using a simulator: the computer is still no substitute for *thinking!* (A major problem with simulators is that new users tend to *believe* the output, *without question*, with its 10+ digits of *precision*, when an experienced designer could quickly ascertain that the results have *very few* digits of *accuracy.*) Of course, such "old school" approaches are of less use at the process and system levels than at the circuit level due to the expenses involved with each "cut and try" iteration.

2.2 Rational Function Approximations

Although CAM circuit models usually come in the form of schematic diagrams, from a mathematical perspective these are thought of as generic linear networks that simply provide some number of poles and zeros. In the Laplace domain, such networks are mathematically matrices that are a function of s, and these functions may be expressed as rational functions, sums of poles and residues, etc. To be useful, circuit models are expected to be stable and passive, and this has specific implications as to the form of the function describing them. We first motivate the use of rational functions to model circuits, and move on to consider the implications of stability and passivity.

The representation of a set of data or a given function in terms of a polynomial is termed a *Taylor Series* (or *Maclurin Series* when the expansion for $f(x)$ is about $x = 0$). Various methods exist for computing the coefficients in a Taylor Series [20], and its application and limitations are well understood. Unfortunately, polynomials are not well suited for modeling electrical circuits; see Appendix B for a discussion. Slightly more sophisticated, a representation for a set of data or a given function in terms of a quotient of rational polynomials is termed a *Padé Approximation.* This representation for an arbitrary function $h(s)$ is as follows

$$h(s) = \frac{\sum\limits_{m=0}^{P} a_m s^m}{\sum\limits_{n=0}^{Q} b_n s^n} \tag{2.1}$$

(Note that the a_m's and b_n's in Eq. (2.1) are generally unique only within a multiplicative constant. Some authors define b_0 or another coefficient as one to remove this ambiguity.) Computing coefficients for Padé Approximations is a significantly more challenging problem than computing those for a Taylor Series, although it has been studied for hundreds of years: the great French mathematician Cauchy published a paper on the topic in 1821 [21]! ([22] provides a more contemporary – and English language – description of the technique, although for those who can read French, Cauchy's collected works are readily available.) Cauchy's paper provides a specific means for determining the rational functions's coefficients, which might be termed "Cauchy's Method," although this term seems to be somewhat abused to refer to *various* different methods by which the coefficients are obtained. A popular "general purpose" approach is the *Epsilon Algorithm* [23], but this method is difficult to extend to vector data in a straightforward manner [24]. Early CAM research included attempts at developing robust rational function fitting routines, it quickly became evident that doing so was beyond the intended scope of the research[3]. As such, CAM uses third-party routines for curve fitting: The "minimum Eigenvalue" approach described in [25] (this approach is roughly the same as Cauchy's Method), and the markedly robust Vector Fitting method developed in [5].

Eq. (2.1) is sometimes used in other, mathematically equivalent forms. The most common of these is a pole-residue form, which results from a partial-fraction expansion of (2.1). This form is

$$h(s) = \sum_{i=1}^{Q} \frac{k_i}{s - p_i} + d + se \tag{2.2}$$

where Q is equal to that in (2.1); if d and e are non-zero, this implies that $P = Q + 1$ in (2.1). Other values of d and e in (2.2) change P and Q in (2.1) self-evidently; the representation shown is used due to the fact that all immittance and transfer functions for passive circuits can be expressed without any additional terms in s; this is proven in Appendix D.2.

To be useful in a simulation, a rational function representing an admittance – or a matrix of such entities representing a linear multi-port network – needs to be used in a manner that results in a circuit that is both stable and passive. We make a distinction here that, while a network may consist of multiple objects (e.g., in CAM this would typically be an augmentation network in the form of a rational function combined with schematic-based equivalent circuit model) and while the *overall* network must be stable and passive, *individual* objects within it need not be: A trivial example would be a "user model" consisting of a -5Ω resistor in series with an "augmentation impedance" of $h(s) = 20 + 3s$ (Ω), resulting in an overall network representing a 15Ω resistor in series with a 478mH inductor, which is obviously stable and passive. Although highly degenerative, this example is not entirely contrived: Embedding negative elements into a model may be used to de-embed the effects of test fixtures and cables. The point here is that enforcing passivity (and thus stability) on every object within a model obviates the question of the model's overall passivity; however, such a restriction is overly constraining and thus not always useful. In any case, tests for stability and passivity are needed with CAM, and these are discussed below.

2.3 Stability

Stability in circuit modeling uses the familiar "bounded input/bounded output" criteria common in system modeling. Specifically, when a non-zero, finite-energy signal is presented to the system, the output must not become unbounded (that is, asymptotically approach infinity at any time) in a manner that indicates infinite energy is produced or consumed. Examining Eq. (2.2), it is clear that obtaining a time-domain response is a simple matter of performing the inverse Laplace transform. For the constant term d, the time-domain function is simply $d \cdot \delta(t)$ (where $\delta(t)$ is the standard

[3] This is a euphemisn for, "we failed." In retrospect, it has become clear that the problem was largely a lack of sufficient mathematical background.

Dirac delta function), which contains finite energy by definition. Similarly, the proportional term (e) is bounded, and while a non-zero value for e is physically meaningful for immittances (e.g., the impedance of an inductor or the admittance of a capacitor), for transfer functions it is generally taken as zero since a non-zero value would imply a source with, e.g., infinite voltage or current output potential, which is non-physical. On the other hand, most rational function curve-fitting routines retain the ability to calculate a (non-zero) value for e, since it may provide some utility in modeling, even with the knowledge that the result cannot be valid as $\omega \to \infty$.

Dealing with the partial fractions in Eq. (2.2) is straightforward. First, since the roots of a polynomial with real coefficients can only be real or occur in complex conjugate pairs, it follows that the k_i's and p_i's in Eq. (2.2) must also be real or come in complex-conjugate pairs; for convenience we set $k_{i+1} = k_i^*$ and $p_{i+1} = p_i^*$. When the pole and residue are real, we have the simple expression

$$\frac{k_i}{s - p_i} \overset{\mathcal{L}}{\Leftrightarrow} k_i e^{p_i t} u(t)$$

which can be seen to be bounded so long as p_i is negative, or – as it is commonly stated – in the left-half of the complex plane. Additionally, note that this response is causal due to the inclusion of the unit step response, $u(t)$. For the complex conjugate case, we have

$$\frac{k_i}{s - p_i} + \frac{k_{i+1}}{s - p_{i+1}} = \frac{k_i}{s - p_i} + \frac{k_i^*}{s - p_i^*} \overset{\mathcal{L}}{\Leftrightarrow} \left(k_i e^{p_i t} + k_i^* e^{p_i^* t}\right) u(t)$$

Expanding the right-hand side into real and imaginary parts, collecting common terms, and equating complex-conjugate exponential sums with sines and cosines produces

$$\left(k_i e^{p_i t} + k_i^* e^{p_i^* t}\right) u(t) = 2e^{p_{re} t}\{k_{re}\cos(p_{im} t) - k_{im}\sin(p_{im} t)\}u(t) \tag{2.3}$$

where $p_{re} = \Re(p_i)$, $p_{im} = \Im(p_i)$, $k_{re} = \Re(k_i)$, and $k_{im} = \Im(k_i)$. Next, using the trigonometric identity

$$a\sin(x) + b\cos(x) = \sqrt{a^2 + b^2}\cos(x - \theta)$$

where $\theta = \tan^{-1}(a/b)$, Eq. (2.3) can be rewritten as

$$2\,|k_i|\,e^{p_{re}}\cos(p_{im} t + \arg(k_i))u(t)$$

(where $\arg(x)$ represents the angle of the complex number x with respect to the real axis) which is seen to be an exponentially decaying sinusoid – assuming p_{re} is negative – with an amplitude set by the magnitude of the residue, the phase offset controlled by the residue's angle, and the sinusoid's frequency and exponential decay rate set by the pole's offset from the $j\omega$ and σ axes, respectively. Again, the result is causal and bounded so long as the pole remains in the left-half plane (LHP).

In curve-fitting (or, more generally, constraint-fitting) routines, enforcing the requirement that all poles are located in the left-half plane may be performed as a "post-processing" step after all the coefficients are calculated. Known techniques for dealing with poles in the right-half plane include simply *removing* the poles [25], as well as "*flipping*" the pole symmetrically about the $j\omega$ axis back into the left-half plane [5]; in both of these rather heuristical methods, the residues are updated accordingly.

2.4 Passivity

Passivity implies that an object can create no more energy than what is put into it. Although this definition is simple enough, dealing with passivity can be challenging due to the difficulty in quantifying the definition when applied to a multi-port circuit. It is important to clarify the difference between stability and passivity: While a stable but non-passive network *in isolation* cannot produce an unstable output for a bounded input, the addition of other, passive elements to such a network may cause the overall system to become unstable. A concrete example is an oscillator circuit where the feedback connection has been brought out to a port and remains disconnected from the circuit's output port: In isolation, any circuit excitation will produce a finite energy output. However, upon closing the feedback path, the circuit begins to oscillate and is unstable. (Real oscillators are unstable upon start-up, but become stable due to decreasing forward path gain as a function of output voltage, limiting devices such as Zener diodes in the output path, or simply "hitting the [power supply] rails" at the output. These are all non-linear behaviors.)

Note that passivity in no way implies a lack of voltage or current gain (or even both at the same time, so long as their phases are far enough apart): The obvious example is the ideal transformer – a passive device, yet one that clearly provides voltage or current gain. Less-intuitive examples include RC filters that provide voltage gain [26], and highly-lossy transmission lines that provide reflection coefficients with magnitudes greater than one [27]. All passive systems are stable and, as such, there's no need to check the stability of a network if it is known to be passive (indeed, in early papers the primary reason to check the stability of a network seems to be its ease relative to checking its passivity! – for more contemporary papers, however, it appears that enforcing stability alone followed by enforcing passivity tends to result in less model accuracy degradation than simple enforcing passivity for a model that's known to be inherently unstable).

The mathematical conditions necessary for passivity can be formulated as follows: For a one-port network (the trivial case such as a single resistor), the power *absorbed* by the network is (using phasor notation) $V_{eff}^{*} \cdot I_{eff}$, with I_{eff} having the standard reference of being positive when conventional current flows *into* the network[4]. In this case, if the real power absorbed by the network is positive

[4]The "eff" subscripts indicates the use of the "effective" or RMS values of the voltage and current; most test equipment utilizes such values, as the formula is then applicable regardless of the exact shape of the waveform

$(= \Re(V^* I))$, the network is passive. If negative, power is instead flowing from the network towards the source, and hence the network is acting as a generator and is not passive. (If zero, the network is completely reactive and also passive, but this is not a physically likely scenario – all real networks have *some* loss, even if it is quite small.) Another way to interpret this scenario is to simply look at the phase difference between voltage and current: If it is between 270° (purely capacitive) and 90° (purely inductive), power flows into the network; between 90° and 270°, power flows out of the network. For n-port networks, the idea is to simply sum the power absorbed at *all* the ports and verify that the real power is greater than zero. Using admittance parameters we compute

$$\begin{aligned} P_{Absorbed} &= \Re(\mathbf{v}^H \cdot \mathbf{i}) \\ &= \frac{1}{2}\left(\mathbf{v}^H \mathbf{i} + \mathbf{v}^T \mathbf{i}^*\right) \\ &= \frac{1}{2}\left(\mathbf{v}^H \mathbf{i} + \mathbf{i}^H \mathbf{v}\right) \\ &= \frac{1}{2}\left(\mathbf{v}^H \mathbf{Y} \mathbf{v} + (\mathbf{Y}\mathbf{v})^H \mathbf{v}\right) \\ &= \mathbf{v}^H \left(\frac{\mathbf{Y} + \mathbf{Y}^H}{2}\right) \mathbf{v} \end{aligned} \tag{2.4}$$

(where $\mathbf{X}^T$ represents the transpose of $\mathbf{X}$ and $\mathbf{X}^H$ represents the complex-conjugate transpose or *Hermitian* of $\mathbf{X}$).

The appearance of the last line in 2.4 is a quadratic form, since $\left(\mathbf{Y} + \mathbf{Y}^H\right)$ is Hermitian (see Appendix C.6 for more information regarding quadratic forms and their relationship to positiveness definiteness.) As such, Eq. (2.4) will be greater than zero when [20]

$$\mathrm{eig}_i\left(\frac{\mathbf{Y}^H(j\omega) + \mathbf{Y}(j\omega)}{2}\right) > 0 \tag{2.5}$$

for all eigenvalues i. This is the form commonly seen in literature, although

$$\mathrm{eig}(\Re(\mathbf{Y})) > 0$$

is perhaps a little clearer.

While Eq. (2.5) provides a passivity test for any arbitrary network represented by admittance parameters, in this work all components were assumed to be reciprocal, implying that $\mathbf{Y} = \mathbf{Y}^T$. Admittance matrices obtained from measurement data were checked to insure this was the case before proceeding – for small asymmetries due to measurement errors, a common technique to

involved. For circuit analysis purposes, sinusoidal waveforms are often assumed, with $V_{eff} = \frac{\sqrt{2}}{2} V_m$, where V_m is the magnitude of the sinusoidal voltage; the resultant formula is then $P = \frac{1}{2}\Re(V_m I_m^*)$. Except as noted, effective voltages and currents are used throughout.

enforce symmetry is to average the two halves (upper and lower triangular regions) of the matrix together.

For impedance parameters, it should come as no surprise that the requirement for passivity is

$$\mathrm{eig}_i\left(\frac{\mathbf{Z}^H(j\omega)+\mathbf{Z}(j\omega)}{2}\right) > 0$$

The derivation follows an identical outline to that used above for admittance parameters. The results for scattering parameters are slightly different, although we begin the same way by noting that the *total* incident power must be greater than the *total* reflected power: $P_{Inc} > P_{Ref}$. Assuming identical real port impedances[5], mathematically this is [28]

$$\mathbf{v}^{+H}\mathbf{v}^{+} > \mathbf{v}^{-H}\mathbf{v}^{-}$$

where the "+" and "-" superscripts indicate the voltages associated with the forward or backwards traveling wave, respectively. Substituting $\mathbf{v}^{-} = \mathbf{S}\mathbf{v}^{+}$ and rearranging terms provides

$$\begin{aligned} \mathbf{v}^{+H}\mathbf{v}^{+} &> (\mathbf{S}\mathbf{v}^{+})^H(\mathbf{S}\mathbf{v}^{+}) \\ \mathbf{v}^{+H}\mathbf{v}^{+} &> \mathbf{v}^{+H}\mathbf{S}^H\mathbf{S}\mathbf{v}^{+} \\ \mathbf{v}^{+H}(\mathbf{I}-\mathbf{S}^H\mathbf{S})\mathbf{v}^{+} &> 0 \end{aligned} \tag{2.6}$$

For the case of a lossless network ($P_{Inc} = P_{Ref}$) we see the well-known result that $\mathbf{S}^H\mathbf{S}$ need be unitary (physically a rare occurrence, but for the sake of analysis often a useful approximation). For the (lossy) passive case we again have a quadratic form, and the expression (2.6) will be true so long as $\mathbf{I}-\mathbf{S}^H\mathbf{S}$ – known as the *dissipation matrix* [29]– is positive definite. Since $\mathbf{I}$ and $\mathbf{S}^H\mathbf{S}$ are both Hermitian, so is their difference, and hence an eigenvalue check suffices to guarantee positive definiteness[6]:

$$\mathrm{eig}(\mathbf{I}-\mathbf{S}^H\mathbf{S}) > 0$$

Using the result that

$$\mathrm{eig}(\alpha_k\mathbf{S}^K + \alpha_{k-1}\mathbf{S}^{K-1} + \cdots + \alpha_1\mathbf{S} + \alpha_0) = \alpha_k\,\mathrm{eig}(\mathbf{S})^K + \alpha_{k-1}\,\mathrm{eig}(\mathbf{S})^{K-1} + \cdots + \alpha_1\,\mathrm{eig}(\mathbf{S}) + \alpha_0$$

(where the eigenvalue on both sides must refer to the same eigenvalue, of course) this is equivalent

[5] In the case of complex port impedances, the power to the load is no longer generally $\mathbf{v}^{+H}\mathbf{v}^{+} - \mathbf{v}^{-H}\mathbf{v}^{-}$, and therefore the analysis becomes more complicated; see [66] for the results.

[6] Since $\mathbf{I}$ is positive definite and $\mathbf{S}^H\mathbf{S}$ is positive semidefinite, their *sum* $\mathbf{I}+\mathbf{S}^H\mathbf{S}$ would also automatically be positive semidefinite. $\mathbf{I}-\mathbf{S}^H\mathbf{S}$ may or may not be, however – the subtraction makes all the 'difference!'

to

$$\begin{aligned} 1 - \mathrm{eig}(\mathbf{S}^H\mathbf{S}) &> 0 \\ \mathrm{eig}(\mathbf{S}^H\mathbf{S}) &< 1 \end{aligned} \tag{2.7}$$

Finally, since $\mathbf{S}^H\mathbf{S}$ is positive semidefinite, its eigenvalues will be greater than or equal to zero, and hence may be expressed as the square of some numbers σ_i, known as the singular values of $\mathbf{S}$. That is

$$\sigma_i = \sqrt{\mathrm{eig}_i(\mathbf{S}^H\mathbf{S})}$$

and therefore a network described by scattering parameters $\mathbf{S}$ is passive at a given frequency $j\omega$ if all of $\mathbf{S}$'s singular values are less than one:

$$\sigma(j\omega) < 1$$

While Eq. (2.7) is quite workable and the introduction of singular values appears to do little but muddy the waters, it is the form commonly seen in literature.

It is surprisingly common to find that a rational function curve fitting routine will calculate coefficients that lead to immittance matrices that are *not* passive. The result – a stable but non-passive network connected to a passive network can create an unstable system – has already been discussed; reference [3] provides some more examples. This should not be too surprising, as the non-passive network represents an energy source, and thus can easily function as an amplifier for the passive network's resonant modes, causing oscillations that quickly grow out of control. Even in system design, the unwary may inadvertently start calling for non-passive devices: [30] discusses naive attempts at "waveform engineering" (precisely defining the voltage and current phases through a transistor, to minimize its power dissipation) in RF power amplifiers that leads to calculated efficiencies greater than 100%. The devil in the details turns out to be the requirement for a non-passive load!

An obvious drawback to Eq. (2.5) is that it applies at only one frequency point. For a device model intended to be used over a given frequency range, how many frequencies within that range need be checked for passivity? How about frequencies outside this range? While enough discrete frequency checks may provide enough confidence in a model to make it usable, clearly it's desirable to instead find an "algebraic" passivity test for transfer functions that would guarantee passivity over *all* frequencies starting with a rational function.

Such a test does exist, albeit with some limitations. [19] details the derivation, which is based on a result from [31] (also available in [32]); its usage is summarized here[7]. First, the transfer

[7]Those looking to confirm the derivations will need a healthy dose of linear algebra and state-space modeling background.

function must be converted to a minimum "state-space" representation[8]:

$$\begin{aligned} \mathbf{x}'(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{aligned} \tag{2.8}$$

Here $\mathbf{A}$ is the *system matrix,* relating how the current state $\mathbf{x}$ affects "itself," $\mathbf{B}$ is the *control matrix* which determines how the current input $\mathbf{u}$ affects the current state, $\mathbf{C}$ is the *output matrix* which translates the current state into an output vector $\mathbf{y}$, and $\mathbf{D}$ is the *feed-forward* matrix, that allows the input to directly influence the output regardless of the state. For a given system, state matrices are not unique, since various permutations of rows and scaling of matrix entries can be made to "cancel out." However, once a state matrix is constructed (by, e.g., the MATLAB "ss" command as used by CAM or any other means) the following statements hold [33]:

- Solving the first line of (2.8) (in the Laplace domain, with $s\mathbf{x}(s)$ replacing $\mathbf{x}'(t)$) and substituting it into the second line allows one to directly calculate the transfer function matrix $\mathbf{H}(\mathbf{s})$, where $\mathbf{y}(s) = \mathbf{H}(s)\mathbf{u}(s)$, as $\mathbf{H}(s) = \mathbf{C}(\mathbf{sI} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$. This may be equated directly to, e.g., $\mathbf{Y}$, $\mathbf{Z}$, or $\mathbf{S}$ parameter matrix.
- $\lim_{\omega\to\infty} \mathbf{H}(j\omega) = \mathbf{D}$, as can be seen from the form of $\mathbf{H}(s)$. This has significant implications as will be seen shortly: If $\mathbf{H}(s)$ represents, e.g., the impedance of a single-port network that becomes a short circuit as $\omega \to \infty$, D (a scalar in the one-port case) will be zero. (If $\mathbf{H}(s)$ represents the scattering parameters of such a network, $D_{\omega\to\infty} = -1$.)
- For a system representing immittance parameters, the system is stable if the real part of $\mathbf{A}$'s eigenvalues are all negative. Other methods such as the Routh-Hurwitz criteria can also check for stability (and require less mathematical effort to compute).
- For a system representing immittance parameters, the system is passive if $\mathbf{H}(s) + \mathbf{H}^*(s) > 0$ for all $\Re(s) > 0$; this a more general version of (2.5).

The algebraic passivity test is as follows. First, construct the appropriate Hamiltonian matrix (see Appendix C.5 for more information regarding Hamiltonian matrices) $\mathbf{P}$ depending on whether the state space system represents immittance parameters or scattering parameters. For immittance

[8]State-space representations cannot represent distributed networks such as those containing transmission lines. While such systems *can* have valid Laplace transforms (e.g., $e^{-\tau s}$ for an ideal transmission line with time delay τ), they *cannot* be represented with rational-polynomial functions of finite order. Since this fact stops no one from *approximating* systems using such functions, it won't stop us from using state-space representations for them either. Yes, circuit modelers *do* live on the edge!

parameters we have

$$\begin{aligned} & \mathbf{Q} \triangleq \mathbf{D} + \mathbf{D}^T \\ \mathbf{P} \quad = & \begin{bmatrix} \mathbf{A} - \mathbf{B}\mathbf{Q}^{-1}\mathbf{C} & -\mathbf{B}\mathbf{Q}^{-1}\mathbf{B}^T \\ \mathbf{C}^T\mathbf{Q}^{-1}\mathbf{C} & -\mathbf{A}^T + \mathbf{C}^T\mathbf{Q}^{-1}\mathbf{B}^T \end{bmatrix} \end{aligned}$$

whereas for scattering parameters we compute

$$\begin{aligned} & \mathbf{R} \triangleq \mathbf{I} - \mathbf{D}^T\mathbf{D} \\ & \mathbf{S} \triangleq \mathbf{I} - \mathbf{D}\mathbf{D}^T \\ \mathbf{P} \quad = & \begin{bmatrix} \mathbf{A} + \mathbf{B}\mathbf{R}^{-1}\mathbf{D}^T\mathbf{C} & \mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T \\ -\mathbf{C}^T\mathbf{S}^{-1}\mathbf{C} & -\mathbf{A}^T - \mathbf{C}^T\mathbf{D}\mathbf{R}^{-1}\mathbf{B}^T \end{bmatrix} \end{aligned}$$

(These differences are due to the passivity criteria differences seen in Eq. (2.5) and (2.7); what the test actually "does" is to provide the frequencies at which the eigenvalues of a given linear matrix inequality become true.) Next, the eigenvalues of **P** are computed. *If any of these eigenvalues are purely imaginary, the system changes, at some frequency, from being passive to non-passive or vice versa, and therefore the system must be non-passive.* Now, the detection of "purely imaginary" eigenvalues begs the question of what to do with small real components calculated due to round-off error; [34] discusses this problem in detail and – given that the eigenvalues are from a Hamiltonian matrix – uses a "structured eigenvalue solver based on symplectic URV decomposition"[9] to obtain more accurate results. In any case, the magnitude of each pure imaginary eigenvalue provides the frequency at which the passivity "violations" occurred; more sophisticated expressions for **P** provide a means of calculating the frequencies at which the system becomes "arbitrarily close" to being non-passive; this may be used as the start of a passivity enforcement routine [19].

There are limitations to the algebraic passivity test: First, since the imaginary eigenvalues indicate *changes* in passivity over all frequencies, one needs to use at least one "single frequency point" test to ascertain that the system isn't non-passive *for all frequencies* ($\omega = 1$ is a good choice). Secondly, the requirement to invert **Q** or **R** and **S**, respectively, is not possible when the system has ports that approach lossless conditions as $\omega \to \infty$ – this implies that $\mathbf{D} = 0$ (immittances) or ± 1 (scattering parameters). Such systems arise more commonly than might be suspected: An example is an ECM's ideal series inductors preventing a port containing various reactances from "seeing" its termination impedance, resulting in a low-pass behavior. In such cases a small amount of artificial loss (or an asymptotic match to the port reference impedance) at high frequencies may be added [35] to avoid ill-conditioning in **P**; this is acceptable from a physics viewpoint since all physical devices become lossy as $\omega \to \infty$.

[9] Such techniques are *well* beyond the scope of this work.

Chapter 3 – The Circuit Augmentation Method

3.1 Direct Augmentation

"Direct" augmentation is the process whereby a user-provided "equivalent circuit" has a circuit network logically interposed to create a "hybrid" combination that, overall, exhibits behavior closer to that of the provided measurement data than the ECM does alone. "Equivalent Circuits," in the sense used here, are simply hand-crafted circuit models that attempt to reasonably approximate the network behavior of an arbitrary circuit object, for instance, a spiral inductor on silicon. Although the model itself is usually created in the form of a schematic utilizing ideal basic circuits elements (e.g., R, L, G, C, lossless transmission lines, etc.), conceivably it could be generated using other techniques such as system model diagrams employing (perhaps) time delays, integrators, differentiators, loads, etc. From the CAM perspective equivalent circuit models simply provide a means to generate "predicted" network behavior for comparison with "measurement data," which may be tabulated measurement data obtained from, e.g., a network analyzer, curve tracer, etc., or may be from a high-accuracy simulator such as a field solver. The data set itself is represented via standard network parameters such as scattering (S or T), immittance (Y/Z), "chain" (ABCD), or "hybrid" (G/H) hybrid parameters. For instance, a spiral inductor's measurement data could be represented as

$$\mathbf{Y}_{Meas}(\omega_n) = \begin{bmatrix} Y_{11}(\omega_n) & Y_{12}(\omega_n) \\ Y_{21}(\omega_n) & Y_{22}(\omega_n) \end{bmatrix}$$

where a field solver has calculated the network parameters of the "input" and "output" sides of the inductor (relative to a common ground) at N discrete frequency points ω_n, $n = 1 \ldots N$. These data are then compared to data at those same frequency points from the equivalent circuit model (ECM). The logical "difference" between these two data sets is used to construct an *augmentation network*, which is combined with the user-provided ECM to create a "hybrid" model whose behavior should accurately match the measurement data; this concept is illustrated in Fig. 3.1. It is this means of generating the hybrid that is referred to as the Circuit Augmentation Method, or CAM, and the parameters associated with the hybrid model as subscripted CAM.

One might question how, exactly, the "difference data" are computed: The answer requires the adoption of a particular network topology, from which the mathematics follows in short order. The most obvious topologies are "network parallel" and "network cascade" setups, along with π and tee "branch" networks; these are detailed below. However – as will be seen in the examples – CAM is a relatively general method, and "variations on the theme" can sometimes be used to advantage when some additional knowledge of the underlying physical structure is available. Despite the ease with which various augmentation topologies can be derived, it is not always obvious which should be chosen: While experienced modeling engineers may have an intuitive feel for the best choice, less experienced individuals may not. However, CAM models seldom require more than a second or so to generate, and therefore various alternatives can be explored until intuition is gleaned.

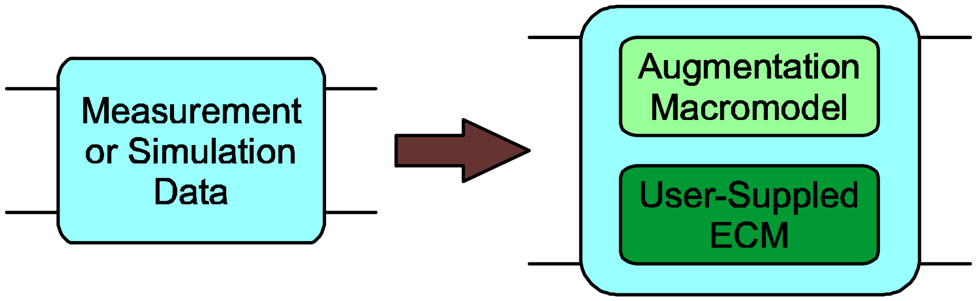


Figure 3.1: The concept of equivalent circuit model augmentation.

3.1.1 Branch Topologies

Branch topology-based augmentations take advantage of the exact equivalences between any *reciprocal* two-port network and simple π- and tee-networks defined with three impedance (or admittance) "legs," denoted "left," "center," and "right," as shown in Fig. 3.2:
The elements in the tee network are impedances, whereas those in the π network are admittances.

For a π-network "decomposition," measurement data are set equal to that provided by the CAM model, which consists of three "augmentation" admittances in parallel with those provided by three ECM admittances, as shown in Fig. 3.3. In this case,

$$\begin{aligned}
Y_{CAM}(s) \overset{\triangle}{=} Y_{MD}(s) &= Y_{ECM}(s) + Y_{Aug}(s) \\
Y_{Aug}(s) &= Y_{MD}(s) - Y_{ECM}(s) \\
\begin{bmatrix} Y^{Aug}_{Left}(s) \\ Y^{Aug}_{Center}(s) \\ Y^{Aug}_{Right}(s) \end{bmatrix} &= \begin{bmatrix} Y^{MD}_{11}(s) + Y^{MD}_{12}(s) \\ -Y^{MD}_{12}(s) \\ Y^{MD}_{22}(s) + Y^{MD}_{12}(s) \end{bmatrix} - \begin{bmatrix} Y^{ECM}_{11}(s) + Y^{ECM}_{12}(s) \\ -Y^{ECM}_{12}(s) \\ Y^{ECM}_{22}(s) + Y^{ECM}_{12}(s) \end{bmatrix}
\end{aligned}$$

After the augmentation data (Y^{Aug}_{Left}, Y^{Aug}_{Center}. and Y^{Aug}_{Right}) are calculated, rational functions are "curve fit" to each branch.

Similarly, for a tee-network decomposition, measurement data are set equal to that provided by

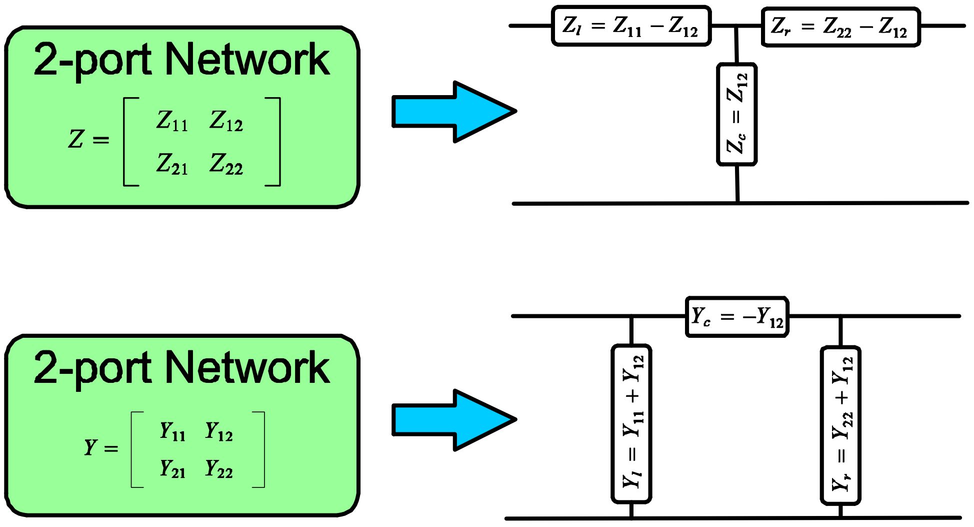


Figure 3.2: Two-port reciprocal network equivalences to tee- and π-topology circuit networks ($Z_{12} = Z_{21}$ and $Y_{12} = Y_{21}$).

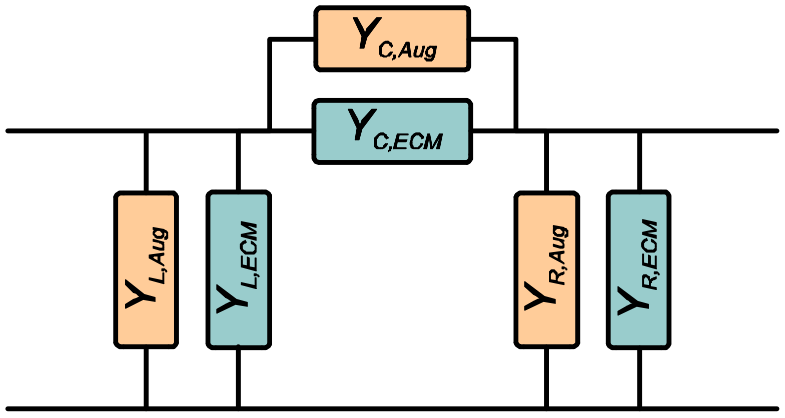


Figure 3.3: Parallel (admittance) augmentation using π-network decomposition.

the CAM model consisting of three augmentation impedances in series with those provided by three ECM impedances, as shown in Fig. 3.4. In this case,

$$\begin{aligned} Z_{CAM}(s) \overset{\triangle}{=} Z_{MD}(s) &= Z_{ECM}(s) + Z_{Aug}(s) \\ Z_{Aug}(s) &= Z_{MD}(s) - Z_{ECM}(s) \\ \begin{bmatrix} Z^{Aug}_{Left}(s) \\ Z^{Aug}_{Center}(s) \\ Z^{Aug}_{Right}(s) \end{bmatrix} &= \begin{bmatrix} Z^{MD}_{11}(s) - Z^{MD}_{12}(s) \\ Z^{MD}_{12}(s) \\ Z^{MD}_{22}(s) - Z^{MD}_{12}(s) \end{bmatrix} - \begin{bmatrix} Z^{ECM}_{11}(s) - Z^{ECM}_{12}(s) \\ Z^{ECM}_{12}(s) \\ Z^{ECM}_{22}(s) - Z^{ECM}_{12}(s) \end{bmatrix} \end{aligned}$$

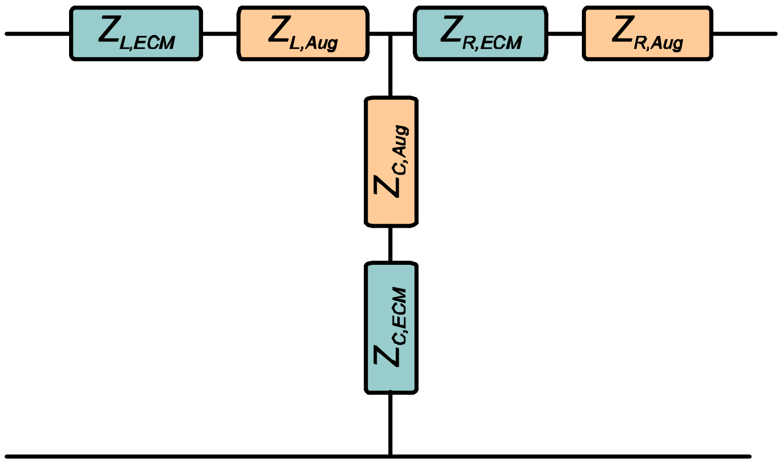


Figure 3.4: Series (impedance) augmentation using tee-network decomposition.

After the augmentation data (Z^{Aug}_{Left}, Z^{Aug}_{Center}. and Z^{Aug}_{Right}) are calculated, rational functions are "curve fit" to each branch.

For each branch in either decomposition, a series impedance or parallel admittance could be used, providing 8 different models. Using the most straightforward choice (all elements taken as admittances for π-networks and as impedances for tee-networks) has generally been found to perform as well as any other selection; scenarios where large differences are seen may be indicative of poor-quality ECMs.

3.1.2 Network Topologies

Augmentation using "network" parameters, such as Y/Z/S/ABCD parameters, involves constructing two networks and calculating the unknown augmentation network's values based on the known measurement data and ECM data sets (which – if need be – are readily converted to a network representation from, e.g., a SPICE or linear RF simulator). CAM software currently supports two network augmentation strategies, "network parallel" and "network cascade." Network parallel is shown in Fig. 3.5. An advantage of network parallel augmentation is that computation of Y_{Aug} is trivial, regardless of the number of ports in the network: By inspection, $Y_{Aug}(s) = Y_{MD}(s) - Y_{ECM}(s)$; a rational function is fit to this data.

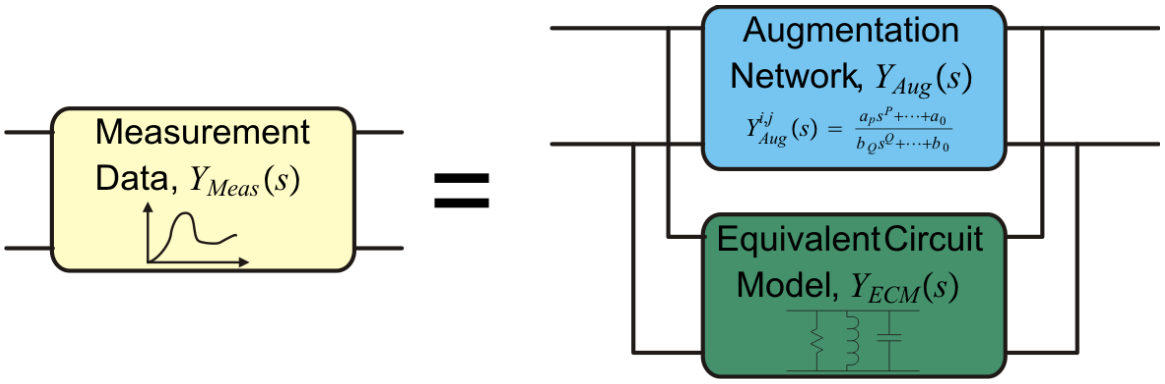

Measurement Data, $Y_{Meas}(s)$
=
Augmentation Network, $Y_{Aug}(s)$
$Y^{i,j}_{Aug}(s) = \frac{a_P s^P + \cdots + a_0}{b_Q s^Q + \cdots + b_0}$
Equivalent Circuit Model, $Y_{ECM}(s)$

Figure 3.5: "Network Parallel" augmentation topology.

The other network topology supported by CAM, network cascade, is shown in Fig. 3.6. Although drawn using T parameters, mathematically ABCD parameters work identically, although T parameters may be preferable in that the implicit (fixed) port reference impedances may make them easier to de-embed and more accurate when converted from S parameters. Additionally, while ABCD parameters can be extended beyond two-port networks – "full differential," or four-port ABCD parameters are not uncommon – T parameters offer a more natural representation for many-port networks. As with the network parallel case, inspection provides $T_{Aug}(s) = T^{-1}_{ECM}(s) \cdot T_{MD}(s)$; a rational function is fit to this data.

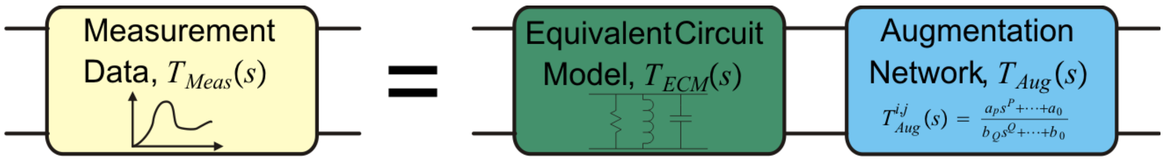

Measurement Data, $T_{Meas}(s)$
=
Equivalent Circuit Model, $T_{ECM}(s)$
Augmentation Network, $T_{Aug}(s)$
$T^{i,j}_{Aug}(s) = \frac{a_P s^P + \cdots + a_0}{b_Q s^Q + \cdots + b_0}$

Figure 3.6: "Network Cascade" augmentation topology. This topology is mathematically identical with ABCD parameters.

Other "variations on the theme" – such as hybrid series/parallel network topologies that directly support G and H parameters – could be constructed. While this has not been thoroughly investigated, it is not expected to provide significantly better results than the simply parallel and cascade topologies. On the other hand, significant changes in the "divide" between the ECM and the augmentation can be quite beneficial, as demonstrated in Section 4.4.

3.2 Augmentation with Perturbation

3.2.1 Requirement for Perturbation

A difficulty that arises in the direct application of the methods described in Section 3.1 is evident if one considers how hand-crafted ECMs are typically created: They are usually constructed to provide low-frequency and, often, DC-accurate responses, using a relatively small number of poles. Obtaining accurate high-frequency responses generally requires more poles, as small reactances become sizable with respect to the low-frequency impedances observed. Yet, adding these poles with a view towards maximizing high-frequency accuracy inherently degrades low-frequency response, since the high-frequency poles affect all frequencies, their influence decaying as the inverse of the distance in the complex plane between the pole and any given frequency on the $j\omega$ axis. Hence, in order to obtain a wideband-accurate result, *low-frequency poles must be perturbed as high-frequency poles are added.* (Using the "latex sheet" model of the Laplace s-plane, it's clear that introducing a pole will generally modify the response along the $j\omega$ axis.) In fact, if the low-frequency accurate ECM is *not* perturbed during an augmentation, obtaining accurate results may require a high-order augmentation that first "cancels out" the low-frequency ECM poles (via pole-zero cancellation) before adding its own low-frequency poles back in, effectively turning CAM into a simple "black box" modeling routine! This result increases overall simulation time, and is contrary to the CAM philosophy.

ECM perturbation often translates into small component value changes on an ECM represented by a schematic; such perturbations will generally change the ECM's poles. Consider the simple scenario of a single-port network represented via admittance data with Q_{Meas} poles. The admittance function, $Y_{Meas}(s)$ can then be expressed as

$$Y_{Meas}(s) = \sum_{i=1}^{Q_{Meas}} \frac{k_i}{s - p_i} + d_{Meas} \tag{3.1}$$

The corresponding CAM model using a parallel admittance for augmentation is then

$$Y_{CAM}(s) = \underbrace{\sum_{i=1}^{Q_{ECM}} \frac{\tilde{k}_i}{s - \tilde{p}_i} + \tilde{d}}_{\text{ECM}} + \underbrace{\sum_{i=1}^{Q_{Aug}} \frac{\hat{k}_i}{s - \hat{p}_i} + \hat{d}}_{\text{Augmentation}}$$

We expect that $Q_{ECM} < Q_{Meas}$ (if this isn't the case, something is generally wrong with the ECM or the measurement data). The best fit of the CAM model to the measurement data occurs when $Q_{ECM} + Q_{Aug} = Q_{Meas}$, implying that the ECM poles *identically* match some of those in the measurement data. However, in most cases such an ECM would not be particularly desirable, since its low-frequency accuracy may be poor – a much likelier scenario is that the ECM has low-frequency poles that are *approximately* the same as those in an ideal model. As such, merely adding the augmentation poles, $\hat{p}_i$ to a low-frequency accurate ECM will disturb the overall response of the CAM model, reducing accuracy. The resolution to this difficulty is to perturb the component values of the ECM to slightly "shift" its poles $\tilde{p}$ into the same position as the measurement data's poles, p. Mathematically, this is

$$Y_{CAM}(s) = \underbrace{\sum_{i=1}^{Q_{ECM}} \frac{(\tilde{k}_i + \Delta\tilde{k}_i)}{s - (\tilde{p}_i + \Delta\tilde{p}_i)} + (\tilde{d} + \Delta\tilde{d})}_{\text{Perturbed ECM}} + \underbrace{\sum_{i=1}^{Q_{Aug}} \frac{\hat{k}_i}{s - \hat{p}_i} + \hat{d}}_{\text{Augmentation}}$$

Although demonstrated for a single-port example, the preceding discussion applies directly to the multi-port case represented by any network parameters. In such cases ECM component value perturbations generally affect all port responses, which constrains the useful range over which the values may be modified.

3.2.2 Automated Perturbational Algorithm

While early versions of the CAM software [36] simply swept over "reasonable" component parameter values in an attempt to optimize the CAM result, the current version utilizes a "general-purpose" global optimizer to intelligently search the solution space of possible ECM component values such that the perturbed ECM's poles $(\tilde{p}_i + \Delta\tilde{p}_i)$ are ideal, or at least optimal within the CAM framework; this significantly reduces the CPU time required to build the model relative to the swept approach, and was first detailed in [37]. While the only means of *guaranteeing* optimal component values is to perform an exhaustive search of the solution space, high-quality optimizers use a mix of analytical and heuristical techniques to return results that are statistically likely to be optimal or near-optimal while searching only a tiny fraction of the solution space [38, 39].

Most general-purpose optimizers require only the provision of a "cost function" or "metric," $f(\mathbf{x})$, and will "wiggle" the contents of the n-dimensional solution vector $\mathbf{x}$ within a lower- and upper-bounds $\mathbf{u}$ and $\mathbf{v}$ in order to minimize the cost function's value, where all quantities are real values. Mathematically stated, this is

$$\begin{aligned} \min \ f(\mathbf{x}) \quad &: \quad \mathbf{x} \in [\mathbf{u}, \mathbf{v}] \\ [\mathbf{u}, \mathbf{v}] \quad &: \quad \{\mathbf{x} \in \Re^n \mid u_i \leq x_i \leq v_i, \ i = 1 \ldots n\} \end{aligned}$$

CAM makes use of the MCS – **M**ulit-level **C**oordinate **S**earch – algorithm detailed in [38] and the DIRECT algorithm detailed in [39]. While quite complex, the basic idea is to examine the gradients of $f(\mathbf{x})$ (within the constraint area provided by $\mathbf{u}$ and $\mathbf{v}$) and "descend" along the steepest gradient to find a minima or maxima. This is similar to the Method of Lagrange Multipliers, discussed in E, with significant complexity added to try to avoid local extrema in favor of the globally optimal result while keeping the number of gradient sample points to a reasonable number for the sake of performance.

Within CAM, the solution vector $\mathbf{x}$ represents ECM component values. With a proposed solution $\mathbf{x}$, it is straightforward to re-compute the network parameter data associated with the ECM and then compute the cost function. CAM typically uses a cost function such as the root-mean-square (RMS) error between the CAM model and the measurement data, with the mean taken over the number of data points (frequencies) where measurement data are available. For instance, with N-port scattering parameters we have

$$f(\mathbf{x}) = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{M} \frac{W(i,j,k)}{M} \cdot \left| S_{i,j}^{Meas}(\omega_k) - S_{i,j}^{CAM}(\omega_k) \right|^2}$$

where $W(i,j,k)$ is an "appropriate" weighting function, which can be used, e.g., to give less importance to diagonal network elements (i.e., trade off reduced accuracy in return loss for greater accuracy in transmission coefficients), avoid "double counting" the diagonal entries (since the scattering parameter matrix of a reciprocal passive component with real port impedances will be symmetrical), etc. $W(i,j,k)$ can also be used to normalize the calculation: While X/Y/Z/G/H network parameters have values ranging between 0 and $\pm\infty$, S and T parameters using real port references (such as the ubiquitous 50Ω termination) are inherently bounded to $|S| \leq 1$ for all passive structures. Finally, changing $W(i,j,k)$ may be used to alter the statistical nature of the error vector $\mathbf{S}_{CAM} - \mathbf{S}_{Meas}$: Setting $W(i,j,k) = 1/\left|S_{i,j}^{Meas}(\omega_k)\right|$ tends to normalize the error *percentage* ($|\mathbf{S}_{CAM} - \mathbf{S}_{Meas}| \,/\, |\mathbf{S}_{Meas}|$), whereas leaving $W(i,j,k) = 1$ attempts to minimize the absolute error, $|\mathbf{S}_{CAM} - \mathbf{S}_{Meas}|$. This topic is closely linked to that of the weighting matrices used in least-squares fitting, which is discussed in C.3.

Fig. 3.7 shows how CAM interfaces to a general-purpose optimizer, including various inputs and outputs. Each time the optimizer proffers a value for $\mathbf{x}$, a *CAM Iteration* is run to compute the cost function; a flowchart of this process is shown in fig 3.8. CAM iterations generally include passivity checks and enforcements for non-passivity of the overall network, since the "optimal" non-passive network may have a significantly different ECM and augmentation than the (required) passive network. (Strictly speaking, these checks can be performed after optimization is finished. While faster, typically the results will not be passive and therefore the optimizer must be run again.) At the end of the optimizer's run, an ECM with specific component values has been combined with a specific topology augmentation; the result should closely match the measurement data.

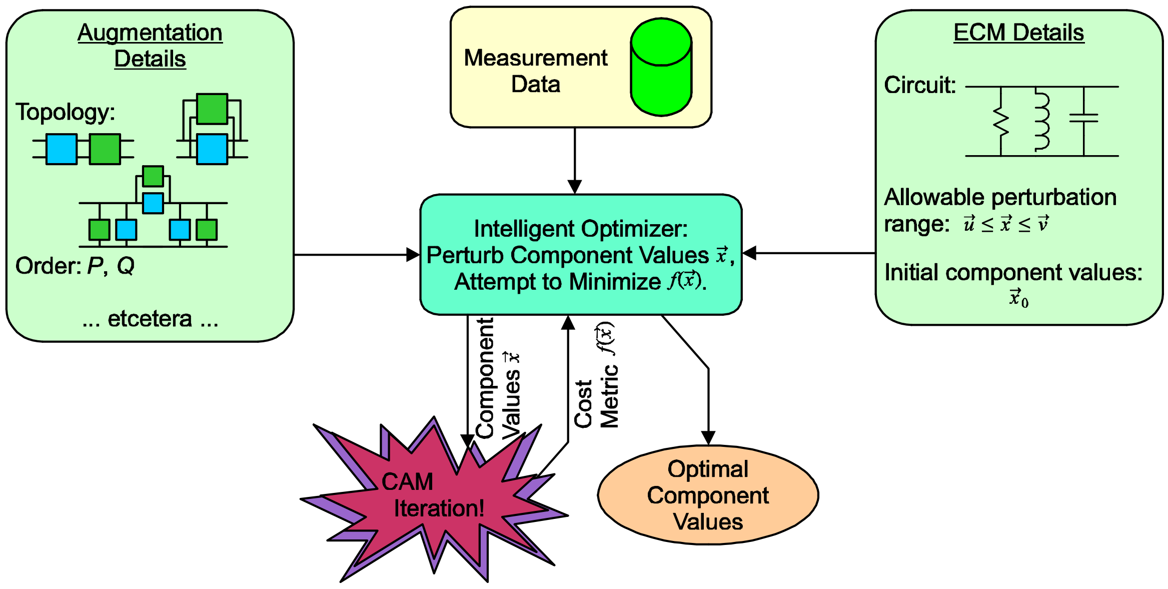

Augmentation Details
Topology:
Order: P, Q
... etcetera ...
Measurement Data
ECM Details
Circuit:
Allowable perturbation range: $\vec{u} \le \vec{x} \le \vec{v}$
Initial component values: $\vec{x}_0$
Intelligent Optimizer: Perturb Component Values $\vec{x}$, Attempt to Minimize $f(\vec{x})$.
Component Values $\vec{x}$
Cost Metric $f(\vec{x})$
CAM Iteration!
Optimal Component Values

Figure 3.7: Optimizer data flow in CAM. The general-purpose optimizer searches a small fraction of the solution space, attempting to find the ideal ECM/augmentation combination.

3.3 Data Fitting

A core operation within CAM is the "curve fitting" of a set of tabulated data to a rational function. As discussed in 2.2, such functions are usually expressed as either a ratio of polynomials, Eq. (2.1), or as a pole-residue summation, Eq. (2.2). CAM uses a "direct" least squares fitting algorithm or as vector fitting routine to perform this operation; this section provides further details about each approach.

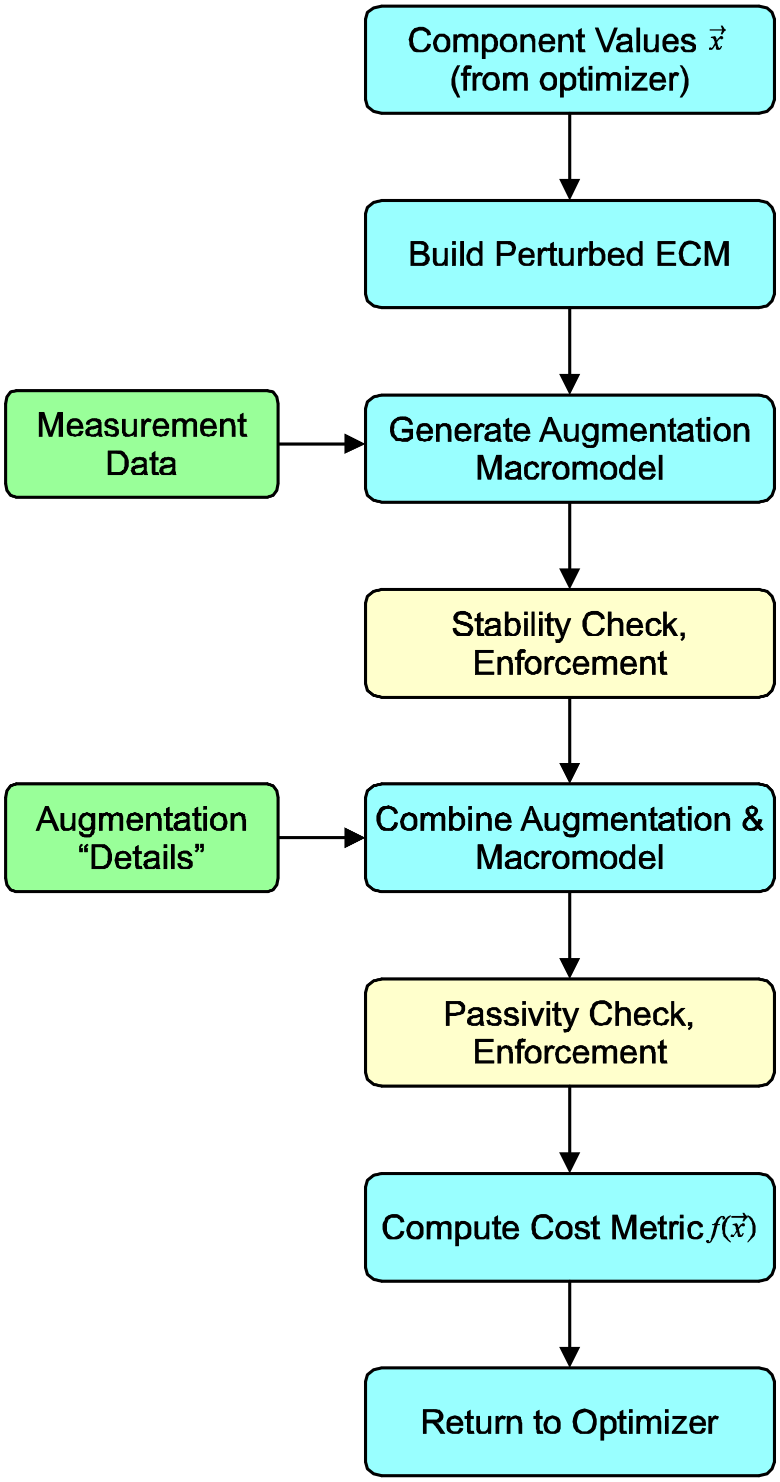

Component Values $\vec{x}$ (from optimizer)
Build Perturbed ECM
Measurement Data
Generate Augmentation Macromodel
Stability Check, Enforcement
Augmentation "Details"
Combine Augmentation & Macromodel
Passivity Check, Enforcement
Compute Cost Metric $f(\vec{x})$
Return to Optimizer

Figure 3.8: CAM iteration flowchart. An ECM using optimizer-provided component values $\mathbf{x}$ is augmented, passivity/stability are tested, and quality of result is computed, to be returned to the optimizer. Process steps in yellow are generally performed but may be optional in certain instances.

3.3.1 Direct Least Squares Fitting

Given a set of tabulated data $h(s)$, we seek the most "direct" approach to obtain the coefficients a_m and b_n in Eq. (2.1) repeated here:

$$h(s) = \frac{\sum\limits_{m=0}^{P} a_m s^m}{\sum\limits_{n=0}^{Q} b_n s^n} \tag{3.2}$$

For real-world problems, the data may be in the form of a vector $\mathbf{h}(s)$; least squares fitting can be applied directly to each element of the vector or to some "hybrid" whereby, e.g., the same poles are used for all vector elements. Furthermore, sometimes it is desirable to *weight* some data more than others if, e.g., high-frequency or low-frequency responses are more important than mid-band responses; this is addressed in further detail in Appendix C.3.

For the simple (scalar, unweighted) case considered herein, however, the problem begins by noting that Eq. (3.2) is non-linear due to the b_n's in the denominator, and is therefore difficult to solve directly. This can be fixed trivially by multiplying both sides by the denominator to obtain

$$\begin{aligned} \sum_{n=0}^{Q} b_n s^n \cdot h(s) &= \sum_{m=0}^{P} a_m s^m \\ h(s) \cdot \sum_{n=0}^{Q} b_n s^n - \sum_{m=0}^{P} a_m s^m &= 0 \end{aligned}$$

which is a standard linear equation. Using the available tabulated data at $s = j\omega_1 \ldots j\omega_K$, one can write a set of k simultaneous equations:

$$\begin{gathered} h(j\omega_1) \cdot (b_0 + b_1(j\omega_1)^1 + \cdots + b_Q(j\omega_1)^Q) - (a_0 + a_1(j\omega_1) + a_2(j\omega_1)^2 + \cdots + a_P(j\omega_1)^P) = 0 \\ \vdots \\ h(j\omega_K) \cdot (b_0 + b_1(j\omega_K)^1 + \cdots + b_Q(j\omega_K)^Q) - (a_0 + a_1(j\omega_K) + a_2(j\omega_K)^2 + \cdots + a_P(j\omega_K)^P) = 0 \end{gathered}$$

which can be written in matrix notation as $\mathbf{Mz} = \mathbf{0}$ where

$$\mathbf{M} \triangleq \left[\begin{array}{cccc|cccc} h(j\omega_1) & h(j\omega_1) \cdot (j\omega_1)^1 & \cdots & h(j\omega_1) \cdot (j\omega_1)^Q & -1 & -(j\omega_1)^1 & \cdots & -(j\omega_1)^P \\ h(j\omega_2) & h(j\omega_2) \cdot (j\omega_2)^1 & \cdots & h(j\omega_2) \cdot (j\omega_2)^Q & -1 & -(j\omega_2)^1 & \cdots & -(j\omega_2)^P \\ \vdots & & & \vdots & \vdots & & & \vdots \\ h(j\omega_K) & h(j\omega_K) \cdot (j\omega_K)^1 & \cdots & h(j\omega_K) \cdot (j\omega_K)^Q & -1 & -(j\omega_K)^1 & \cdots & -(j\omega_K)^P \end{array} \right] \tag{3.3}$$

and

$$\mathbf{z} \triangleq \begin{bmatrix} b_0 & b_1 & b_2 & \cdots & b_Q & a_0 & a_1 & a_2 & \cdots & a_P \end{bmatrix}^T$$

If this equation were solved directly, the coefficients returned would typically be complex when $h(s)$ is complex. Since physical circuits can only have poles and zeros that are real or occur in complex conjugate pairs, the a_n and b_n coefficients must be real (see Appendix D.2 for more information). An easy way to enforce this requirement is to split the matrix into two sub-matrices, the upper-half containing the real part of (3.3) and the lower-half containing the imaginary part. The new equation is

$$\begin{bmatrix} \Re(\mathbf{M}) \\ \Im(\mathbf{M}) \end{bmatrix} \mathbf{z} = \mathbf{0} \tag{3.4}$$

which can be seen to be in the standard "$\mathbf{Ax} = \mathbf{0}$" form. This equation is solved in a least-squares manner, beginning as described in Appendix C.1 by transforming Eq. (3.4) into

$$\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T \cdot \mathbf{0} = \mathbf{0}$$

At this point we'd like to solve the system of equations, but there's a problem: If $\mathbf{A}^T\mathbf{Ax} = \mathbf{0}$, then surely $\mathbf{x}$ is zero, isn't it? Indeed, zero is a solution, but this so-called "trivial solution" is *not* the one we're after. To have a non-trivial solution, $\mathbf{A}$ must be singular and – preferably – have a null space with rank one (i.e., $\mathbf{A}$ has nullity one) so that there is a single uniquely defined, non-zero solution vector $\mathbf{x}$. The means of obtaining $\mathbf{A}$ requires a bit of analysis – detailed in [40] – with the result being surprisingly simple: The best (in the standard least-squares sense that $|\mathbf{Ax}|$ is as close to zero as possible) non-zero solution $\mathbf{x}$ is the eigenvector corresponding to the minimum eigenvalue in the equation

$$\mathbf{A}^T\mathbf{Ax} = \lambda_{\min}\mathbf{x} \tag{3.5}$$

(Note that one coefficient such as b_0 in (3.2) may be set to a fixed constant such as 1, as mentioned in Section 2.2. In such a case, Eq. (3.4) has a non-zero right-hand side and, at least conceptually, may be solved without resorting to eigenvalue techniques.)

Unfortunately, the preceding technique for fitting data to a rational function is inherently weak, with the source of the problem being the increasing powers of s $(= j\omega)$ seen in (3.3). Intuitively, the problem is that the coefficients for the higher-order terms of the polynomial receive radically higher "weighting" in the system of equations that the lower-order terms, and therefore the mean values of the matrix's columns end up with enormous variations from column to column. (Since the problem shows up with the *columns*, there's no easy means of re-normalizing the matrix.) This sort of "power" variation creates a matrix similar in form to that of a Vandermonde matrix, which has a condition number[1] that's approximately proportional to the inverse of the Hilbert determinant[2] of order N, where N is the size of the matrix $\mathbf{A}$ [41]. For example, the Hilbert determinant for $N = 1$ is 1, whereas it's 4.6×10^{-4} for $N = 3$, 3.7×10^{-12} for $N = 5$, and 4.8×10^{-25} for $N = 7$. In other words, the system of equations is inherently ill-conditioned, and attempting to use the direct method to curve-fit rational functions with orders higher than 5-6 often leads to poor quality results.

Many techniques have been developed to overcome the limitations of this naive approach (although the "obvious" fix of changing the computation to use double-precision numbers is *not* considered a proper "solution!"). Various methods include the use of "quasi-orthogonal polynomials" [41], "total least squares" methods [42], and those based on Chebyshev polynomial basis functions [43]. In some cases the original problem is posed in terms of fitting a rational function to another function *and its derivatives* known at a single point (as opposed to knowing the function's value as multiple points); such approaches would require additional steps to first numerically compute the derivatives based on the tabulated data.

3.3.2 Vector Fitting

A relatively recent and robust method of rational function curve fitting is that of *vector fitting*, first introduced by Bjørn Gustavsen and Adam Semlyen in 1997 [44]. The "core" was improved and generalized over the next two years before further publication in 1999 [5]; since that time, Gustavsen has maintained the freely-available vector fitting code, making improvements, providing passivity enforcement routines, and fixing bugs. To date, several dozen papers reference the vector fitting code and it has been incorporated into commercial products.

Vector fitting eliminates the inherently ill-conditioned matrix obtained via direct least squares fitting by first formulating the rational function in the pole-residue form, Eq. (2.2):

$$h(s) = \sum_{i=1}^{Q} \frac{k_i}{s - p_i} + d + se \tag{3.6}$$

Here, Q is prescribed but the k_i's, p_i's, d and e are unknown. Vector fitting begins by *assuming* the various poles have values, $\bar{p}_i$, that are "reasonably" close to their actual values, p_i. The (largely heuristical) strategy for choosing these initial poles is based on whether $h(s)$ is relatively "smooth" or has multiple resonances: For smooth functions, real poles spaced linearly or logarithmically over the σ axis of the S plane are used (where $s = \sigma + j\omega$ as usual), extending over the same values as the frequency range to be modeled by $h(s)$ (e.g., if $h(s)$ is to be modeled between 1Hz and 100kHz, one might set $\bar{p}_1 = -1 \cdot 2\pi$ (rad/s), $\bar{p}_Q = -100 \cdot 10^3 \cdot 2\pi$ (rad/s), and distribute the remaining poles

[1]See Appenfix C.2 for more information on condition numbers.
[2]See Appendix C.5 for more information on Hilbert matrices.

in-between). For functions with resonances, complex pole pairs are placed such that the imaginary part of the poles covers the frequency range of interest whereas the real part is set to the imaginary part divided by 100. For the preceding example, the initial poles would be $\bar{p}_1 = 2\pi(-1/100 + j1)$, $\bar{p}_2 = 2\pi(-1/100 - j1)$, $\bar{p}_{Q-1} = 2\pi(-10^3 + j100 \cdot 10^3)$, and $\bar{p}_Q = 2\pi(-10^3 - j100 \cdot 10^3)$ (all rad/s).

Once the starting poles are chosen, a "weighting" or "scaling" function, $w(s)$ is specified. It is expressed as a rational function in pole-residue form with unknown residues $\bar{k}_i$ but the *same* (known) poles $\bar{p}_i$ as assumed for $h(s)$. That is,

$$w(s) = \sum_{i=1}^{Q} \frac{\bar{k}_i}{s - \bar{p}_i} + 1$$

with the constant "1" introduced to avoid the ambiguity in scaling factors, as discussed in Section 2.2. Next, the product of the weighting function and $h(s)$ is set equal to a third function, $c(s)$, which is constructed in pole-residue form again using the same assumed poles $\bar{p}_i$ as for $h(s)$ and $w(s)$:

$$\begin{aligned} w(s)h(s) &= c(s) & (3.7) \\ \left(\sum_{i=1}^{Q} \frac{\bar{k}_i}{s - \bar{p}_i} + 1\right) \cdot h(s) &= \sum_{i=1}^{Q} \frac{\tilde{k}_i}{s - \bar{p}_i} + \tilde{d} + s\tilde{e} & (3.8) \end{aligned}$$

Notice that this is a linear problem in all unknowns ($\bar{k}_i$, $\tilde{k}_i$, $\tilde{d}$, and $\tilde{e}$). Writing Eq. (3.8) at several frequency points creates an overdetermined system of equations in the standard $\mathbf{Ax} = \mathbf{b}$ form; this is solved using standard least squares fitting as described in Appendix C.1, or by more robust techniques such as those based on singular value decomposition.

At this point we've gone through several machinations with seemingly little gain, as we've yet to ascertain the poles and residues or Eq. (3.6). Deliverance is near, however, as seen by first writing out $c(s)$ and $w(s)$ in pole-zero product forms:

$$\begin{aligned} w(s) &= \sum_{i=1}^{Q} \frac{\bar{k}_i}{s - \bar{p}_i} + 1 = \prod_{i=1}^{Q} \frac{s - \bar{z}_i}{s - \bar{p}_i} \\ c(s) &= \sum_{i=1}^{Q} \frac{\tilde{k}_i}{s - \bar{p}_i} + \tilde{d} + s\tilde{e} = \tilde{e} \cdot \prod_{i=1}^{Q} \frac{s - \tilde{z}_i}{s - \bar{p}_i} \end{aligned}$$

Note that $\bar{z}_i$ and $\tilde{z}_i$ are implicit once $\bar{k}_i$ and $\tilde{k}_i$ are obtained from solving Eq. (3.8). Next, observe from Eq. (3.7) that $h(s) = c(s)/w(s)$ which is

$$h(s) = \frac{c(s)}{w(s)} = \frac{\tilde{e} \cdot \prod_{i=1}^{Q} \frac{s-\tilde{z}_i}{s-\bar{p}_i}}{\prod_{i=1}^{Q} \frac{s-\bar{z}_i}{s-\bar{p}_i}} = \tilde{e} \cdot \prod_{i=1}^{Q} \frac{(s-\tilde{z}_i)}{(s-\bar{p}_i)} \cdot \frac{(s-\bar{p}_i)}{(s-\bar{z}_i)} = \tilde{e} \cdot \prod_{i=1}^{Q} \frac{(s-\tilde{z}_i)}{(s-\bar{z}_i)} \tag{3.9}$$

Comparing Eq. (3.9) to Eq. (3.6) shows that *the zeros of $w(s)$ are the poles of $h(s)$*! Additionally, since the assumed poles $\bar{p}_i$ cancel out, the actual values chosen for them impacts only the "degree of conditioning" when solving (3.8) and not (directly) the solution itself. In brief, we've solved the non-linear problem of obtaining $h(s)$'s poles by instead solving the linear problem in Eq. (3.8). While the problem is solved using least squares methods, the solution can be used in an iterative process whereby the first "solution" poles are used as the second "assumed" poles, etc. This technique is similar to that of Santhanan-Koerner iteration, using partial fractions as basis functions rather than polynomials [45].

There are various details of the technique that have been glossed over or outright ignored: Determining the zeros, $\tilde{z}_i$, from Eq. (3.8) is done by solving an eigenvalue problem, taking into account the requirement that the poles of $h(s)$ need to be complex conjugates. Secondly, once the poles of $h(s)$ are found, the residues of Eq. (3.6) still need to be computed. While this could be done directly using Eq. (3.9), a more accurate approach is to substitute the new poles of $h(s)$ into Eq. (3.6) and directly solve for the residues again using standard least squares fitting. Finally, as the name implies, "vector fitting" applies to vector data just as readily as it does with scalar data using the derivation shown; this is a straightforward extension implemented by changing $h(s)$ to $\mathbf{h}(s)$ in Eq. (3.6) and proceeding as before. More details on these procedures can be found in the primary reference, [5].

Chapter 4 – Results

4.1 Lumped Element Test Circuit

To illustrate the CAM concept – as well as to debug the CAM code! – a test circuit was created to generate "measurement data," after which component changes were made to test CAM's ability to recreate the original circuit. The test circuit is shown in Fig. 4.1. The modifications to the test circuit to create an equivalent circuit model are the removal of the upper branches of the test circuit along with significant changes in the component values; this is shown in Fig. 4.2. Notice that the lower half of the network forms a low-pass structure whereas the upper-half forms a high-pass structure. Given the nature of these modifications, the obvious topology for augmentation is the "network parallel" style, shown in Fig. 4.3. With this setup, it is expected that CAM should be able to regenerate the original test circuit's measurements with near-perfection.

Fig. 4.4 shows S_{11} and S_{22} magnitude plots for the ideal circuit, the ECM, and the ECM with a 3rd-order parallel network augmentation (*without* component value perturbation). The RMS error for this augmented network is approximately one-sixth the error of the original ECM. If perturbation *is* performed – specifically, on the two capacitors and two inductors modified from the ideal test circuit's schematic, the results improve dramatically, as expected. Table 4.1 shows the perturbed values of the four perturbed circuit elements, demonstrating that they have been effectively transformed back to their original values. Finally, Fig. 4.5 compares the scattering parameters of the ideal circuit with those of the augmented *and* perturbed ECM; the plots are virtually indistinguishable.

Component	**ECM Value**	**Perturbed Value**	**Ideal Value**
C1	290fF	244.9fF	245fF
C2	390fF	419.9fF	420fF
L1	390pH	360.2pH	360pH
L2	970pH	1.100nH	1.1nH

Table 4.1: Test Circuit Perturbation Results

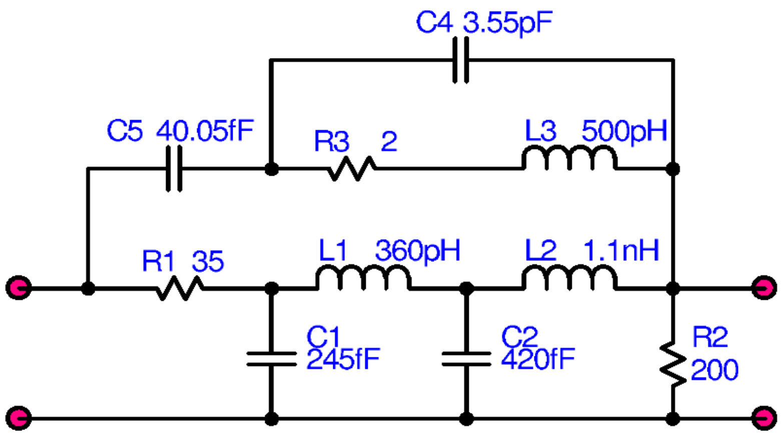


Figure 4.1: CAM test circuit for generation of "measurement" data.

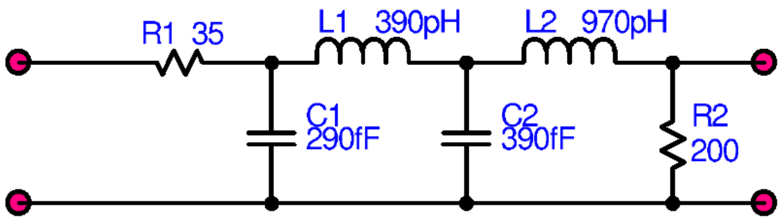


Figure 4.2: CAM test circuit with upper branches removed and component values modified; this is the ECM.

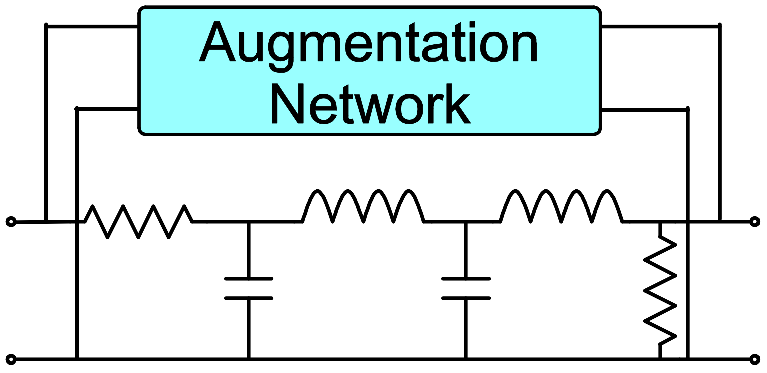
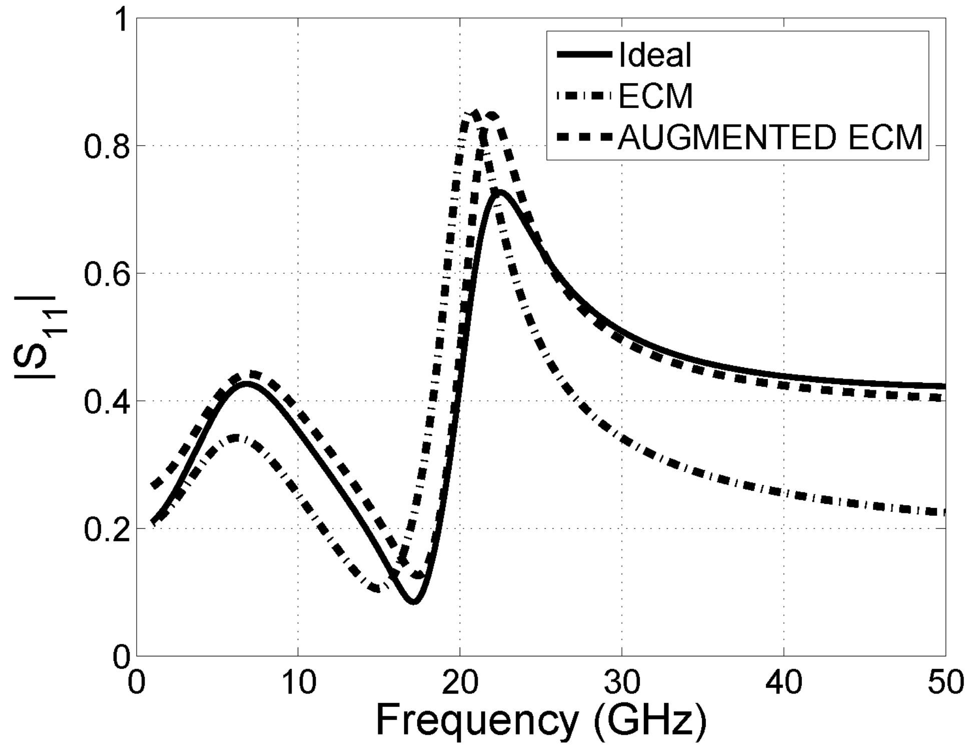


Figure 4.3: ECM showing placement of augmentation network.



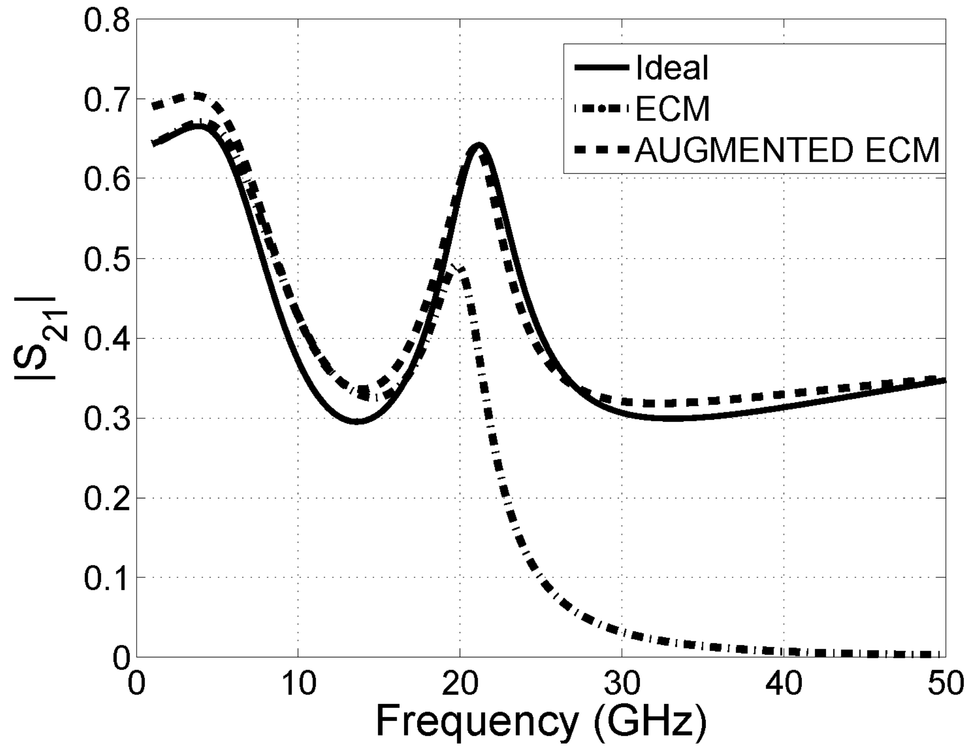
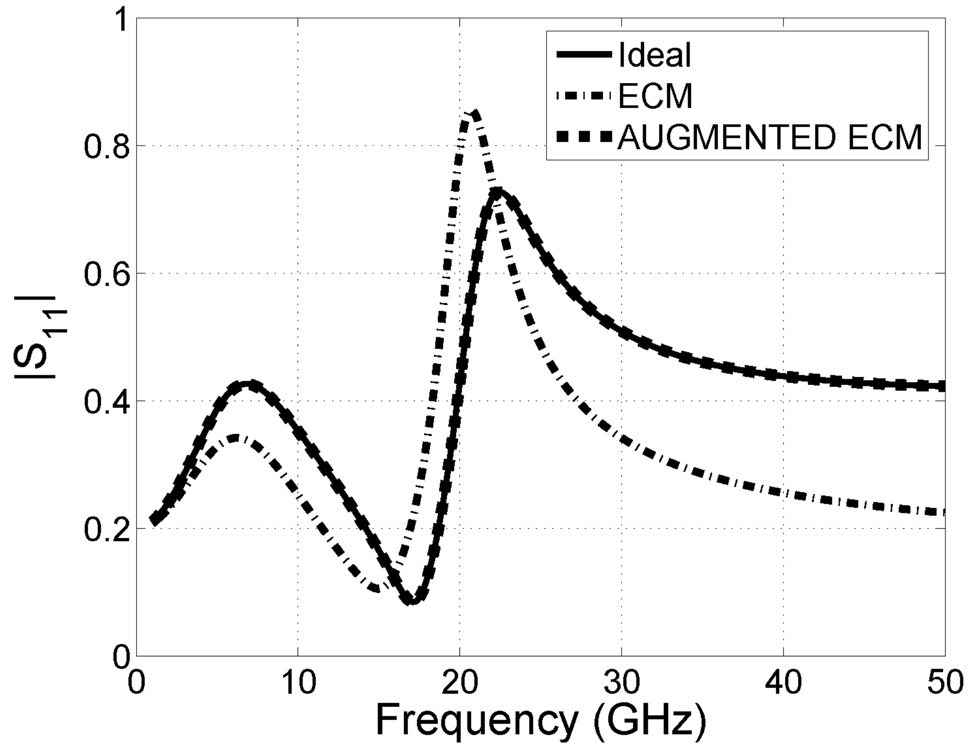


Figure 4.4: Scattering parameter comparison between original ("ideal") test circuit, ECM, and augmented ECM.



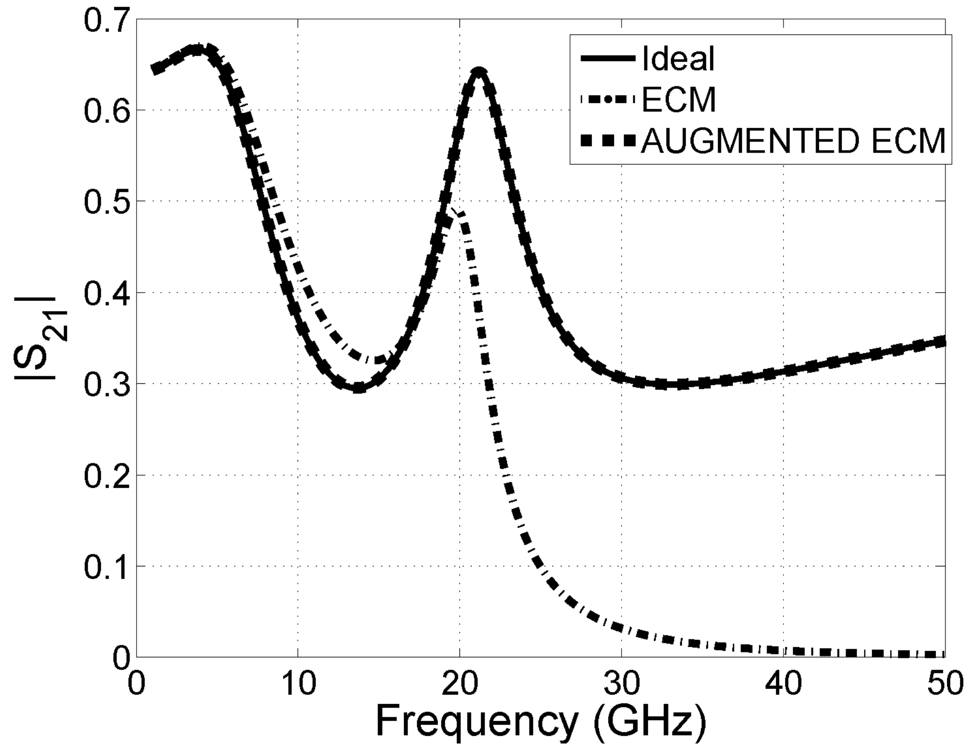


Figure 4.5: Scattering parameter comparison between original ("ideal") test circuit, ECM, and ECM with perturbation *and* augmentation.

4.2 CMOS Spiral Inductor

Reliably producing high-Q inductors within an integrated circuit fabrication process is quite desirable from the point of view of minimizing the cost of RF power amplifiers, mixers, and other ICs that typically require low-loss matching or filtering networks. Producing such inductors has proven difficult given the lossy substrates and (typically) moderate-conductance metal layers available in most IC processes [46,47]. While on-chip inductors are commercially viable and produced in quantity today, their quality is still far below that available from discrete components, and this drawback has emphasized the need for accurate equivalent circuit models.

CAM provides a means of generating accurate ECMs while retaining the traditional model used for inductors. A typical on-chip spiral inductor as well as a low-frequency ECM is show in Fig. 4.6. While this model has intuitive appeal, its high-frequency behavior diverges significantly from inductor's actual performance, and – at least for on-chip inductors with lossy silicon substrates – is only applicable over narrow bands. Part of the inaccuracy rises from a lack of modeling several significant magnetic-field loss effects, as detailed in [48]. Additionally, parts of the model itself are somewhat specious, or at least misnomers: the capacitor C_C, generally known as "the interwinding capacitance," is *not* a significant function of interwinding spacing [49]. Better models can be obtained by using frequency dependent loss in the series branch of Fig. 4.6, adding multiple series sections as in [46] to better approximate the distributed nature of the structure, or moving directly to a transmission line-based model as in [49].

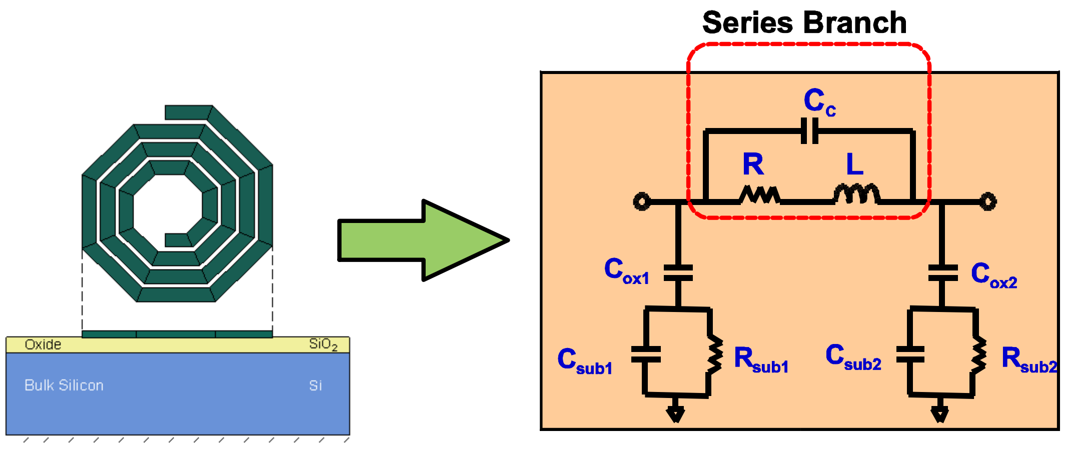


Figure 4.6: Illustration of an on-chip spiral inductor structure and the associated ECM. The series branch is highlighted.

Fig. 4.6 highlights the series branch of the spiral inductor ECM, which is regarded as the most difficult to model [46]; this is the branch that CAM will model. From Y parameters, the series branch impedance (shown on the figure) can be immediately calculated as described in Section 3.1.1. For a particular spiral inductor, a standard "least-squares" curve fit generates "optimal" series branch component values of R=1.527Ω, L=1.301nH, and C=1.896fF; these values are used for the ECM. Next, augmentation alone, using one zero and two poles, is applied; the results are quite poor as seen in Fig. 4.7. Allowing ECM component value perturbation generates a model with acceptable accuracy, as seen in Fig. 4.7; the perturbed component values are R=1.238Ω, L=1.349nH, and C=2.280fF.

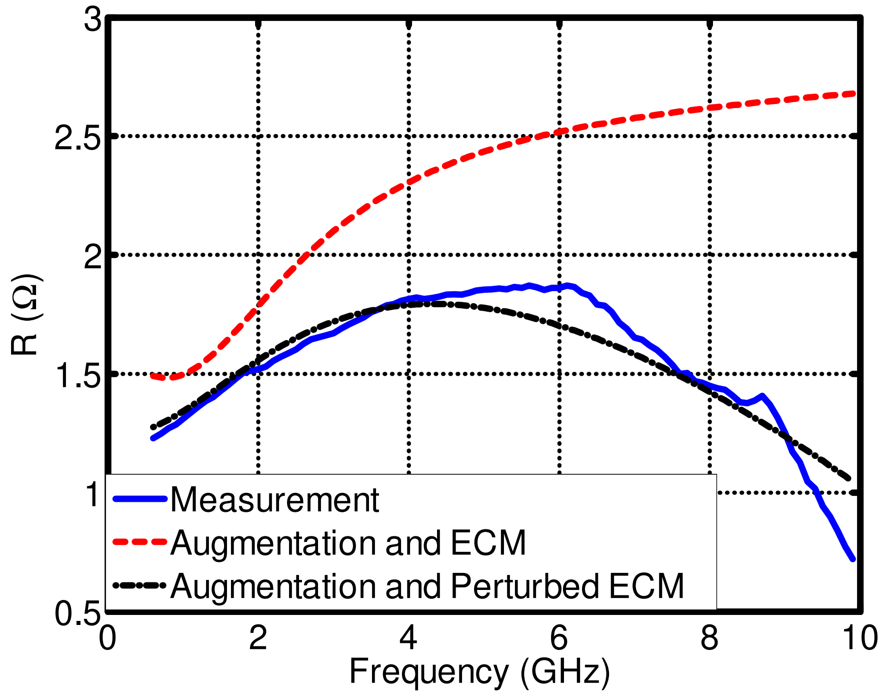
This spiral inductor example was utilized during the early stages of CAM research and the resultant overall model was not checked for passivity. In retrospect, there is reason to believe that the augmentation is *not* passive, but the example nevertheless illustrates the CAM concept nicely. This example drove the need for the use of a *global* optimizer during the perturbation stage of CAM: Examination of mean-square errors (using S parameters) between the perturbed and augmented ECM relative to just the ECM itself, as a function of the R, L, and C component values, showed numerous local extrema that non-global optimizers would converge to. A further challenge for a global optimizer lies in the observation that some local extrema had extremely large gradients. As one may infer, finding the "ideal" component values is quite challenging.

4.3 Probe Tip Hybrid Matching Network

While manufacturers of the day's fastest CPU or highest bandwidth wireless networking card get all the press for pushing the limits of technology, the lowly test-equipment manufacturer is a prime participant in making such designs a reality. As soon as the venture capitalist's check clears, one of the first things a company asked to build a GHz-range device does is to go out and spend a five- or six-figure sum on multi-GHz oscilloscopes, network analyzers, or similar test equipment. A sometimes-overlooked component of such test equipment is the importance of the *probe* used to gather high-speed signals, which contributes significantly to test equipment's overall behavior, including the limits of its high-frequency performance[1]. A model of part of a high-performance (multi-GHz) oscilloscope probe "tip hybrid" from Tektronix is shown in Fig. 4.8. The overall hybrid circuit consists of a small buffer amplifier connected through 125Ω coax at port one that drives an oscilloscope input; the portion of the circuit shown in the figure consists of a resistor/capacitor matching network between the port 1 output and the port 2 input, which is the input of the probe itself connected only to, e.g., a needle point for ease of probing.

An initial ECM designed by an experienced modeling engineer for the probe's matching network is shown in Fig. 4.9; ideal transmission lines have been used to model pure delays. One of the strengths

[1]One ironic limitation of better technology is that manufacturers successfully building multi-GHz oscilloscopes sometimes have to *artificially* limit the bandwidth of their designs targeted at the entry-level (<=1GHz) market segments!



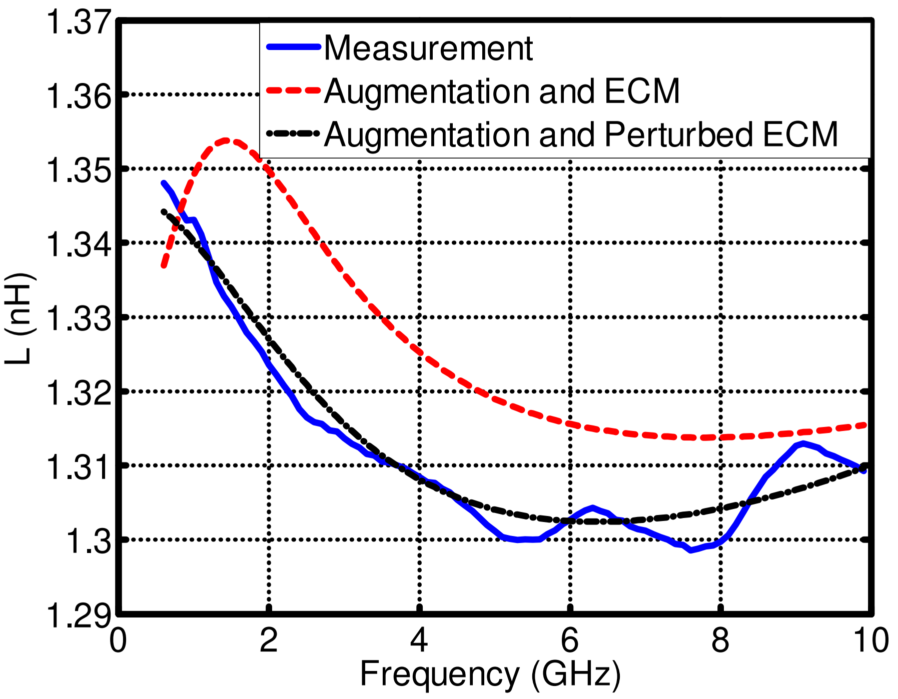


Figure 4.7: On-chip spiral inductor results using augmentation (only) vs. augmentation with perturbation.

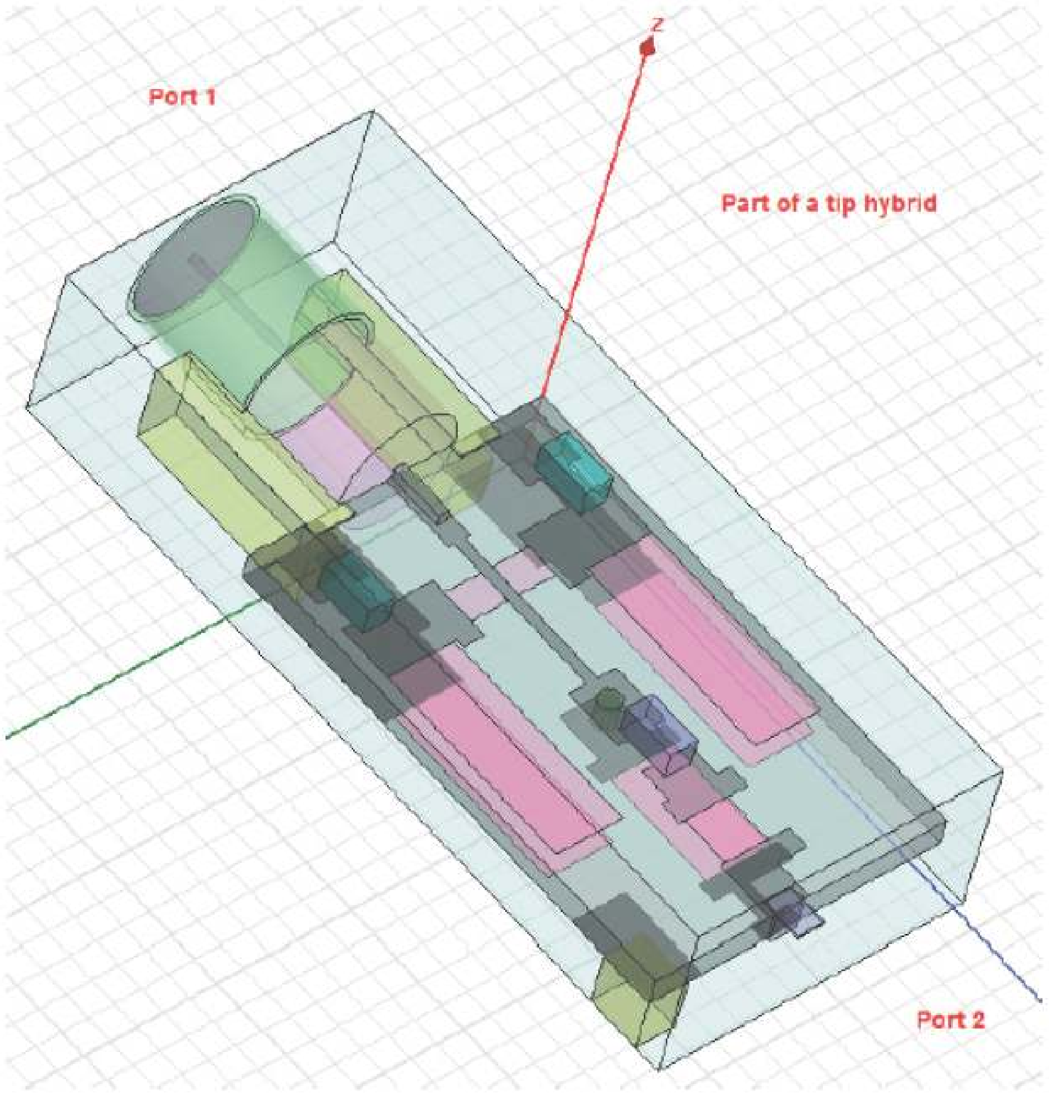


Figure 4.8: Physical CAD model for Tektronix oscilloscope probe tip matching network.

of CAM is the ability to freely mix lumped elements with distributed elements during augmentation and perturbation. The "measurement data" for the matching network was obtained from a 3D full-wave electromagnetic simulation, performed using the finite-element based solver HFSS [50]. As shown in Fig. 4.10, the initial ECM doesn't match the measurement data particularly well, especially at higher frequencies. Augmentation, using "network parallel" topology with six poles and zeros, provides a much better match, but perturbation provides an even better improvement. (In this case, component values as well as transmission line lengths (delays) and characteristic impedances were allowed to be perturbed a maximum of $\pm 50\%$.) Fig. 4.11 plots the same data on a Smith chart to emphasize the quality of the results in both magnitude *and* phase, since in measurement applications small phase errors are often more important than small magnitude errors, and generally more difficult to correct. The results are quantified in Table 4.2. As seen, the augmented ECM decreases the RMS error relative to the simulation ("measurement") data by a factor of 30, and perturbation provides an additional factor of approximately 5.

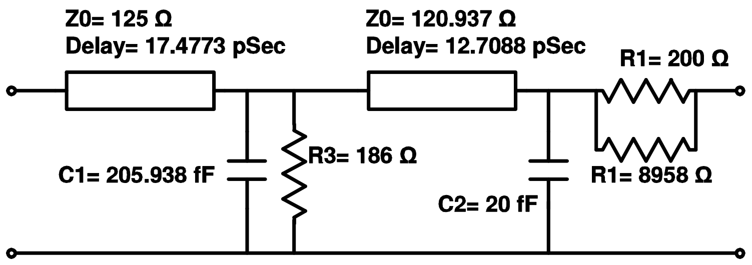


Figure 4.9: Hand-engineered equivalent circuit model for the probe tip hybrid's input matching network.

Circuit	**RMS Error** (Relative to Measurement Data)
ECM	$257.6 \cdot 10^{-3}$
Augmented ECM	$8.4 \cdot 10^{-3}$
Augmented and Perturbed ECM	$1.7 \cdot 10^{-3}$

Table 4.2: RMS error vs. model type for tip matching network

4.4 Distributed and Lossy Test Circuit

As already demonstrated, CAM is useful for modeling circuits containing distributed elements. While time-domain simulation of lossless (ideal) transmission lines is straightforward to implement, significant care is required to correctly simulate lossy lines while still maintaining high simulation

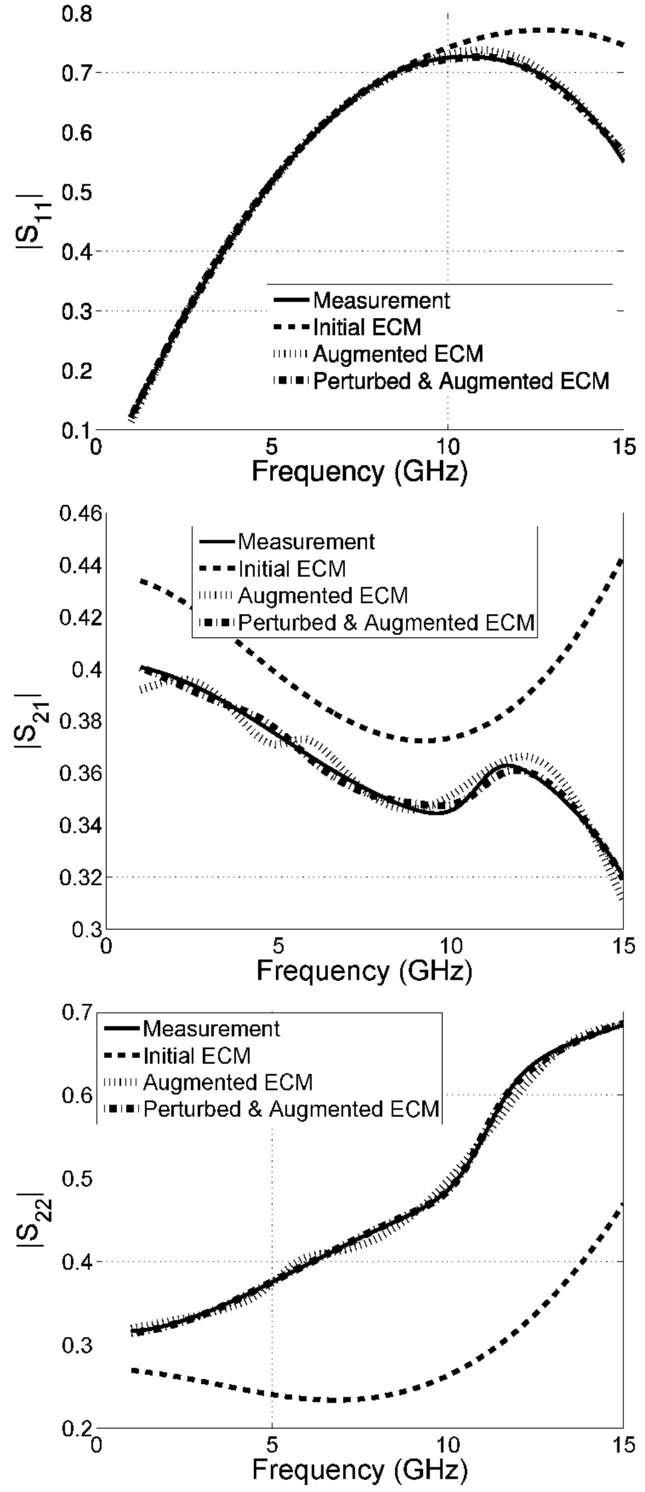


Figure 4.10: Comparison of the probe tip matching network's measurement data to CAM data (this is a reciprocal network, so $S_{12} = S_{21}$).

.

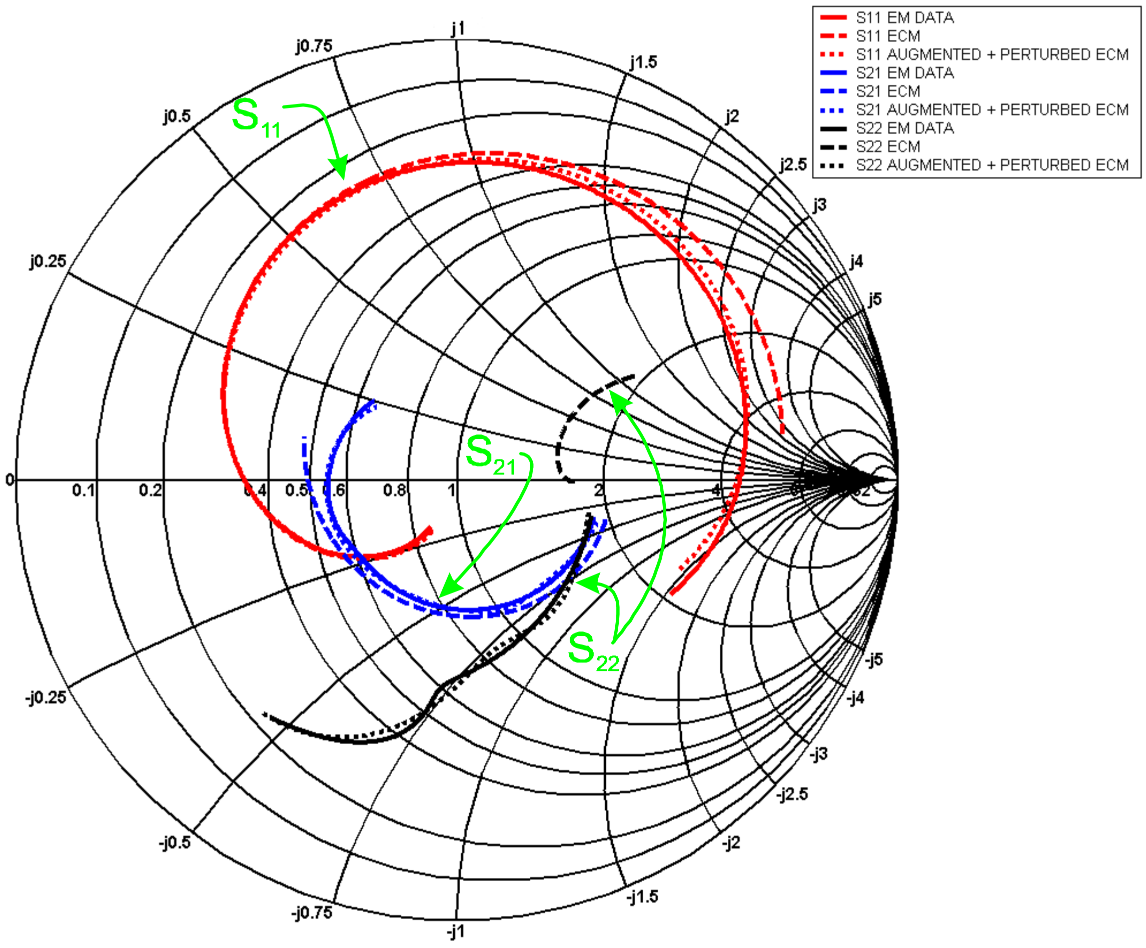


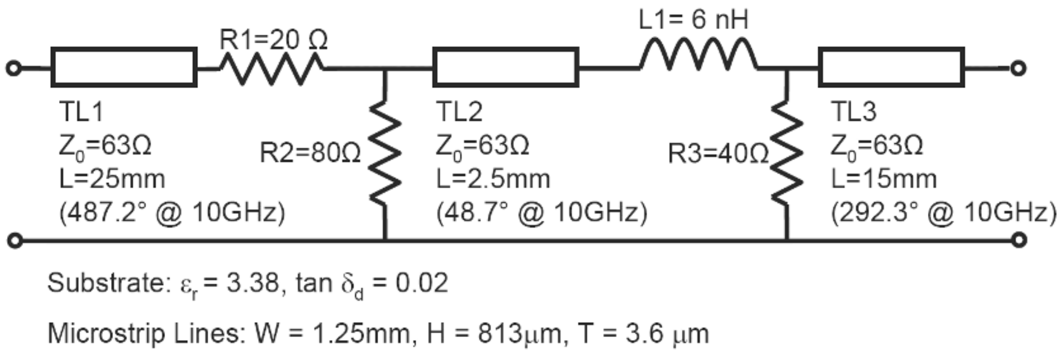
Figure 4.11: Probe tip matching network results, indicating excellent magnitude *and* phase obtained with CAM.

efficiency [51]; the difficulty is heightened when the loss is significantly frequency dependent, as in the case of microstrip transmission lines patterned on many inexpensive substrates. One means of side-stepping the difficulty is to use a pair of ideal transmission lines on the input and output of a network and use CAM to model the "inner" circuit *as well as the loss of the transmission lines.* Consider the test circuit shown in Fig. 4.12(a): The microstrip transmission lines have significant electrical length (especially the outermost lines) at 10GHz where the circuit is meant to be used. The CAM approach here is slightly different than those detailed so far: As illustrated in Fig. 4.12(b), we replace TL1 and TL3 with ideal transmission lines, and use a rational function "macromodel" intended to capture *all* other circuit behavior seen in Fig. 4.12(a). In essence, the ideal transmission lines have become the ECM.

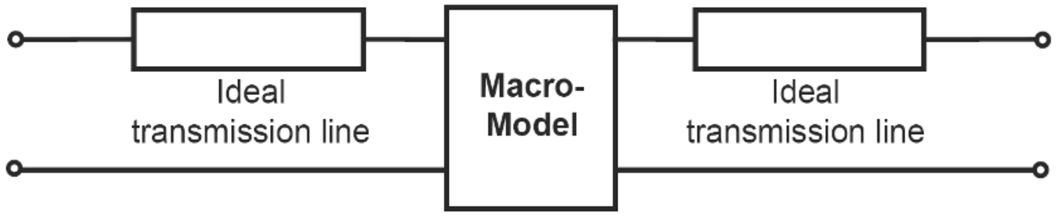
The ideal transmission lines in Fig. 4.12(b) are given the same physical length and characteristic impedance as those in Fig. 4.12(a); their dielectric constant is set to microstrip line's low-frequency effective dielectric constant. The characteristic impedances and physical lengths are allowed to be perturbed while the macromodel augments the ECM using a 4th-order rational function. Fig. 4.13 shows the results, comparing a 22nd-order "direct" curve fit to that obtained via CAM – the CAM result is quite comparable, arguably even a skosh superior from the viewpoint of having a more evenly distributed error function. Table 4.3 summarizes these results, including a comparison with a 21st-order direct fit which is significantly worse than the 4th-order CAM model. One final benefit of the CAM approach is the ability to largely ignore numerical round-off and ill-conditioning issues given the low order of the augmentation, whereas attempting to increase the number of poles used in a direct fit did not result in significantly better matches, due to such difficulties.

Methodology	**TL1** Z_0	**TL1 Length**	**TL3** Z_0	**TL3 Length**	**RMS Error** (Vs. Fig. 4.12(a))
21^{st}-order Direct Fit	63Ω	25mm	63Ω	15mm	$291 \cdot 10^{-3}$
22^{nd}-order Direct Fit	63Ω	25mm	63Ω	15mm	$6.55 \cdot 10^{-3}$
4^{th}-order CAM	63.33Ω	28.47mm	63.24Ω	12.73mm	$7.85 \cdot 10^{-3}$

Table 4.3: Distributed, lossy test circuit results as a function of curve fitting methodology

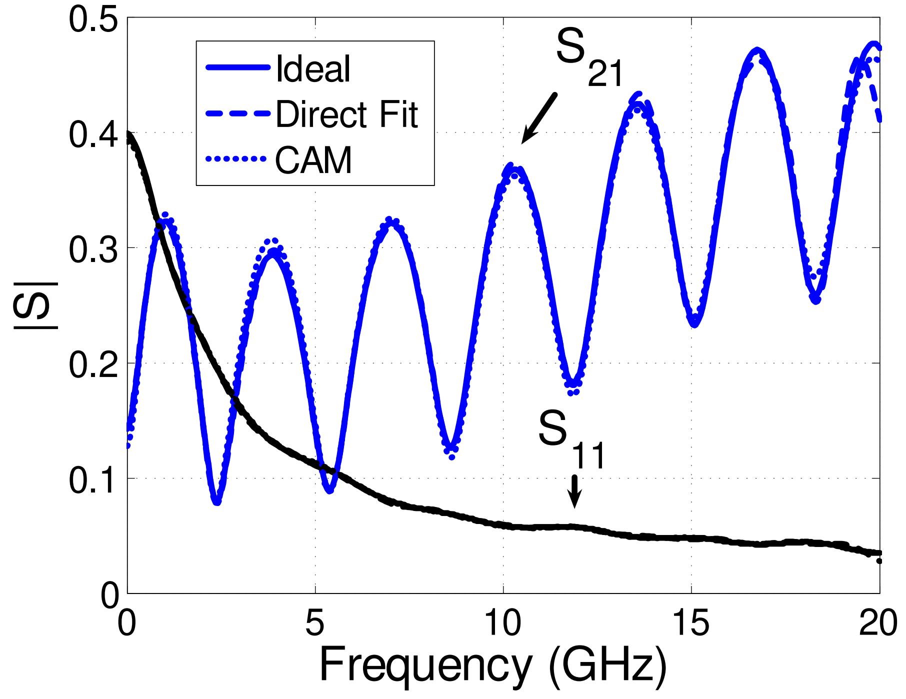


(a)

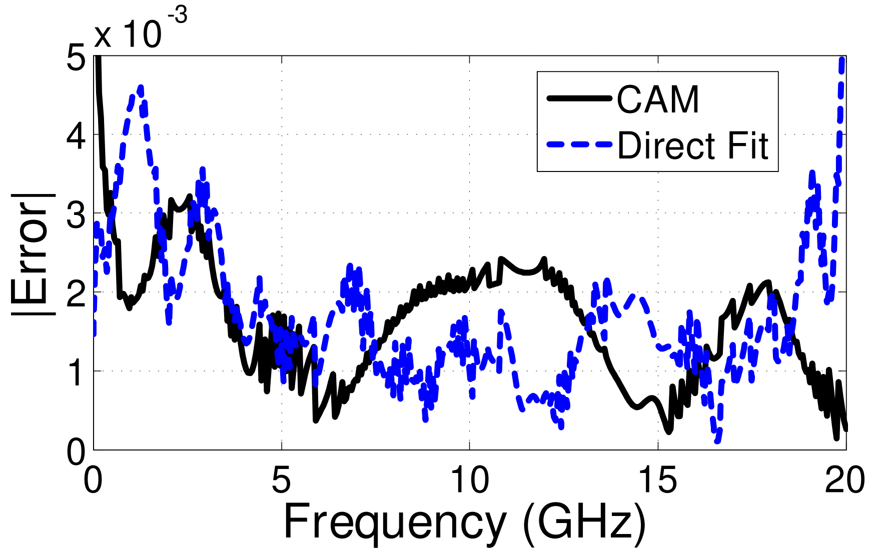


(b)

Figure 4.12: Distributed, lossy test circuit: (a) Test circuit containing distributed, lossy elements, represented by ideal transmissions lines shunted by loss components. (b) CAM model toplogy.



(a)



(b)

Figure 4.13: Distributed, lossy test circuit: (a) Comparison between measurement data and high-order direct fit vs. 4th-order augmented/perturbed CAM model. (b) Comparison of error magnitude of S_{21}.

Chapter 5 – A Guided Tour of the MATLAB CAM Software

CAM research was largely performed using MATLAB software to test out and validate the underlying concepts using data files provided by National Semiconductor and Tektronix. This software slowly evolved into a menu-driven suite of procedures and function calls that – after re-factoring – has become powerful and readily extensible by others (for instance, a GUI "wrapper" is now available; contact the OSU Microwaves Group for more information on this option). In this section, a quick "walkthrough" of the CAM software is provided, demonstrating its utility and some of its finer points.

When CAM is first executed (either by double-clicking on the compiled executable or by running `cam.m` from the MATLAB command line), a splash screen is shown followed by the main menu:

```
******************************************************************************
*                                                                            *
* Oregon State University Microwaves Group Circuit Augmentation Modeling Tool *
* Version 0.9 beta     Contact: andreas@eecs.oregonstate.edu   (541) 737-3153 *
*                                                                            *
******************************************************************************

Enter bracketed text to change parameters or execute command:

Measurement Data
================
[M]  File name: (None -- this must be specified prior to modeling)
[F]  Data filtering: Skip 0 points, use 0 point median filter, use 0
     point moving average filter
   ** Data interpolated to no more than 100 data points.

Equivalent Circuit Model
========================
[E]  File name: (None -- if left unspecified, pure 'black box' synthesis
       will be used)

Curves/Augmentation/Perturbation Setup
```

```
======================================
[O]  Order of numerator/denominator curve fitting polynomials: 3/3


[C]  Curve fitting routine: (None -- this must be specified prior to modeling)


[A]  Augmentation topology: (None -- this must be specified prior to modeling)


[P]  Perturbation optimizer: (None -- no ECM perturbation will be performed)

Control
=======
[GO] Create model!

Other Functions
===============
[PL] Configure data plotters

[S]  Save modeling parameters to a file
[L]  Load modeling parameters from a file
[H]  Help/About
[X]  Exit

Enter command:
```

For the sake of simplicity, we'll perform a "black-box" fit to a measurement data file. (That is, there will be no ECM.) To begin, menu option "M" ("Measurement Data") is selected, providing a standard file requestor to specify the measurement data file, which is named "50res.s2p". Next a "plotter" – a graph of a particular set of parameters relating to the data provided and generated – is added with the "PL" ("Plotters") command; we request that S_{11}, S_{21}, and S_{22} are shown:

```
Plotting configuration:

(No plotters are currently defined...)

Commands
========
[A] Add new plotter
```

```
[E] Edit existing plotter
[D] Delete existing plotter
[X] Return to main menu

Enter command: a

Available plotters:

 1. S Parameters (linear)
 2. S Parameters (dB)
 3. Y Parameters
 4. Z Parameters
 5. Smith Chart S Parameters

Select plotter: 1

Enter plot title: CAM Demonstration S Parameters

Expected number of ports in data (hit return to use default of 2):

Select responses to plot; use Y/N, T/F or 1/0:

 S11:Y S12:N
 S21:Y S22:Y
```

After exiting this sub-menu, we see the original data plotted in the form requested; this is shown in Fig. 5.1. Now back at the main menu, we configure the "augmentation" (which is, for this demo, the complete black box fit) to use a rational function with a fourth-order numerator and fifth-order denominator using the "O" ("Order") command. We also select the augmentation topology to be "Network Parallel" with the "A" ("Augmentation Topology") command (although for a black-box fit the option chosen doesn't really matter), and the curve fitting routine to be vector fitting with the "C" ("Curve Fitting Routine") command. We then use the "CS" ("Curve Fitter Settings") command to indicate that our data is relatively smooth and that four vector fitting iterations should be performed (the "CS" commands are specific to the type of curve fitting routine chosen). Finally, we issue the "go" (Go!) command to execute the black-box synthesis algorithm. In short order, the results are displayed, as shown in Fig. 5.2. S_{11} is a very good match, whereas S_{21} is a little

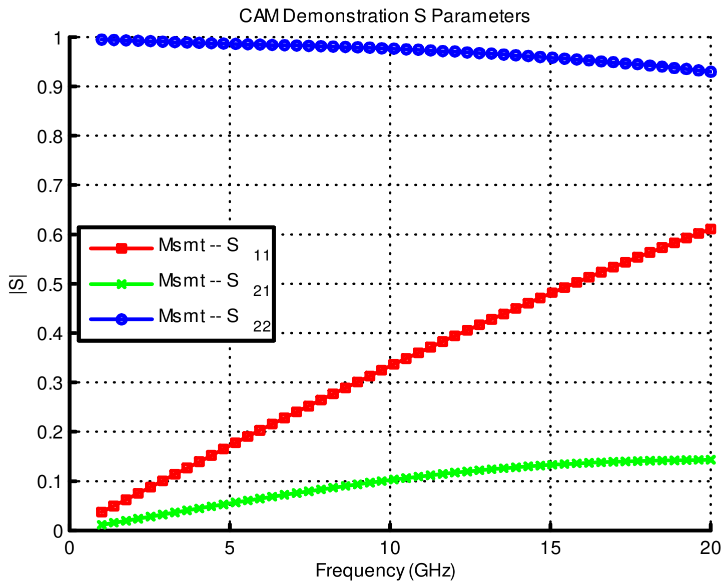


Figure 5.1: S parameter "measurement" ("Msmt") data for CAM software demonstration.

off at higher frequencies and S_{22} contains signification variations relative to the measurement data throughout the measurement frequencies.

After fitting, the CAM software's main menu displays the mean-square error between the CAM model (the block box fit, in this case) and the measurement data; 0.051 in this case. Increasing the augmentation order to use a 5th order numerator reduces this to $3.164 \cdot 10^{-3}$ and visibly improves the appearance of the fit. Adding a Smith chart plot confirms this, as shown Fig. 5.3. Finally, the CAM software can output a SPICE netlist corresponding to the CAM model with the "SM" ("Save Model") command; the augmentation is implemented using Laplace-type controlled sources that correspond to the rational function calculated for the augmentation. This rational function and the SPICE augmentation model can be displayed with the "AR" ("Augmentation Results") command:

```
Augmentation Results -- Network Parallel Topology

Interpolating polynomials in Y:

     0.01906*s^5+7.02e+009*s^4+2.477e+021*s^3+(2.212e+032+j1.252e+016)*s^2+ ...
Y11: ---------------------------------------------------------------------------
```

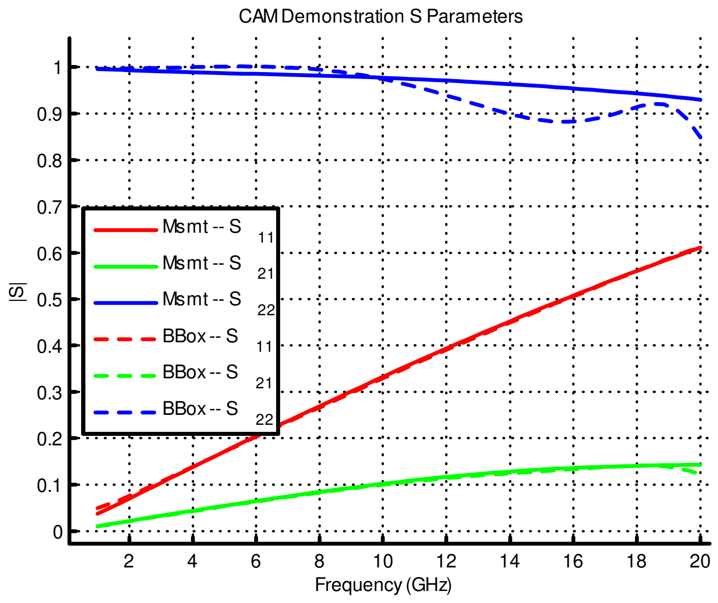


Figure 5.2: Black box fit ("BBox") to measurement data ("Msmt") for CAM software demonstration.

```
     1*s^5+3.698e+011*s^4+5.419e+022*s^3+(8.005e+033-j2.882e+017)*s^2+ ...

     (-0.006289)*s^5+(-1.677e+006)*s^4+(-1.283e+020)*s^3+(-2.278e+031+j1.813e+015)*s^2+ ...
Y12: ------------------------------------------------------------------------------------
     1*s^5+3.698e+011*s^4+5.419e+022*s^3+(8.005e+033-j2.882e+017)*s^2+ ...

     (-0.006289)*s^5+(-1.677e+006)*s^4+(-1.283e+020)*s^3+(-2.278e+031+j1.813e+015)*s^2+ ...
Y21: ------------------------------------------------------------------------------------
     1*s^5+3.698e+011*s^4+5.419e+022*s^3+(8.005e+033-j2.882e+017)*s^2+ ...

     0.0352*s^5+5.794e+009*s^4+1.475e+021*s^3+(1.194e+032-j1.915e+016)*s^2+ ...
Y22: ------------------------------------------------------------------------------------
     1*s^5+3.698e+011*s^4+5.419e+022*s^3+(8.005e+033-j2.882e+017)*s^2+ ...

Augmentation SPICE Netlist (Y-parameters):

.subckt 2PortAug 1 2
```

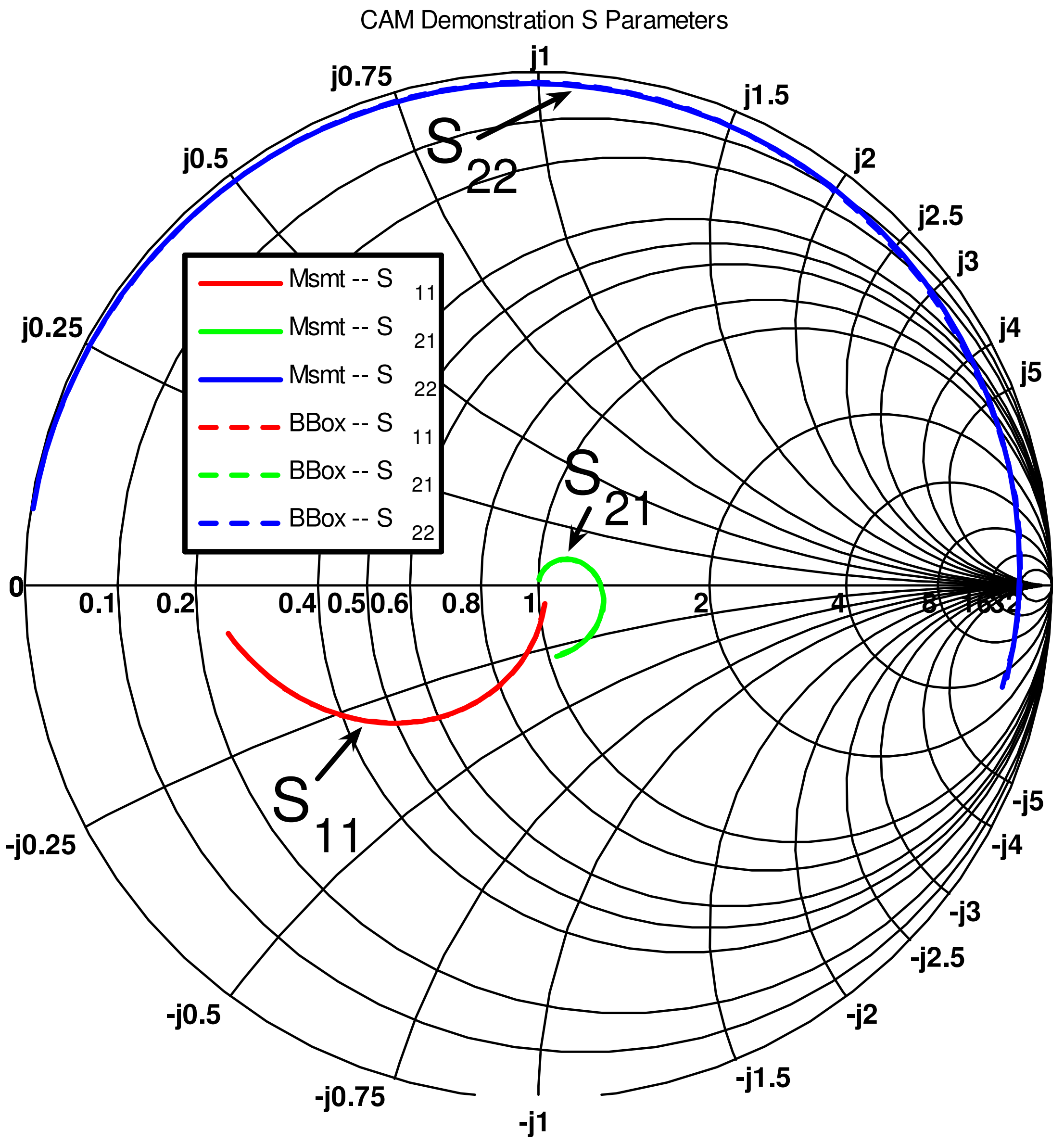


Figure 5.3: Smith chart comparison of CAM demonstration measurement data to that synthesized by vector fitting with a 5th-order numerator and denominator.

```
Gy11 1 0 LAPLACE 1 0 0.0190628061155509 7019506155.39351 2.47727921883448e+021 ...
Gy12 1 0 LAPLACE 2 0 -0.00628890276845647 -1676855.94955683 -1.28261884243961e+020 ...
Gy21 2 0 LAPLACE 1 0 -0.00628890276845647 -1676855.94955635 -1.28261884243961e+020 ...
Gy22 2 0 LAPLACE 2 0 0.0352043200039831 5793739103.99805 1.47461196859803e+021 ...
.ends
```

(Lines ending with "..." have been truncated to fit on the page.)

With this demonstration as a starting point, the user is encouraged to "play" with the CAM software to become more familiar with its abilities. Configuring perturbation and equivalent circuit models is a straightforward extension using the "P" ("Perturbation") and "E" ("Equivalent Circuit Model") commands.

Chapter 6 – Conclusions and Suggestions for Future Research

6.1 Conclusions

This work has developed a new, automated means of creating high-accuracy, broadband models that are compatible with time-domain simulators such as SPICE. This *circuit augmentation method*, CAM, seeks to merge the physical information provided by a user-provided equivalent circuit model and that obtained from a high-quality, general-purpose computer-based modeling algorithm. CAM seeks to "fill in the gaps" where ECMs fail, often in the realm of high-frequency accuracy or providing fits over wider bandwidths for band-limited designs. Unlike many traditional modeling approaches, CAM is not negatively impacted by distributed elements or highly frequency-dependent losses. CAM benefits directly from the tremendous speed of a computer, making heretofore intractable problems plausible when the solution space can be decimated many times over by the application of *human* intelligence (via the provision of a high-quality ECM), and by the iterative process that component value perturbation provides. This "meeting of the minds" can create a better solution than either approach alone, and the input of man vs. machine can be controlled by setting the allowable ECM perturbation range.

We have applied CAM to numerous passive structures with encouraging results. Run times are typically in the minutes, which is acceptable for a research-oriented vehicle. The quality of curve-fitting routines continues to improve, as does the quality of general-purpose optimizers; together these improvements may superlinearly increase the performance of CAM over time. This new methodology could be useful for modeling a wide range of passive structures on-chip and in electronics packaging, including bond wires, matching networks, solder-bump interconnects, and so on.

6.2 Suggestions for Future Research

As CAM stands today it does not yet appear to be of significant value to the practicing engineer; further research will be required if the method is ever to move out of the insular world of academia and reach the level of utility that practicing engineers demand. Some interesting additional area of research, related to CAM, are as follows:

- CAM has introduced numerous topologies for augmenting ECMs, but it still relies on user selection or brute-force testing to find the "best" topology. Is there a more intelligent, automated means to perform this selection? What produces better results – focusing on relatively "large scale" augmentations as has been done to date, or perhaps focusing on very "fine scale" augmentations where, e.g., every component in the ECM becomes a "super component." This idea, suggested by Erick Naviasky at Cadence, might replace resistors with a network whereby P=Q, inductors with networks where P=Q+1 and capacitors with networks where P=Q-1 (with relatively low values for P and Q), so as to preserve the resistive, inductive, or capacitive behaviors at low frequencies.

- Can an automated means of selecting the augmentation order be found? To date, almost all work has consisted of simply increasing P and Q until "good" results were obtained. Some early research concentrated on the behavior (specifically, the condition numbers) of the matrices involved in curve fitting; this provides a reasonably good indication of when increasing P or Q beyond their current values is unlikely to provide further improvements in augmentation results. Is there a more intelligent way to find the optimal values of P and Q besides this "brute force" approach?

- Noise modeling is of obvious importance for devices used in noise-sensitive applications, such as the first amplifiers encountered in a receiver's RF chain. While much of this noise is due to active components, discrepancies between simulated and fabricated devices are often significant, easily exceeding 3dBc/Hz [52]: Perhaps an augmentation approach could be used to close this gap?

- Historically, a very useful technique has been to compute a "sensitivity analysis" of a circuit; this provides the percentage change in a "desired" output (such as a voltage, a transfer gain such as S_{21}, etc.) to the percentage in a component value (such as the value of a resistor or inductor). This analysis turns out to be surprisingly straightforward to perform, using "adjoint" networks [53] and is included in all major SPICE simulators. Could this data be used to help select an augmentation topology? Or augmentation locations? How about using it to help set proportional limits on the percentage changes allowed in component values during perturbation?

- A significant recent work [54] that builds on CAM provides an elegant technique to calculate the necessary serial or parallel immittance needed in combination with a specified ECM component (e.g., a R, L, C, voltage source, etc.) to optimally (in a least-squares manner) "shift" a network's n-port response to that prescribed by its measurement data. An important aspect of this work is that the problem formulation is completely linear: Global optimizers need not apply! Results are obtained numerically as a function of frequency; these can be readily fit to a low-order rational function or synthesized into low-component count circuits. The method has been extended to support up to three or four multiple augmentation immittances simultaneously and is quite powerful; for a larger number of augmentation immittances one can iterate through three of four components at a time – the referenced paper synthesizes six immittances for a small-signal FET model. This is a considerably more general technique than the branch augmentation scenarios described in Section 3.1.1, and perhaps could be extended to automatically ascertain the best immittances to augment.

There are many more interesting circuit modeling avenues to explore, and hybrid methods such as CAM have perhaps opened the door to approaches that may one day reduce the level of complexity needed for circuit models without reducing simulation accuracy or increasing simulation time.

APPENDICES

Appendix A – A Brief History of Circuit Simulators

Research into efficient circuit modeling occurred simultaneously with research into faster simulation techniques, as discussed in Section 2.1; this section provides a brief history of the development of the simulators themselves.

As anyone with an interest in circuit simulators is well aware, the University of California at Berkeley came out with what would become the first "killer application" for circuit simulation: SPICE ("Simulation Program with Integrated Circuit Emphasis"), by Larry W. Nagel, Ron Rohrer and many others. SPICE was largely based on the program CANCER ("Computer Analysis of Nonlinear Circuits, Excluding Radiation" – a snub from the liberal '60s Berkeley to the government, which was funding many "Cold War" contracts that required radiation-hardness analysis) before it. CANCER was an "all students" project for a class taught by Rohrer, and the rule was that if department professor Donald Pederson – who was a pioneer in transistor research and established the first integrated circuit lab at Berkeley in 1960 – liked the program, everyone passed! Nagel was given the job of demonstrating CANCER to Pederson who – much to his classmates' relief, since the class was *supposed* to have been about circuit *synthesis* and had somehow managed to turn into one on circuit *simulation* – was satisfied. CANCER was ~6,000 lines of FORTRAN, and was the first simulator to use sparse matrix techniques.

CANCER became Nagel's master's project, and SPICE 1 was his doctoral project with first Rohrer and then Pederson as his thesis advisor. SPICE 1 was released in stages between 1971-1973, although widespread acceptance came with the release of SPICE 2 in 1975, polished and advanced to the point of being usable and extensible by anyone familiar with FORTRAN. Despite Nagel's own first interest being IC design, his PhD thesis [55] was the seminal tome on such programs and perennially linked him to simulator development long after he left Berkeley.[1]

The increasing availability of computing power for circuit design gave rise to the need for accurate component models – SPICE 1 had models for diodes and bipolar transistors but not MOSFETs, which are significantly more difficult to model accurately. Changes between SPICE 1 and 2 included the addition of JFET and MOSFET models, as well as a change from Ebers-Moll bipolar transistor models to the more accurate Gummel-Poon models. Berkeley was still a center of intense simulation and modeling research; by the late-1980's Don Pederson's group came up with innovative simulator techniques such as CODECS, a **Co**upled **De**vice and **C**ircuit **S**imulator, which extended SPICE to include some "technology" or "T-CAD" simulation, generally taken to be the device and/or process

[1] Nagel's own recollection of the history of SPICE is a fascinating read; it is available at http://www.designers-guide.org/Perspective/life-of-spice.pdf.

layers shown in Section 2.1. Largely designed by Karti Mayaram [56], Mayaram's research later continued at Oregon State University.

Meanwhile, SPICE had become a mainstream tool for the electrical engineer. Version 2G6 released in 1978 was the last version written in FORTRAN and contained numerous model improvements that had been integrated over the past decade. For a number of years, SPICE research waned, as many considered it a "solved problem," when in actuality, it was anything but! In 1983, Berkeley's Tomas Quarles took up the gauntlet and converted SPICE 2G6 to RATFOR (**RAT**ional **FOR**TRAN, sometimes described as "FORTRAN with improvements stolen from C") for his master's project. For his PhD project, he started yet again and converted the code to C, releasing SPICE 3 in 1989 [57]. Even before this point, various companies had created their own internal SPICEs: Tektronix had TekSpice, Bell Labs had ADVICE, TI had TISPICE, and Motorola had MCSPICE – a preponderance of software known as, "alphabet SPICE." Additionally, commercial entities had begun selling simulators based largely on the freely-available Berkeley source code: HSPICE, presently owned by Synopsys (it seems to change ownership every handful of years) and now the "golden standard" of IC simulation, started life based on SPICE 2E after significant performance optimizations were added by Shawn and Kim Hailey[2], running on a time-shared Cray supercomputer that supported vector operations in hardware.

The early- to mid-1990's was largely a period of incremental improvements in circuit simulators and a realization that, for all of its utility, SPICE was inherently ill-suited to many design problems such as RF IC designs, due to the slow speed and convergence difficulties of the core SPICE "transient" simulation needed to model large-signal non-linear circuits. Much research centered on specific simulation techniques to overcome these limitations; developments included various "shooting method" techniques such as harmonic balance simulations that quickly provide steady-state non-linear simulation results based on a small, finite number of fixed amplitude and frequency continuous wave excitation sources. Along with enhancements to the simulation "engine," advances in circuit modeling techniques provided another means to significantly decrease simulation time while maintaining acceptable accuracy. Many such techniques attempt to extract a "dominant pole" representation of circuit behavior, which speeds up circuit simulation due to the elimination of superfluous poles; AWE (Asymptotic Waveform Evaluation, [58]) and PVL (Padé via Lanczos, [59]), fall into this category. For passive systems, a detailed overview of many popular high-efficiency simulation techniques can be found in [3].

The late 1990's and the 2000's saw further refinements with a large emphasis on seamlessly "co-simulating" large designs. Although the complete simulation of a complicated circuit such

[2]The Hailey twins have a very colorful history, and might be described as a pair of highly successful "electronic cowboys" of a then-young – 1970s and 1980s – Silicon Valley, long before the sheriff came to town; some of their exploits at the time would today be more liable to land them in a penitentiary rather than their vacation home in Hawaii! A fascinating interview transcript with them is available at http://silicongenesis.stanford.edu/transcripts/hailey.htm.

as an RF front end from a high-level "block diagram" all the way down to the process level is not yet feasible, many contemporary design tools such as Microwave Office [60] provide nearly seamless merging between system, circuit, and device level simulators – including those implementing harmonic balance, finite-element, and other specialized techniques.

Perhaps ironically, the 1990's onward also saw the gradual changeover at most educational institutions from using Berkeley SPICE to instead using freely-provided or heavily-subsidized versions of commercial SPICE software. Similarly, companies that had developed their own SPICEs also began to phase out their usage in favor of commercial offerings. Such changes are indicative of the "coming of age" of circuit simulation and its significance in today's technology-oriented economy.

Appendix B – An Examination of Polynomials for Circuit Modeling

Almost any non-trivial electronic circuit will quickly end up with transfer functions that are rational functions in s. This is not a surprise given the behavioral difference between capacitors and inductors with respect to s, and the "$1/(1/a + 1/b)$" form of impedances in parallel or admittances in series. Hence, it is intuitively reasonable to use curve fitting with rational functions to create circuit models. Nevertheless, one might consider the utility of simple polynomials of the form

$$h(s) = \sum_{m=0}^{P} a_m s^m$$

given the easier means of fitting a curve to a polynomial (rather than a rational function) and the potential increase in simulation speed due to the lack of denominator terms requiring evaluation as well as the lack of a division operation, which is generally much slower than multiplication or addition. The difficulty in such a construction is that P must often be increased well beyond the sum of order of the numerator and denominator in a rational function to provide a good curve fit. For instance, consider the simple function $y(x) = ((x-35)(x-65))/(x-250)$. In this case, over the span of $x = [0 \ldots 100]$, the 3rd-order polynomial $y(x) = -26.733 \cdot 10^{-6} \cdot x^3 - 1.2488 \cdot 10^{-3} \cdot x^2 + 332.04 \cdot 10^{-3} \cdot x - 8.9392$ creates a reasonably good approximation to $y(x)$ as shown in Fig. B.1.

On the other hand, once the poles of $y(x)$ move closer to the function's domain, their influence becomes greater and a polynomial fit becomes difficult, as shown in Fig. B.2, where $y(x) = ((x-35)(x-65))/(x-115)$. The 3rd-order fit $y(x) = -568.76 \cdot 10^{-6} \cdot x^3 + 553.03 \cdot 10^{-3} \cdot x^2 - 972.57 \cdot 10^{-3} \cdot x - 10.198$ produces a poor fit, and even the 5th-order fit $y(x) = -203.31 \cdot 10^{-9} \cdot x^5 - 40.068 \cdot 10^{-6} \cdot x^4 - 2.9239 \cdot 10^{-3} \cdot x^3 + 85.2 \cdot 10^{-3} \cdot x^2 - 293.74 \cdot 10^{-3} \cdot x - 17.375$ is a little off at $y(0) = 19.783$.

A "rule of thumb" is that polynomials are best at approximating data where the analytic continuation of the function to be modeled contains no poles that are "close" to the intended domain of the interpolating polynomial. Although the preceding example has an obvious pole nearby (at $x = 115$), in the general case the poles may not lie on the real axis, in which case the suitability of an interpolating polynomial is always immediately obvious. Hence, while polynomials are useful for many curve-fitting requirements, they are not particularly applicable to modeling data resulting from physical circuits. On the other hand, one might consider other basis functions such as Chebyshev or Legendre polynomials for inclusion in a CAM augmentation network: Both can be readily expanded in terms of regular polynomials (and hence simulated efficiently), but have boundary conditions that may allow for better conditioning during the curve-fitting process.

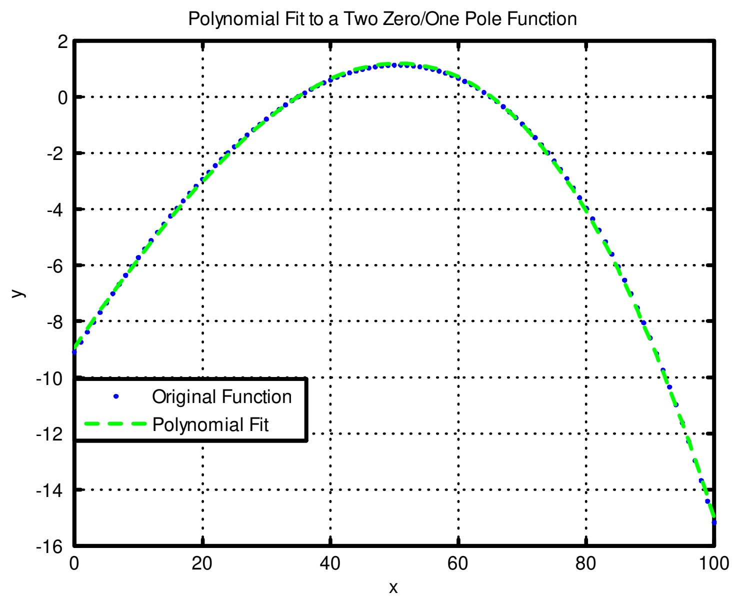


Figure B.1: 3rd-order polynomial curve fit to $((x-35)(x-65))/(x-250)$

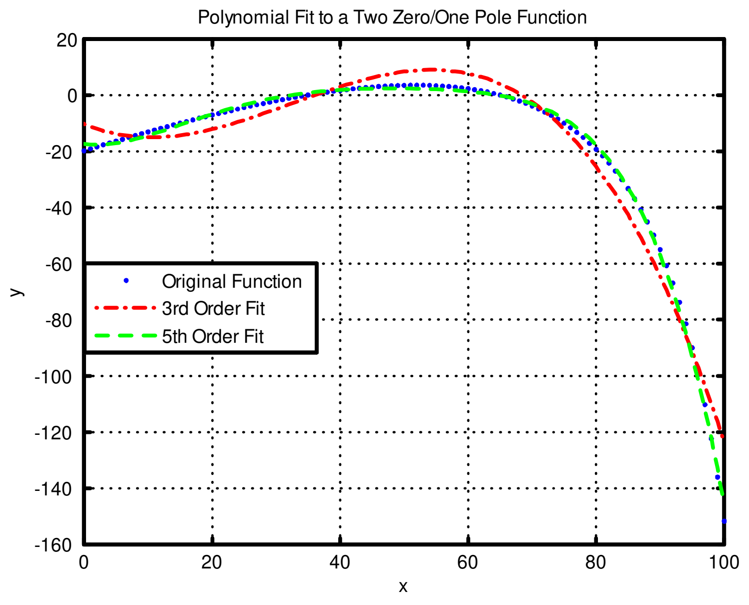


Figure B.2: 3rd-order and 5th-order polynomial curves fit to $((x-35)(x-65))/(x-115)$.

Appendix C – Useful Results from Linear Algebra

In this section only, we use $\|\mathbf{x}\|_2$ to denote the standard (Euclidean) 2-norm for vectors, so that single vertical bars can be used to denote the determinant of a matrix (e.g., $|\mathbf{M}|$), and to minimize ambiguity when discussing norms. In all other sections, $|\mathbf{x}|$ refers to the standard Euclidean 2-norm, as other norms aren't used and the "$|\mathbf{x}|$" form is felt to be more readable due to its compactness.

C.1 Least-Squares Fitting

With real-world problems, more often than not an exact solution to $\mathbf{Ax} = \mathbf{b}$ does not exist: Often $\mathbf{A}$ isn't square to begin with (i.e., the number of unknowns differs from the number of equations present) or, even if it is, the equations are inconsistent: $\mathbf{b}$ will have come from measurement data that contain (hopefully small) errors and are likely to have suffered numerical round-off. In such cases, a common approach[1] is to find the vector $\hat{x}$ such that $\mathbf{A}\hat{x}$ is as "close as possible" to $\mathbf{b}$. (This approach is so common that MATLAB does so by default when asked to "solve" a system of equations, although the method used is considerably more sophisticated than the approach described herein.) That is, we seek to minimize $\|\mathbf{A\hat{x}} - \mathbf{b}\|_2$. This task is equivalent to minimizing $\|\mathbf{A\hat{x}} - \mathbf{b}\|_2^2$, and therefore $\mathbf{\hat{x}}$ is known as the "least squares solution" to $\mathbf{Ax} = \mathbf{b}$.

Obtaining $\hat{x}$ is relatively straightforward. Consider Fig. C.1, meant to illustrate least squares fitting for a 3x3 system of equations: The shaded area (a plane) represents the span of the column space of $\mathbf{A}$, indicating that $\mathbf{A}$'s rank is only 2 and therefore $\mathbf{Ax} = \mathbf{b}$ cannot be solved exactly unless $\mathbf{b}$ happens to lie in the plane illustrated; this is not the case for the vector $\mathbf{b}$ shown and therefore we instead seek the least squares solution $\mathbf{\hat{x}}$. The dashed lines around the shading indicate that the span of $\mathbf{A}$ extends to ∞ in all directions. $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$ are the columns of $\mathbf{A}$ that provide a basis for its span, that is, $\mathbf{A} = [\mathbf{a1}|\mathbf{a2}|\mathbf{a3}]$ For any given $\mathbf{x}$, Fig. C.1 shows the error vector $\mathbf{b} - \mathbf{Ax}$; *it is this vector that must be made as small as possible by judiciously choosing $\hat{x}$.* As shown in Fig. C.2, when $\mathbf{x} = \mathbf{\hat{x}}$ the error vector is orthogonal (at right angles to) the span of $\mathbf{A}$ and is as small as possible.

Since the error vector $\mathbf{b} - \mathbf{Ax}$ is orthogonal to $\mathbf{A}$, it is orthogonal to any and all of A's basis

[1]So common that MATLAB does so by default when asked to solve a system of equations, although the method used is more sophisticated than the approach herein. This robust implementation benefits CAM, as solving systems of equations is a very common CAM operation.

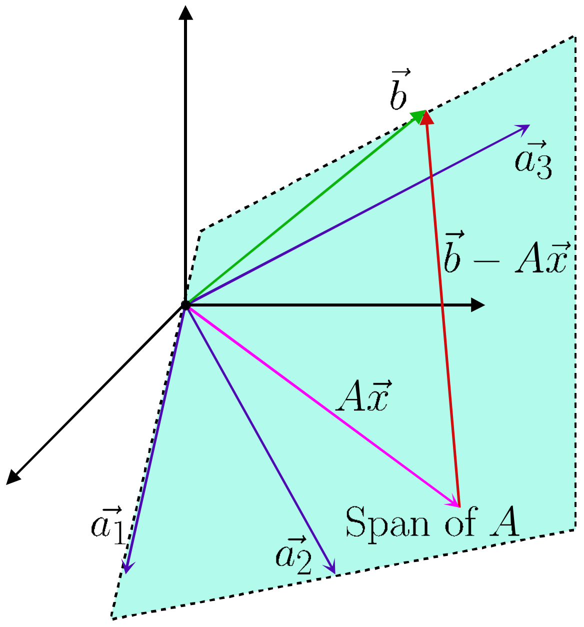


Figure C.1: Error vector between arbitrary **Ax** and **b**.

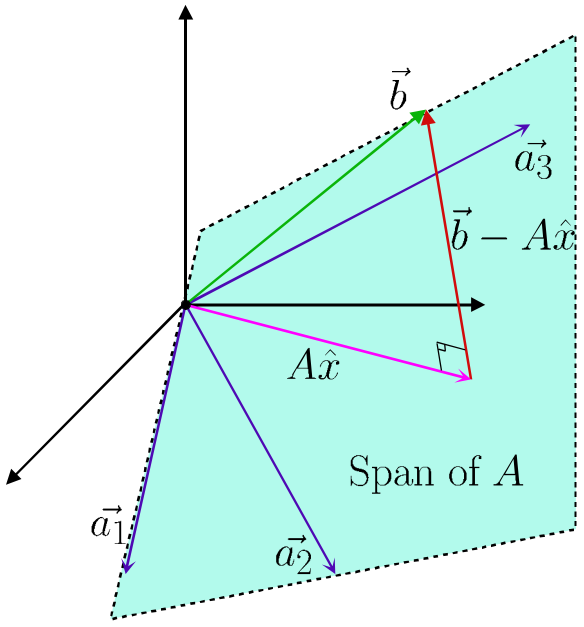


Figure C.2: Minimizing the error vector.

vectors. As such we can write

$$\begin{aligned}\mathbf{a}_1 \cdot (\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}) &= 0\\ \mathbf{a}_2 \cdot (\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}) &= 0\\ &\vdots\\ \mathbf{a}_n \cdot (\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}) &= 0\end{aligned}$$

where $n = 3$ for the example given, but applies in general for any $n \times n$ system of equations. Writing this in matrix notation we have

$$\begin{bmatrix}\mathbf{a}_1^T\\ \vdots\\ \mathbf{a}_n^T\end{bmatrix}[\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}] = 0$$

which is

$$\begin{aligned}\mathbf{A}^T(\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}) &= 0\\ \mathbf{A}^T\mathbf{A}\hat{\mathbf{x}} &= \mathbf{A}^T\mathbf{b}\\ \hat{\mathbf{x}} &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \qquad (C.1)\end{aligned}$$

In other words, to find the least squares solution to $\mathbf{Ax} = \mathbf{b}$, we multiply both sides of the equation by $\mathbf{A}^T$ and proceed to solve for what is now $\hat{\mathbf{x}}$. This solution exists whenever $\mathbf{A}^T\mathbf{A}$ is invertible, which can be shown to be the case so long as $\mathbf{A}$ has independent columns or (equivalently) full *column* rank – see [61]; this reference also has a more formal derivation of the result. As an aside, $\mathbf{A}^T\mathbf{A}$ is additionally square and symmetric, regardless of the original size $(m \times n)$ of $\mathbf{A}$; this desirable property is made use of in more sophisticated least squares algorithms.

Eq. (C.1) can also be obtained directly from calculus by setting the derivative of an "error" function, $\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}$, equal to zero and solving for $\hat{\mathbf{x}}$; the approach presented above is felt to be somewhat more illustrative and elegant.[2]

C.2 Condition Numbers

In a *well-conditioned* system of equations $\mathbf{Ax} = \mathbf{b}$, small changes in the "measurement" data $\mathbf{b}$ shouldn't give rise to unexpectedly large changes in the "model" $\mathbf{x}$: the *condition number* of the matrix $\mathbf{A}$ quantifies this change. Specifically, assume that contaminated measurement data $\hat{\mathbf{b}}$ are used to obtain a contaminated model $\hat{\mathbf{x}}$, that is, we solve $\mathbf{A}\hat{\mathbf{x}} = \hat{\mathbf{b}}$. We would like to relate $\|\mathbf{x} - \hat{\mathbf{x}}\|$ to

[2]It's times like this when one grudgingly agrees with Dr. Gilbert Strang that linear algebra's beauty is often underappreciated. But then someone reminds you that studying symplectic vector spaces or manifolds perhaps isn't *quite* as much fun as going to Disneyworld... or just building a radio and listening to Def Leppard.

$\|\mathbf{b} - \hat{\mathbf{b}}\|$, and do so with some algebraic manipulation. Start by subtracting $\mathbf{A}\hat{\mathbf{x}} = \hat{\mathbf{b}}$ from $\mathbf{A}\mathbf{x} = \mathbf{b}$:

$$\begin{aligned}
\mathbf{A}\mathbf{x} - \mathbf{A}\hat{\mathbf{x}} &= \mathbf{b} - \hat{\mathbf{b}} \\
\mathbf{x} - \hat{\mathbf{x}} &= \mathbf{A}^{-1}(\mathbf{b} - \hat{\mathbf{b}}) \\
\|\mathbf{x} - \hat{\mathbf{x}}\| &= \left\|\mathbf{A}^{-1}(\mathbf{b} - \hat{\mathbf{b}})\right\|
\end{aligned}$$

Assuming a *compatible* norm is chosen (see Section C.4), this becomes

$$\|\mathbf{x} - \hat{\mathbf{x}}\| \leq \left\|\mathbf{A}^{-1}\right\| \left\|\mathbf{b} - \hat{\mathbf{b}}\right\|$$

We would like to normalize the two sides of the inequality to $\|\mathbf{x}\|$ and$\|\mathbf{b}\|$, respectively, to obtain a unitless sensitivity measure. First, divide both sides by $\|\mathbf{b}\|$:

$$\begin{aligned}
\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|}{\|\mathbf{b}\|} = \frac{\|\mathbf{x} - \hat{\mathbf{x}}\|}{\|\mathbf{A}\mathbf{x}\|} &\leq \frac{\left\|\mathbf{A}^{-1}\right\| \left\|\mathbf{b} - \hat{\mathbf{b}}\right\|}{\|\mathbf{b}\|} \\
\|\mathbf{x} - \hat{\mathbf{x}}\| &\leq \frac{\|\mathbf{A}\mathbf{x}\| \left\|\mathbf{A}^{-1}\right\| \left\|\mathbf{b} - \hat{\mathbf{b}}\right\|}{\|\mathbf{b}\|} \\
\|\mathbf{x} - \hat{\mathbf{x}}\| &\leq \frac{\|\mathbf{A}\| \|\mathbf{x}\| \left\|\mathbf{A}^{-1}\right\| \left\|\mathbf{b} - \hat{\mathbf{b}}\right\|}{\|\mathbf{b}\|} \\
\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|}{\|\mathbf{x}\|} &\leq \|\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\| \cdot \frac{\left\|\mathbf{b} - \hat{\mathbf{b}}\right\|}{\|\mathbf{b}\|}
\end{aligned}$$

$\|\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\|$ is the condition number of matrix $\mathbf{A}$, and sets an upper limit on the errors in the solution $\mathbf{x}$ dues to errors in the right-hand side, although in some systems cond(A) grossly overestimates the actual error. Practically speaking, $\log_{10} \mathrm{cond}(\mathbf{A})$ provides some indication of the number of trailing digits in $\mathbf{x}$ that may be inaccurate. When this reaches a significant fraction of the digits of precision in the system (e.g., 16 for MATLAB), all solutions $\mathbf{x}$ become highly suspect and may no longer retain *any* significant accuracy (results might as well be compared with, e.g., $\hat{\mathbf{x}}$ = rand(size($\mathbf{A}$))...!).

C.3 Weighting Matrices

The fact that an inconsistent set of equations has *no* exact solution, yet we're pressed to provide some sort of "best approximation" of $\mathbf{x}$ in $\mathbf{A}\mathbf{x} = \mathbf{b}$**,** provides the opportunity for endless definition (and journal articles) of the "best" solution $\hat{\mathbf{x}}$ for any given problem. Before exploring a slightly different solution for $\hat{\mathbf{x}}$, recall that the least squares solution derived above minimizes $\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\|_2^2$.

Mathematically we can expand the 2-norm and write out that the least squares solution obtains

$$\min \sum_{i=1}^{m} [(\mathbf{Ax})_i - b_i]^2 \tag{C.2}$$

where $\mathbf{A}$ is $m \times n$. Now consider the possible sources of error in our "measurement" data $\mathbf{b}$[3]: First, assume that the errors are independent and normally distributed with mean 0 and standard deviation σ_i; then $\mathbf{b}$ will have a mean of $\mathbf{Ax}$ and a standard deviation $\boldsymbol{\sigma}_i$. Recall (or look up in, e.g., [62]) the probability density function of the normal distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(x-\mu)^2/\sigma^2}$$

where μ is the distribution's mean and σ is its standard deviation. The conditional probability density function for each b_i (one entry in the measurement vector) for a given "model" $\mathbf{x}$ is then

$$f(b_i|\mathbf{x}) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{1}{2}(b_i-(A\mathbf{x})_i)^2/\sigma_i^2}$$

Now, assume that the solution $\mathbf{x}$ we seek should maximize the overall probability density function $f(\mathbf{b}|\mathbf{x})$ (i.e., the contaminated data $\mathbf{b}$ observed are most likely if we choose this particular $\mathbf{x}$); such a solution is not only intuitively reasonable but also has desirable statistical properties [63]. From the assumed independence of the errors we have

$$f(\mathbf{b}|\mathbf{x}) = f(b_1|\mathbf{x}) \cdot f(b_2|\mathbf{x}) \cdot \cdots \cdot f(b_m|\mathbf{x})$$

and seek

$$\begin{aligned}
\max\ f(\mathbf{b}|\mathbf{x}) &= \max\ f(b_1|\mathbf{x}) \cdot f(b_2|\mathbf{x}) \cdot \cdots \cdot f(b_m|\mathbf{x}) \\
&= \max\ \prod_{i=1}^{m} \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{1}{2}(b_i-(A\mathbf{x})_i)^2/\sigma_i^2} \\
&= \max\ \underbrace{\frac{1}{\sqrt[m/2]{2\pi} \cdot \prod_{1}^{m} \sigma_i}}_{\text{Constant}} \cdot \prod_{i=1}^{m} e^{-\frac{1}{2}(b_i-(A\mathbf{x})_i)^2/\sigma_i^2}
\end{aligned}$$

where the constant can be removed directly. Taking the logarithm (allowable as it is monotonically

[3]In books on system modeling, a standard linear equation used is $\mathbf{Gm} = \mathbf{d}$ where $\mathbf{m}$ is the **model** for the system, $\mathbf{d}$ is the **data** that model produced, and the **kernel** $\mathbf{G}$ (which likely translates into a word starting with "G" in *some* language) that ties the two together; we'll keep using $\mathbf{Ax} = \mathbf{b}$ for the sake of consistency.

increasing) converts the multiplicative series into a summation

$$
\begin{aligned}
&= \max \ln \prod_{i=1}^{m} e^{-\dfrac{\left[b_i - (\mathbf{Ax})_i\right]^2}{2\sigma_i^2}} \\
&= \max \ -\sum_{i=1}^{m} \frac{\left[b_i - (\mathbf{Ax})_i\right]^2}{2\sigma_i^2} \\
&= \min \ \sum_{i=1}^{m} \frac{\left[(\mathbf{Ax})_i - b_i\right]^2}{\sigma_i^2}
\end{aligned}
$$

Notice the similarity with Eq. (C.2). Thus – under the assumption of normally distributed errors in **b** as detailed above – the most likely solution **x** is obtained if we first **weight** each equation (row i of **A** and **b**) with $1/\sigma_i$. More formally, we form a **weighting matrix**

$$\mathbf{W} = \mathrm{diag}(1/\sigma_1, 1/\sigma_1, \cdots, 1/\sigma_m)$$

and solve the system of equations

$$\mathbf{WAx} = \mathbf{Wb}$$

using least squares fitting. From Eq. (C.1) we then have

$$
\begin{aligned}
\hat{x} &= \left((\mathbf{WA})^T\mathbf{WA}\right)^{-1}(\mathbf{WA})^T\mathbf{b} \\
&= (\mathbf{A}^T\mathbf{W}^2\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}^T\mathbf{b}
\end{aligned}
$$

In real-world systems, $\boldsymbol{\sigma}$ may not be known (due to the difficultly of or simple lack of its measurement!). A heuristical approach is to assume that σ_i is proportional to b_i and simply use $\mathbf{W} = \mathrm{diag}(\mathbf{b})$; this assumption implies that measurement error is proportional to the measurement value itself – true for many instruments, where error limits are often given as percentages of the measured values. This approach tends to improve results (and condition numbers); maybe people have experimentally come up with this form of "normalization" on their own – CAM uses this approach internally.

C.4 Norms

Although most people are familiar with "the Euclidean norm" in the context of vectors, the application of norms to matrices is more obscure and overlooked. Confusion often arises due to the use of norms without being aware of what *type* of norm is in use. Strictly speaking, for vectors, *the norm of* **x** (designated by $\|\mathbf{x}\|$) is *any* function that satisfies the following properties

$$\begin{aligned}\|\alpha\mathbf{x}\| &= |\alpha|\|\mathbf{x}\| \\ \|\mathbf{x}_1+\mathbf{x}_2\| &\leq \|\mathbf{x}_1\|+\|\mathbf{x}_2\| \\ \|\mathbf{x}\| &\geq 0 \\ \|\mathbf{x}\| &= 0 \text{ iff } x=0\end{aligned}$$

(where α is a scalar). The so-called *p-norm* is typically used for vectors; it is defined as

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{N} |x_i|^p\right)^{1/p}$$

(where $|x_i|$ is the absolute value of x_i) This gives the standard 1-norm – the sum of the absolute value of the vector's components –, the 2- or Euclidean-norm – the length of the vector –, and the infinite-norm where $p \to \infty$ emphasizes the largest entry within $\mathbf{x}$ to pick out $\max |x_i|$.

For matrices, any norm may be defined so long as

$$\begin{aligned}\|\alpha\mathbf{A}\| &= |\alpha|\|\mathbf{A}\| \\ \|\mathbf{A}+\mathbf{B}\| &\leq \|\mathbf{A}\|+\|\mathbf{B}\| \\ \|\mathbf{A}\| &\geq 0 \\ \|\mathbf{A}\| &= 0 \text{ iff } \mathbf{A}=0 \\ \|\mathbf{AB}\| &\leq \|\mathbf{A}\|\|\mathbf{B}\|\end{aligned}$$

Norms between matrices and vectors are *compatible* so long as $\|\mathbf{Ax}\| \leq \|\mathbf{A}\|\|\mathbf{x}\|$; this definition is used in many derivations such as that of condition numbers.

P-norms for matrices have the definition of

$$\|\mathbf{A}\|_p = \{\ \max \|\mathbf{Ax}\|_p : \|\mathbf{x}\|_p = 1\ \}$$

which is compatible with the P-norms for vectors. Unfortunately, $\|\mathbf{A}\|_p$ is generally difficult to obtain for $p \neq \{1, 2, \infty\}$ (see [61]). Even for $p = 2$, $\|\mathbf{A}\|_p$ is the largest singular value of $\mathbf{A}$, which is typically quite time-consuming to obtain. As such, a more common matrix norm is the *Frobenius*[4]

[4]Ferdinand Georg Frobenius, October 1849-August 1917, a German theoretical mathematician whose main contributions were to the theory of differential equations and group theory. Frobenius was recognized as so gifted that he was allowed to begin teaching in Berlin even before finishing his post-doctorial thesis (*Habilitationsschrift* in German).

norm, which has the markedly simpler definition

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{i,j}|^2} = \sqrt{\mathrm{Tr}(\mathbf{A}^H\mathbf{A})}$$

The Frobenius norm is compatible with the 2-norm for vectors, and as such is sometimes referred to as the *Euclidean Norm* for matrices, which unfortunately has the potential to create confusion with the matrix 2-norm.

C.5 Special Matrices

Many matrices will, under certain conditions, have various interesting properties that make them "special." This list provides the definitions of various special matrices encountered within CAM, although the "interesting properties" are highly abridged:

- Hermitian Matrix: A Hermitian matrix is a matrix where the "diagonally opposite" entries are complex conjugates. That is, for each matrix entry $A_{j,k}$, $A_{k,j} = A^*_{j,k}$. This implies that the matrix's diagonal must be real. "The Hermitian of $\mathbf{A}$" or "$\mathbf{A}$-Hermitian," $\mathbf{A}^H$, is meant to refer to $\mathbf{A}^{*^T}$ (that is, the complex conjugate transpose of $\mathbf{A}$). Hermitian matrices are the logical extension to real symmetric matrices when the entries are no longer purely real. This extension unfortunately provides for a certain degree of confusion, in that it is not uncommon to see literature refer to "the transpose of $\mathbf{A}$" to mean the Hermitian of A. As with real symmetric matrices, Hermitian matrices have real eigenvalues.

- Hilbert Matrix: Hilbert[5] matrices have the simple definition that each matrix element $H_{i,j}$ is equal to $1/(i+j-1)$. I.e., the Hilbert matrix of size 2 is simply $\begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{bmatrix}$. Hilbert matrices are often considered to be canonical examples of ill-conditioning, as their condition numbers grow almost exponentially with order: $O(e^{3.5255n}/\sqrt{n})$. Hilbert matrices are clearly symmetric and positive definite, and their determinants and inverses can be expressed in closed form. The most direct form of setting up a system of equations to curve fit tabulated data to a polynomial

[5] David Hilbert, 1862-1943. Hilbert was a brilliant German mathematician, researching a broad range of theoretical mathematical areas, including invariant sets, geometry, and functional analysis. He rang in the 20th century by proposing 23 then-unsolved problems in Paris. The solutions trickled in, providing great mathematical insight and understanding long after Hilbert's death (even today, a few problems remain unresolved). Hilbert was a "formalist," meaning that he considered mathematics a "game" that man invented for himself, consisting of axioms and theorems that were rigorously founded and logically extended. From this vantage point, he was fond of proclaiming that no problem was unsolvable: "Für den Mathematiker gibt es kein Ignorabimus... Wir müssen wissen, wir werden wissen" – "For the mathematician there is no ignorabimus... We must know, we shall know." Hilbert's later years were spent at the University of Göttingen; he retired in 1930, shortly before the Nazi party began gutting Göttingen of Jews and other intellectuals in 1933.

(or rational function) generates a matrix that is approximately (depending on the exact data used) Hilbertian in form.

- Hamiltonian Matrix: A Hamiltonian matrix is a real matrix $\mathbf{H}$ of size $2n$ by $2n$ such that $\mathbf{J} \cdot \mathbf{H}$, where $\mathbf{J} = \begin{bmatrix} o & \mathbf{I}_n \\ -\mathbf{I}_n & 0 \end{bmatrix}$, is symmetric. $\mathbf{J}$ can be seen to be a skew-symmetric matrix, and (as can be verified directly) $\mathbf{J}^{-1} = -\mathbf{J} = \mathbf{J}^T$ – hence $\mathbf{J}$ is orthogonal as well. In other words, $\mathbf{JH} = -\mathbf{H}^T\mathbf{J}$. It can be shown that any Hamiltonian matrix $\mathbf{H}$ can be written as $\mathbf{H} = \begin{bmatrix} \mathbf{F} & \mathbf{G} \\ \mathbf{Q} & -\mathbf{F}^T \end{bmatrix}$ where $\mathbf{G}$ and $\mathbf{Q}$ are symmetric matrices. The name "Hamiltonian" derives from the form of the results when classical (Newtonian) dynamics has its equations of motions generalized using Hamiltonian mechanics.

- Symplectic Matrix: A symplectic matrix is a real or complex matrix $\mathbf{M}$ of size $2n$ by $2n$ such that $\mathbf{M}^T\mathbf{\Omega}\mathbf{M} = \mathbf{\Omega}$, where $\mathbf{\Omega}$ is a skew-symmetric matrix that must be non-singular. Often $\mathbf{\Omega}$ is chosen to be $\mathbf{J}$ as used in Hamiltonian matrices, $\begin{bmatrix} o & \mathbf{I}_n \\ -\mathbf{I}_n & 0 \end{bmatrix}$, which gives symplectic matrices properties similar to those of Hamiltonian matrices. It can be shown that all symplectic matrices have a determinant of 1.

The use of Hamiltonian and Symplectic matrices is an example of specialized results from other fields being put to productive use in circuit modeling theory. These results are specialized enough that it's highly unlikely one would find a discussion of their application in a *circuits* textbook, but rather only in published papers relating to circuit modeling. A resource such as [64] is helpful, as the typical undergraduate or graduate engineering student will not have been previously exposed to this material.

C.6 Quadratic Forms and Positive Definiteness

Quadratic forms are mathematical expressions of the form

$$\mathbf{x}^H \cdot \mathbf{M} {\cdot} \mathbf{x} \tag{C.3}$$

where $\mathbf{M}$ is an arbitrary matrix. This expression produces a single, scalar result. By themselves, quadratic forms provide little more than a compact way to write polynomial expressions; (C.3) can be expanded as

$$\mathbf{x}^H\mathbf{M}\mathbf{x} = \sum_{j=1}^{n}\sum_{k=1}^{n} \mathrm{M}_{j,k}\mathbf{x}_j^*\mathbf{x}_k \tag{C.4}$$

(The form of this summation is what leads to the *quadratic form* nomenclature; *quadratus* is Latin for *square*.) For example, consider the case where $\mathbf{x}$ and $\mathbf{M}$ are real. For a 2x2 matrix $\mathbf{M}$ we have

$$\mathbf{x}^H\mathbf{M}\mathbf{x} = \mathrm{M}_{11}x_1^2 + \mathrm{M}_{12}x_1x_2 + \mathrm{M}_{21}x_2x_1 + \mathrm{M}_{22}x_2^2 \tag{C.5}$$

which can be seen to provide a scalar polynomial function $f(x_1, x_2)$ when $\mathbf{M}$ is constant. Under various scenarios – usually involving restrictions on $\mathbf{M}$ – various useful results can be shown:

- When $\mathbf{M}$ is Hermitian, the expression $\mathbf{x}^H\mathbf{M}\mathbf{x}$ produces a *real* result – regardless of the signs and pure real or complex nature of $\mathbf{x}$ or $\mathbf{M}$. The proof of this is shown in Section C.7.
- When $\mathbf{M}$ is Hermitian, the expression $\mathbf{x}^H\mathbf{M}\mathbf{x}$ is greater than or equal to zero when the eigenvalues of $\mathbf{M}$ are themselves greater than or equal to zero for all values of $\mathbf{x}$. (Since $\mathbf{M}$ is Hermitian, its eigenvalues will be real.) In such a case, $\mathbf{M}$ is said to be *positive semi-definite* which is sometimes written as $\mathbf{M} > 0$[6]. This notation should not be taken to imply anything concerning the entries of $\mathbf{M}$ itself, however: The matrix $\mathbf{M} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$ is *not* positive semi-definite (e.g., for $\mathbf{x} = \begin{bmatrix} -3 \\ 5 \end{bmatrix}$, $\mathbf{x}^H\mathbf{M}\mathbf{x} = -26$), whereas the matrix $\mathbf{M} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$ *is* positive semi-definite (its eigenvalues are 1 and 3).
- If $\mathbf{x}^H\mathbf{M}\mathbf{x}$ is strictly greater than zero, $\mathbf{M}$ is termed *positive definite*. The eigenvalues of $\mathbf{M}$ being strictly greater than zero is necessary and sufficient for $\mathbf{M}$ to be positive definite.
- The product $\mathbf{M}^H\mathbf{M}$ is itself Hermitian (since $\left(\mathbf{M}^H\mathbf{M}\right)^H = \mathbf{M}^H\mathbf{M}^{H^H} = \mathbf{M}^H\mathbf{M}!$), as well as positive semi-definite, since $\mathbf{x}^H(\mathbf{M}^H\mathbf{M})\mathbf{x} = (\mathbf{M}\mathbf{x})^H\mathbf{M}\mathbf{x}$, which is seen to be the Euclidean norm squared of the vector $\mathbf{M}\mathbf{x}$ and hence must be greater than or equal to zero.
- An arbitrary matrix $\mathbf{M}$ is positive (semi-) definite if the *Hermitian part* of $\mathbf{M}$ is positive (semi-) definite. The Hermitian part of a matrix, denoted $\mathbf{M}_H$, is $\mathbf{M}_H = \frac{1}{2}\left(\mathbf{M} + \mathbf{M}^H\right)$. $\mathbf{M}_H$ is itself Hermitian, as can be verified by direct examination. Similar to how a function can always be equated to the sum of an odd and even function – and doing so can have significant utility, an arbitrary matrix can always be equated to the sum of a Hermitian ($\mathbf{M}_a = \mathbf{M}^H$) and Skew-Hermitian ($\mathbf{M}_b = -\mathbf{M}^H$) matrix. As mentioned previously, Hermitian matrices have purely real eigenvalues; Skew-Hermitian matrices have purely imaginary eigenvalues.

The bullet points listed without proofs can generally be shown using material from, e.g., [20] or [65], along with a little elbow grease to take proofs based on real symmetric matrices and extending them to the Hermitian case. An unfortunate fact about the literature surrounding quadratic forms

[6]You are not alone if you find this notation a little misleading! One would think the "0" would at least be written as "$\mathbf{0}$"(bolded to indicate a matrix).

and positive definiteness is that calculations often *assume* a Hermitian matrix without stating as much.

C.7 Proof: Quadratic Form of Hermitian Matrix Produces a Real Result

Consider the expression

$$\mathbf{x}^H \cdot \mathbf{M}{\cdot}\mathbf{x} \tag{C.6}$$

When $\mathbf{M}$ is a Hermitian matrix, the result is real even when $\mathbf{x}$ is complex. To prove this, begin by expanding the expression into a a double summation as follows

$$\mathbf{x}^H \cdot \mathbf{M}{\cdot}\mathbf{x} = \sum_{j=1}^{n} \sum_{k=1}^{n} \mathbf{M}_{j,k} \mathbf{x}_j^* \mathbf{x}_k$$

In the summation, when $j = k$, we have entities of the form $\mathbf{M}_{j,j}\mathbf{x}_j^*\mathbf{x}_j$, which is real given that $\mathbf{M}_{jj}$ must be real for a Hermitian matrix. For the "off-diagonal" ($j \neq k$) terms, we pair the "j,k" entry with the "k,j" entry to obtain the real quantity

$$\begin{aligned}
& \mathbf{M}_{j,k}\mathbf{x}_j^*\mathbf{x}_k + \mathbf{M}_{k,j}\mathbf{x}_k^*\mathbf{x}_j \\
= & \mathbf{M}_{j,k}\mathbf{x}_j^*\mathbf{x}_k + \mathbf{M}_{j,k}^*\mathbf{x}_k^*\mathbf{x}_j \\
= & \mathbf{M}_{j,k}\mathbf{x}_j^*\mathbf{x}_k + \left(\mathbf{M}_{j,k}\mathbf{x}_j^*\mathbf{x}_k\right)^* \\
= & 2\Re\left(\mathbf{M}_{j,k}\mathbf{x}_j^*\mathbf{x}_k\right)
\end{aligned}$$

Hence, Eq. (C.6) is real.

In the case where $\mathbf{M}$ is *not* Hermitian, a pair of useful results that can be proven using similar reasoning are

$$\begin{aligned}
\Re(\mathbf{x}^H\mathbf{M}\mathbf{x}) &= \mathbf{x}^H\Re(\mathbf{M})\mathbf{x} \\
\Im(\mathbf{x}^H\mathbf{M}\mathbf{x}) &= \mathbf{x}^H\Im(\mathbf{M})\mathbf{x}
\end{aligned}$$

Appendix D – Results from Circuit Network Theory

D.1 Driving Point Immittance Functions

Immittance is a generic name for either an impedance or admittance. Their functions, $h(s) = z(s)$ or $h(s) = y(s)$ can be viewed as any other transfer function, albeit with the unusual convention that both the "input" and "output" (input current and output voltage for $z(s) = v(s)/i(s)$ or input voltage and output current for $y(s) = i(s)/v(s)$) are taken at the same point in the system. (Using standard network parameters, *driving point* immittances are found along the diagonals of the Y or Z matrices, whereas the off-diagonal entries are termed *transfer* immittances.) For a network made of only resistors, inductors, and capacitors, the driving point immittance transfer function (2.1), repeated here

$$h(s) = \frac{\sum\limits_{m=0}^{P} a_m s^m}{\sum\limits_{n=0}^{Q} b_n s^n}$$

will have $|P - Q| \leq 1$ regardless of the network's topology. This can be derived by starting with a single R, L, or C at the measurement point of $h(s)$ and proceeding to add all other elements until the network is complete. The network at any given point is termed resistive, inductive, or capacitive if $h(s)$ approaches a constant multiplied by 1, s, or $1/s$ as $s \to \infty$ with $s = j\omega$. The constant reflects the particular component values used, but does not affect the order of the network as components are combined (other than for the non-physical cases of zero-valued or infinite-valued components).

As a second component is added to the first, if the resulting immittance is still asymptotically bounded as 1, s, or $1/s$, via induction any arbitrarily large network will also remain likewise bounded and therefore $|P - Q| \leq 1$. For instance, a 1F capacitor in parallel with a 1H inductor has impedance $z(s) = 1/(s + 1/s) = s/(s^2 + 1) \Rightarrow z(s) \propto 1/s$ (Ω) whereas the series combination as seen in Fig. D.1 has $z(s) = s + 1/s = (s^2 + 1)/s \Rightarrow z(s) \propto s$ (Ω).

Table D.1 shows the results of all possible series and parallel "second" (or n+1) elements added to the "first" (or n). As seen, all results are still bounded to 1, s, or $1/s$, and as such $|P - Q| \leq 1$.

If a given physical network has a stable driving point impedance function, the same network must have a stable driving point admittance function as well. Hence, not only must $h(s)$ have no poles in the right-half of the $s-$plane, it must not have any zeros in the RHP either, as the zeros of $z(s)$ are the poles of $y(s)$ and vice-versa.

First Element	Asymptotic Form	Add Series			Add Parallel		
		R	**L**	**C**	**R**	**L**	**C**
Resistive	1	1	s	$1/s$	1	s	$1/s$
Inductive	s	s	s	s	1	s	$1/s$
Capacitive	$1/s$	1	s	$1/s$	$1/s$	$1/s$	$1/s$

Table D.1: Asymptotic network behavior as additional components are added to an asymptotically resistive, inductive, or capacitive network.

D.2 Transfer Functions

Transfer functions are often represented in the *pole-residue* form, Eq. (2.2), repeated here

$$h(s) = \sum_{i=1}^{Q} \frac{k_i}{s - p_i} + d + se$$

where the residues, k_i, and the poles, p_i, are either real or occur in complex-conjugate pairs (i.e., $k_{i+1} = k_i^*$ and $p_{i+1} = p_i^*$); d and e are real. Restricting our attention to the case where all k_i's have a non-zero real component, Eq. (2.2) can be readily manipulated into a rational function with real coefficients, taking one of the following forms:

$$h(s) = \begin{cases} \dfrac{\sum_{m=0}^{Q-1} a_m s^m}{\sum_{n=0}^{Q} b_n s^n} & \text{for } d = 0, e = 0 \\ \dfrac{\sum_{m=0}^{Q} a_m s^m}{\sum_{n=0}^{Q} b_n s^n} & \text{for } d \neq 0, e = 0 \\ \dfrac{\sum_{m=0}^{Q+1} a_m s^m}{\sum_{n=0}^{Q} b_n s^n} & \text{for } d \neq 0, e \neq 0 \end{cases} \tag{D.1}$$

Observe the utility of the d and e coefficients to control $h(s)$'s limit as one of 0, a constant, or ∞ as $s \to \infty$.

Complex conjugates combine as follows

$$\frac{k}{s-p} + \frac{k^*}{s-p^*} = \frac{2\Re(k)s - 2(\Re(k)\Re(p) + \Im(k)\Im(p))}{s^2 - 2\Re(p)s + |p|^2}$$

which can be seen to have purely real coefficients. If k is purely imaginary, however, this result has a zero coefficient in s, i.e., the numerator's order is *not* one less than that of the denominator (as occurs when k has a real component – the basis for Eq. (D.1)). A trivial example of such a case is Fig. D.1 where $V_{out}/V_{in} = 1/(s^2+1)$.

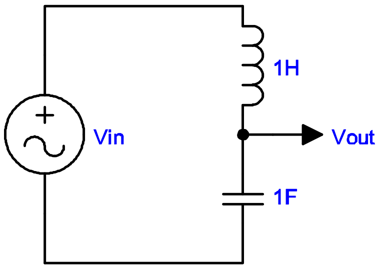


Figure D.1: Simple L-C voltage divider; $Vout/Vin = 1/(s^2+1)$.

As such, in the *general* case we have

$$\sum_{i=1}^{Q} \frac{k_i}{s-p_i} + d + se = \frac{\sum\limits_{m=0}^{P} a_m s^m}{\sum\limits_{n=0}^{Q} b_n s^n}$$

where the relationship between P and Q is bounded by $P - Q \leq 1$. With additional information, this bound can be tightened, e.g., for driving-point immittance functions, it can be shown (see the previous section) that $|P-Q| \leq 1$. In many cases $h(s) \to \infty$ is non-physical (e.g., for **S** parameters) and $P - Q \leq 0$ is ensured during modeling (e.g., by setting $e = 0$ in (2.2)).

D.3 Conversion Between **Y**, **Z**, and **S** Parameters

While two-port conversion formulas are readily found in almost all references on network or microwave theory and synthesis such as [28], the more general, multi-port conversion formulas are less common, and are therefore reproduced here for the convenience of the reader. This is especially true when the port references are complex, as many commonly published formulas are not valid in this case[1]. The derivations of these formulae are available in [66].

When converting to or from scattering parameters, one must know the port reference impedances

[1]All CAM work was performed with data utilizing purely real port impedances, however.

desired or used. To begin, define two diagonal matrices $\mathbf{Z}_{Ref}$ and $\mathbf{U}$ as follows:

$$\mathbf{Z}_{Ref} \overset{\triangle}{=} \mathrm{diag}(Z^i_{Ref})$$
$$\mathbf{U} \overset{\triangle}{=} \mathrm{diag}\left(\frac{\sqrt{\Re(Z^i_{Ref})}}{\left|Z^i_{Ref}\right|}\right)$$

where Z^i_{Ref} is the reference impedance at port i

The conversion formulas are then as follows:

From $\Rightarrow$ **To** $\Downarrow$	**Admittance Parameters** $\mathbf{Y}$	**Impedance Parameters** $\mathbf{Z}$	**Scattering Parameters** $\mathbf{S}$
$\mathbf{Y}$	$\mathbf{Y}=\mathbf{Y}$	$\mathbf{Y}=\mathbf{Z}^{-1}$	$\mathbf{Y}=\mathbf{Z}^{-1}_{Ref}\cdot(\mathbf{U}+\mathbf{SU})^{-1}$ $\cdot(\mathbf{U}-\mathbf{SU})$
$\mathbf{Z}$	$\mathbf{Z}=\mathbf{Y}^{-1}$	$\mathbf{Z}=\mathbf{Z}$	$\mathbf{Z}=(\mathbf{U}-\mathbf{SU})^{-1}$ $\cdot(\mathbf{U}+\mathbf{SU})\cdot\mathbf{Z}_{Ref}$
$\mathbf{S}$	$\mathbf{S}=\mathbf{U}\cdot(\mathbf{Y}^{-1}-\mathbf{Z}_{Ref})$ $\cdot(\mathbf{Y}^{-1}+\mathbf{Z}_{Ref})^{-1}\cdot\mathbf{U}^{-1}$	$\mathbf{S}=\mathbf{U}\cdot(\mathbf{Z}-\mathbf{Z}_{Ref})$ $\cdot(\mathbf{Z}+\mathbf{Z}_{Ref})^{-1}\cdot\mathbf{U}^{-1}$	$\mathbf{S}=\mathbf{S}$

In the preceding, $\mathbf{I}$ is the identity matrix of size n (the number of ports). CAM includes MATLAB functions to perform these conversions both numerically as well as symbolically, in terms of rational function coefficients.

Appendix E – Lagrange Multipliers

The use of Lagrange Multipliers allows one to minimize an equation *subject to another equation's constraint*. This is quite similar to what CAM does during the perturbation stage of its algorithm, with the significant difference that the network parameters of an ECM are generally not available in terms of a closed-form expression. Nevertheless, general-purpose optimizers are loosely based on some of the same ideas as those that go into the use of Lagrange Multipliers, and for this reason we discuss the method in-depth. Mathematically stated, Method of Lagrange Multipliers solves the problem

$$\{\min\ f(\mathbf{x}) : g(\mathbf{x}) = 0\}$$

where $g(\mathbf{x}) = 0$ is the constraint equation. The solution to minimizing $f(\mathbf{x})$ can be motivated with the following thought experiment: Assume one is made to close one's eyes and walk along a path (*the constraint*) that encircles a set of rolling hills, constantly finding oneself walking uphill one minute and downhill the next. How do you locate the minimum height (*the function*) of the hill *along the path prescribed*? Let $f(\mathbf{x})$ represent the height of the hill in any location. For an $\mathbf{x}$ on the path, if moving a little bit forwards or back along the path doesn't change the height, one has discovered what must be either a height extremum or a saddle point. This implies the gradient of $f(\mathbf{x})$ must be orthogonal to the path $g(\mathbf{x})$ (i.e., walking at a right-angle to the path would be the steepest ascent or descent you could make on the hill). One final trick: Since we constrained $g(\mathbf{x})$ to be equal to zero, its gradient must *also* be orthogonal to the path itself! Mathematically, then, a minimum of $f(\mathbf{x})$ can only occur when its gradient "lines up with" (is a multiple of) $g(\mathbf{x})$'s:

$$\nabla f(\mathbf{x}) = \alpha \nabla g(\mathbf{x}) \tag{E.1}$$

where α is an arbitrary constant. This equation is commonly seen in the form of

$$\nabla f(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) = 0 \tag{E.2}$$

with λ $(= -\alpha)$ referred to as the Lagrange multiplier. Again, Eq. (E.2) is requisite but not sufficient for a minimum: it may also locate a maximum or a saddle point.

A straightforward extension to the method of Lagrange multipliers is solving

$$\{\min\ f(\mathbf{x}) : g(\mathbf{x}) \leq 0\}$$

To extend the analogy, $g(\mathbf{x})$ now defines one or more *areas* on the hills where one might roam. Extrema of $f(\mathbf{x})$ will either occur *within* an area – in which case $\nabla f(\mathbf{x}) = 0$ and $\lambda = 0$ as well – or it will occur along the *perimeter* of an area, in which case the same reasoning as before applies. In both cases, Eq. (E.2) holds true, but additional restrictions are needed to insure the sign of $g(\mathbf{x})$: If we move a little in the direction *opposite* of $\nabla g(\mathbf{x})$, $g(\mathbf{x})$ will either decrease to something now less than zero or remain constant; if $f(\mathbf{x})$ *increases* in such a case, we've found our minimum. In other words, a minima can only occur when moving in a given direction decreases $g(\mathbf{x})$ but increases $f(\mathbf{x})$; this implies that the gradient of f must be opposite that of g or, equivalently, λ in Eq. (E.2) must be negative (or zero). Similarly, a constraint of $g(\mathbf{x}) \geq 0$ would require $\lambda \geq 0$.[1] In all cases, such conditions are requisite but not sufficient to finding a minima.

Obtaining the solution $\mathbf{x}$ that minimizes $f(\mathbf{x})$ is typically performed by finding the extrema of the *unconstrained* function

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x}) \tag{E.3}$$

in the usual manner, by setting all partial derivatives equal to zero or, in vector notation,

$$\nabla L(\mathbf{x}, \lambda) = \nabla f(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) = 0$$

which of course is just Eq. (E.2); substituting the solution $\mathbf{x}$ back into Eq. (E.3) provides $f(\mathbf{x})$, as $g(\mathbf{x}) = 0$ at these points.

[1]Eq. E.2 is sometimes written as $\nabla f(\mathbf{x}) = \lambda \nabla g(\mathbf{x})$, in which case the necessary signs of λ are reversed.

Bibliography

[1] S. Lin and E.S. Kuh, “Transient Simulation of Lossy Interconnects Based on the Recursive Convolution Formulation,” *IEEE Trans. Circuits Syst.*, vol. 39, pp. 879-892, Nov. 1992.

[2] A. Semlyen, A. Dabuleanu, "Fast and Accurate Switching Transient Calculation on Transmission Lines With Ground Return Using Recursive Convolution," *IEEE Trans. Power Apparatus Syst.*, vol. PAS-94, pp. 56-1575, 1975.

[3] R. Achar, M. S. Nakhla, “Simulation of High-Speed Interconnects,” *Proc. IEEE*, vol. 89, no. 5, pp. 693-728, May 2001.

[4] M. Celik and A. C. Cangellaris, “Simulation of Dispersive Multiconductor Transmission Lines by Padé Approximation via the Lanczos process,” *IEEE Trans. Microwave Theory Tech.*, vol. 44, No. 12, pp. 2525-2535, Dec. 1996.

[5] B. Gustavsen and A. Semlyen, “Rational Approximation of Frequency Domain Responses by Vector Fitting,” *IEEE Trans. Power Delivery*, vol. 14, no. 3, pp. 1052-1061, July 1999.

[6] C. P. Coelho, J. R. Phillips, and L. M. Silveira, “Robust Rational Approximation Algorithm for Model Generation,” *Proc. ACM/IEEE Design Automat. Conf.*, New Orleans, 1999.

[7] L. M. Silveira, I. M. Elfadel, J. K. White, M. Chilukuri, and K.S. Kundert, “An Efficient Approach to Transmission Line Simulation Using Measured or Tabulated S-parameter Data,” *Proc. ACM/IEEE Design Automat. Conf.*, pp. 634-639, June 1994.

[8] A. Dounavis, E. Gad, R. Achar, and M. S. Nakhla, “Passive Model Reduction of Multiport Distributed Interconnects,” *IEEE Trans. Microwave Theory Tech.*, vol. 48, no. 12, pp. 2325-2334, Dec. 2000.

[9] M. Elzinga, K.L. Virga, L. Zhao, J.L. Prince, “Pole-Residue Formulation for Transient Simulation of High-Frequency Interconnects Using Householder LS Curve-Fitting Techniques,” *IEEE Trans. Adv. Packag.*, vol. 23, no. 2, pp. 142-147, May 2000.

[10] W.T. Beyene and J.E. Schutt-Aine, “Efficient transient simulation of high- speed interconnects characterized by sampled data,” *IEEE Trans. Comp., Packag. Manufact. Technol. B*, vol. 21, pp. 105-114, Feb. 1998.

[11] E. C. Levy, “Complex-Curve Fitting,” *IRE Trans. Automat. Contr.*, vol. 4, pp. 37-43, May 1959.

[12] W. Cauer, "Ausgangsseitig leerlaufende Filter," *Elek. Nachr.-Tech.*, vol. 16, no. 6, pp. 161-163, 1939.

[13] Ernst A. Guillemin, *Synthesis of Passive Networks*, John Wiley & Sons, 1957.

[14] T. Mangold and P. Russer, "Full-Wave Modeling and Automatic Equivalent-Circuit Generation of Millimeter-Wave Planar and Multilayer Structures," *IEEE Trans. Microwave Theory Tech.*, vol. 47, no. 6, pp. 851-858, June 1999.

[15] S. Min and M. Swaminathan, “Efficient Construction of Two-Port Passive Macromodels for Resonant Networks,” *Proc. IEEE 10th Topical Meeting Elect. Perform. Electron. Packag. (EPEP’2001)*, pp. 229-232, Oct. 2001.

[16] L. Daniel, and J. Phillips,“Model Order Reduction for Strictly Passive and Causal Distributed Systems,” *Proc. ACM/IEEE Design Automat. Conf.*, June, 2002.

[17] D. Saraswat, R. Achar, M.S. Nakhla, “A Fast Algorithm and Practical Considerations for Passive Macromodeling of Measured/Simulated Data,” *IEEE Trans. Adv. Packag.*, vol. 27, no. 1, pp. 57-70, Feb. 2004.

[18] B. Gustavsen and A. Semlyen, “Enforcing Passivity for Admittance Matrices Approximated by Rational Functions,” *IEEE Trans. Power Systems*, vol. 16, no. 1, Feb. 2001.

[19] S. Grivet-Talocia, “Passivity Enforcement via Perturbation of Hamiltonian Matrices,” *IEEE Trans Circuits Syst. I*, vol. 51, no. 9, Sept. 2004.

[20] Peter V. O’Neil, *Advanced Engineering Mathematics*, 6th edition, Thomson Engineering, March 2006.

[21] A. L. Cauchy, "Sur La Formule de Lagrange Relative a l’Interpolation," *Analyse Algebrique*, Paris, 1821.

[22] Krishnamoorthy Kottapalli, Tapan K. Sarket, Yinbo Hua, Edkund K. Miller, Gerald J. Burker, "Accurate Computation of Wide-Band Response of Electromagnetic Systems Utilizing Narrow-Band Information," *IEEE Trans. Microwave Theory Tech.,* vol. 39, no. 4, pp. 682-687, April 1991.

[23] P. R. Graves-Morris, D. E. Robers, and A. Salam, "The Epsilon Algorithm and Related Topics," *Journal of Computation and Applied Mathematics*, vol. 122, no 1-2, Oct. 2000.

[24] I. D. Cooper and P. R. Graves-Morris, "The Rise and Fall of the Vector Epsilon Algorithm," *Journal of Numerical Algorithms*, vol. 5, no. 6, June 1993.

[25] K. L. Choi and M. Swaminathan, "Development of Model Libraries for Embedded Passives Using Network Synthesis," *IEEE Trans. Circuit Syst. II*, vol. 47, pp. 247-269, April 2000.

[26] Vatché Vorpérian, *Fast Analytical Techniques for Electrical and Electronic Circuits,* Cambridge University Press, 2002.

[27] Robert A. Chipman, *Theory and Problems of Transmission Lines,* McGraw-Hill, 1968.

[28] David M. Pozar, *Microwave Engineering*, 3rd edition, John Wiley and Sons, 2004.

[29] Ralph S. Carson, *High-Frequency Amplifiers*, John Wiley & Sons, 2nd. edn., July 1982.

[30] Steve C. Cripps, *RF Power Amplifiers for Wireless Communications,* 2nd edition, Artech House, 2006.

[31] S. Boyd, V. Balakrishnan, and P. Kabamba, "A Bi-section Method for Computing the H_∞ Norm of a Transfer Matrix and Related Problems," *Linear Matrix Inequalities in System and Control Theory*, SIAM, 1994.

[32] Stephen Boyd, Laurent El Ghaoul, Eric Feron, and Venkataramanan Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, Society for Industrial & Applied Mathematics (SIAM), June 1997.

[33] Richard C. Dorf, *Modern Control Systems*, 10th edition, Prentice Hall, 2004.

[34] Christian Schröder, Tatjana Stykel, "Passivation of LTI Systems," Preprint 368, DFG Research Center Matheon, Technische Universität Berlin, 2007.

[35] Stefano Grivet-Talocia, personal correspondence.

[36] J. Kolstad, C. Blevins, J. M. Dunn, and A. Weisshaar, "A New Modeling Methodology for Passive Components Based on Black-Box Augmentation Combined with Equivalent Circuit Perturbation," *Proc. IEEE 13th Topical Meeting Elect. Perform. Electron. Packag. (EPEP 2004)*, Oct. 2004, pp. 259-262.

[37] J. Kolstad, C. Blevins, J. M. Dunn, and A. Weisshaar, "A New Circuit Augmentation Method for Modeling of Interconnects and Passive Components," *IEEE Trans. Adv. Packag.*, vol. 29, no. 1, pp 66-77, Feb. 2006.

[38] W. Huyer and A. Neumaier, "Global Optimization by Multilevel Coordinate Search," *J. Global Optimization*, vol. 14, pp. 331-355, 1999.

[39] D. R. Jones, “The DIRECT Global Optimization Algorithm” *Encyclopedia of Optimization*, 1999.

[40] R. S. Adve, T. K. Sarkar, S. M. Rao, E. K. Miller, and D. R. Pflug, "Application of the Cauchy Method for Extrapolating/Interpolating Narrow-Band System Responses," *IEEE Trans. on Microwave Theory and Tech.*, vol. 45, no. 5, pp. 837-845, May 1997.

[41] R. W. Hammin, *Numerical Methods for Scientists and Engineers*, Dover Publications, 2nd. edn., 1973.

[42] Gene H. Golub, Charles F. Van Loan, *Matrix Computations*, John Hopkins University Press, 3rd. edn., Oct. 1996.

[43] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C++*, Cambridge University Press, 2nd. edn., Feb. 2002.

[44] B. Gustavsen and A. Semlyen, "Simulation of Transmission Line Transients Using Vector Fitting and Modal Decomposition," PE-347-PWRD-0-01-1997, *IEEE/PES Winter Meeting,* 1997.

[45] C.K. Sanathanan, and J. Koerner, “Transfer function synthesis as a ratio of two complex polynomials”, *IEEE Trans. Automatic Control*, vol. 8, no. 1, pp. 56-58, Jan. 1963.

[46] A. Watson, D. Melendy, P. Francis, K. Hwang, A. Weisshaar, “A Comprehensive Compact Modeling Methodology for Spiral Inductors in Silicon-Based RFICs,” *IEEE Trans. Microwave Theory Tech.*, vol. 52, no. 3, pp. 849-857, March 2004.

[47] Adam C. Watson, *Analysis and Modeling of Single-Ended and Differential Spiral Inductors in Silicon-Based RFICs*, Oregon State University, Microwaves Group, December 2003.

[48] Daniel Melendy, "Modelling of On-Chip Spiral Inductors for Silicon RFICs," Masters Thesis, Oregon State University, November 2002.

[49] R. W. Rhea, "A Multimode High-Frequency Inductor Model," *Applied Microwave and Wireless,* pp. 70-80, Nov./Dec. 1997.

[50] High-Frequency Structure Simulator, Ansoft Corporation, Pittsburgh, PA.

[51] S. Grivet-Talocia, Huang Hao-Mind, A. E. Ruehli, F. Canavero, I. M. Elfadel, "Transient analysis of lossy transmission lines: an efficient approach based on the method of Characteristics," *IEEE Trans. Adv. Packag.*, vol. 27, no. 1, pp. 45-56, Feb. 2004.

[52] Y. Ou, N. Barton, R. Fetche, N. Seshan, T. Fiez, U. Moon, and K. Mayaram, "Phase Noise Simulation and Estimation Methods: A Comparative Study," *IEEE Trans. CAS II*, pp. 635-638, Sept. 2002.

[53] Gabor C. Temes, and Jack W. LaPatra, *Introduction to Circuit Synthesis and Design,* McGraw Hill Higher Education, August, 1977.

[54] D. Paul, M. S. Nakhla, R. Achar, and A. Weisshaar, "An Automated Algorithm for Broadband Modeling of High-Frequency Microwave Devices," *IEEE Microwave Symposium Digest*, IEEE MTT-S Intl., pp. 1767-1770, June 2006.

[55] L. W. Nagel, *SPICE2: A Computer Program to Simulate Semiconductor Circuits,* Berkeley, University of California, Electronic Research Laboratory. ERL-M520, 1975.

[56] Kartikeya Mayaram, "CODECS: A Mixed-Level Circuit and Device Simulator," Memo No. UCB/ERL M88/77. Electronics Research Laboratory, University of California, Berkeley, Nov. 1988.

[57] L. T. Quarles, "The SPICE 3 Implementation Guide," Technical Report No. ERL-M89/44, University of California, Berkeley, Technical Report. No. ERL-M89/44, 1989.

[58] Lawrence T. Pillage, Ronald A. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis," *IEEE Trans. on Computer-Aided Design*, vol. 9, no. 4, pp. 352-366, April 1999.

[59] Peter Feldmann, "Efficient Linear Circuit Analysis by Padé approximation via the Lanczos Process," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 5, pp. 639-649, May 1995.

[60] Microwave Office, Applied Wave Research, El Segundo, CA.

[61] Gilbert Strang, *Introduction to Linear Algebra*, 3rd edition, Wellesley Cambridge Press, March 2003.

[62] Richard C. Aster, Brian Borchers, and Clifford H. Thurber, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, 2005.

[63] G. Casella and R. L. Berger, *Statistical Inference*, 2nd edition, Duxbury, Pacific Grove, CA, 2002.

[64] Dennis S. Berstein, *Matrix Mathematics: Theory, Facts, and Formulas with Application to Linear Systems Theory*, Princeton University Press, Feb. 22, 2005.

[65] Gilbert Strang, *Linear Algebra and Its Applications*, 4th edition, Brooks Cole, July 2005.

[66] Roger B. Marks, Dylan F. Williams, "A General Waveguide Circuit Theory," *J. Res. Natl. Inst. Stand. Technol.,* vol. 97, no. 4, pp. 533-562, July-Aug. 1992.