Adjoint Sensitivity Analysis of High Frequency Structures with MATLAB®

Mohamed Bakr, Atef Elsherbeni, and Veysel Demir
Adjoint Sensitivity Analysis of High Frequency Structures with MATLAB®
The ACES Series on Computational Electromagnetics and Engineering (CEME)

Andrew F. Peterson, PhD – Series Editor

The volumes in this series encompass the development and application of numerical techniques to electrical systems, including the modeling of electromagnetic phenomena over all frequency ranges and closely related techniques for acoustic and optical analysis. The scope includes the use of computation for engineering design and optimization, as well as the application of commercial modeling tools to practical problems. The series will include titles for undergraduate and graduate education, research monographs for reference, and practitioner guides and handbooks.

Series Editorial Board
Andrew F. Peterson – Series Editor, Georgia Institute of Technology
Atef Z. Elsherbeni – Associate Series Editor, Colorado School of Mines
James C. Rautio, Associate Series Editor, Sonnet Software, Inc.
Branislav M. Notaros, Associate Series Editor, Colorado State University

Titles in the Series


Yu, Yang, and Li – VALU, AVX and GPU Acceleration Techniques for Parallel FDTD Methods (2013)

Adjoint Sensitivity Analysis of High Frequency Structures with MATLAB®

ACES Series

Mohamed Bakr
McMaster University

Atef Elsherbeni
Colorado School of Mines

Veysel Demir
Northern Illinois University

ScITECH Publishing
an imprint of the IET

theiet.org
To my parents, my wife, Mahetab, and my children Omar, Youssef, Jannah, and Noor

— M. Bakr

To the memory of my parents, my wife, Magda, my daughters, Dalia and Donia, and my son Tamer

— A. Elsherbeni

To my parents Abdurrahman and Aysan, my wife Minmei, my daughter Laureen, and my son Ronnie

— V. Demir
## Contents

Preface ix  
Acknowledgments xi  
Ebook Version and MATLAB Codes xii  

1 Introduction to sensitivity analysis approaches 1  
  1.1 Introduction 1  
  1.2 Finite difference approximations 2  
  1.3 Adjoint sensitivity analysis of linear systems 6  
References 25  

2 Introduction to FDTD 27  
  2.1 Basic equations 27  
  2.2 FDTD updating equations for three-dimensional space 29  
  2.3 FDTD updating equations for two-dimensional space 37  
  2.4 FDTD updating equations for one-dimensional space 40  
  2.5 Dispersive material modeling 41  
    2.5.1 Modeling Lorentz medium using ADE technique 42  
    2.5.2 Modeling Drude medium using ADE technique 44  
  2.6 Anisotropic material modeling 44  
References 47  

3 The adjoint variable method for frequency-independent constitutive parameters 49  
  3.1 Introduction 49  
  3.2 The 1D case 50  
  3.3 The 2D TM case 62  
  3.4 The 3D AVM algorithm 85  
References 98  

4 Sensitivity analysis for frequency-dependent objective functions 99  
  4.1 The monochromatic case 99  
  4.2 The wideband case 107  
  4.3 The self-adjoint case 122  
References 132
Sensitivity analysis is a powerful tool in modeling and computer-aided design optimization of high-frequency structures. It determines how a response or an objective function is sensitive to different parameters. These parameters include shape parameters or material parameters of different components. The responses may be time-domain responses such as the electric field measured by a probe at a specific point in space. They can also represent a frequency-domain response such as the S-parameters of an antenna over the band of interest. The acquired sensitivity information can be used to construct local surrogate models that approximate a time-intensive electromagnetic simulation. It can also be used to guide the steps taken by an optimization algorithm as it aims at satisfying the design specification. First-order sensitivities are the ones most commonly used. Second-order sensitivities are also useful. They are rarely used though because of the associated computational cost using classical finite difference approaches.

The adjoint variable method (AVM) offers an efficient way for estimating first- and second-order sensitivities. Using at most one extra simulation, the complete first-order sensitivities of a response or an objective function are estimated. The classical finite difference approaches require at least $n$ extra simulations, where $n$ is the number of parameters. The AVM also reduces the number of simulations required for estimating the second-order (Hessian) sensitivities from $O(n^2)$ to only $n$. The reduction in the computational cost is significant for high-frequency structures with intensive simulation time or for structures with a large number of parameters.

There are many publications in the area of adjoint sensitivity analysis. The feedback we received over the years from fellow researchers in the scientific community that they find these algorithms complicated. Researchers in other disciplines such as power engineering and biomedical engineering, who would benefit greatly from this work, find it difficult to understand and program these algorithms. Some commercial solvers are starting to offer some limited AVM capabilities. This, however, does not satisfy researchers with advanced applications.

We took it on ourselves to write a book with a simple and unified approach to different AVM techniques. We decided to implement all these techniques using the popular FDTD numerical approach. This will hopefully help the largest possible number of researchers understand and use these techniques. The approach adopted in this book is to first explain the theory in detail and with as much clarifications as possible. Then we move to implement the theory on simple examples using MATLAB. We give extensive codes with detailed explanation to clarify any confusing points. The readers will realize that published results that use more
complex structures can be implemented using the supplied codes after making some changes. The target here is to stress and illustrate the basic concepts rather than handle complex structures.

We discuss in this book problems that utilize different types of materials and responses. We show how to estimate the sensitivities of scalar objective functions, transient responses, and frequency-domain responses such as the $S$-parameters. We address problems with isotropic, anisotropic, dispersive, and nonlinear materials. We solve problems with 1D, 2D, or 3D FDTD solvers. Our target is to show the reader the universal nature of the adjoint sensitivity techniques and the wide applications it can be utilized in.

The book does not assume prior knowledge of the FDTD technique or of sensitivity analysis approaches. Chapter 1 gives a simple introduction to AVM approaches using simple electric circuits. Chapter 2 is dedicated to give an introduction to the FDTD methods and its application in modeling different types of materials. Chapter 3 introduces the foundation needed to apply the AVM approach using FDTD. The other chapters apply the AVM approach to estimate first- and/or second-order sensitivities of simple high-frequency structures with different types of materials and for different responses. The codes presented in this book are for educational purposes. They can be optimized to further improve the memory storage and the computational time.

We hope that the readers will find this book useful and will be able to build on the supplied codes to produce interesting modeling and optimization results.
Acknowledgments

The authors would like to acknowledge Dr. Natalia Nikolova from McMaster University for her work on adjoint sensitivities. The first author collaborated with her for a number of years on developing adjoint sensitivity analysis tools. This initial work is what led to more advanced publications on wideband adjoint sensitivity analysis of dispersive, anisotropic, and nonlinear high-frequency structures.

The authors would also like to acknowledge Dr. Tsuyoshi Nomura with Toyota Labs, previously with the Toyota Research Center of North America (TEMA). Dr. Nomura was involved in developing the initial wideband adjoint sensitivity analysis techniques for dispersive media using the Transmission Line Matrix (TLM) method.

We would like to recognize the work of the former graduate students: Dr. Yu Zhang, Dr. Osman Ahmed, Dr. Peter Basl, and Mr. Mohamed Negm. Their hard work was instrumental in forming the ideas that gave rise to this book. We would also like to recognize Joseph Denier, Mahmoud Maghrabi, and Ahmed Elsharabasy for their help in developing and testing some of the codes and figures in this book.

We are grateful to our parents and families for being supportive of our endeavor. Their patience while this material was prepared is greatly appreciated.

Finally, we thank God the Almighty for giving us the strength to see this work through. We hope that he accepts our sincere intention of making this book useful and helpful for others.

Bakr, Elsherbeni, and Demir
November 2016
An ebook version of this book is available on the IET Digital Library at http://digital-library.theiet.org/content/books/ew/sbew525e.

The MATLAB codes used in this book are available at the same link. When you have visited the link, click on the “Supplementary material” tab to download the files.
Chapter 1

Introduction to sensitivity analysis approaches

1.1 Introduction

One of the important aspects of computer-aided engineering is sensitivity analysis (SA). It allows the designer to understand how the response of the considered system changes through changing different parameters. The higher the sensitivity of a response relative to a parameter, the more important this parameter becomes. If a response has lower sensitivity relative to a parameter, then this parameter should not be used to adjust the response to match the design specifications.

There are many applications to SA. Derivative-based optimization techniques [1] utilize sensitivity information to update the values of the optimized parameters at every optimization iteration. A typical optimization step in a sensitivity-based optimization algorithm is given by

$$p(k + 1) = p(k) + \alpha s(k)$$

(1.1)

where $s(k)$ is the search direction, and $\alpha$ is the step taken in that direction. Staring from an initial set of values of the optimization parameters $p(0)$, the algorithm creates a sequence of points in the parameter space. At the $k$th iteration, starting from the point $p(k)$, the algorithm takes a step $\alpha$ in a search direction $s(k)$. This direction, in most cases, utilizes some sensitivity information. For example, the steepest descent method [2] uses $s(k) = -g(k)$, where $g(k)$ is the first-order sensitivities (gradient) of the objective function to be minimized. Conjugate gradient techniques [3] utilize a search direction $s(k)$ that is derived using the gradient.

Also, first- and second-order sensitivity information can be used to build a Taylor-based surrogate model of time-intensive simulations [3]. Using the gradient information and second-order sensitivities (Hessian) information, an approximate Taylor expansion of the response of interest is constructed. This model is given by

$$f(p + \Delta p) \approx f(p) + g^T \Delta p + 0.5 \Delta p^T H \Delta p,$$

(1.2)

where the superscript $T$ is used throughout the book to denote the transpose of a vector. The cost of constructing the surrogate model (1.2) can be significant. For a structure with $n$ parameters, the gradient requires $O(n)$ extra simulations, while the Hessian matrix $H$ requires $O(n^2)$ extra simulations using classical finite difference methods. Once the surrogate model (1.2) is available, it can be used to predict the system response without having to carry out any extra simulation of the original
system. The surrogate model approach is particularly helpful for systems whose simulation time is extensive.

One of the other interesting applications for sensitivity information is yield and statistical analyses. Yield analysis allows us to predict, given the random changes in the values of the parameters after fabrication, the percentage of the fabricated systems that meet the design specifications. The yield integral is usually evaluated through a Monte Carlo approach [4] where the system is repeatedly simulated with randomly perturbed parameter values. Having the sensitivity information readily available enables faster evaluation of the yield integral.

Classically, the sensitivity information is obtained through finite difference approximations. By perturbing the parameter values, estimates of the first- and second-order sensitivities are obtained. The main problem with this approach is the computational cost associated with it. The number of required simulations scales linearly with the number of parameters for first-order derivatives and scales quadratically for second-order sensitivities. This makes the cost of estimating the response sensitivities prohibitive for time-intensive simulations or for systems with large number of parameters.

Adjoint SA approaches aim at addressing these challenges. Utilizing only one system simulation, all first-order adjoint sensitivities are estimated regardless of their number. Adjoint approaches utilize an adjoint simulation of the original system. Using responses of the original simulation and the adjoint simulation, all components of the gradient vector are estimated.

In this chapter, we give an initial introduction to adjoint techniques. We first review the classical approaches for derivative estimation and illustrate their computational cost. Approaches such as forward finite differences (FFD), backward finite differences (BFD), and central finite differences (CFD) are addressed. These approaches are utilized in later chapters as a reference to compare the estimated adjoint sensitivities against. We then present an adjoint sensitivity formulation that is suitable for analyzing electric circuits. The adjoint SA of electrical circuits and conductor transmission lines was addressed by several researchers, for example, [5,6]. This analysis serves as a smooth introduction to the basic concepts involved in adjoint analysis of high-frequency structures. The same theorem is extended to high-frequency electromagnetic simulations as will be illustrated in the following chapters. We illustrate the theory presented in this chapter with circuit examples.

1.2 Finite difference approximations

We start by considering the classical finite difference approximations of derivatives. We assume throughout the book that the parameters of interest are given by the vector $p = [p_1, p_2 \ldots p_n]^T$, where $n$ is the number of the parameters. These parameters may represent the values of resistors, inductors, and capacitors of an electric circuit. They may also represent the values of different dimensions and material properties of a high-frequency structure. The response of the system may be the voltage across an output resistor, the power delivered to a load, or the power
dissipated in the circuit. It may also be the S-parameters, the temporal electric field values, or the radiation pattern of an electromagnetic structure. We denote such a response by \( f = f(p) \). The system response \( f \) is thus dependent on the parameter values. The first-order sensitivities of \( f(p) \) with respect to the parameters are obtained by perturbing the parameters one at a time. The derivative of \( f \) with respect to the \( i \)th parameter is given by

\[
\frac{\partial f}{\partial p_i} = \lim_{\Delta p_i \to 0} \frac{f(p_1, p_2, \ldots, p_i + \Delta p_i, \ldots, p_n) - f(p_1, p_2, \ldots, p_i, \ldots, p_n)}{\Delta p_i}. \tag{1.3}
\]

Formula (1.3) defines the \( i \)th component of the gradient vector

\[
\nabla f = \left[ \frac{\partial f}{\partial p_1} \frac{\partial f}{\partial p_2} \cdots \frac{\partial f}{\partial p_n} \right]^T. \tag{1.4}
\]

The limit in (1.3) implies that the perturbation in the \( i \)th parameter \( \Delta p_i \) should be as small as possible to get an accurate result. Formula (1.3) is approximated by the forward difference formula

\[
\frac{\partial f}{\partial p_i} \approx \frac{f(p_1, p_2, \ldots, p_i + \Delta p_i, \ldots, p_n) - f(p_1, p_2, \ldots, p_i, \ldots, p_n)}{\Delta p_i}. \tag{1.5}
\]

Formula (1.5) requires two simulations of the system under consideration. The first one is the nominal simulation at the current parameter values resulting in the response \( f(p_1, p_2, \ldots, p_i, \ldots, p_n) \). The second simulation involves perturbing the \( i \)th parameter in the forward direction (increasing its value) to give \( f(p_1, p_2, \ldots, p_i + \Delta p_i, \ldots, p_n) \). Formula (1.5) is denoted as the FFD formula because of the way the parameter is perturbed. Even though this formula recommends using as small perturbation \( \Delta p_i \) as possible, in practice, this perturbation cannot be arbitrarily small. Numerical simulators used to model different circuits and systems usually have an accuracy limit. Reducing the perturbation \( \Delta p_i \) below a certain value results in corrupted sensitivity information.

There are other formulas for estimating the first-order derivatives that can be derived using Taylor’s expansion. These formulations utilize different perturbations for estimating the sensitivities. In addition to the FFD formulation, the BFD and CFD are also often utilized. These two formulas are given, respectively, by

\[
\frac{\partial f}{\partial p_i} \approx \frac{f(p_1, p_2, \ldots, p_i, \ldots, p_n) - f(p_1, p_2, \ldots, p_i - \Delta p_i, \ldots, p_n)}{\Delta p_i}, \tag{1.6}
\]

\[
\frac{\partial f}{\partial p_i} \approx \frac{f(p_1, p_2, \ldots, p_i + \Delta p_i, \ldots, p_n) - f(p_1, p_2, \ldots, p_i - \Delta p_i, \ldots, p_n)}{2\Delta p_i}. \tag{1.7}
\]

The BFD formula perturbs the \( i \)th parameter in the backward direction thus decreasing its value. Similar to the FFD formula, it requires only one extra simulation per parameter. Both FFD and BFD require \( n \) extra simulations each to estimate all sensitivities of the function \( f \) with respect to the parameters \( p \). The CFD
formula (1.7) requires perturbing each parameter in both the forward and backward directions. It thus requires \(2n\) extra simulations to estimate all \(n\) sensitivities. It can be shown that the CFD is more accurate than the FFD and BFD.

**Example 1.1** Estimate the sensitivity of the single-variable function \(f(p) = 3p^2 - p + 1\) at the point \(p = 4.0\) both analytically and numerically.

The function \(f\) is a scalar function of a single scalar parameter \(p\). The analytical derivative of this function at a general point \(p\) is obtained by differentiating with respect to the function argument

\[
f'(p) = 6p - 1
\]

At the point \(p = 4.0\), the derivative is given by

\[
f'(4) = 6 \times 4 - 1 = 23
\]

The same derivative can be estimated using finite difference approximations. Assuming a perturbation of \(\Delta p = 0.05\), the forward and BFD estimates are given by

\[
f'(4) \approx f(4.05) - f(4) = \frac{3 \times (4.05)^2 - 4.05 + 1 - 3 \times (4)^2 + 4 - 1}{0.05} = 23.149999
\]

\[
f'(4) \approx f(4) - f(3.95) = \frac{3 \times (4)^2 - 4 + 1 - 3 \times (3.95)^2 + 3.95 - 1}{0.05} = 22.849999
\]

It is obvious that the FFD approximation overestimates the derivative, while the BFD underestimates it. Each one of these estimates requires only one extra simulation in addition to the original one at \(p = 4.0\). The more costly CFD estimate is given by

\[
f'(4) \approx f(4.05) - f(3.95) = \frac{3 \times (4.05)^2 - 4.05 + 1 - 3 \times (3.95)^2 + 3.95 - 1}{2 \times 0.05} = 22.999999
\]

It is obvious the CFD estimates the sensitivity with very little error at the expense of utilizing two simulations. The accuracy of all three estimates can be improved by using a smaller perturbation \(\Delta p\). However, for functions estimated using numerical solvers, \(\Delta p\) cannot be made arbitrarily small. The numerical noise of the solver starts to contaminate the estimated responses and derivatives below a certain \(\Delta p\). The perturbation \(\Delta p\) is usually selected between 1% and 5% of the nominal parameter value.

We next consider a three-parameter function and show how numerical and analytical sensitivities are estimated for this function.
**Example 1.2** Estimate the gradient of the function

\[ f(p) = p_1 p_2 p_3 + 2p_1^2 p_2 + 2p_2^2 p_3 + 2p_1 p_3^2 \]

at the point \( p_o = [2.0 \quad 3.2 \quad 1.75]^T \) using FFD, BFD, and CFD. Compare your answers to the analytical case. Utilize the perturbations \( \Delta p = [0.05 \quad 0.1 \quad 0.05]^T \).

The analytical derivatives of this function at an arbitrary point \( p \) are given by

\[
\begin{align*}
\frac{\partial f}{\partial p_1} &= p_2 p_3 + 2(2p_1 p_2 + p_3^2), \\
\frac{\partial f}{\partial p_2} &= p_1 p_3 + 2(p_1^2 + 2p_2 p_3), \\
\frac{\partial f}{\partial p_3} &= p_1 p_2 + 2(2p_1 p_3 + p_2^2).
\end{align*}
\]

At the point \( p_o = [2.0 \quad 3.2 \quad 1.75]^T \), the analytical gradient is given by

\[

\nabla f_{p_o} = \begin{bmatrix}
\frac{\partial f}{\partial p_1} \\
\frac{\partial f}{\partial p_2} \\
\frac{\partial f}{\partial p_3}
\end{bmatrix}_{p_o} = \begin{bmatrix} 37.325 \\ 33.900 \\ 40.880 \end{bmatrix}
\]

To evaluate this gradient using forward, backward, and central differences, we independently perturb each parameter and then use the formulas (1.5)–(1.7) to evaluate the sensitivities. The MATLAB® Listing M1.1 implements these formulas. The output from this code is

\[
\begin{align*}
\text{gradient\_analytic} &= 37.325000000000003 \\
&\quad 33.900000000000006 \\
&\quad 40.880000000000003 \\
\text{gradient\_forward} &= 37.644999999999982 \\
&\quad 34.249999999999972 \\
&\quad 41.080000000000041 \\
\text{gradient\_backward} &= 37.005000000000337 \\
&\quad 33.550000000000182 \\
&\quad 40.680000000000121
\end{align*}
\]
It is obvious from the results that the gradient estimated using the more expensive CFDs match better than the analytical gradient as expected.

1.3 Adjoint sensitivity analysis of linear systems

In the previous section, we addressed the classical approaches for sensitivity estimation. In this section, we focus on wideband adjoint SA of electric circuits. Through solving the differential equations governing the system in the time-domain, all transient responses are obtained. These responses are time-varying.
quantities whose values change by changing different parameters of the problem. Iterative numerical approaches are usually used for solving such problems. The adjoint approaches related to these systems offer a smooth introduction to the high-frequency case. For a good review of the narrow band case for linear circuits, the reader is referred to Chapter 10 of [7].

Without loss of generality, we assume that the considered system or circuit is governed by the following system of differential equations:

\[ T \dddot{E} + M \dot{E} + N E + KE = G, \]  

where \( E \in \mathbb{R}^n \) is the vector of state responses. \( \dot{E}, \ddot{E}, \) and \( \dddot{E} \) are its first-, second-, and third-order time derivatives. These state responses may represent voltages across capacitors or currents through inductors. The square matrices \( T, M, N, \) and \( K \) are the system matrices. These matrices are assumed to be functions of the parameters \( p. \) The vector \( G \in \mathbb{R}^m \) is the vector of time varying excitation of the system.

Our target is to find the sensitivities of a given objective function that depends on the temporal response \( E(t). \) This objective function may be given by the integral [8]:

\[ f = \int_0^{T_m} \psi(p, E) \, dt, \]  

where \( \psi \) is the kernel of the integral, and \( T_m \) is the simulation time. The kernel function may have an explicit dependence on \( p, \) and it may have also an implicit dependence on \( p \) through the temporal vector \( E. \) \( T_m \) is the time over which the system response is observed. Our target is to find the gradient of \( f \) with respect to the parameters \( p. \) From (1.9), the \( i \)th component of the gradient is given by

\[ \frac{\partial f}{\partial p_i} = \int_0^{T_m} \frac{\partial \psi}{\partial p_i} \, dt + \int_0^{T_m} \left( \frac{\partial \psi}{\partial E} \right)^T \frac{T}{\partial p_i} \, dt, \quad i = 1, 2, \ldots, n \]  

(1.10)

The first integral in (1.10) calculates the explicit derivative which is assumed to be known. The superscript \( e \) indicates the explicit derivative. The second integral requires the derivative of \( E \) with respect to \( p \) which is unknown. The classical way of calculating (1.10) involves solving (1.8) for perturbed values of the parameters \( p. \) At least \( n \) extra solutions of the system (1.8) are required to estimate all sensitivities of the function \( f. \)

The adjoint variable method estimates the derivatives in (1.10) using at most one extra simulation. The derivation of the method is simple and it is instructive to go over this simple derivation. This derivation starts by differentiating both sides of (1.8) with respect to the \( i \)th parameter \( p_i \) to get

\[ \frac{\partial T}{\partial p_i} \dddot{E} + T \frac{\partial \dot{E}}{\partial p_i} + M \frac{\partial \dddot{E}}{\partial p_i} + N \frac{\partial \dot{E}}{\partial p_i} + K \frac{\partial E}{\partial p_i} = \frac{\partial G}{\partial p_i}. \]  

(1.11)

In (1.11), we assume that the system matrices are analytical functions of the parameters. If these matrices are only available numerically, as in the case in many
applications, then finite differences can be used to approximate the system matrices derivatives with respect to the parameters. Multiplying both sides of (1.11) by the transpose of the temporal adjoint variable $\lambda$, whose value is to be determined later, we obtain the expression

$$
\int_0^{T_m} \left( \lambda^T T \frac{\partial \tilde{E}}{\partial p_i} + \lambda^T M \frac{\partial \tilde{E}}{\partial p_i} + \lambda^T N \frac{\partial \tilde{E}}{\partial p_i} + \lambda^T K \frac{\partial \tilde{E}}{\partial p_i} \right) dt = \int_0^{T_m} \lambda^T R_i dt,
$$

where $R_i = \frac{\partial G}{\partial p_i} - \frac{\partial T}{\partial p_i} \tilde{E} - \frac{\partial M}{\partial p_i} \tilde{E} - \frac{\partial N}{\partial p_i} \tilde{E} - \frac{\partial K}{\partial p_i} E$ \hspace{1cm} (1.12)

is the $i$th residue vector. Notice that the residue vector $R_i$ is known $\forall i$. This is because the vector $E$, and its derivatives are known once the original system (1.8) is solved. Also, the derivatives of the system matrices with respect to the parameters are known either analytically or numerically. The derivation proceeds by integrating the second-order term by parts twice and integrating the first-order derivatives once to get

$$
\lambda^T T \frac{\partial \tilde{E}}{\partial p_i} \bigg|_0^{T_m} - \lambda^T M \frac{\partial \tilde{E}}{\partial p_i} \bigg|_0^{T_m} + \lambda^T M \frac{\partial \tilde{E}}{\partial p_i} \bigg|_0^{T_m} - \int_0^{T_m} \lambda^T T \frac{\partial E}{\partial p_i} dt + \lambda^T M \frac{\partial \tilde{E}}{\partial p_i} \bigg|_0^{T_m} - \int_0^{T_m} \lambda^T M \frac{\partial \tilde{E}}{\partial p_i} dt + \int_0^{T_m} \lambda^T K \frac{\partial \tilde{E}}{\partial p_i} dt = \int_0^{T_m} \lambda^T R_i dt.
$$

The system (1.8) is assumed to start with zero initial conditions. This implies that energy storage elements such as capacitors and inductors have initial zero energy. It follows that $E(0) = \dot{E}(0) = \ddot{E}(0) = 0$. Also, to simplify (1.13), we impose zero terminal values on the adjoint variable, that is, $\lambda(T_m) = \dot{\lambda}(T_m) = \ddot{\lambda}(T_m) = 0$. Equation (1.13) is thus simplified into

$$
\int_0^{T_m} \left( -\ddot{\lambda}^T T + \dot{\lambda}^T M - \dot{\lambda}^T N + \lambda^T K \right) \frac{\partial E}{\partial p_i} dt = \int_0^{T_m} \lambda^T R_i dt.
$$

Comparing (1.14) to the implicit integral in (1.10), we see that if the adjoint vector satisfies

$$
-\ddot{\lambda}^T T + \dot{\lambda}^T M - \dot{\lambda}^T N + \lambda^T K = \left( \frac{\partial \psi}{\partial E} \right)^T \Rightarrow -T^T \ddot{\lambda} + M^T \dot{\lambda} - N^T \lambda + K^T \lambda = \frac{\partial \psi}{\partial E},
$$

then the sensitivities (1.10) are given by

$$
\frac{\partial f}{\partial p_i} = \int_0^{T_m} \frac{\partial \psi}{\partial p_i} dt + \int_0^{T_m} \lambda^T R_i dt, \forall i.
$$

(1.16)
It follows that by solving the original system (1.8) with zero initial values and then solving the adjoint system (1.15) with zero terminal values, we can estimate the sensitivities with respect to all the parameters regardless of their number. Notice that the \( i \)th residue \( R_i \), is known using the original simulation for all time and for all parameters.

Few things to notice about the procedure for estimating adjoint sensitivities: first, the excitation of the adjoint problem (1.15) is the temporal vector \( \partial \psi / \partial E \) which is estimated during the original system. Second, the adjoint system in (1.15) is similar to the original system with the adjoint matrices given by

\[
\hat{T} = -T^T, \quad \hat{M} = M^T, \quad \hat{N} = -N, \quad \text{and} \quad \hat{K} = K. \tag{1.17}
\]

The same numerical solver used for solving (1.8) can thus be used for solving (1.15) with the appropriate matrices. Actually, if we make a change of variables in (1.15) to use the adjoint time coordinate \( \tau = T_m - t \), the sign of the first- and third-order derivatives are reversed and (1.15) is written as

\[
T^T \ddot{\lambda} + M^T \dot{\lambda} + N^T \dot{\lambda} + K^T \lambda = \frac{\partial \psi(T_m - \tau)}{\partial E}, \tag{1.18}
\]

where all temporal derivatives in (1.18) are with respect to \( \tau \). In (1.18), we denote the adjoint excitation by \( Q(\tau) \). The same solver used to solve (1.8) with respect to \( t \) can thus solve (1.18) with respect to \( \tau \), the backward time coordinate of the adjoint problem.

Finally, the inner product in (1.16) is usually approximated by the summation

\[
\int_0^{T_m} \lambda^T R_i dt \approx \Delta t \sum_k \lambda^T(k\Delta t)R_i(k\Delta t). \tag{1.19}
\]

The expression (1.19) implies carrying out an inner product between the adjoint response and each temporal residue and then summing over all \( N_T \) time steps where \( T_m = N_T \Delta t \). The values of the original and adjoint responses are usually obtained through numerical techniques at discrete multiples of the time step \( \Delta t \). The following example illustrates the time-domain adjoint SA.

**Example 1.3** Consider the circuit shown in Figure 1.1. Find the derivatives of the energy dissipated in the resistor with respect to all parameters \( p = [R \ C \ L]^T \) using time-domain adjoint SA. The sensitivities are estimated at the values \( p_o = [50 \Omega \ 1.0 \mu F \ 1.0 \text{ mH}]^T \). The simulation time is \( T_m = 5.0 \text{ ms} \).

First, we write the differential equation governing this circuit using loop and nodal equations to obtain

\[
\begin{align*}
vs(t) &= L \frac{\partial i_L}{\partial t} + v_c \\
i_L &= i_R + i_C = \frac{v_c}{R} + C \frac{\partial v_c}{\partial t}
\end{align*}
\tag{1.20}
\]
Defining the state variables as the current through the inductor and the voltage through the capacitor \( E = [i_L, v_C]^T \), the two equations in (1.20) can be written in the matrix form

\[
\begin{bmatrix}
L & 0 \\
0 & C
\end{bmatrix}
\begin{bmatrix}
\frac{di_L}{dt} \\
\frac{dv_c}{dt}
\end{bmatrix}
+ \begin{bmatrix}
0 & 1 \\
-1 & \frac{1}{R}
\end{bmatrix}
\begin{bmatrix}
i_L \\
v_c
\end{bmatrix}
= \begin{bmatrix}
v_s \\
0
\end{bmatrix}.
\]

This system has the form (1.8) with the parameter-dependent system matrices

\[
T = M = \begin{bmatrix}
0 & 0 \\
0 & 0
\end{bmatrix}, \quad N = \begin{bmatrix}
L & 0 \\
0 & C
\end{bmatrix}, \quad K = \begin{bmatrix}
0 & 1 \\
-1 & \frac{1}{R}
\end{bmatrix}, \quad G = \begin{bmatrix}
v_s \\
0
\end{bmatrix}.
\]

The objective function of this problem is given by

\[
f = \int_0^{T_m} \psi(R, E) \, dt = \int_0^{T_m} \frac{v_c^2}{R} \, dt.
\]

It should be noted that this objective function has an explicit dependence on \( R \) as this variable appears within the integral. Its explicit derivative has to be evaluated separately and is given by the integral:

\[
\frac{\partial f}{\partial R} = \int_0^{T_m} -\frac{v_c^2}{R^2} \, dt \approx -\Delta t \sum_k \frac{v_c^2(k \Delta t)}{R^2}.
\]

According to the adjoint variable theory presented earlier, the corresponding adjoint system required to evaluate the sensitivities of the objective function (1.23) relative to all parameters is given by

\[
- \begin{bmatrix}
L & 0 \\
0 & C
\end{bmatrix}
\begin{bmatrix}
\frac{d\lambda_1}{dt} \\
\frac{d\lambda_2}{dt}
\end{bmatrix}
+ \begin{bmatrix}
0 & 1 \\
1 & \frac{1}{R}
\end{bmatrix}
\begin{bmatrix}
\lambda_1 \\
\lambda_2
\end{bmatrix}
= \frac{\partial \psi}{\partial E},
\]

Figure 1.1 The circuit of Example 1.3
where the adjoint excitation is given by

\[
\frac{\partial \psi}{\partial E} = \begin{bmatrix}
0 \\
\frac{2 v_c(t)}{R}
\end{bmatrix}.
\]  

(1.26)

By changing the time variable from \( t \) to \( \tau = T_m - t \), this adjoint system (1.25) can be written as

\[
\begin{bmatrix}
L & 0 \\
0 & C
\end{bmatrix} \begin{bmatrix}
\frac{d\lambda_1}{d\tau} \\
\frac{d\lambda_2}{d\tau}
\end{bmatrix} + \begin{bmatrix}
0 & -1 \\
1 & \frac{1}{R}
\end{bmatrix} \begin{bmatrix}
\lambda_1 \\
\lambda_2
\end{bmatrix} = \begin{bmatrix}
Q(t) = \frac{\partial \psi(T_m - \tau)}{\partial E}
\end{bmatrix}.
\]

(1.27)

The adjoint excitation in (1.27) is the same excitation determined in (1.26) but is applied backward in time. The same solver used to solve the original problem (1.21) at every value of the time variable \( t \) can also be used to solve (1.27) at every value of the adjoint time variable \( \tau \). It should be clear that the adjoint simulation is solved backward in time starting from time \( t = T_m \) (\( \tau = 0 \)).

The value of the adjoint excitation in (1.27) is determined during the original simulation. The original simulation (1.21) and the adjoint simulation (1.27) are solved for the original response \( E(t) \) and the adjoint response \( \lambda(t) \). The residues with respect to all parameters in (1.12) are time-varying vectors that are determined during the original simulation. These residues are given by

\[
R_1 = \frac{\partial K}{\partial R} E = \begin{bmatrix}
0 & 0 \\
0 & -\frac{1}{R^2}
\end{bmatrix} \begin{bmatrix}
i_L(t) \\
v_C(t)
\end{bmatrix} = \begin{bmatrix}
0 \\
-\frac{v_C(t)}{R^2}
\end{bmatrix},
\]

\[
R_2 = \frac{\partial N}{\partial L} \dot{E} = \begin{bmatrix}
0 & 0 \\
0 & 1
\end{bmatrix} \begin{bmatrix}
\frac{d i_L}{dt} \\
\frac{d v_C}{dt}
\end{bmatrix} = \begin{bmatrix}
0 \\
\frac{d v_C}{dt}
\end{bmatrix},
\]

\[
R_3 = \frac{\partial N}{\partial L} \dot{E} = \begin{bmatrix}
1 & 0 \\
0 & 0
\end{bmatrix} \begin{bmatrix}
\frac{d i_L}{dt} \\
\frac{d v_C}{dt}
\end{bmatrix} = \begin{bmatrix}
\frac{d i_L}{dt} \\
0
\end{bmatrix},
\]

(1.28)

where \( R_i \) is the residue vector related to the \( i \)th parameter where \( p = [R \quad C \quad L]^T \). Each one of these residues has only one non-zero component. The integration (1.19) is then executed by taking the inner product of the respective residue and adjoint response at the same time step and then integrating them over the simulation time.

A point that remains is how to solve the two systems (1.21) and (1.27) with one of them running forward in time and the other running backward in time. There are actually several approaches for solving this problem. In this example, we utilize second-order accurate finite differences to obtain a time marching scheme. We
discretize the first-order derivative of the state variables using a CFD approximation. The current value of the state variables is expressed as an average between the previous and the next value. Following these two steps, the system (1.21) is given by

\[
N \frac{(E^+ - E^-)}{2\Delta t} + K \frac{(E^+ + E^-)}{2} = G,
\]  

(1.29)

where \( E^+ = E((k+1)\Delta t) \), \( E^- = E((k-1)\Delta t) \), and \( G = G(k\Delta t) \). In the second term in (1.29), an averaging expression to express the field at the current time step in terms of the previous and next field values. Reorganizing (1.29), we obtain the time marching scheme:

\[
\left( \frac{N}{2\Delta t} + \frac{K}{2} \right) E^+ = \left( \frac{N}{2\Delta t} - \frac{K}{2} \right) E^- + G.
\]  

(1.30)

Using (1.30), the values of the state variables at time \( t = (k+1)\Delta t \) are evaluated using the current value of the excitation and the previous state values. Both sides of (1.30) are then multiplied by the inverse of the time-invariant matrix \((N/2\Delta t) + K/2\) to solve for the new values for the state variables. The system (1.30) is solved with zero initial conditions. The procedure illustrated in this example is implemented in the MATLAB Listing M1.2. The procedure implemented by this code in general but we use specific values for the circuit parameters for illustration.

Few things are to be noticed regarding the code in the MATLAB Listing M1.2. First, the adjoint excitation determined during the original simulation as a function of \( t \) is flipped in time using the MATLAB function \texttt{flipdim()} . This ensures that the adjoint excitation is applied backward in time. Second, the function \texttt{Integrate()} is the one that carries out the time marching scheme (1.30). It returns the two-state variables for all time steps. It is called once for the original system with a Gaussian excitation. It is called again with transposed system matrices and the adjoint excitation to give the adjoint response. The original and adjoint responses are shown in Figures 1.2 and 1.3, respectively. The \texttt{Integrate()} function is given in the MATLAB Listing M1.3. Finally, in the last three lines of the MATLAB Listing M1.2, the residue associated with every variable are multiplied by the adjoint response and integrated over time. The function \texttt{flipdim()} is used again to express the adjoint responses as functions of the time variable \( t \) of the original simulation. For the resistance \( R \), the total derivative is the sum of the explicit derivative and the derivative estimated using adjoint sensitivities. The MATLAB Listing M1.2 gives the following output:

\[
\begin{align*}
derivative_R &= -2.371624070479185e-07 \\
derivative_C &= 0.109583908562999 \\
derivative_L &= 6.491187422887372e-05
\end{align*}
\]
To verify these sensitivities, we utilize the more expensive finite-difference approximations. The MATLAB Listing M1.4 estimates the forward, backward, and CFD by perturbing the system parameters one at a time and then accordingly update the system matrices. The numerical solver implemented in the `Integrate()` function is then invoked to calculate the perturbed state variables. The objective function (1.23) is estimated through a summation. The results of running this listing are
Figure 1.2 The original response of the circuit of Example 1.3: (a) the inductor current $i_L$ and (b) the capacitor voltage $v_c$ as functions of the simulation time step.

Figure 1.3 The adjoint responses of the circuit of Example 1.3: (a) the adjoint response $\lambda_1$ and (b) the adjoint response $\lambda_2$ as functions of the time step.

gradient_forward =
-0.000000225928056
0.109701551691227
0.000064035676836

gradient_backward =
-0.000000249570090
0.109472801020545
0.000066228849919
These finite difference estimates match very well the ones obtained using adjoint SA. The CFD approach requires six extra simulations, while the adjoint approach required only one extra simulation.
Example 1.4 We consider the RLC circuit shown in Figure 1.4. We aim to find the
sensitivities of the objective function \( f = \int_0^{T_m} i_{\text{out}}^2 dt \) with respect to the circuit parameters \( p = [R_o L_1 L_2 C_1 C_2 C_3]^T \) at the values \( p_o = [50 \ \Omega \ 1.0 \ \text{mH} \ 1.0 \ \text{mF} \ 1.0 \ \mu\text{F} \ 1.0 \ \mu\text{F}]^T \). The simulation time is \( T_m = 5.0 \ \text{ms} \). This circuit
is governed by the four independent equations:

\[
L_1 \frac{di_{\text{in}}}{dt} + i_{\text{in}}R_o + v_1 + v_2 = v_s(t),
\]

\[
C_3 \frac{d(v_1 + v_2)}{dt} + C_1 \frac{dv_1}{dt} - i_{\text{in}} = 0,
\]

\[
C_1 \frac{dv_1}{dt} - C_2 \frac{dv_2}{dt} - i_{\text{out}} = 0,
\]

\[
L_2 \frac{di_{\text{out}}}{dt} + i_{\text{out}}R_o - v_2 = 0.
\]
As evident from (1.31) to (1.34), this system has four state variables $E = [i_{in}(t) \ v_1(t) \ v_2(t) \ i_{out}(t)]^T$. Reorganizing (1.31)–(1.34) in a matrix form, we have

$$
\begin{bmatrix}
L_1 & 0 & 0 & 0 \\
0 & (C_1 + C_3) & C_3 & 0 \\
0 & C_1 & -C_2 & 0 \\
0 & 0 & 0 & L_2 \\
\end{bmatrix}
\begin{bmatrix}
i_{in} \\
\dot{v}_1 \\
\dot{v}_2 \\
i_{out} \\
\end{bmatrix}
+
\begin{bmatrix}
R_0 & 1 & 1 & 0 \\
-1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 \\
0 & 0 & -1 & R_o \\
\end{bmatrix}
\begin{bmatrix}
i_{in} \\
v_1 \\
v_2 \\
i_{out} \\
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}
$$

(1.35)

Comparing the system (1.35) to the system (1.8), we have the matrices

$$
T = M = 0, \quad N = 
\begin{bmatrix}
L_1 & 0 & 0 & 0 \\
0 & (C_1 + C_3) & C_3 & 0 \\
0 & C_1 & -C_2 & 0 \\
0 & 0 & 0 & L_2 \\
\end{bmatrix},
$$

$$
K = 
\begin{bmatrix}
R_0 & 1 & 1 & 0 \\
-1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 \\
0 & 0 & -1 & R_o \\
\end{bmatrix}, \quad \text{and} \quad G = 
\begin{bmatrix}
v_1(t) \\
0 \\
0 \\
0 \\
\end{bmatrix}.
$$

(1.36)

According to the adjoint variable theorem presented earlier, the corresponding adjoint system is given by

$$
-\begin{bmatrix}
L_1 & 0 & 0 & 0 \\
0 & (C_1 + C_3) & C_1 & 0 \\
0 & C_3 & -C_2 & 0 \\
0 & 0 & 0 & L_2 \\
\end{bmatrix}
\begin{bmatrix}
\dot{\lambda}_1 \\
\dot{\lambda}_2 \\
\dot{\lambda}_3 \\
\dot{\lambda}_4 \\
\end{bmatrix}
+
\begin{bmatrix}
R_0 & -1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & -1 \\
0 & 0 & -1 & R_o \\
\end{bmatrix}
\begin{bmatrix}
\lambda_1 \\
\lambda_2 \\
\lambda_3 \\
\lambda_4 \\
\end{bmatrix}
= 
\begin{bmatrix}
2i_{out}(t) \\
0 \\
0 \\
0 \\
\end{bmatrix}
$$

(1.37)
where \( i_{\text{out}}(t) \) is determined during the original simulation (1.35). By changing the time variable from \( t \) to \( \tau = T_m - t \), the adjoint system (1.37) can be rewritten as

\[
\begin{bmatrix}
L_1 & 0 & 0 & 0 \\
0 & (C_1 + C_3) & C_1 & 0 \\
0 & C_3 & -C_2 & 0 \\
0 & 0 & 0 & L_2 \\
\end{bmatrix}
\begin{bmatrix}
\frac{d\lambda_1}{dt} \\
\frac{d\lambda_2}{dt} \\
\frac{d\lambda_3}{dt} \\
\frac{d\lambda_4}{dt} \\
\end{bmatrix}
+ \begin{bmatrix}
R_0 & -1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & -1 \\
0 & 0 & -1 & R_0 \\
\end{bmatrix}
\begin{bmatrix}
\lambda_1 \\
\lambda_2 \\
\lambda_3 \\
\lambda_4 \\
\end{bmatrix}
= \begin{bmatrix}
2i_{\text{out}}(T_m - \tau) \\
0 \\
0 \\
0 \\
\end{bmatrix}. 
\]

(1.38)

This system has the same form as the original system (1.35). The same solver used for solving (1.35) can be used for solving (1.38) but with different matrices and excitation vector. The stored value of \( i_{\text{out}}(t) \) is applied backward in time in the adjoint simulation (1.38). The original simulation (1.35) and the adjoint simulation (1.38) are solved for the original response \( E(t) \) and the adjoint response \( \dot{E}(\tau) \). The residues in (1.12) are time-varying vectors that are determined during the original simulation and are given by

\[
R_1 = \frac{\partial K}{\partial R_o} E = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
i_{\text{in}} \\
v_1 \\
v_2 \\
i_{\text{out}} \\
\end{bmatrix}
= \begin{bmatrix}
i_{\text{in}} \\
0 \\
0 \\
i_{\text{out}} \\
\end{bmatrix}, 
\]

(1.39)

\[
R_2 = \frac{\partial N}{\partial L_1} \dot{E} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
i_{\text{in}} \\
v_1 \\
v_2 \\
i_{\text{out}} \\
\end{bmatrix}
= \begin{bmatrix}
i_{\text{in}} \\
0 \\
0 \\
i_{\text{out}} \\
\end{bmatrix}, 
\]

(1.40)

\[
R_3 = \frac{\partial N}{\partial L_2} \dot{E} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
i_{\text{in}} \\
v_1 \\
v_2 \\
i_{\text{out}} \\
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}, 
\]

(1.41)

\[
R_4 = \frac{\partial N}{\partial C_1} \dot{E} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
i_{\text{in}} \\
v_1 \\
v_2 \\
i_{\text{out}} \\
\end{bmatrix}
= \begin{bmatrix}
0 \\
v_1 \\
v_2 \\
v_1 \\
\end{bmatrix}, 
\]

(1.42)
\[ R_5 = \frac{\partial N}{\partial C_2} \dot{E} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix} \begin{bmatrix}
i_{\text{in}} \\
\dot{v}_1 \\
\dot{v}_2 \\
i_{\text{out}}
\end{bmatrix} = \begin{bmatrix}
0 \\
0 \\
-\dot{v}_1 \\
0
\end{bmatrix}, \quad (1.43) \]

\[ R_6 = \frac{\partial N}{\partial C_3} \dot{E} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix} \begin{bmatrix}
i_{\text{in}} \\
\dot{v}_1 \\
\dot{v}_2 \\
i_{\text{out}}
\end{bmatrix} = \begin{bmatrix}
0 \\
\dot{v}_1 + \dot{v}_2 \\
0 \\
0
\end{bmatrix}, \quad (1.44) \]

where \( R_i \) is the residue vector related to the \( i \)th parameter \( p_i \) where \( p = [R_0 \ L_1 \ L_2 \ C_1 \ C_2 \ C_3]^T \). These residues are determined through the original responses obtained by solving the system (1.35). They are used to evaluate all the sensitivities through inner product with the adjoint variable as in (1.19). We use in this example a time marching scheme identical to the one given by (1.30). MATLAB Listing M1.5 implements the adjoint sensitivity approach for this example. The output from this code is

\[
\begin{align*}
\text{derivative}_R_{o} &= -3.563629900427726e-12 \\
\text{derivative}_L_{1} &= 2.095857091808219e-08 \\
\text{derivative}_L_{2} &= 2.095857091808009e-08 \\
\text{derivative}_C_{1} &= 0.002336530747689 \\
\text{derivative}_C_{2} &= -4.000022427415373e-05 \\
\text{derivative}_C_{3} &= -4.000022427413188e-05
\end{align*}
\]

Figures 1.5 and 1.6 show the original and adjoint responses obtained using the MATLAB Listing M1.5, respectively. The results obtained using the MATLAB Listing M1.5 are verified using CFD. The MATLAB Listing M1.6 implements a central difference approach where each parameter is perturbed in the forward and backward directions.

Notice that the MATLAB Listing M1.6 implements perturbations in the \( i \)th parameter only by using the command \( p=p_0+\text{identity_matrix}(;,:i) \cdot \text{perturbations} \) which multiplies the vector of perturbations with the \( i \)th row.
MATLAB Listing M1.5

% MATLAB Listing M1.5
% Adjoint sensitivity analysis of example 1.4

clear all;

number_of_states=4; % 4 state variables in this example
Ro=0.05e3; C1=0.5e-6; C2=1.0e-6; C3=1.0e-6; L1=1.0e-3; L2=1.0e-3 % These are the simulation parameters

start_time=0; % start of simulation
end_time=5.0e-3; % end of simulation

number_of_time_steps=50000; % number of simulation time steps

simulation_time=linspace(start_time,end_time,number_of_time_steps); % total simulation time
dt=(end_time-start_time)/(number_of_time_steps-1); % the time step
excitation_centre=(end_time+start_time)/2; % centre of excitation
excitation_spread=(end_time+start_time)/10; % spread of excitation

source_waveform=1.0*exp(-((simulation_time-

% determine the original excitation vector

original_excitation=[1,1];

% build the N and K matrices

N=[L1 0 0 0;
   0 (C1+C3) C3 0;
   0 C1 -C2 0;
   0 0 0 L2];

K=[Ro 1 1 0;
   -1 0 0 0;
   0 0 0 -1;
   0 0 0 Ro];

% call the solver that returns the four state variables

original_response=[zeros(4,number_of_time_steps);

adjoint_excitation=zeros(4,number_of_time_steps);

flipped_adjoint_excitation=flipdim(adjoint_excitation,2);

% adjoint excitation is applied in a backward way

% N matrix of adjoint system

N_adjoint=N';

% K matrix of adjoint system

K_adjoint=K';

% Call adjoint simulator.

adjoint_response=adjoint_simulator(N_adjoint, K_adjoint, flipped_adjoint_excitation);

% now we evaluate the sensitivities by integrating the product of the adjoint response and the residues.

derivative_L1=dt*(-1.0*diff(original_response(1,:))/dt)*

(flipdim(adjoint_response(1,:),2))'

* (original_response(4,:)*

flipdim(adjoint_response(4,:),2))'

derivative_L2=dt*(-1.0*diff(original_response(1,:))/dt)*

(flipdim(adjoint_response(1,:),2))'

* (original_response(4,:)*

flipdim(adjoint_response(4,:),2))'

derivative_C1= dt*(-1.0*diff(original_response(2,:))/dt)*

(flipdim(adjoint_response(2,:),2))'

* (original_response(3,:)*

flipdim(adjoint_response(3,:),2))'

derivative_C2= dt*(-1.0*diff(original_response(3,:))/dt)*

(flipdim(adjoint_response(3,:),2))'

* (original_response(3,:)*

flipdim(adjoint_response(3,:),2))'

derivative_C3= dt*(-1.0*diff(original_response(3,:))/dt)*

(flipdim(adjoint_response(3,:),2))'

* (original_response(3,:)*

flipdim(adjoint_response(3,:),2))'
of the identity matrix. The same procedure is used in the backward perturbation. The output of the MATLAB Listing M1.6 is

\[
\text{gradient} = \\
-0.0000000000003562 \\
0.000000020960478 \\
0.000000020960478 \\
0.002336240830582 \\
-0.000039955340266 \\
-0.000039955340265
\]

Figure 1.5 The original responses of the circuit of Example 1.4: (a) the current \(i_{\text{in}}\), (b) the voltage \(v_1\), (c) the voltage \(v_2\), and (d) the current \(i_{\text{out}}\) as functions of the simulation time step.
A comparison between this vector and the sensitivities estimated using adjoint SA reveals good match. The CFD approach requires 12 extra simulations, while the AVM approach requires only one extra simulation.

To further demonstrate the strength of adjoint SA, we compare the adjoint sensitivities with those obtained using CFD for a sweep of the parameter $C_1$. The results are shown in Figure 1.7. Very good match is achieved for the considered sweep of values of the parameter $C_1$.

**Figure 1.6** The adjoint responses of the circuit of Example 1.4: (a) the adjoint variable $\lambda_1$, (b) the adjoint variable $\lambda_2$, (c) the adjoint variable $\lambda_3$, and (d) the adjoint variable $\lambda_4$ as functions of the adjoint simulation time variable $\tau$.
MATLAB Listing M1.6

```matlab
%MATLAB Listing M1.6.
%CFD approximations for example 1.4
clear all;
number_of_states=4; %4 state variables in this example
number_of_parameters=6; %this is the number of p
Ro=0.05; L1=1.0e-3; L2=1.0e-3; C1=0.5e-6; C2=1.0e-6; C3=1.0e-6; %parameters
start_time=0; %start of simulation
dt=(end_time-start_time)/(number_of_time_steps); %time vector
end_time=5.0e-3; %end of simulation
number_of_time_steps=50000; %number of simulation time steps
simulation_time= linspace(start_time,end_time,number_of_time_steps); %time vector
dt=(end_time-start_time)/(number_of_time_steps-1); %this is the time step
excitation_centre=(end_time-start_time)/2; %parameter for excitation
excitation_spread=(end_time-start_time)/15; %spread of excitation waveform
%we use Gaussian waveform with amplitude of 1 volts
source_waveform=1.0*exp(-((simulation_time-
excitation_centre)^2)/(2.0*excitation_spread^2)); %determine the original excitation vector
original_excitation=(1:size(source_waveform,1)); %vector of nominal parameter values
perturbations=0.05*p0; %this is the perturbation vector
gradient=zeros(number_of_parameters,1); %initialize the gradient vector
identity_matrix=eye(number_of_parameters); %an identity matrix
for l=1:number_of_parameters
    p=p0*identity_matrix(:,1); %perturbations; %forward perturb only the lth parameter
    N=[p(2) 0 0 0]; %build the perturbed N and K matrices
    p=(p(1)+p(6)) p(6) 0;
    p(4) -p(5) 0;
    p(4) 0 0 0;
    K=[-1 1 0; 0 0 -1; 0 0 -1];
    forward_response=Integrate(N,original_excitation,dt);
    %get original responses
    forward_objective=dt*forward_response(4,:)*forward_response(4,:);
    p=p0*identity_matrix(:,1); %perturbations; %backward perturb the lth parameter
    N=[p(2) 0 0 0]; %build the perturbed N and K matrices
    p=(p(1)+p(6)) p(6) 0;
    p(4) -p(5) 0;
    p(4) 0 0 0;
    K=[-1 1 0; 0 0 -1; 0 0 -1];
    %call the solver that returns the four state variables
    backward_response=Integrate(N,original_excitation,dt);
    %get original responses
    backward_objective=dt*backward_response(4,:)*backward_response(4,:);
    gradient(l,1)=(forward_objective-backward_objective)/(2*perturbations(l));
end
```
Figure 1.7 The adjoint variable sensitivities (○) and the CFD sensitivities (−) of Example 1.4 for a sweep of the parameter $C_1$. 

---

Adjoint sensitivity analysis of high frequency structures with MATLAB®
References


Chapter 2
Introduction to FDTD

Computational electromagnetics (CEM) has evolved rapidly during the past decade to a point where now extremely accurate predictions can be given for a variety of electromagnetic problems, including the scattering cross-section of radar targets and the precise design of antennas and microwave devices. In general, commonly used CEM methods are based on the applications of Maxwell’s equations and the appropriate boundary conditions associated with the problem to be solved.

The faster and more powerful computational resources are allowed for more advanced development of time-domain CEM models. More focus is directed toward differential equation time-domain approaches as they are easier to formulate and to adapt in computer simulation models without complex mathematics. They also provide more physical insight to the characteristics of the problems.

The basic formulation of the commonly used time-domain differential equation approach, namely the finite-difference time-domain (FDTD) method for CEM applications, is covered in this chapter. This formulation sets the stage for the adjoint variable methods covered in the subsequent chapters. Most of the materials in this chapter is extracted from the authors’ recent book [1].

The FDTD method is based on simple formulations that do not require complex asymptotic or Green’s functions. Although it solves the problem in time, it can provide frequency-domain responses over a wide band using the Fourier transform. It can easily handle composite geometries consisting of different types of materials including dielectric, magnetic, frequency-dependent, nonlinear, and anisotropic materials and can easily be adapted for parallel computations. These features of the FDTD method have made it the most attractive technique of CEM for many microwave devices and antenna applications.

2.1 Basic equations

The starting point for the construction of an FDTD algorithm is Maxwell’s equations in time domain. The differential time-domain Maxwell’s equations needed to specify the field behavior over time are

\[ \nabla \times H = \frac{\partial D}{\partial t} + J, \quad (2.1) \]
where \( \mathbf{E} \) is the electric field strength vector in volts per meter, \( \mathbf{D} \) is the electric displacement vector in coulombs per square meter, \( \mathbf{H} \) is the magnetic field strength vector in amperes per meter, \( \mathbf{B} \) is the magnetic flux density vector in webers per square meter, \( \mathbf{J} \) is the electric current density vector in amperes per square meter, and \( \rho_e \) is the electric charge density in coulombs per cubic meter. In the above equations, both the magnetic current and charge densities are considered zero. The constitutive relations are necessary to supplement Maxwell’s equations and characterize the material media. Constitutive relations for linear, isotropic, and non-dispersive materials can be written as

\[ \mathbf{D} = \varepsilon \mathbf{E}, \]
\[ \mathbf{B} = \mu \mathbf{H}, \]

where \( \varepsilon \) is the permittivity, and \( \mu \) is the permeability of the material. In free space, \( \varepsilon = \varepsilon_0 \approx 8.854 \times 10^{-12} \text{ F/m} \), \( \mu = \mu_0 = 4\pi \times 10^{-7} \text{ H/m} \). The electric current density \( \mathbf{J} = \mathbf{J}_c + \mathbf{J}_i \) is the sum of the conduction current density \( \mathbf{J}_c = \sigma_e \mathbf{E} \) and the impressed current density \( \mathbf{J}_i \). Here, \( \sigma_e \) is the electric conductivity in Siemens per meter. Decomposing the current densities in (2.1) to conduction and impressed components and using the constitutive relations, we can rewrite Maxwell’s curl equations as

\[ \nabla \times \mathbf{H} = \varepsilon \frac{\partial \mathbf{E}}{\partial t} + \sigma_e \mathbf{E} + \mathbf{J}_i, \]
\[ \nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t}. \]

This formulation treats only the electromagnetic fields \( \mathbf{E} \) and \( \mathbf{H} \) and not the fluxes \( \mathbf{D} \) and \( \mathbf{B} \). Although only the curl equations are used and the divergence equations are not part of the FDTD formalism, the divergence equations can be used as a test on the predicted field response. After forming \( \mathbf{D} = \varepsilon \mathbf{E} \) and \( \mathbf{B} = \mu \mathbf{H} \) from the predicted \( \mathbf{E} \) and \( \mathbf{H} \) fields, the resulting \( \mathbf{D} \) and \( \mathbf{B} \) must satisfy the divergence equations.

Equations (2.7) and (2.8) are two vector equations where each vector equation can be decomposed into three scalar equations for three-dimensional space as represented with the following six scalar equations for the Cartesian coordinate system \((x, y, z)\):

\[ \frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon_x} \left( \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} - \sigma_e E_x - J_{ix} \right), \]
The material parameters \( \varepsilon_x, \varepsilon_y, \) and \( \varepsilon_z \) are associated with electric field components \( E_x, E_y, \) and \( E_z \) through constitutive relations \( D_x = \varepsilon_x E_x, D_y = \varepsilon_y E_y, \) and \( D_z = \varepsilon_z E_z, \) respectively. Similarly, the material parameters \( \mu_x, \mu_y, \) and \( \mu_z \) are associated with magnetic field components \( H_x, H_y, \) and \( H_z \) through constitutive relations \( B_x = \mu_x H_x, B_y = \mu_y H_y, \) and \( B_z = \mu_z H_z, \) respectively. In (2.9)–(2.14), we use the constitutive relationships \( D = \varepsilon E \) and \( B = \mu H \) where \( \varepsilon \) and \( \mu \) are diagonal tensors.

The FDTD algorithm divides the problem geometry into a spatial grid where electric and magnetic field components are placed at certain discrete positions in space, and it solves Maxwell’s equations in time at discrete time instances. This can be implemented by first approximating the time and space derivatives appearing in Maxwell’s equations by finite differences. A set of equations that calculate the values of fields at a future time instant from the values of fields at past time instants are thus constructed. These steps create a time marching algorithm that simulates the progression of the fields in time.

### 2.2 FDTD updating equations for three-dimensional space

In 1966, Yee originated a set of finite-difference equations for the time-dependent Maxwell’s curl equations system [2]. These equations can be represented in discrete form, both in space and time, employing the second-order accurate central difference formulas. For a function \( f(x) \), its derivative, with respect to the independent variable \( x \), can be approximated by

\[
f'(x) \approx \frac{f(x + \Delta x/2) - f(x - \Delta x/2)}{\Delta x}. \tag{2.15}
\]

The electric and magnetic field components are sampled at discrete positions both in time and space. The FDTD technique divides the three-dimensional problem geometry into cells to form a grid. Figure 2.1 illustrates an FDTD grid composed of
(\(N_x \times N_y \times N_z\)) cells. A unit cell of this grid is called a Yee cell. Using Yee cells, a stepped or “staircase” approximation of the surface and internal geometry of the structure of interest is made with a space resolution set by the size of the unit cell in each direction.

The discrete spatial positions of the field components have a specific arrangement in the Yee cell, as demonstrated in Figure 2.2. The electric field vector components are placed at the centers of the edges of the Yee cells and oriented parallel to the respective edges. The magnetic field vector components are placed at the centers of the faces of the Yee cells and are oriented normal to the respective faces. This provides a simple picture of three-dimensional space being filled by an interlinked array of Faraday’s law and Ampere’s law contours. It can be easily noticed in Figure 2.2 that each magnetic field vector is surrounded by four electric field vectors that are curling around the magnetic field vector, thus simulating Faraday’s law. Similarly, if the neighboring cells are also added to the picture, it would be apparent that each electric field vector is surrounded by four magnetic

\[\Delta x, \Delta y, \Delta z\]

\[E_x(i,j,k), H_x(i,j,k), H_y(i,j,k), H_z(i,j,k)\]

Figure 2.1 An FDTD computational space composed of \((N_x \times N_y \times N_z)\) Yee cells

Figure 2.2 Arrangement of field components on a Yee cell indexed as \((i,j,k)\)
field vectors that are curling around the electric field vector, thus simulating Ampere’s law.

Figure 2.2 shows the indices of the field components, which are indexed as $(i,j,k)$, associated with a cell indexed as $(i,j,k)$. For a computational domain composed of uniform Yee cells having dimension $\Delta x$ in the $x$ direction, $\Delta y$ in the $y$ direction, and $\Delta z$ in the $z$ direction, the actual positions of the field components with respect to an origin coinciding with the position of the node $(i,j,k)$ can easily be calculated as

\[ E_x(i,j,k) \Rightarrow ((i - 0.5)\Delta x, (j - 1)\Delta y, (k - 1)\Delta z), \]
\[ E_y(i,j,k) \Rightarrow ((i - 1)\Delta x, (j - 0.5)\Delta y, (k - 1)\Delta z), \]
\[ E_z(i,j,k) \Rightarrow ((i - 1)\Delta x, (j - 1)\Delta y, (k - 0.5)\Delta z), \]
\[ H_x(i,j,k) \Rightarrow ((i - 1)\Delta x, (j - 0.5)\Delta y, (k - 0.5)\Delta z), \]
\[ H_y(i,j,k) \Rightarrow ((i - 0.5)\Delta x, (j - 1)\Delta y, (k - 0.5)\Delta z), \]
\[ H_z(i,j,k) \Rightarrow ((i - 0.5)\Delta x, (j - 0.5)\Delta y, (k - 1)\Delta z). \]

The FDTD algorithm samples and calculates the fields at discrete time instants; however, the electric and magnetic field components are not sampled at the same time instants. For a time-sampling period $\Delta t$, the electric field components are sampled at time instants $0, \Delta t, 2\Delta t, \ldots, n\Delta t, \ldots$; however, the magnetic field components are sampled at time instants $(1/2)\Delta t, (1 + (1/2))\Delta t, \ldots, (n + (1/2))\Delta t, \ldots$.

Therefore, the electric field components are calculated at integer time steps, and magnetic field components are calculated at half-integer time steps, and they are offset from each other by $\Delta t/2$. The field components need to be referred by not only their spatial indices, which indicate their positions in space, but also by their temporal indices, which indicate their time instants. Therefore, a superscript notation is adopted to indicate the time instant. For instance, the $z$ component of an electric field vector positioned at $((i - 1)\Delta x, (j - 1)\Delta y, (k - 0.5)\Delta z)$ and sampled at time instant $n\Delta t$ is referred to as $E^z_n(i,j,k)$. Similarly, the $y$ component of a magnetic field vector positioned at $((i - 0.5)\Delta x, (j - 1)\Delta y, (k - 0.5)\Delta z)$ and sampled at time instant $(n + 1/2)\Delta t$ is referred to as $H^y_{n+1/2}(i,j,k)$.

The material parameters (permittivity, permeability, and electric conductivity) are distributed over the FDTD grid and are associated with field components. Therefore, they are indexed the same way as their respective field components. For instance, Figure 2.3 illustrates the indices for the permittivity and permeability parameters. The electric conductivity is distributed and indexed the same way as the permittivity.

Having adopted an indexing scheme for the discrete samples of field components in both time and space, Maxwell’s curl equations (2.9)–(2.14) that are given in scalar form can be expressed in terms of finite differences. For instance, consider (2.9) where the derivatives can be approximated by using the central difference formula as given by (2.15). In this approximation, the position of $E_x(i,j,k)$ is the center point for the central difference formula in space, and time instant $(n + (1/2))\Delta t$ is the center point in time. Considering the field component positions
Figure 2.3 Material parameters indexed on a Yee cell

Figure 2.4 Field components around $E_x(i,j,k)$

given in Figure 2.4, we can write

$$
\frac{E_x^{n+1}(i,j,k) - E_x^n(i,j,k)}{\Delta t} = \frac{1}{\varepsilon_x(i,j,k)} \left( \frac{H_z^{n+1/2}(i,j,k) - H_z^{n+1/2}(i,j-1,k)}{\Delta y} \right.
- \frac{1}{\varepsilon_x(i,j,k)} \left( \frac{H_y^{n+1/2}(i,j,k) - H_y^{n+1/2}(i,j,k-1)}{\Delta z} \right)
- \left. \frac{\sigma_x(i,j,k)}{\varepsilon_x(i,j,k)} \right) E_x^{n+1/2}(i,j,k) - \frac{1}{\varepsilon_x(i,j,k)} J_x^{n+1/2}(i,j,k). \quad (2.16)
$$
The electric field components are to be defined and calculated at integer time steps; however, the right-hand side of (2.16) includes an electric field term at time instant \((n + (1/2))\Delta t\), that is, \(E_{x}^{n+(1/2)}(i,j,k)\). This term can be written as the average of the terms at time instants \((n + 1)\Delta t\) and \(n\Delta t\), such that

\[
E_{x}^{n+(1/2)}(i,j,k) = \frac{E_{x}^{n+1}(i,j,k) + E_{x}^{n}(i,j,k)}{2}.
\]  

(2.17)

Using (2.17) in (2.16) and arranging the terms such that the future term \(E_{x}^{n+1}(i,j,k)\) is kept on the left-hand side of the equation and the rest of the terms are moved to the right-hand side of the equation, we can write after some manipulations [1]:

\[
E_{x}^{n+1}(i,j,k) = \frac{2\varepsilon_{x}(i,j,k) - \Delta\sigma_{x}(i,j,k)}{2\varepsilon_{x}(i,j,k) + \Delta\sigma_{x}(i,j,k)} E_{x}^{n}(i,j,k)
\]

+ \(\frac{2\Delta t}{(2\varepsilon_{x}(i,j,k) + \Delta\sigma_{x}(i,j,k))\Delta y} \left( H_{z}^{n+(1/2)}(i,j,k) - H_{z}^{n+(1/2)}(i,j - 1,k) \right)\)

- \(\frac{2\Delta t}{(2\varepsilon_{x}(i,j,k) + \Delta\sigma_{x}(i,j,k))\Delta z} \left( H_{y}^{n+(1/2)}(i,j,k) - H_{y}^{n+(1/2)}(i,j,k - 1) \right)\)

- \(\frac{2\Delta t}{2\varepsilon_{x}(i,j,k) + \Delta\sigma_{x}(i,j,k)} J_{ix}^{n+(1/2)}(i,j,k)\).

(2.18)

The form of (2.18) demonstrates how the future value of an electric field component can be calculated by using the past values of the electric field component, the magnetic field components, and the source components. This form of an equation is called an FDTD updating equation. Updating equations can easily be obtained for calculating \(E_{x}^{n+1}(i,j,k)\) starting from (2.10) and \(E_{y}^{n+1}(i,j,k)\) starting from (2.11) following the same methodology that has been used to obtain (2.18).

Similarly, updating equations can be obtained for magnetic field components following the same methodology. However, while applying the central difference formula to the time derivative of the magnetic field components, the central point in time shall be taken as \(n\Delta t\). For instance, (2.12) can be approximated using finite differences based on the field positions (as shown in Figure 2.5) as

\[
\frac{H_{x}^{n+(1/2)}(i,j,k) - H_{x}^{n-(1/2)}(i,j,k)}{\Delta t} = \frac{1}{\mu_{x}(i,j,k)}
\]

\(\times \left( \frac{E_{y}^{n}(i,j,k + 1) - E_{y}^{n}(i,j,k)}{\Delta z} - \frac{E_{y}^{n}(i,j,k + 1) - E_{y}^{n}(i,j,k)}{\Delta y} \right) \)

(2.19)

After some manipulations, the future term of \(H_{x}^{n+(1/2)}(i,j,k)\) in (2.19) can be moved to the left-hand side and the other terms can be moved to the right-hand
side, such that

\[
H_n^{n+1/2}(i,j,k) = H_n^{n-1/2}(i,j,k) + \frac{\Delta t}{\mu_z(i,j,k)} \left( \frac{E^n_x(i,j,k+1) - E^n_x(i,j,k)}{\Delta z} - \frac{E^n_y(i,j+1,k) - E^n_y(i,j,k)}{\Delta y} \right).
\]

(2.20)

This equation is the updating equation for \(H_n^{n+1/2}(i,j,k)\). Similarly, updating equations can be derived for \(H_n^{n+1/2}(i,j,k)\) starting from (2.13), and for \(H_n^{n+1/2}(i,j,k)\), starting from (2.14) following the same methodology used to obtain (2.20).

Thus, (2.9)–(2.14) can be expressed using finite differences and can be arranged to construct the following six FDTD updating equations for the six components of electromagnetic fields by introduction of respective coefficient terms

\[
E_n^{n+1}(i,j,k) = C_{exe}(i,j,k) \times E_n^n(i,j,k) + C_{exh}(i,j,k)
\]

\[
\times \begin{pmatrix}
\frac{H_{n+1/2}^z(i,j,k) - H_{n+1/2}^z(i,j-1,k)}{\Delta y} + \frac{H_{n+1/2}^z(i,j,k) - H_{n+1/2}^z(i,j,k-1)}{\Delta z} \\
-J_{ix}^{n+1/2}(i,j,k)
\end{pmatrix},
\]

(2.21)
Having derived the FDTD updating equations, a time-marching algorithm can be constructed as illustrated in Figure 2.6. The first step in this algorithm is setting up
the problem space including the objects, material types, and sources then defining any other parameters that will be used during the FDTD computation. Next, the coefficient terms appearing in (2.21)–(2.26) are calculated and stored as arrays before the iteration is started. The field components are defined as arrays as well and initialized with zeros since the initial values of the fields in the problem space in most cases are zeros. Fields are induced in the problem space due to sources as the iteration proceeds. At every step of the time-marching iteration, the magnetic field components are updated for time instant \((n+0.5)\Delta t\) using (2.24)–(2.26). The electric field components are then updated for time instant \((n+1)\Delta t\) using (2.21)–(2.23). The problem space has a finite size, and specific boundary conditions can be enforced on the boundaries of the problem space. Therefore, the field components on the boundaries of the problem space are treated according to the type of boundary conditions during the iteration. The types of boundary conditions and the techniques used to integrate them into the FDTD algorithm are discussed in detail in [1]. After the fields are updated and boundary conditions are enforced, the current values of any desired field components can be captured and stored as output data. These data can be used for real-time processing or postprocessing to calculate some other desired parameters. The FDTD iterations continue until some stopping criteria are achieved.
2.3 FDTD updating equations for two-dimensional space

The FDTD updating equations given in (2.21)–(2.26) are used to solve three-dimensional problems. In the two-dimensional case, where there is no variation in the problem geometry and field distributions in one of the dimensions, a simplified set of updating equations can be obtained. Since there is no variation in one of the dimensions, the derivative terms in (2.9)–(2.14) with respect to that dimension vanish. For instance, if the problem is independent of the \( z \) dimension, we get

\[
\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon_x} \left( \frac{\partial H_z}{\partial y} - \sigma_x^e E_x - J_{ix} \right),
\]

(2.27)

\[
\frac{\partial E_y}{\partial t} = -\frac{1}{\varepsilon_y} \left( \frac{\partial H_z}{\partial x} + \sigma_y^e E_y + J_{iy} \right),
\]

(2.28)

\[
\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon_z} \left( \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - \sigma_z^e E_z - J_{iz} \right),
\]

(2.29)

\[
\frac{\partial H_x}{\partial t} = -\frac{\partial E_z}{\mu_x \partial y},
\]

(2.30)

\[
\frac{\partial H_y}{\partial t} = \frac{\partial E_z}{\mu_y \partial x},
\]

(2.31)

\[
\frac{\partial H_z}{\partial t} = \frac{1}{\mu_z} \left( \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x} \right).
\]

(2.32)

One should notice that (2.27), (2.28), and (2.32) are dependent only on the terms \( E_x, E_y, \) and \( H_z \); whereas (2.29), (2.30), and (2.31) are dependent only on the terms \( E_z, H_x, \) and \( H_y \). Therefore, the above six equations can be treated as two separate sets of three equations each. In the first set (2.27), (2.28), and (2.32), all the electric field components are transverse to the reference dimension \( z \); therefore, this set of equations constitutes the transverse electric to \( z \) case—\( \text{TE}_z \). In the second set, (2.29), (2.30), and (2.31), all the magnetic field components are transverse to the reference dimension \( z \); therefore, this set of equations constitutes the transverse magnetic to \( z \) case—\( \text{TM}_z \). Most two-dimensional problems can be decomposed into two separate problems, each including separate field components that are \( \text{TE}_z \) and \( \text{TM}_z \). These two problems can be solved separately, and the solution of the main problem is the sum of the two solutions.

The FDTD updating equations for the \( \text{TE}_z \) case can be obtained by applying the central difference formula to the equations constituting the \( \text{TE}_z \) case based on the field positions shown in Figure 2.7. This figure is obtained by projection of the
Yee cells in Figure 2.2 on the \(xy\) plane in the \(z\) direction. The FDTD updating equations for the TE\(_z\) case are therefore obtained as

\[
E_x^{n+1}(i,j) = C_{ex}(i,j) \times E_x^n(i,j) + C_{exh}(i,j) \\
\times \left( \frac{H_z^{n+(1/2)}(i,j) - H_z^{n+(1/2)}(i,j-1)}{\Delta y} - j_l^{n+(1/2)}(i,j) \right),
\]

(2.33)

\[
E_y^{n+1}(i,j) = C_{eye}(i,j) \times E_y^n(i,j) + C_{eyh}(i,j) \\
\times \left( -\frac{H_z^{n+(1/2)}(i,j) - H_z^{n+(1/2)}(i-1,j)}{\Delta x} - j_l^{n+(1/2)}(i,j) \right),
\]

(2.34)

\[
H_z^{n+(1/2)}(i,j) = H_z^{n-(1/2)}(i,j) + C_{hze}(i,j) \\
\times \left( \frac{E_x^n(i,j+1) - E_x^n(i,j)}{\Delta y} - \frac{E_x^n(i+1,j) - E_x^n(i,j)}{\Delta x} \right).
\]

(2.35)

\(E_x^n\) and \(E_y^n\) denote the electric field components in the \(x\) and \(y\) directions respectively. \(H_z^n\) represents the magnetic field component in the \(z\) direction. \(C_{ex}, C_{exh}, C_{ey}, C_{eyh}, C_{hze}\) are coefficients that depend on the material properties. \(\Delta x, \Delta y\) are the spatial discretization steps in the \(x\) and \(y\) directions respectively.

Figure 2.7 Two-dimensional TE\(_z\) FDTD field components
Similar procedure can be followed to obtain the updating equations for the TM$_z$ case while considering the field component positions as shown in Figure 2.8. The resulting updating equations are

\[
E_{z}^{n+1}(i,j) = C_{ee}(i,j) 	imes E_{z}^{n}(i,j) + C_{eh}(i,j) \times \left( \frac{ H_{x}^{n+1/2}(i,j) - H_{x}^{n+1/2}(i-1,j) - H_{x}^{n+1/2}(i,j-1) + H_{x}^{n+1/2}(i,j-1) }{\Delta y} - \frac{ E_{z}^{n}(i+1,j) + E_{z}^{n}(i,j) }{\Delta x} \right),
\]

(2.36)

\[
H_{x}^{n+1/2}(i,j) = H_{x}^{n-1/2}(i,j) + C_{hee}(i,j) \left( -\frac{ E_{z}^{n}(i,j) + E_{z}^{n}(i+1,j) }{\Delta y} \right),
\]

(2.37)

\[
H_{y}^{n+1/2}(i,j) = H_{y}^{n-1/2}(i,j) + C_{hey}(i,j) \times \left( \frac{ E_{z}^{n}(i+1,j) - E_{z}^{n}(i,j) }{\Delta x} \right).
\]

(2.38)
2.4 FDTD updating equations for one-dimensional space

In the one-dimensional case, there is no variation in the problem geometry and field distributions in two of the dimensions. For instance, if the $y$ and $z$ dimensions both have no variation, the derivatives with respect to the $y$ and $z$ dimensions vanish in Maxwell’s curl equations. Therefore, the two-dimensional curl (2.27)–(2.32) reduces to

\[
\frac{\partial E_x}{\partial t} = -\frac{1}{\varepsilon_x} \left( \sigma_x E_x + J_x \right), \quad (2.39)
\]

\[
\frac{\partial E_y}{\partial t} = -\frac{1}{\varepsilon_y} \left( \frac{\partial H_z}{\partial x} + \sigma_y E_y + J_y \right), \quad (2.40)
\]

\[
\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon_z} \left( \frac{\partial H_y}{\partial x} - \sigma_z E_z - J_z \right), \quad (2.41)
\]

\[
\frac{\partial H_x}{\partial t} = 0, \quad (2.42)
\]

\[
\frac{\partial H_y}{\partial t} = \frac{\partial E_z}{\mu_y \partial x}, \quad (2.43)
\]

\[
\frac{\partial H_z}{\partial t} = -\frac{\partial E_y}{\mu_z \partial x}. \quad (2.44)
\]

It should be noted that (2.39) and (2.42) include time derivatives but not space derivatives. Therefore, these equations do not represent propagating fields. The field components $E_x$ and $H_x$, which only exist in these two equations, do not propagate. The other four equations represent propagating fields, and both the electric and magnetic field components existing in these equations are transverse to the $x$ dimension. Therefore, transverse electric and magnetic to $x$ (TEM$_x$) fields exist and propagate as plane waves in the one-dimensional case under consideration.

Similar to the two-dimensional case, the one-dimensional case can be broken down into two separate cases. Equations (2.40) and (2.44), which include only the terms $E_y$ and $H_z$, are decoupled from (2.41) and (2.43), that include only the terms $E_z$ and $H_y$. FDTD updating equations can be obtained for (2.40) and (2.44) using the central difference formula based on the field positioning in one-dimensional space as illustrated in Figure 2.9, such that

\[
E_y^{n+1}(i) = C_{eye}(i) \times E_y^n(i) + C_{eyh}(i) \times \left( -\frac{H_z^{n+1/2}(i) - H_z^{n+1/2}(i-1)}{\Delta x} - J_y^{n+1/2}(i) \right), \quad (2.45)
\]

\[
H_z^{n+1/2}(i) = H_z^{n-1/2}(i) + C_{hze}(i) \left( -\frac{E_y^n(i+1) - E_y^n(i)}{\Delta x} \right). \quad (2.46)
\]
Similarly, FDTD updating equations can be obtained for (2.41) and (2.43) using the central difference formula based on the field positioning in one-dimensional space as illustrated in Figure 2.10, such that

\[
E_{z}^{n+1}(i) = C_{ecz}(i) \times E_{z}^{n}(i) + C_{ecz}(i) \times \left( \frac{H_{y}^{n+(1/2)}(i) - H_{y}^{n-(1/2)}(i-1)}{\Delta x} - J_{iz}^{n+(1/2)}(i) \right), \tag{2.47}
\]

\[
H_{y}^{n+(1/2)}(i) = H_{y}^{n-(1/2)}(i) + C_{hye}(i) \times \left( \frac{E_{z}^{n}(i+1) - E_{z}^{n}(i)}{\Delta x} \right). \tag{2.48}
\]

### 2.5 Dispersive material modeling

The electromagnetic material properties, permittivity, permeability, and electrical conductivity were treated as constants in the previous sections. In many applications, these parameters are relatively constant over the frequency band of interest and simulations based on constant parameter assumption yield sufficiently accurate results. On the other hand, in some applications, the values of material parameters vary significantly as functions of frequency over the frequency band of interest. One of the attractive features of the FDTD method is that it allows for modeling behavior of complex media such as dispersion, anisotropy, and nonlinearity. Algorithms can be developed to model the electromagnetic properties of these media within the discrete time domain solution. In this section, we will briefly introduce the appropriate FDTD formulation for dispersive materials using two different dispersion profiles, the Lorentz and Drude models.

Dispersion modeling has been one of the key topics in FDTD research over the years. Many algorithms have been introduced by researchers. Most of these
algorithms are classified mainly under the auxiliary differential equation (ADE) [3–7], recursive convolution [8–12], and Z-transform [13,14] techniques. An in-depth review of these techniques for modeling dispersive and other complex media in the FDTD method is presented in [15].

In general, frequency dependence of permittivity, permeability, or conductivity can be expressed as a sum of rational functions of angular frequency $\omega$. The most common types of these rational functions are referred to as Debye, Lorentz, or Drude model. Debye models are commonly used to approximate the frequency behavior of biological tissues and soil permittivities. Lorentz models describe the frequency behavior of some metamaterials such as double-negative media close to resonances. Drude models are useful in describing the behavior of metals at optical frequencies, and they can sometimes be augmented by Lorentz terms [15]. It should be noted that some applications employ some other dispersion models described in the literature besides these three models. For instance, Cole–Cole is another model, yet a more general one than Debye, for biological applications, while Condon is a model used to describe chiral media.

In the following sections, we will present the integration of Lorentz and Drude models into FDTD by using the ADE technique. The detailed implementation of the resulting algorithms in the associated MATLAB$^\text{®}$ FDTD codes can be found in [1].

### 2.5.1 Modeling Lorentz medium using ADE technique

In this section, we illustrate an algorithm to model Lorentz medium based on the ADE technique. It should be noted that the procedure to develop an ADE and find a solution to the developed ADE equation is not unique. Here, we discuss the procedure presented in [7].

Ampere’s law can be expressed for a dispersive medium in the frequency domain with $N_p$ poles, as

$$\nabla \times \tilde{H} = j\omega\varepsilon_0\varepsilon_\infty \tilde{E} + \sigma \tilde{E} + j\omega \sum_{m=1}^{N_p} \tilde{Q}_m, \quad (2.49)$$

while in time domain, it takes the form

$$\nabla \times \dot{H} = \varepsilon_0\varepsilon_\infty \frac{\partial}{\partial t} \tilde{E} + \sigma \tilde{E} + \sum_{m=1}^{N_p} \frac{\partial}{\partial t} \tilde{Q}_m. \quad (2.50)$$

In the frequency domain, the term $\tilde{Q}_m$ can be written for Lorentz model as

$$\tilde{Q}_m = \frac{A_m\varepsilon_0}{\omega_m^2 + 2j\omega\delta_m - \omega^2} \tilde{E} = \frac{\psi_m}{\omega_m^2 + 2j\omega\delta_m - \omega^2} \tilde{E}, \quad (2.51)$$

where $\omega_m$ is the pole location and $\delta_m$ is the damping factor of the $m$th term. Here, we define $\psi_m = A_m\varepsilon_0(\varepsilon_s - \varepsilon_\infty)\omega_m^3$ for brevity. The $Q_m$ term is then constructed.
using (2.51) such that

\[
\left( \frac{\partial^2}{\partial t^2} + 2\delta_m \frac{\partial}{\partial t} + \omega_m^2 \right) Q_m = \psi_mE,
\]

which can be expressed in discrete time using central difference approximation at time step \(n\) as

\[
\frac{Q_m^{n+1} - 2Q_m^n + Q_m^{n-1}}{\Delta t^2} + \delta_m \frac{Q_m^{n+1} - Q_m^{n-1}}{\Delta t} + \omega_m^2 Q_m^n = \psi_mE^n. \tag{2.52}
\]

Equation (2.52) can be arranged to calculate the future value of \(Q_m\), such that

\[
Q_m^{n+1} = \frac{2 - (\Delta t)^2 \omega_m^2}{(\delta_m \Delta t + 1)} Q_m^n + \frac{(\delta_m \Delta t - 1)}{(\delta_m \Delta t + 1)} Q_m^{n-1} + \frac{(\Delta t)^2 \psi_m}{(\delta_m \Delta t + 1)} E^n. \tag{2.53}
\]

Meanwhile, (2.50) can be represented in discrete time at time step \(n + 0.5\) as

\[
\nabla \times \mathbf{H}^{n+0.5} = \varepsilon_0\varepsilon_\infty \frac{E^{n+1} - E^n}{\Delta t} + \sigma \frac{E^{n+1} + E^n}{2} + \frac{1}{\Delta t} \sum_{m=1}^{N_p} \left( Q_m^{n+1} - Q_m^n \right), \tag{2.54}
\]

which can be arranged, by moving the \(E^{n+1}\) term to left-hand side, to obtain an updating equation for the electric field as

\[
E^{n+1} = \frac{2\Delta t}{2\varepsilon_0\varepsilon_\infty + \Delta t \sigma} \nabla \times \mathbf{H}^{n+0.5} - \frac{2\varepsilon_0\varepsilon_\infty - \Delta t \sigma}{2\varepsilon_0\varepsilon_\infty + \Delta t \sigma} E^n + \frac{1}{\Delta t} \sum_{m=1}^{N_p} \left( Q_m^{n+1} - Q_m^n \right). \tag{2.55}
\]

An algorithm can be constructed using (2.53) and (2.55) as illustrated in Figure 2.11. At every time step, magnetic field components are updated as usual. Next, the new values of \(Q_m\) are calculated using their past values and the past values of the electric field components following (2.53). Then, electric field

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{update_sequence_fields_lorentz_drude.png}
\caption{Update sequence of fields in the presented Lorentz and Drude modeling algorithms}
\end{figure}
components are updated using the past values of electric and magnetic field components as well as the new and past values of \( Q_m \) following (2.55). One should notice that this formulation requires additional arrays to store \( Q_m^n \) and \( Q_m^{n-1} \) in order to implement (2.53).

### 2.5.2 Modeling Drude medium using ADE technique

The ADE technique introduced in [7], discussed above, is employed to model a Drude medium in [16]. In this section, we illustrate the formulation presented in [16]. Here, we consider (2.50) as the starting point to develop an algorithm for Drude medium. In the frequency domain, the term \( \tilde{Q}_m \) can be written for Drude model as

\[
\tilde{Q}_m = \frac{\varepsilon_0 \omega_m^2}{\omega^2 - j\omega \gamma_m} \tilde{E},
\]

(2.56)

where \( \omega_m \) is the plasma frequency, and \( \gamma_m \) is the inverse of the pole relaxation time. An ADE can be constructed from (2.56) as

\[
\left( \frac{\partial^2}{\partial t^2} + \gamma_m \frac{\partial}{\partial t} \right) Q_m = \varepsilon_0 \omega_m^2 E.
\]

(2.57)

Then, (2.57) can be expressed in discrete time using central difference approximation at time step \( n \), and rearranged to calculate the future value of \( Q_m \) as

\[
Q_m^{n+1} = \frac{4}{\Delta t \gamma_m + 2} Q_m^n + \frac{\Delta t \gamma_m - 2}{\Delta t \gamma_m + 2} Q_m^{n-1} + \frac{2 \Delta t^2 \varepsilon_0 \omega_m^2}{\Delta t \gamma_m + 2} E^n.
\]

(2.58)

Equation (2.58) is in the same form as (2.53), while its coefficients are different. Therefore, we can use (2.58) and (2.55) to construct an ADE algorithm to model a Drude medium, which is essentially the same as the one constructed for Lorentz medium as shown in Figure 2.11. At every time step, magnetic field components are updated as usual. Next, the new values of \( Q_m \) are calculated following (2.58). Then, electric field components are updated following (2.55).

### 2.6 Anisotropic material modeling

We consider in this book adjoint sensitivity analysis for high-frequency structures with different types of media. One of the considered classes of materials is anisotropic material. For an anisotropic medium, the electric flux is related to the electric field \( E \) by a permittivity tensor \( \varepsilon \) and the magnetic flux \( B \) is related to the magnetic field \( H \) by a permeability tensor \( \mu \) as presented in [17,18]. Maxwell’s equations for anisotropic media can then be written as in (2.1)–(2.4), along with similar constitution relations such that

\[
B = \mu H, \quad D = \varepsilon E,
\]

(2.59)
where

\[
\varepsilon = \begin{bmatrix}
\varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\
\varepsilon_{yx} & \varepsilon_{yy} & \varepsilon_{yz} \\
\varepsilon_{zx} & \varepsilon_{zy} & \varepsilon_{zz}
\end{bmatrix}, \quad \mu = \begin{bmatrix}
\mu_{xx} & \mu_{xy} & \mu_{xz} \\
\mu_{yx} & \mu_{yy} & \mu_{yz} \\
\mu_{zx} & \mu_{zy} & \mu_{zz}
\end{bmatrix},
\]

(2.60)

and the inverse of the permittivity and permeability tensors are defined as

\[
[\varepsilon]^{-1} = \frac{1}{\Delta_\varepsilon} \begin{bmatrix}
P_{xx} & P_{xy} & P_{xz} \\
P_{yx} & P_{yy} & P_{yz} \\
P_{zx} & P_{zy} & P_{zz}
\end{bmatrix}, \quad [\mu]^{-1} = \frac{1}{\Delta_\mu} \begin{bmatrix}
R_{xx} & R_{xy} & R_{xz} \\
R_{yx} & R_{yy} & R_{yz} \\
R_{zx} & R_{zy} & R_{zz}
\end{bmatrix},
\]

(2.61)

with

\[
\Delta_\varepsilon = \varepsilon_0 \left( \varepsilon_{xx} \varepsilon_{yy} \varepsilon_{zz} + \varepsilon_{xy} \varepsilon_{yz} \varepsilon_{zx} + \varepsilon_{xz} \varepsilon_{yx} \varepsilon_{zy} - \varepsilon_{xx} \varepsilon_{yy} \varepsilon_{zz} - \varepsilon_{xx} \varepsilon_{yz} \varepsilon_{zy} - \varepsilon_{xx} \varepsilon_{zx} \varepsilon_{zy} \right),
\]

\[
\Delta_\mu = \mu_0 \left( \mu_{xx} \mu_{yy} \mu_{zz} + \mu_{xy} \mu_{yz} \mu_{zx} + \mu_{xz} \mu_{yx} \mu_{zy} - \mu_{xx} \mu_{yy} \mu_{zz} - \mu_{xx} \mu_{yz} \mu_{zy} - \mu_{xx} \mu_{zx} \mu_{zy} \right),
\]

(2.62)

The expressions in (2.61) are used for the evaluation of the \( E \) field components from the components of \( D \) such that

\[
E_{x}^{n+1}(i,j,k) = \frac{P_{xx}}{\Delta_\varepsilon} D_{x}^{n+1}(i,j,k) + \frac{P_{xy}}{\Delta_\varepsilon} D_{y, \text{avg},ex}^{n+1}(i,j,k) + \frac{P_{xz}}{\Delta_\varepsilon} D_{z, \text{avg},ex}^{n+1}(i,j,k),
\]

\[
E_{y}^{n+1}(i,j,k) = \frac{P_{yx}}{\Delta_\varepsilon} D_{x, \text{avg},ey}^{n+1}(i,j,k) + \frac{P_{yy}}{\Delta_\varepsilon} D_{y}^{n+1}(i,j,k) + \frac{P_{yz}}{\Delta_\varepsilon} D_{z, \text{avg},ey}^{n+1}(i,j,k),
\]

\[
E_{z}^{n+1}(i,j,k) = \frac{P_{zx}}{\Delta_\varepsilon} D_{x, \text{avg},ez}^{n+1}(i,j,k) + \frac{P_{zy}}{\Delta_\varepsilon} D_{y, \text{avg},ez}^{n+1}(i,j,k) + \frac{P_{zz}}{\Delta_\varepsilon} D_{z}^{n+1}(i,j,k).
\]

(2.63)

In the first line of (2.63), the \( D_{y} \) and \( D_{z} \) field components are needed to update \( E_{x} \). These field components, however, are not readily available at the location of \( E_{x} \). Therefore, average values for \( D_{y} \) and \( D_{z} \), denoted as \( D_{y, \text{avg},ex} \) and \( D_{z, \text{avg},ex} \), are used.
at the location of $E_x$. These average values are calculated over four points surrounding the desired location of $E_x$ in the FDTD grid such as

$$D_{y,\text{avg,} ex}^{n+1}(i, j, k) = \frac{[D_{y}^{n+1}(i + 1, j, k) + D_{y}^{n+1}(i - 1, j, k) + D_{y}^{n+1}(i, j, k) + D_{y}^{n+1}(i, j, k - 1)]}{4},$$

$$D_{z,\text{avg,} ex}^{n+1}(i, j, k) = \frac{[D_{z}^{n+1}(i + 1, j, k) + D_{z}^{n+1}(i - 1, j, k) + D_{z}^{n+1}(i, j, k) + D_{z}^{n+1}(i, j, k + 1)]}{4}. $$

The same averaging procedure is required for $D_x$ and $D_z$ in the second line of (2.63) and for $D_x$ and $D_y$ in the third line of (2.63). In a similar procedure, the $H$ field components from the components of $B$ can be obtained as follows:

$$H_{x}^{n+0.5}(i, j, k) = \frac{R_{x}}{\Delta t}B_{x,\text{avg,}hx}^{n+0.5}(i, j, k) + \frac{R_{xy}}{\Delta \mu}B_{y,\text{avg,}hx}^{n+0.5}(i, j, k) + \frac{R_{xz}}{\Delta \mu}B_{z,\text{avg,}hx}^{n+0.5}(i, j, k),$$

$$H_{y}^{n+0.5}(i, j, k) = \frac{R_{y}}{\Delta t}B_{x,\text{avg,}hy}^{n+0.5}(i, j, k) + \frac{R_{xy}}{\Delta \mu}B_{y,\text{avg,}hy}^{n+0.5}(i, j, k) + \frac{R_{yz}}{\Delta \mu}B_{z,\text{avg,}hy}^{n+0.5}(i, j, k),$$

$$H_{z}^{n+0.5}(i, j, k) = \frac{R_{z}}{\Delta t}B_{x,\text{avg,}hz}^{n+0.5}(i, j, k) + \frac{R_{xy}}{\Delta \mu}B_{y,\text{avg,}hz}^{n+0.5}(i, j, k) + \frac{R_{yz}}{\Delta \mu}B_{z,\text{avg,}hz}^{n+0.5}(i, j, k).$$

In the first line of (2.65), the $B_y$ and $B_z$ field components are not located at the position of $H_x$ in the FDTD grids. Hence, an average of $B_y$ and $B_z$ over four points surrounding the desired location of $H_x$ in the FDTD grid are needed such that

$$B_{y,\text{avg,}hx}^{n+0.5}(i, j, k) = \frac{[B_{y}^{n+0.5}(i - 1, j, k) + B_{y}^{n+0.5}(i + 1, j, k) + B_{y}^{n+0.5}(i, j, k) + B_{y}^{n+0.5}(i, j + 1, k)]}{4},$$

$$B_{z,\text{avg,}hx}^{n+0.5}(i, j, k) = \frac{[B_{z}^{n+0.5}(i - 1, j, k) + B_{z}^{n+0.5}(i + 1, j, k) + B_{z}^{n+0.5}(i, j, k) + B_{z}^{n+0.5}(i, j + 1, k)]}{4}. $$

The same averaging procedure is required for $B_x$ and $B_z$ in the second line of (2.65) and for $B_x$ and $B_y$ in the third line of (2.65).
In the FDTD time-marching loop, first step is to compute the magnetic field intensity $H$ from $E$ using regular updating equations everywhere except for the anisotropic regions and compute the magnetic flux $B$ from $E$ using an updating equation based on (2.2) in the anisotropic regions. Then, the $B$ fields are averaged such as in (2.66) and they are used to compute $H$ using (2.65) in the anisotropic regions. In the second half of an iteration of the FDTD time marching loop, $E$ field is computed using $H$ using conventional updating equations everywhere except for the anisotropic regions. Then, $D$ field is computed using $H$ using an updating equation based on (2.1) in the anisotropic regions. These $D$ fields are then averaged as in (2.64), and they are used to compute $E$ using (2.63) in the anisotropic regions.

References


Chapter 3

The adjoint variable method for frequency-independent constitutive parameters

3.1 Introduction

In Chapter 1, we reviewed wideband adjoint sensitivity analysis approach for circuits. We showed that for an arbitrary response of a linear electric circuit, the sensitivities of the response with respect to all parameters are estimated through one extra circuit simulation. We explained that we have to store a temporal residue vector for each parameter of interest. The products of temporal residue vectors and the adjoint variable vector are integrated over time to obtain the sensitivity estimates.

In this chapter, we show how adjoint sensitivity analysis techniques can be applied to computational electromagnetics problems. We focus in this chapter on linear, isotropic, and nondispersive electromagnetic structures. In this case, the properties of all materials in the computational domain do not depend on the field magnitude, the excitation polarization, or the excited frequency band. We adopt a step-by-step approach for introducing the basic concepts. First, we address the 1D finite difference time domain (FDTD) case and show how adjoint sensitivities are estimated. This case is illustrated through a numerical example. We then address the 2D TMz case. A full derivation of the adjoint variable method is given. We show that using a similar formulation to the 1D case, the sensitivities of a general objective function with respect to all parameters are estimated using one extra simulation. The 2D transverse electric to z (TEz) is similar to the 2D transverse magnetic to z (TMz) and will not be addressed here. Two numerical examples are presented to illustrate the 2D case. Finally, we address the full 3D case and show that the same concept applies to full 3D problems.

We utilize the time-domain Maxwell’s equations in our derivation. Previous derivations given for the adjoint variable method (AVM) utilizing the FDTD method exploit the wave equation formulation [1–4]. We opted though to develop a simpler derivation that the reader will find easier to understand. We assume here an objective function of the form

\[ F = \int_0^{T_m} \psi(V, p) dt, \]  

(3.1)

where \( V \) is the vector of state variables (electric and magnetic fields at different cells), and \( p \) is the vector of parameters. The objective function integral (3.1) is
usually evaluated over a small subset of the computational domain. This subset is labeled as the observation domain. The parameters in $p$ include dimensions of different discontinuities and material properties. Contrary to the case discussed in Chapter 1, some of the parameters in $p$ may not assume continuous values. For example, on-grid length of a discontinuity in the $x$-direction changes by only multiples of the cell size $\Delta x$. In several addressed cases, we will thus assume that the perturbation made in some of the parameters cannot be made arbitrarily small.

The analytical derivative of the objective function (3.1) with respect to the $i$th parameter $p_i$ is given by

$$\frac{\partial F}{\partial p_i} = \int_0^{T_n} \frac{\partial^2 \psi}{\partial p_i} dt + \int_0^{T_n} \left( \frac{\partial \psi}{\partial V} \right)^T \frac{\partial V}{\partial p_i} dt, \quad i = 1, 2, \ldots, n,$$

(3.2)

where the first integral expresses the explicit dependence on the parameter. This explicit dependence is nonzero only if the expression of the kernel function $\psi$ contains $p_i$. The value of the first integral in (3.2) is readily available using $\psi$. We will thus focus in this chapter on how to evaluate the second (implicit) integral in (3.2).

### 3.2 The 1D case

We consider first the 1D case. As explained in Chapter 2, a plane wave propagating in the positive $z$-direction with $E_x$ and $H_y$ components in a region with electric and magnetic current densities is governed by the following two equations:

$$- \frac{\partial H_y}{\partial z} = \varepsilon \frac{\partial E_x}{\partial t} + J_x + J_{ix},$$

(3.3)

$$\frac{\partial E_x}{\partial z} = -\mu \frac{\partial H_y}{\partial t} - M_y - M_{iy},$$

(3.4)

where $M_y = \sigma^m H_y$ is the magnetic conduction current density in the $y$-direction, with $\sigma^m$ being the magnetic conductivity. $M_{iy}$ is the impressed magnetic current density. Similarly, $J_x = \sigma^e E_x$ is the electric conduction current density in the $x$-direction, with $\sigma^e$ being the electric conductivity. $J_{ix}$ is the impressed electric current density. Reorganizing (3.3) and (3.4) into a matrix form, we have

$$\begin{bmatrix} \varepsilon & 0 \\ 0 & -\mu \end{bmatrix} \begin{bmatrix} \dot{E}_x \\ \dot{H}_y \end{bmatrix} + \begin{bmatrix} \sigma^e & 0 \\ 0 & -\sigma^m \end{bmatrix} \begin{bmatrix} E_x \\ H_y \end{bmatrix} + \begin{bmatrix} 0 & \partial / \partial z \\ -\partial / \partial z & 0 \end{bmatrix} \begin{bmatrix} E_x \\ H_y \end{bmatrix} = \begin{bmatrix} -J_{ix} \\ M_{iy} \end{bmatrix}$$

(3.5)

Equation (3.5) is written at a single point in space with constitutive parameters ($\varepsilon$, $\mu$, $\sigma^e$, $\sigma^m$). Yee’s cell, however, utilizes staggered electric and magnetic fields. Equation (3.5) is adapted for this case. Consider the 1D domain shown in Figure 3.1. This domain is assumed to be long enough with respect to the total simulation time and is terminated by electric walls from both sides. We assume that
any reflections from the electric wall boundaries will not affect the measurements at the observation domain because of the length of the computational domain. In this case, the electric field is sampled at multiples of $\Delta z$ and $\Delta t$. It follows that we have $E_{x,i}^n = E_x(i\Delta x, n\Delta t), i = 1, 2, \ldots, N_E, n = 0, 1, 2, \ldots, N_T$. $N_E$ is the number of non-boundary electric field components. Similarly, the magnetic field components are shifted in time and step from the electric field ones. They are defined by $H_{y,j}^n = H_y((j - 0.5)\Delta x, (n + 0.5)\Delta t), j = 1, 2, \ldots, N_H, n = 0, 1, 2, \ldots, N_T$. We then define the vectors of electric and magnetic field components over the entire domain as

$$E^n = \begin{bmatrix} E_{x,1}^n & E_{x,2}^n & \cdots & E_{x,N_E}^n \end{bmatrix}^T,$$

$$H^n = \begin{bmatrix} H_{y,1}^n & H_{y,2}^n & \cdots & H_{y,N_H}^n \end{bmatrix}^T.$$

Equation (3.5) can be written for the whole domain as

$$\begin{bmatrix} \varepsilon & 0 \\ 0 & -\mu \end{bmatrix} \begin{bmatrix} \dot{E}^n \\ \dot{H}^n \end{bmatrix} + \begin{bmatrix} \sigma^e & 0 \\ 0 & -\sigma^m \end{bmatrix} \begin{bmatrix} E^n \\ H^n \end{bmatrix} + K_c \begin{bmatrix} E^n \\ H^n \end{bmatrix} = \begin{bmatrix} -J_{ix} \\ M_{iy} \end{bmatrix},$$

where $\varepsilon$, $\mu$, $\sigma^e$, and $\sigma^m$ are diagonal matrices with each diagonal element corresponding to a single computational cell. The square matrix $K_c$ discretizes the $\partial/\partial z$ operator into the difference between two successive field components for both the electric and magnetic field components. It can be shown that the components of the matrix $K_c$ are constants and independent of parameters. This matrix can also be shown to be a symmetric matrix. This property will prove useful in later derivations. The time derivatives of the first set of equations in (3.8) is evaluated at multiples of the time step $\Delta t$, whereas the second set of time derivatives are estimated at odd multiples of ($\Delta t/2$). This is consistent with Yee’s cell staggering of fields in both time and space.

The system of (3.8) can be written in the compact form

$$N \dot{V} + KV = G,$$

where $V = \begin{bmatrix} E^{nT} \\ H^{nT} \end{bmatrix}^T$ and

$$N = \begin{bmatrix} \varepsilon & 0 \\ 0 & -\mu \end{bmatrix}, \quad K = \begin{bmatrix} \sigma^e & 0 \\ 0 & -\sigma^m \end{bmatrix} + K_c = K_m + K_c, \quad G = \begin{bmatrix} -J_{ix} \\ M_{iy} \end{bmatrix}.$$
Note that the matrix $K$ is symmetric and is composed of the summation of two symmetric matrices $K_m$ and $K_c$. $K_m$ is a function of the electric conductivity in the domain, whereas $K_c$ is a matrix of constant coefficients.

To derive the adjoint system corresponding to (3.9), we follow a slightly different approach than the one adopted in Chapter 1. This derivation is similar to the one given in [5]. We assume that the $i$th parameter $p_i$ is perturbed by $\Delta p_i$. This results in perturbing the system matrices to $N + \Delta N_i$ and $K + \Delta K_i$. Note that the matrix $K_c$ is unchanged. Only the matrix $K_m$ is perturbed. The perturbation $\Delta p_i$ also results in perturbing the state variables (electric and magnetic field components in the whole domain) to $V + \Delta V_i$. The system (3.9) then becomes

$$(N + \Delta N_i)(\dot{V} + \Delta \dot{V}_i) + (K + \Delta K_i)(V + \Delta V_i) = G.$$ \hspace{1cm} (3.11)

Subtracting (3.9) from (3.11) and assuming that the excitation does not change with the perturbation in the $i$th parameter, we have the perturbed system

$$(N + \Delta N_i)\Delta \dot{V}_i + (K + \Delta K_i)\Delta V_i = - \Delta N_i \dot{V} - \Delta K_i V = - \Delta R_i,$$ \hspace{1cm} (3.12)

where $\Delta R_i = \Delta N_i \dot{V} + \Delta K_i V$ is the temporal residue vector associated with the $i$th parameter. Note that, unlike the derivation in Chapter 1, these residues may not be analytical functions of the parameters. Dividing both sides by a small enough $\Delta p_i$, we have the approximate expression:

$$(N + \Delta N_i)\frac{\partial \dot{V}}{\partial p_i} + (K + \Delta K_i)\frac{\partial V}{\partial p_i} = - \frac{\Delta R_i}{\Delta p_i}.$$ \hspace{1cm} (3.13)

Multiplying both sides of (3.13) by a yet-to-be determined adjoint vector $\lambda$ and integrating by parts, we have

$$\int_0^{T_m} \lambda^T (N + \Delta N_i) \frac{\partial \dot{V}}{\partial p_i} + \lambda^T (K + \Delta K_i) \frac{\partial V}{\partial p_i} \, dt = \int_0^{T_m} \lambda^T \left( - \frac{\Delta R_i}{\Delta p_i} \right) \, dt$$ \hspace{1cm} (3.14)

$$\lambda^T (N + \Delta N_i) \frac{\partial V}{\partial p_i} \Bigg|_0^{T_m} - \int_0^{T_m} \lambda^T (N + \Delta N_i) \frac{\partial V}{\partial p_i} \, dt + \int_0^{T_m} \lambda^T (K + \Delta K_i) \frac{\partial V}{\partial p_i} \, dt$$

$$= - \int_0^{T_m} \lambda^T \left( \frac{\Delta R_i}{\Delta p_i} \right) \, dt.$$ \hspace{1cm} (3.15)

Choosing the adjoint variable to have the zero terminal values $\lambda(T_m) = 0$ eliminates the first term in (3.15). The electric and magnetic fields are assumed to have zero initial values in the whole domain regardless of any change in the parameters. Equating the left-hand side of (3.15) to the second term of (3.2), we have

$$- \dot{\lambda}^T (N + \Delta N_i) + \lambda^T (K + \Delta K_i) = \left( \frac{\partial \psi}{\partial V} \right)^T,$$ \hspace{1cm} (3.16)
which is equivalent to

\[-(N + \Delta N_i)^T \lambda + (K + \Delta K_i)^T \lambda = \left( \frac{\partial \psi}{\partial V} \right) \].

The system (3.17) is labeled as the adjoint system. Its excitation, which is given by the right-hand side vector, is determined during the original simulation (3.9). Upon solving the system (3.17) for the values of the adjoint fields \( \lambda \) at every time step, the sensitivities of the objective function with respect to the \( i \)th parameter are given by

\[ \frac{\partial F}{\partial p_i} = \int_0^{T_m} \frac{\partial \psi}{\partial p_i} dt - \int_0^{T_m} \lambda^T (\Delta R_i) dt. \] (3.18)

Few things to notice regarding the adjoint system (3.17). First, the system matrices are parameter dependent. This implies that the vector of adjoint variable \( \lambda \) will be different for each parameter. For a system with \( n \) parameters, \( n \) adjoint simulations are needed to estimate all the sensitivities. This implies that no saving is achieved over the finite difference approximations. Luckily, because the perturbations \( \Delta p_i, \forall i \) are small enough, we can assume that the perturbations in the system matrices in (3.17) are negligible and do not disturb much the field distribution. Only one adjoint simulation is carried out for all parameters. This adjoint simulation is given by

\[-N^T \lambda + K^T \lambda = \left( \frac{\partial \psi}{\partial V} \right). \] (3.19)

The second thing to note about the adjoint system (3.19) is that there is a negative sign associated with the first-order derivative of \( \lambda \) with respect to the time variable \( t \). By utilizing the adjoint variable time \( \tau = T_m - t \), the system (3.19) can be written as

\[ N^T \frac{d \lambda}{d \tau} + K^T \lambda = Q(\tau) = \left[ \frac{\partial \psi(T_m - \tau)}{\partial V} \right]. \] (3.20)

Note that in (3.20), the excitation is applied backward in time. The excitation applied at time \( \tau = 0 \) (the beginning of the adjoint simulation) is the one stored at the end of the original simulation (\( t = T_m \)). Also, the system matrices in (3.20) are the transpose of those of the original system (3.9). Because of the way we structured the governing equations in (3.8), the system matrices are symmetric. The adjoint system (3.20) can thus be expressed in the expanded form

\[
\begin{bmatrix}
\varepsilon & 0 \\
0 & -\mu
\end{bmatrix}
\begin{bmatrix}
\dot{\lambda}_E \\
\dot{\lambda}_H
\end{bmatrix} +
\begin{bmatrix}
\sigma^e & 0 \\
0 & -\sigma^m
\end{bmatrix}
\begin{bmatrix}
\lambda_E \\
\lambda_H
\end{bmatrix} +
K_e
\begin{bmatrix}
\lambda_E \\
\lambda_H
\end{bmatrix} = \left[ \frac{\partial \psi(T_m - \tau)}{\partial V} \right],
\] (3.21)

where \( \lambda = [\lambda_E^T \lambda_H^T]^T \) are the adjoint electric and magnetic fields. The derivatives in (3.21) are all with respect to the adjoint time variable \( \tau \). The adjoint system (3.21) represents an FDTD simulation with the same material properties
but with different excitation. The excitation utilized in (3.20) is determined during the original FDTD simulation (3.8). It is applied in a backward way as per (3.20).

The steps presented by (3.1)–(3.21) are summarized by the following algorithm steps:

1. **Parameterization:** Determine the FDTD cells affected by a forward perturbation $\Delta p_i$ in the $i$th parameter. We call this set of cells $S_{pi}^+$, the forward perturbation region of the $i$th parameter. Determine also the set of cells affected by a backward perturbation in the $i$th parameter $S_{pi}^-$. This set is denoted as the backward perturbation region of the $i$th parameter.

2. **Original simulation:** Carry out the original simulation (3.8) using the FDTD time marching scheme explained in Chapter 2. In every time step, store the residue vectors $\Delta R_i = \Delta N_i \hat{V} + \Delta K_i V, i = 1, 2, \ldots, n$. Note that these residues require knowledge of the original response $V$ and its first-order derivative $\dot{V}$. Both are available through the original simulation. They also require knowledge of how the system matrices change relative to a forward perturbation $\Delta p_i$ in the $i$th parameter. The set $S_{pi}^+$ determines the set of FDTD cells affected by such a perturbation and thus the components of the systems matrices affected by this perturbation. These components correspond to the nonzero residue values. Store the adjoint excitation $\hat{y}/\hat{V}$ in the observation domain. This vector offers the adjoint electric sources in the adjoint simulation (3.20).

3. **Adjoint simulation:** Carry out the adjoint simulation (3.20). Store the adjoint fields at all locations where the residue vectors have nonzero values. Carry out the inner product in (3.18) to estimate the sensitivities.

The way the adjoint fields are stored needs some clarification. Consider a small dielectric discontinuity of length $L$ inside an air-filled computational domain. If we perturb this discontinuity in length by only one cell $\Delta p = \Delta z$, we see that the residue vector has only one nonzero component as indicated by the gray cell $S_{p}^+$ in Figure 3.2(a). To accurately solve the adjoint field, we should solve the perturbed system (3.17) and then store the field inside the gray area as well. However, we do not solve the perturbed adjoint system and instead solve the nominal system shown in Figure 3.2(b). The adjoint fields inside the forward perturbed region $S_{p}^+$ are then approximated by the fields inside the region $S_{p}^-$ of the unperturbed adjoint system. This approach is called one-to-one mapping [1], and it was shown to give good results in adjoint sensitivity estimation. This mapping is based on establishing a one-to-one correspondence between the perturbed and unperturbed systems.

When estimating the sensitivities with respect to the material properties in Figure 3.2, we note that the whole discontinuity is affected by this change. It follows that $S_{p}^+ = S_{p}^-$ for material properties and is equal to the whole domain of the discontinuity. In relatively large discontinuities, this may require significant memory storage.
Example 3.1 As an example for the 1D FDTD case, we consider the dielectric slab problem shown in Figure 3.3. We take into account two parameters \( p = [L \ \varepsilon_r]^T \), the length of the discontinuity in the \( z \)-direction, and the relative permittivity of the discontinuity material. The utilized discretization is \( \Delta z = 0.5 \) mm. The domain is excited with a Gaussian-modulated \( x \)-polarized electric current density source at \( z = 20\Delta z \). This excitation waveform is determined by calling the function `get_gaussian()`. The original excitation is shown in Figure 3.4(a). The domain has a length of \( 600\Delta z \). The considered objective function is given by

\[
F = 0.5 \int_0^{T_m} E_z^2 dt, \tag{3.22}
\]

where the electric field in (3.22) is observed at \( z = 580\Delta z \), which is our observation domain. The objective function (3.22) is a measure of the energy delivered to the output port. The domain is terminated from both sides by a Mur’s boundary condition [6]. This boundary condition updates the fields at the edges of the domain to imitate the fields resulting from an infinite domain.
The MATLAB® implementation for this problem is given by the script `avm_slab`. This script starts by calling a settings file containing all the information about the problem. This settings file is given in MATLAB Listing M3.1. The dielectric slab with relative permittivity $\varepsilon_r = 3$ starts at $100\Delta z$ and has a length of $40\Delta z$. We make use of the function `fdtd_1d()`, whose theory is explained in Chapter 2, to calculate the electric field value at every point in the computational domain for all time steps.

Figure 3.4 The original Gaussian-modulated excitation (a) and the adjoint excitation (b) for the dielectric slab example
The AVM algorithm starts by simulating the original structure. Because of the assumed objective function in (3.22), the corresponding adjoint excitation is given by

$$\frac{\partial \psi(n\Delta t)}{\partial V} = \begin{bmatrix} 0 & \cdots & E_{580} & \cdots & 0 \end{bmatrix}^T,$$

where this vector has only one nonzero value at the 580th component (corresponding to the observation domain). It follows that we store the electric field only at this point. It is then utilized as an excitation for the adjoint simulation. As stated earlier, this adjoint excitation is applied backward in time. MATLAB Listing M3.2 shows the implementation of the AVM approach.

Few things to note regarding MATLAB Listing M3.2: We assume forward perturbation in both parameters. When the slab length is perturbed by only one cell (1\Delta z), the dielectric constant at the cell with an index \texttt{slab_stop}+1 changes from that of air to that of the dielectric slab. The command \texttt{s_p1_f = [slab_stop + 1]'} thus sets the set \(S^+_1\) related to the first parameter. Similarly, when the second parameter, the slab’s dielectric constant changes, all cells comprising the slab undergo a dielectric change. The command \texttt{s_p2_f = [slab_start:slab_stop]'} thus sets the set \(S^+_2\) of perturbed cells associated with the second parameter. Once the original simulation is carried out, the temporal fields at the two sets of cells \(S^+_1\) and \(S^+_2\) are stored to be utilized later in evaluating the residue vector for each parameter.
The MATLAB Listing M3.2 also evaluates the residues for the two considered parameters. Because we assume that the discontinuity is lossless, changing the length of the discontinuity, or its relative dielectric constant change only the dielectric constant at some cells. It follows from the original FDTD system (3.8) that only the matrix $N$ is affected by these perturbations. By perturbing the length $L$ by only one cell in the forward direction, we change the dielectric constant at this specific cell. The residue is thus given by

$$
\frac{\Delta N}{\Delta \rho_1} \hat{V} = \frac{1}{\Delta z} \begin{bmatrix}
\Delta \varepsilon_i & 0 & 0 \\
0 & 0 & -\varepsilon_i \hat{E}^n \\
0 & -\varepsilon_i \hat{H}^n & 0
\end{bmatrix} = \frac{1}{\Delta z} \begin{bmatrix}
0 \\
\Delta \varepsilon \hat{E}^n_x \\
0
\end{bmatrix},
$$

(3.24)

where this change happens only at one cell (the $l$th cell) at the edge of the discontinuity. Also, $\Delta \varepsilon = \varepsilon_r - 1$ as this cell changes its dielectric constant from that of air to that of the discontinuity. The residue for the first parameter in (3.24) has thus only one nonzero component. Equation (3.24) is used also to evaluate the residue vector for the second parameter. It should be noted though that this residue vector has 40 components as all cells of the slab are affected by perturbing the dielectric constant. For the second parameter, the perturbation in the dielectric constant is taken as $\Delta \varepsilon = 0.1$. The residues of the first and second parameters are evaluated through the commands as

```matlab
residue_p1_f = (epsilon_slab - epsilon_air) * epsilon_0 * diff(dir_p1_f)/dt;
residue_p2_f = diff(dir_p2_f) * dp2 * epsilon_0/dt;
```
where the MATLAB command `diff()` is used to differentiate the stored electric field components. The adjoint excitation (3.23) is implemented through the command

```matlab
adjoint_excitation = -1.0*flipud(td_original(:,response_domain))
```

As mentioned earlier, the adjoint excitation is applied backward in time. This is implemented using the MATLAB command `flipud()`. Also, because the adjoint excitation (3.23) is introduced in the solver as a current source, a comparison with (3.8) reveals that a negative sign is also needed. The utilized adjoint excitation is shown in Figure 3.4(b). The last two lines of MATLAB Listing M3.2 implements the second integral in (3.18) for each residue vector. The output from MATLAB Listing M3.2 is

```
sensitivity_p1 = -2.8916e-13
sensitivity_p2 = -3.7169e-16
```

To verify these results, the central finite difference approach was also implemented at the response level. MATLAB Listing M3.3 implements the central finite differences (CFD) calculations by perturbing both parameters in the forward and backward directions. The output from this code is given by

```
sensitivity_p1_cfd = -2.8213e-13
sensitivity_p2_cfd = -3.7157e-16
```
The two results show good match. The relative error between the two results is within 2.4%. The CFD approach requires four extra simulations whereas the AVM approach requires only one extra simulation.

The derivation (3.11)–(3.18) does not assume a certain polarity for the virtual perturbation $\Delta p_i$. The implementation given in MATLAB Listing M3.2 assumes forward perturbation of both parameters ($\Delta p_i > 0$). We can, however, assume that both parameters are perturbed in the backward direction. The achieved AVM sensitivities are labeled as backward adjoint sensitivities. These sensitivities require the same adjoint simulation. The only difference is that the residues are stored in different domain cells for both the original simulation and the adjoint simulation. An implementation of the backward AVM approach is given in MATLAB Listing M3.4. The output from this code is

```matlab
Sensitivities_p1_bavm = -2.8304e-13
sensitivities_p2_bavm = -3.7169e-16
```

This result also matches very well that of the CFD results. The backward AVM approach requires also the same adjoint simulation. A central AVM (CAVM) approach utilizes the average of the forward AVM result and the backward AVM result. Evaluating CAVM has been proposed with a wave equation formulation [3]. The CAVM requires more memory storage than the forward or backward AVM approaches.
To compare the CAVM approach to the more expensive CFD approach over a wider range of parameters, we sweep the length of the slab (the parameter `slab_length`) in the MATLAB Listing M3.1 from 40 to 500 and plot the sensitivities obtained using AVM and CFD. We also include the results for the forward finite differences (FFD) and backward finite difference (BFD). The results in Figure 3.5

![Figure 3.5 A comparison between the CAVM sensitivities (x) as compared to the CFD sensitivities (−), the FFD sensitivities (⋆), and the BFD estimates (•) for a sweep of the parameter p1 of the dielectric slab example](image-url)
show good match between the CAVM approach (the average of the forward AVM and backward AVM) and the more expensive CFD. Both the FFD and the BFD deviate slightly from both the CAVM and the CFD. Throughout this book, we will compare our adjoint sensitivity results with either the CFD or the FFD approximations.

3.3 The 2D TM case

The same concepts applied in Section 3.2 to the 1D FDTD case apply also to the 2D case. As explained in Chapter 2, when we assume that the fields do not change in one direction (say the z-direction), Maxwell’s equations can be decomposed into two uncoupled sets of equations. The first one has components $E_z, H_x,$ and $H_y$ and is denoted as the TMz case. The second one has components $H_z, E_x,$ and $E_y$ and is denoted as the TEz case. These two types of field solutions exist independent of one another. We focus in this section on the TMz case and give a complete derivation of the corresponding adjoint system in the presence of electric and magnetic current densities as well as electric and magnetic conductivities.

For the TMz case, the governing equations in the time domain are given by

$$
\begin{align*}
\mu \frac{\partial H_x}{\partial t} &= -\frac{\partial E_z}{\partial y} - \sigma^m H_x - M_{ix}, \\
\mu \frac{\partial H_y}{\partial t} &= \frac{\partial E_z}{\partial y} - \sigma^m H_y - M_{iy}, \\
\varepsilon \frac{\partial E_z}{\partial t} &= \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - \sigma^e E_z - J_{iz}.
\end{align*}
$$

Here, we assume that the materials composing the computational domain are nondispersive and isotropic. Equations (3.25) are valid for any point in the computational domain. Arranging these equations in a matrix form, we have

$$
\begin{bmatrix}
-\mu & 0 & 0 \\
0 & -\mu & 0 \\
0 & 0 & \varepsilon
\end{bmatrix}
\begin{bmatrix}
\dot{H}_x \\
\dot{H}_y \\
\dot{E}_z
\end{bmatrix}
+ 
\begin{bmatrix}
0 & 0 & -\partial/\partial y \\
0 & 0 & \partial/\partial x \\
\partial/\partial y & -\partial/\partial x & 0
\end{bmatrix}
\begin{bmatrix}
H_x \\
H_y \\
E_z
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
0 \\
0
\end{bmatrix}
+ 
\begin{bmatrix}
-\sigma^m & 0 & 0 \\
0 & -\sigma^m & 0 \\
0 & 0 & \sigma^e
\end{bmatrix}
\begin{bmatrix}
H_x \\
H_y \\
E_z
\end{bmatrix}
= 
\begin{bmatrix}
M_{ix} \\
M_{iy} \\
-J_{iz}
\end{bmatrix}.
$$

The first and second equations in (3.26) are multiplied by a negative sign. The reason for organizing the system of equations this way will become clear later. We discretize this system of equations using Yee’s cell configuration as shown in Figure 2.7. For simplicity, we assume that the domain is very large and is
terminated by electric walls in all four directions. The assumption of a very large domain guarantees that any reflections from the boundaries will not affect the fields measured inside the observation domain. In a realistic problem though, a Perfectly Matched Layer (PML) is used to terminate the domain and to imitate the infinite extension of the domain in all directions.

For the TMz case, the electric field $E_z$ is sampled at multiples of $\Delta t$ at the grid points indicated in Figure 2.7. The magnetic fields are sampled at odd multiples of $0.5\Delta t$ and are staggered from the electric field components as explained earlier in Chapter 2. The number of FDTD cells in the $x$- and $y$-directions is $N_x$ and $N_y$, respectively. Writing (3.26) for the whole computational domain, we have

$$
\begin{bmatrix}
-\mu & 0 & 0 \\
0 & -\mu & 0 \\
0 & 0 & \varepsilon
\end{bmatrix}
\begin{bmatrix}
\dot{H}_x \\
\dot{H}_y \\
\dot{E}_z
\end{bmatrix}
+
\begin{bmatrix}
-\sigma^m & 0 & 0 \\
0 & -\sigma^m & 0 \\
0 & 0 & \sigma^e
\end{bmatrix}
\begin{bmatrix}
H_x \\
H_y \\
E_z
\end{bmatrix}
+ K_c
\begin{bmatrix}
H_x \\
H_y \\
E_z
\end{bmatrix}
=
\begin{bmatrix}
M_{ix} \\
M_{iy} \\
-J_{iz}
\end{bmatrix}
$$

where $H_x \in \mathbb{R}^{(N_x+1)N_y}$ is a column vector of all $H_x$ components in the domain, $H_y \in \mathbb{R}^{(N_y+1)N_x}$ is a column vector of all $H_y$ components in the domain, and $E_z \in \mathbb{R}^{(N_x+1)(N_y+1)}$ is a column vector of all $E_z$ components in the domain. These three vectors constitute the vector of state variables $V = \begin{bmatrix} H_x^T & H_y^T & E_z^T \end{bmatrix}^T$ for the TMz case. $\mu$, $\varepsilon$, $\sigma^m$, and $\sigma^e$ are diagonal matrices with the corresponding material components. $K_c$ is a square matrix with material-independent constant coefficients that represents the discretization of the curl operator. With the signs used in (3.26), it can be shown that this matrix is also symmetric. This property is useful in deriving the adjoint system. The vectors $M_{ix}$, $M_{iy}$, and $J_{iz}$ are the vectors of impressed magnetic and electric current density excitations in the domain. The differential system (3.27) has the form (3.9) with the system matrices and vectors:

$$
N = \begin{bmatrix}
-\mu & 0 & 0 \\
0 & -\mu & 0 \\
0 & 0 & \varepsilon
\end{bmatrix}, \quad K = \begin{bmatrix}
-\sigma^m & 0 & 0 \\
0 & -\sigma^m & 0 \\
0 & 0 & \sigma^e
\end{bmatrix} + K_c, \quad \text{and} \quad G = \begin{bmatrix} M_{ix} \\
M_{iy} \\
-J_{iz}
\end{bmatrix}.
$$

(3.28)

Following a similar derivation to that given by (3.11)–(3.19), the corresponding adjoint system used to evaluate the sensitivities of an objective function of the form (3.1) is given by

$$
-N^T \dot{\lambda} + K^T \lambda = \begin{bmatrix} \frac{\partial \psi}{\partial V} \end{bmatrix}
$$

(3.29)
where \( \lambda \) is the vector of adjoint state variables. Changing the time variable from \( t \) to the adjoint variable time \( \tau = T_m - t \), the system (3.29) can be written as

\[
\begin{bmatrix}
-\mu & 0 & 0 \\
0 & -\mu & 0 \\
0 & 0 & \varepsilon
\end{bmatrix}
\begin{bmatrix}
\dot{\lambda}_{Hx} \\
\dot{\lambda}_{Hy} \\
\dot{\lambda}_{Ez}
\end{bmatrix}
+ \begin{bmatrix}
-\sigma^m & 0 & 0 \\
0 & -\sigma^m & 0 \\
0 & 0 & \sigma^e
\end{bmatrix}
\begin{bmatrix}
\lambda_{Hx} \\
\lambda_{Hy} \\
\lambda_{Ez}
\end{bmatrix}
+ K_e
\begin{bmatrix}
\lambda_{Hx} \\
\lambda_{Hy} \\
\lambda_{Ez}
\end{bmatrix}
= \frac{\partial \psi}{\partial V}.
\]

(3.30)

In (3.30), the vectors \( \lambda_{Hx}, \lambda_{Hy}, \) and \( \lambda_{Ez} \) are the magnetic adjoint fields in the \( x \)-direction, the magnetic adjoint fields in the \( y \)-direction, and the electric adjoint fields in the \( z \)-direction. These vectors represent the adjoint fields over the whole computational domain. The adjoint system (3.30) has the same material distribution as the original system in (3.27). The derivatives in (3.30) are with respect to the adjoint time variable \( \tau \). The only difference between the original system (3.27) and the adjoint system (3.30) is that the excitation vector that depends on the objective function. It follows that the same FDTD solver utilized in solving the original problem (3.27) is also used to solve the adjoint system (3.30) after making the necessary excitation change.

As was explained for the 1D case, the adjoint system (3.30) is parameter independent and is thus an approximate one. The exact adjoint system, given by (3.17), would require \( n \) adjoint simulations. Using the parameter-independent adjoint system (3.30) requires approximating the adjoint responses obtained from the exact adjoint systems (3.17) by those obtained from the system (3.30). A one-to-one mapping, similar to the one applied for the 1D case, is utilized in this case [5]. Figure 3.6 illustrates this mapping for the 2D TMz example case. Consider a lossy-dielectric discontinuity in the parameter space. The exact adjoint system required for the parameter \( L \) is shown in Figure 3.6(b). In this adjoint system, the parameter \( L \) is perturbed by the smallest possible on-grid perturbation \( \Delta y \). It is required in this adjoint system that we store the adjoint responses in the forward perturbation subdomain \( S_L^+ \). However, we do not actually carry out the adjoint simulation shown in Figure 3.6(b). Instead, we carry out the parameter-independent adjoint simulation shown in Figure 3.7(c). The adjoint fields in the forward perturbation subdomain \( S_L^+ \) in Figure 3.6(b) are approximated by the fields in the backward perturbation subdomain \( S_L^- \) in Figure 3.6(c). The same concept applies to all geometrical parameters.

For material parameters, the approach is illustrated in Figure 3.7. The considered material property in this figure is the permittivity. The original simulation is shown in Figure 3.7(a). The exact adjoint simulation, shown in Figure 3.7(b), requires making a small change to the permittivity. The parameter-independent adjoint simulation, shown in Figure 3.7(c), does not perturb any of the parameters. The field in the domain affected by the perturbation in Figure 3.7(b), shown by the area \( S_e^+ \), is approximated by the adjoint field in the area \( S_e^- \) of Figure 3.7(c). Taking the limit as the perturbation in the permittivity approaches zero, \( S_e^+ \) becomes identical to the exact adjoint simulation shown in Figure 3.7(c). Our adjoint field
Calculations are thus exact for the material parameters. The two sets of forward and backward perturbation subdomains occupy the complete volume of the discontinuity.

Similar to the 1D case, the residues are calculated during the original simulation at the cells affected by the perturbation. The $i$th residue related to the $i$th parameter is given by

$$D_R^i = D_N^i \hat{V} + D_K^i V, \quad i = 1, 2, \ldots, n.$$  \hspace{1cm} (3.31)

Figure 3.6 The one-to-one mapping for shape parameters: (a) the original simulation where the objective function is calculated over the observation domain, (b) the exact adjoint problem for the parameter $L$ where the adjoint fields are stored over the forward perturbation subdomain $S_L^+$, and (c) the approximate adjoint simulation where the exact adjoint fields in (b) are approximated by the adjoint fields over the backward perturbation subdomain $S_L^-$. The perfectly matched boundaries are not shown.
We analyze this term further for the 2D case. Considering that the matrix $K_c$ is parameter-independent, the residue (3.31) can be expanded as

$$\Delta R_i = \begin{bmatrix} -\Delta \mu & 0 & 0 \\ 0 & -\Delta \mu & 0 \\ 0 & 0 & \Delta \varepsilon \end{bmatrix} \begin{bmatrix} \dot{H}_x \\ \dot{H}_y \\ \dot{E}_z \end{bmatrix} + \begin{bmatrix} -\Delta \sigma_{i}^m & 0 & 0 \\ 0 & -\Delta \sigma_{i}^m & 0 \\ 0 & 0 & \Delta \sigma_{i}^e \end{bmatrix} \begin{bmatrix} H_x \\ H_y \\ E_z \end{bmatrix},$$

(3.32)

where $\Delta \mu$, $\Delta \varepsilon$, $\Delta \sigma_{i}^m$, and $\Delta \sigma_{i}^e$ are the changes in the material properties due to a perturbation $\Delta p_i$ in the $i$th parameter. Because all the perturbation matrices in (3.32) are diagonal, this system implies that original field components and their derivatives are multiplied by weights or “stamps” that represents the change in the $i$th parameter. These stamps are nonzero only within the assumed perturbation regions of the parameters.

A subtle difference exists between the 1D AVM algorithm introduced earlier and the 2D and 3D AVM implementations using FDTD. This subtle difference can be explained in Figure 3.8 where a lossy-dielectric discontinuity is located in the computational domain. Note that the electric field component $E_z$ is multiplied by the permittivity and conductivity at its on-grid location. For interior nodes marked with dots in Figure 3.8, the permittivity is that of the dielectric $\varepsilon = \varepsilon_0 \varepsilon_r$. For edge nodes marked with crosses, the effective dielectric constant in the $z$-direction is the average between that of air (the assumed surrounding medium) and that of the dielectric material $\varepsilon = 0.5 \varepsilon_0 \varepsilon_r + 0.5 \varepsilon_0$. For corner nodes marked with diamonds, the effective permittivity is a weighted sum of the dielectric and air according to their...
area and is given by $\varepsilon = 0.25\varepsilon_0\varepsilon_r + 0.75\varepsilon_0$. We consider in Figure 3.8 the case where the length is perturbed in the forward direction by the smallest possible on-grid perturbation $\Delta L = \Delta \tau$. In this case, all nodes within the domain $S$ undergo a material change. These nodes will thus have corresponding nonzero components in the residue vector (3.32). The edge nodes marked with a cross in Figure 3.8 will have their permittivity changed from $\varepsilon = 0.5\varepsilon_0\varepsilon_r + 0.5\varepsilon_0$ to the new value $\varepsilon = \varepsilon_0\varepsilon_r$. The permittivity perturbation is thus $\Delta \varepsilon = \varepsilon_0\varepsilon_r - (0.5\varepsilon_0\varepsilon_r + 0.5\varepsilon_0) = 0.5\varepsilon_0\varepsilon_r - 0.5\varepsilon_0 = 0.5\varepsilon_0(\varepsilon_r - 1)$. This is the stamp that will be used in calculating the residue (3.32). Similarly, the corner nodes marked with a diamond in Figure 3.8 will have their permittivity change from $\varepsilon = 0.25\varepsilon_0\varepsilon_r + 0.75\varepsilon_0$ to $\varepsilon = 0.5\varepsilon_0\varepsilon_r$. The permittivity perturbation at these nodes is thus given by $\Delta \varepsilon = 0.5\varepsilon_0\varepsilon_r + 0.5\varepsilon_0 - (0.25\varepsilon_0\varepsilon_r + 0.75\varepsilon_0) = 0.25\varepsilon_0\varepsilon_r - 0.25\varepsilon_0 = 0.25\varepsilon_0(\varepsilon_r - 1)$. In addition to these cells, Figure 3.8 shows that nodes just outside the discontinuity (marked with black squares and triangles) also underwent a material change. In the unperturbed structure (with discontinuity length $L$), these cells are fully surrounded by air and thus their permittivity was $\varepsilon = \varepsilon_0$. For the perturbed structure (with discontinuity length $L + \Delta L$), these cells are surrounded by both air and dielectric. The nodes with black squares have new permittivity $\varepsilon = 0.5\varepsilon_0\varepsilon_r + 0.5\varepsilon_0$. The permittivity perturbation at these nodes is thus given by $\Delta \varepsilon = 0.5\varepsilon_0\varepsilon_r + 0.5\varepsilon_0 - (\varepsilon_0) = 0.5\varepsilon_0\varepsilon_r - 0.5\varepsilon_0 = 0.5\varepsilon_0(\varepsilon_r - 1)$. For the corner nodes with black triangles, the perturbation is $\Delta \varepsilon = 0.25\varepsilon_0\varepsilon_r + 0.75\varepsilon_0 - \varepsilon_0 = 0.25\varepsilon_0(\varepsilon_r - 1)$. In total, if the discontinuity has a width of 6 cells as shown in Figure 3.8, 14 electric field components are affected and the corresponding residue in (3.32) will have 14 nonzero components. The same approach applies to other material properties as well. Note that the edges have half the perturbation $(\varepsilon_0\varepsilon_r - \varepsilon_0)$, whereas the corner nodes have quarter of this perturbation. We utilize this later in our implementation of this approach.

In order to evaluate the inner product (3.18) for each parameter, the residue (3.32) must be divided by the perturbation in the $i$th parameter $\Delta p_i$. If $p_i$ is a material-related parameter, we can make $\Delta p_i$ arbitrarily small and this leads to an analytical calculation of the stamps. For example, for the relative permittivity

![Figure 3.8](image.png)
parameter, we have

$$\frac{\Delta R}{\Delta \varepsilon_r} \approx \frac{dR}{d\varepsilon_r} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{H}_x \\ \dot{H}_y \\ \dot{E}_z \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} H_x \\ H_y \\ E_z \end{bmatrix},$$  (3.33)

where the diagonal matrix $\partial \varepsilon_r / \partial \varepsilon_r$ contains components belonging to the set $\{0, \varepsilon_0, 0.5\varepsilon_0, 0.25\varepsilon_0\}$ depending whether the node is outside the affected discontinuity, in its interior, at its edges, or at one of its corners as explained earlier. Similarly, for the electric conductivity we can also write

$$\frac{\Delta R}{\Delta \sigma_e} \approx \frac{dR}{d\sigma_e} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{H}_x \\ \dot{H}_y \\ \dot{E}_z \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \beta \end{bmatrix} \begin{bmatrix} H_x \\ H_y \\ E_z \end{bmatrix},$$  (3.34)

where the diagonal matrix $\beta$ contains components belonging to the set $\{0, 1, 0.5, \text{ and } 0.25\}$ depending on whether the node is outside the affected discontinuity, in its interior, at its edges, or at one of its corners. Similar to the permittivity case, we assume that the surrounding is air and the conductivity at any node is a weighted sum between the conductivity of the discontinuity and of that of air ($\sigma_e = 0$). The same concept applies when calculating the stamps for all material parameters.

For shape parameters, the situation is different. The smallest possible perturbation is limited in a numerical solver by the utilized spatial discretization. For example, if the length of a nonmagnetic discontinuity is perturbed by $\Delta y$ in the $y$-direction, the corresponding residue is evaluated through the formula

$$\frac{\Delta R_i}{\Delta p_i} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \Delta \varepsilon_i/\Delta p_i \end{bmatrix} \begin{bmatrix} \dot{H}_x \\ \dot{H}_y \\ \dot{E}_z \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \Delta \sigma_e/\Delta p_i \end{bmatrix} \begin{bmatrix} H_x \\ H_y \\ E_z \end{bmatrix}. $$ (3.35)

In this case, both the $N$ and $K$ matrices have nonzero stamps. The concepts explained in (3.33)–(3.35) can be generalized for all other material and shape parameters.

The weights or “stamps” to be used for each parameter are determined beforehand as a part of the parameterization step. This step determines the field components affected by the perturbation $\Delta p_i$, $i = 1, 2, \ldots, n$ and the stamp multiplying each field component. During the original simulation, each residue is calculated using (3.32) by multiplying the field components with their corresponding stamps. These residues are then stored at each time step.

The AVM algorithm for the 2D TMz FDTD case is similar to the 1D case and can be summarized in the following steps:

1. Parameterization: Determine the FDTD cells affected by the perturbation $\Delta p_i$ in the $i$th parameter in the forward direction $S^+_p$ and in the backward direction $S^-_p$. Determine how many field components will be stored for each parameter and their corresponding stamps.
2. Original simulation: Carry out the original simulation (3.27) using the FDTD time marching scheme explained in Chapter 2. In every time step, store the residue vectors $\partial R_i/\partial p_i$ for material parameters or $\Delta R_i/\Delta p_i$ for shape parameters, $\forall i$, by multiplying each affected field component by its corresponding stamp. Store the adjoint excitation $\partial \psi/\partial V$ in the observation domain. This will be used as adjoint electric and magnetic current sources in the adjoint simulation (3.30).

3. Adjoint simulation: Carry out the adjoint simulation (3.30). Store the adjoint fields at all locations where the residue vectors have nonzero values. The one-to-one mapping is utilized in this case. Carry out the inner product in (3.18) to estimate the sensitivities.

Example 3.2 Consider the 2D FDTD structure shown in Figure 3.9. The domain is excited with a plane wave at the input plane with the temporal profile shown in Figure 3.10. The discretizations utilized are $\Delta x = \Delta y = 1.0$ mm. The utilized time step is $\Delta t = 2.1228$ ps. The objective function is obtained by integrating the square of the electric field on the line $y = 61 \Delta y$. It is thus given by

$$F = \int_0^{T_w} \int_\Omega E_z^2 \, dt,$$

(3.36)

where $\Omega$ is the line $y = 61 \Delta y$ extending from $x = -30 \, \Delta x$ to $x = 30 \, \Delta x$. The domain is surrounded by a CPML from all four sides. Transmitted and reflected wave decomposition are also carried out inside this domain [7]. Our target is to evaluate...
the sensitivities of the objective function (3.36) with respect to the parameters of the dielectric discontinuity $p = [\varepsilon_r \sigma^e x_{\text{min}} x_{\text{max}} y_{\text{min}} y_{\text{max}}]^T$. These parameters have the nominal values $p = [2.0 \ 0 \ -5.0 \text{ mm} \ 6.0 \text{ mm} \ 28.0 \text{ mm} \ 40.0 \text{ mm}]^T$.

For simplicity, we assume that we deal only with rectangular discontinuities. The first rectangle represents the air background and is assumed to be constant. This is why the loop starts with an index of 2.

To carry out an AVM analysis for this problem, we adapt a 2D FDTD codes from [7]. MATLAB Listing M3.5 shows the basic script `TestAVM` that invokes both the original and the adjoint simulation. The script `fddt_solve_original` carries out the original simulation, stores the residues associated with each parameter, and determines the excitation of the adjoint simulation. This script is shown in MATLAB Listing M3.6.
A number of functions have been added to implement the adjoint sensitivity analysis. The first one is the `initialize_residue_storage`. This function determines the special sets $S^+_{pi}, i = 1, 2, \ldots, 6$. This script is shown in MATLAB Listing M3.7. Notice that the script M3.7 starts with the second rectangular discontinuity as the first rectangle constitutes the surrounding air background that is independent of all parameter.

The function `get_node_indices()` returns the indices of the different sides of each rectangular discontinuity after the CPML and air buffer cells have been taken into account. Based on the given implementation of this function, a rectangular discontinuity spans cells with corner indices $(ni.is, ni.js)$ to $(ni.ie-1, ni.je-1)$. This implies that the sets $S^+_{p1}$ and $S^+_{p2}$ are associated with the two material parameters $\varepsilon_r$ and $\sigma_e$ to occupy the same spatial domain as the discontinuity itself. The third parameter is $x_{\text{min}}$ of the discontinuity. Increasing this parameter by the on-grid perturbation $Dx$ results in changing the material properties of one column of cells with corner indices $(ni.is, ni.js)$ to $(ni.is, ni.je-1)$. Similarly, the fourth parameter is $x_{\text{max}}$ of the discontinuity. Increasing this parameter by the on-grid perturbation $Dx$ results in changing the material properties of the cells with indices $(ni.ie, ni.js)$ to $(ni.ie, ni.je-1)$. This is the line of cells just outside the discontinuity in the $+ve x$-direction. The same applies to the fifth and sixth parameters ($y_{\text{min}}$ and $y_{\text{max}}$ of the discontinuity).

Once the domain subsets $S^+_{pi}, i = 1, 2, \ldots, 6$ have been determined, the next step is to assign the stamps associated with the field components in each subset.
This is achieved by the script `initialize_residue_stamps` which is shown in MATLAB Listing M3.8. As we consider only changes in the permittivity and electric conductivity (as is the case in problems with nonmagnetic materials), we need only to create stamps for the electric field \(E_z\). Considering (3.32), we note that we have two stamps. There is the \(N\)-matrix stamp which is dependent on the change in the dielectric constant. This stamp is multiplied by the derivative of the electric field at every node affected by the parameter change. There is also the \(K\)-matrix stamp which is dependent on the change in the electric conductivity. This stamp
is multiplied by all components of the electric field itself (not its derivative) affected by the parameter perturbation. This is why in the MATLAB Listing M3.8, we store two vectors of stamps. The first one is \texttt{residue\_sampled\_field(parameter\_index).N\_stamp\_Ez} and the second one is \texttt{residue\_sampled\_field(parameter\_index).K\_stamp\_Ez}. Note that both are stored as vectors even though the product in (3.32) is a matrix–vector product. This is because all the matrices in (3.32) are diagonal matrices and every field component is only multiplied by just one weight (stamp). The variables \texttt{N\_x} and \texttt{N\_y} are the number of stored \(E_z\) components in the \(x\)- and \(y\)-directions, respectively. These variables are dependent on the rectangular discontinuity and the parameter index of each rectangular discontinuity. They are calculated inside the script \texttt{build\_Ez\_stamp} given in the MATLAB Listing M3.9.

The script M3.9 starts by evaluating the number of stored layers of fields in the \(x\)- and \(y\)-direction corresponding to the perturbation in every parameter. If the parameter is the permittivity or the electric conductivity, \(E_z\) is stored throughout the whole discontinuity. If the parameter is a shape parameter (\(x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, \text{ and } y_{\text{max}}\)), we store only the electric field in the perturbation domain. Note that two layers of \(E_z\) are stored even though we are only perturbing one layer of cells as explained earlier. Once the number of field components is determined, the script M3.9 moves to determine the stamp based on the parameter type. If the parameter is a material parameter \((i = 1, 2)\), then the whole discontinuity is affected. For a discontinuity with size \(n_x\) by \(n_y\), two stamps are created for the \(N\) and \(K\) matrices each with a size of \((n_x + 1)(n_y + 1)\). As explained earlier, the weights change according to whether the \(E_z\) component is an interior component, an edge component, or a corner component.

The script \texttt{get\_tangential\_stamp()} creates the stamps for the four shape parameters. It receives as an input the number of \(E_z\) components in the \(x\)-direction \texttt{N\_x} and the number of \(E_z\) components in the \(y\)-direction \texttt{N\_y} of the perturbation subdomain. The size of the returned stamp vector is thus \(N\_x*N\_y\).
This function receives only the relative perturbation that occurs in the material parameters as a result of the shape parameter perturbation. The components of the stamp, as explained earlier, will be one half or one quarter of this perturbation depending on the node location. Note if $x_{\text{max}}$ is perturbed in the forward direction, some cells are converted from air to a dielectric and the relative dielectric perturbation is $\frac{\epsilon_0 \epsilon_r}{\epsilon_0} / D_x$. If $x_{\text{min}}$ is perturbed in the forward direction, some cells are converted from dielectric to air and the relative dielectric perturbation is $\frac{\epsilon_0}{\epsilon_0 \epsilon_r} / D_x$. The same concept applies to the change in the conductivity associated with the $K$-matrix stamp.

Once the perturbation subdomains $S_{pi}^+$ and their corresponding stamps have been determined for all parameters, the algorithm carries out the original FDTD time marching loop. This loop is shown in MATLAB Listing M3.10. It updates the magnetic fields, and then it updates the electric fields as discussed in Chapter 2 except for two differences. The objective function value is updated at every time step where the integral (3.36) is approximated by a summation. The second difference is that the residues given by (3.33)–(3.35) are also calculated using the script `store_original_fields`. This script is shown in MATLAB Listing M3.11.
The script M3.11 starts by first recovering the indices of the perturbation domain associated with the $i$th parameter of the $j$th rectangular discontinuity. It then retrieves the stamps for the $N$ and $K$ matrices associated with this same parameter. The second integral in (3.18) is approximated by the summation

$$\int_0^{T_m} T_s \left( \frac{\Delta R_i}{\Delta p_i} \right) dt \approx \Delta t \sum_k \lambda^T_k \Delta R_i,k,$$ (3.37)

where both the adjoint field and the residue are estimated at the middle of each time interval at odd multiples of 0.5$\Delta t$. Because the electric field is calculated at multiples of $\Delta t$, averaging between the current electric field values and previous electric field values is carried out. Figure 3.11 shows the time instants in both the original and adjoint simulations at which the original residues and the adjoint responses are stored. The command

```matlab
residue_sampled_field(parameter_index).residue_Ez(time_step,:) =((1/dt)*(N_stamp_Ez.*(current_Ez-previous_Ez)))+(0.5*K_stamp_Ez.*(current_Ez+previous_Ez));
```

evaluates the residue expression (3.35). The derivative relative to time is approximated by a central difference approximation at odd multiples of 0.5$\Delta t$. The value of

MATLAB Listing M3.10

```matlab
%MATLAB Listing M3.10
disp(['Starting the time marching loop']);
disp(['Total number of time steps: ' ...
num2str(number_of_time_steps)]);

start_time = cputime;
current_time = 0;

for time_step = 1:number_of_time_steps
    update_incident_fields_2d;
    update_magnetic_fields_2d;
    update_impressed_M;
    update_magnetic_fields_for_CPML_2d;
    capture_sampled_magnetic_fields_2d;
    update_electric_fields_2d;
    update_impressed_J;
    update_electric_fields_for_CPML_2d;
    store_original_fields; %store residues
    capture_sampled_electric_fields_2d;
    update_objective_function; %update objective function
    display_samples_parameters;
end

disp(['Total simulation time is ' ...
num2str(total_time_in_minutes) ' minutes.']);
end_time = cputime;
```
the electric field multiplying the $K$-matrix stamp is evaluated at the middle of the interval by using an averaging of the current and previous $E_z$ values.

Once the original simulation is done, the adjoint simulation is carried out. MATLAB Listing M3.12 shows the script `fdtd_solve_adjoint` which carries out this simulation.

The adjoint simulation, as explained earlier, uses the same system matrices as the original simulation. This means that the same material distribution and boundary conditions are used in both simulations. The only difference is in the

![Figure 3.11 Illustration of the time instants at which the residues and adjoint variables are stored during the original and adjoint simulations](image)
The data structures used to define original sources are eliminated and new adjoint sources are created. These adjoint sources are related to the considered objective function. In this example, the objective function (3.36) can be written as

$$F = \int_0^{T_m} \left( E_{z,1}^2 + E_{z,2}^2 + \cdots + E_{z,N_z}^2 \right) dt,$$

where in (3.38) we sum the square of the $E_z$ field components at all cells at the $x$-directed observation domain. The adjoint source is the derivative of the kernel of the integral (3.38) with respect to the state variables. These state variables contain all the $E_z$ field components in the whole domain in addition to the magnetic field components. The adjoint source utilized is thus given by

$$\left( \frac{\partial \psi}{\partial \boldsymbol{V}} \right) = 2 \begin{bmatrix} 0 \\ 0 \\ \zeta \end{bmatrix}.$$  

As the objective function does not depend on the magnetic fields, the components in the adjoint excitation vector corresponding to the magnetic fields are zeros. The vector $\zeta$ corresponds to the excited electric current densities. It has nonzero value only for the components corresponding to the $E_z$ components used in calculating (3.38). Because the derivative of the square of the field value is double the field value, the nonzero components in the vector $\zeta$ in (3.39) store the electric field values at the observation domain. Also, for the vector of adjoint excitation (3.39), the corresponding impressed adjoint current densities are given by

$$\begin{bmatrix} M_{ix} \\ M_{iy} \\ -J_{iz} \end{bmatrix} = 2 \begin{bmatrix} 0 \\ 0 \\ \zeta \end{bmatrix}.$$  

It follows that only an impressed adjoint electric current density is excited. The script initialize_adjoint_domain, shown in MATLAB Listing M.13, creates the adjoint sources as explained in (3.38)–(3.40). As the impressed currents are added at odd multiples of $0.5\Delta t$, the values of the stored original fields are averaged to create the adjoint sources. The number of created adjoint impressed
electric current sources is equal to the number of original electric probes. The command

\[
\text{impressed} \_\text{J}(\text{ind}).\text{waveform} = -1.0 \times 2.0 \times \text{excitation\_value\_center}(1, \text{end}:-1:1)
\]

stores the complete waveform of the respective impressed adjoint electric current source. Time reversal is used because the adjoint sources are applied in a backward way as explained earlier. The last three commands in this script initialize the updating coefficients required for the adjoint electric current density sources and clears parameters associated with the original sources.

Once the adjoint sources are determined using the stored fields in the original simulation, the next step is to determine the domains in which the adjoint fields are stored. The script that implements this functionality is initialize_
adjoint_storage (shown in MATLAB Listing M3.14). The reader should notice that this script is very similar to the script used to determine the original field storage domains M3.7. The main difference is the shift in the storage domains related to the shape parameters. While the script M3.7 determines the set $S^+_P$, the script M3.14 determines the set $S^+_c$ for shape parameters. The reason for this
shift is that we approximate the adjoint fields in the perturbed adjoint problem within the domains $S^+_p$ by the adjoint fields in the nonperturbed adjoint problem with the domain $S^-_p$. The reader may notice that for the material parameters of each rectangular discontinuity, there is no difference between the two scripts as the two subdomains $S^+_p$ and $S^-_p$ are the same. When the perturbed parameter is $x_{\text{min}}$ of the rectangular discontinuity, the stored adjoint fields and their corresponding original residues are shifted by $-1\Delta x$ in the $x$-direction. When the perturbed parameter is $x_{\text{max}}$ of the rectangular discontinuity, the shift is also $-1\Delta x$ in the $x$-direction. The same applies for the $y$-direction boundaries of all rectangular discontinuities.

The main time-marching loop of the adjoint simulation run_adjoint_fdtd_time_marching_loop_2d, given in MATLAB Listing M3.15, is similar to the one of the original simulation with two key differences. First, the sources in the adjoint problem are electric current density sources, and this is why the function update_impressed_J is invoked with the adjoint current sources initialized by the script M3.13. The second difference is that within the adjoint simulation loop, we carry out the inner product given by (3.37). This inner product is carried out in the script update_adjoint_product that is shown in the Listing M3.16. This code retrieves the adjoint fields in the region $S^-_p$ of the $i$th parameter of the $j$th rectangular discontinuity. As we dot multiply the adjoint fields and the original residues at odd multiples of $0.5\Delta t$, the script averages the adjoint fields over the current and previous time steps. The command

\[
\text{residue}_E_z(\text{number_of_time_steps-time_step+1,:});
\]
recovers the residue stored in the original simulation at the same time instant. Time reversal is applied because the adjoint simulation is running forward with respect to the adjoint time $t$, whereas the original simulation is running forward with respect to the original time variable $t$ where $t = T_m - t$. The summation (3.37) is carried out by adding the contribution available at every time step. The command

\[
\text{sensitivities}(j-1,i) = \text{sensitivities}(j-1,i) - \frac{dt}{dt} \left( \text{residue}_E \times \text{average}_E \text{adjoint}' \right)
\]

multiplies the adjoint fields and the residues at the current time instant and add them to the respective component of the sensitivity storage. This is carried out for all rectangular discontinuities.

Executing MATLAB Listing M3.5 runs the original simulation followed by the adjoint simulation. Figure 3.12 shows two snap shots of the $E_z$ field in both
Figure 3.12 Snapshots of the 2D example: (a) the original simulation and (b) the adjoint simulation
the original and adjoint simulations. It is obvious from Figure 3.12(a) that the excitation for the original problem is a plane wave, whereas Figure 3.12(b) shows that the adjoint excitation is at the observation output line. The output from this script is

```
sensitivities =
1.0e-07*
Columns 1 through 4
0.001444473586861 -0.037682187048431 -0.130245090290955 0.130246328710791
Columns 5 through 6
-0.110305913204953 0.126220289349769
```

To check the accuracy of these sensitivities, we utilize FFD. The original simulation is invoked six extra times where only one parameter is perturbed at a time. The utilized perturbations are \( \Delta p = [0.1 \ 0.001 \text{ S/m} \ 0.001 \text{ mm} \ 0.001 \text{ mm} \ 0.001 \text{ mm} \ 0.001 \text{ mm}]^T \). The estimated FFD sensitivities are

```
1.448592171237444e-10 -3.770392207320875e-09 -1.307812539532651e-08
1.30111113277248e-08 -1.130936203748738e-08 1.296782457551731e-08
```

The two gradients show good match. The FFD requires six extra simulations while the AVM approach requires only one extra adjoint simulation. More accurate sensitivities can be achieved using the more expensive CFD as explained earlier. The reader is encouraged to use the supplied code to estimate the gradient vector using BFDs and central finite differences and compare against the AVM estimates.

To further demonstrate the accuracy of this approach, we compare the AVM sensitivities vs. the FFD sensitivities for a sweep of the relative permittivity from 2.0 to 3.0. The sensitivities of the objective function with respect to all parameters are shown in Figure 3.13. Good agreement is observed for the swept relative permittivity value. The largest relative error is 3%. This error occurs for the \( y_{\text{max}} \) Parameter. The reason for this is the utilized one-to-one approximation. The more nonlinear the change in the field around the discontinuity as a result of the parameter perturbation, the more the loss of accuracy in this approximation. This can be remedied by using a finer grid mesh or a graded mesh with more fine resolution around the discontinuity. The reader can verify that the CFD estimates for this example yields results that are in better agreement with the AVM ones.
Figure 3.13  The sensitivities of the 2D example for a sweep of the relative dielectric constant of the dielectric discontinuity: the AVM sensitivities (○) vs. the FFD sensitivities (−)
3.4 The 3D AVM algorithm

The derivation of an AVM algorithm for the full 3D case is addressed in this section. For every FDTD cell, we have six field components. The good news is that the 3D case has good degree of similarity with the 2D TMz case addressed in the previous section. Many of the concepts presented in the 2D case are still valid. We have though to take into account the existence of more field components per cell and the volumetric nature of the problem. For an isotropic and nondispersive medium, Maxwell’s equations for the 3D case were presented in Chapter 2. Writing these equations in a matrix form at any point in the computational domain with the presence of electric and magnetic current densities and electric and magnetic conductivities, we have

\[
\begin{bmatrix}
\varepsilon \\
\varepsilon \\
\varepsilon \\
-\mu \\
-\mu \\
-\mu
\end{bmatrix} \begin{bmatrix}
\dot{E}_x \\
\dot{E}_y \\
\dot{E}_z \\
\dot{H}_x \\
\dot{H}_y \\
\dot{H}_z
\end{bmatrix} + \begin{bmatrix}
\sigma^e \\
\sigma^e \\
\sigma^e \\
-\sigma^m \\
-\sigma^m \\
-\sigma^m
\end{bmatrix} \begin{bmatrix}
\frac{\partial}{\partial z} & \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \\
-\frac{\partial}{\partial z} & \frac{\partial}{\partial x} & -\frac{\partial}{\partial y} \\
\frac{\partial}{\partial y} & -\frac{\partial}{\partial x} & \frac{\partial}{\partial z}
\end{bmatrix} \begin{bmatrix}
E_x \\
E_y \\
E_z \\
H_x \\
H_y \\
H_z
\end{bmatrix} = \begin{bmatrix}
-\frac{\partial}{\partial t} & 0 & 0 \\
0 & -\frac{\partial}{\partial t} & 0 \\
0 & 0 & -\frac{\partial}{\partial t}
\end{bmatrix} \begin{bmatrix}
\frac{\partial}{\partial z} & \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \\
-\frac{\partial}{\partial z} & \frac{\partial}{\partial x} & -\frac{\partial}{\partial y} \\
\frac{\partial}{\partial y} & -\frac{\partial}{\partial x} & \frac{\partial}{\partial z}
\end{bmatrix} \begin{bmatrix}
J_i \\
M_i \\
M_i
\end{bmatrix},
\]

(3.41)

Utilizing Yee’s discretization and writing this equation for all cells in the computational domain, we have

\[
\begin{bmatrix}
\varepsilon & 0 \\
0 & -\mu
\end{bmatrix} \begin{bmatrix}
\dot{E} \\
\dot{H}
\end{bmatrix} + \begin{bmatrix}
\sigma^e & 0 \\
0 & -\sigma^m
\end{bmatrix} \begin{bmatrix}
E \\
H
\end{bmatrix} + K_c \begin{bmatrix}
E \\
H
\end{bmatrix} = \begin{bmatrix}
-J_i \\
M_i
\end{bmatrix},
\]

(3.42)

where \( E \) and \( H \) are the vectors of electric field components and magnetic field components for the whole domain, respectively. \( \varepsilon, \mu, \sigma^e, \) and \( \sigma^m \) are diagonal matrices representing the material properties over the whole domain. \( K_c \) is a symmetric matrix of parameter-independent coefficients that represents the finite difference approximation of the curl operators. The vectors \( J_i \) and \( M_i \) represent the impressed electric and magnetic current densities over the whole domain.
The system (3.42) can be written in the form (3.9) with the system matrices and excitation:

\[ \begin{bmatrix} \varepsilon & 0 \\ 0 & -\mu \end{bmatrix} \hat{\mathbf{E}} + \begin{bmatrix} \sigma^e & 0 \\ 0 & -\sigma^m \end{bmatrix} \hat{\mathbf{H}} + \begin{bmatrix} \mathbf{J} \\ \mathbf{M} \end{bmatrix} \]

(3.43)

Here, the system parameters are \( V = [\mathbf{E}^T \quad \mathbf{H}^T]^T \). The system (3.42) is denoted as the original system. It is solved using Yee’s time marching scheme explained in Chapter 2. For an objective function of the form (3.1), and following similar derivation steps similar to the 1D and 2D TMz cases, the corresponding adjoint system is given by

\[ -\mathbf{N}^T \dot{\lambda} + \mathbf{K}^T \lambda = \left( \frac{\partial \psi}{\partial V} \right). \]

(3.44)

Also, by changing the time variable from \( t \) to \( \tau \), the adjoint variable time parameter, the negative sign of the first temporal derivative is eliminated. Taking into account that the matrices \( \mathbf{N} \) and \( \mathbf{K} \) are symmetric, we end up with the adjoint system

\[ \mathbf{N} \dot{\lambda} + \mathbf{K} \lambda = \left( \frac{\partial \psi}{\partial V} \right). \]

(3.45)

The system (3.45) is the adjoint system. It has the same \( \mathbf{N} \) and \( \mathbf{K} \) matrices as the original simulation and thus the same material and boundary conditions. The two simulations differ only in their excitation. The same solver used in solving the 3D original simulation (3.42) can be also used to solve the adjoint system (3.45). This adjoint system can also be put in the more familiar form

\[ \begin{bmatrix} \varepsilon & 0 \\ 0 & -\mu \end{bmatrix} \begin{bmatrix} \hat{\mathbf{E}}^* \\ \hat{\mathbf{H}}^* \end{bmatrix} + \begin{bmatrix} \sigma^e & 0 \\ 0 & -\sigma^m \end{bmatrix} \begin{bmatrix} \lambda^* \\ \lambda^* \end{bmatrix} + \begin{bmatrix} \mathbf{J} \\ \mathbf{M} \end{bmatrix} = \left( \frac{\partial \psi}{\partial V} \right), \]

(3.46)

where \( \lambda^* \) and \( \lambda^* \) are the vectors of adjoint electric and magnetic fields. The derivatives in (3.46) are with respect to the adjoint time variable \( \tau \). The adjoint excitation in the right-hand side of (3.46) is determined during the original simulation and then applied backward in time (forward with respect to \( \tau \)).

As was explained for the 1D and 2D cases, the adjoint system (3.45) is parameter independent and is thus an approximate one. The exact adjoint systems are given by (3.17) and would require a total of \( n \) extra adjoint simulations. Using the parameter-independent adjoint system (3.45) requires approximating the adjoint fields obtained from the exact adjoint systems (3.17) by the fields obtained from the approximate system (3.45). The same one-to-one mapping applied for the 2D TMz is also used for the 3D case. The main difference is that the mapped fields are within a volume. Figure 3.14 illustrates this mapping for the 3D case. Consider a lossy-dielectric discontinuity in the parameter space whose length in the \( z \) direction is \( L \) as shown in Figure 3.14(a). The exact adjoint system required for the parameter \( L \) is shown in Figure 3.14(b). In this adjoint system, the parameter \( L \) is perturbed by the smallest possible on-grid perturbation \( \Delta z \). It is required in this adjoint system to
store the adjoint responses in the volume $S_L^+$. However, we do not actually carry out the adjoint simulation shown in Figure 3.14(b). Instead, we carry out the parameter-independent adjoint simulation shown in Figure 3.14(c). The adjoint fields in the volume $S_L^+$ in Figure 3.14(b) are approximated by the fields in the perturbation volume $S_L^-$ in Figure 3.14(c). The same concept applies to all geometrical parameters. For material parameters, the approach is similar to the 2D case. The perturbation in the material parameters can be made infinitesimally small. This makes the adjoint system (3.46) an exact one and thus no approximation is required.

Similar to the 1D and 2D cases, the residues are calculated in the original simulation at the cells affected by the perturbation in the parameters. The residue
related to the \( i \)th parameter is given by (3.31). We analyze this term further for the 3D case. Considering that the matrix \( \mathbf{K}_c \) is parameter independent, the residue (3.31) is expanded as

\[
\Delta R_i = \begin{bmatrix} \Delta \varepsilon_i & 0 \\ 0 & -\Delta \mu_i \end{bmatrix} \begin{bmatrix} \dot{E} \\ \dot{H} \end{bmatrix} + \begin{bmatrix} \Delta \sigma^e_i & 0 \\ 0 & -\Delta \sigma^m_i \end{bmatrix} \begin{bmatrix} E \\ H \end{bmatrix},
\]

where \( \Delta \mu_i, \Delta \varepsilon_i, \Delta \sigma^e_i, \) and \( \Delta \sigma^m_i \) are the changes in the material properties corresponding to a perturbation \( \Delta \rho_i \) in the \( i \)th parameter. As all the perturbation matrices in (3.47) are diagonal, this system implies that every field component in the original simulation and its time derivative are multiplied by a weight or a “stamp” that represents the change in the \( i \)th parameter.

Because of the way the fields are organized in Yee’s cell, different field components have different stamps depending on their position within the cell and within the perturbed discontinuity. This is illustrated in Figure 3.15 where a lossy-dielectric discontinuity is located in the computational domain. We assume that the considered parameter is the parameter \( z_{\text{max}} \) of the discontinuity. This parameter is perturbed by the smallest on-grid perturbation to determine the corresponding residue. The electric field components \( E_x, E_y, \) and \( E_z \) are multiplied by the respective permittivity and conductivity values. First, we consider the tangential field component \( E_x \) at the surface \( z = z_{\text{max}} \) (see Figure 3.15(a)). At the two surfaces, \( y = y_{\text{min}} \) and \( y = y_{\text{max}} \), the effective permittivity for the component \( E_x \) is \( \varepsilon = 0.25\varepsilon_0\varepsilon_r + 0.75\varepsilon_0 \). For all other \( E_x \) components on this surface of the discontinuity,
the field component $E_y$ has the effective permittivity $\varepsilon = 0.5\varepsilon_0\varepsilon_r + 0.5\varepsilon_0$. A similar permittivity interpolation scheme applies to the tangential $E_y$ component at the surface $z = z_{\text{max}}$. At the two surfaces, $x = x_{\text{min}}$ and $x = x_{\text{max}}$, the effective permittivity for the component $E_y$ is $\varepsilon = 0.25\varepsilon_0\varepsilon_r + 0.75\varepsilon_0$. For all other $E_y$ components on the surface $z = z_{\text{max}}$ of the discontinuity, we have the effective permittivity $\varepsilon = 0.5\varepsilon_0\varepsilon_r + 0.5\varepsilon_0$.

Now, we assume that a perturbation of $1\Delta z$ is applied to the parameter $z = z_{\text{max}}$ (see Figure 3.15(b)). The tangential field component $E_x$ at the intersection of the surface $z = z_{\text{max}}$ and the two surfaces $y = y_{\text{min}}$ and $y = y_{\text{max}}$ will have the perturbed effective permittivity $\varepsilon = 0.5\varepsilon_0\varepsilon_r + 0.5\varepsilon_0$. The perturbation in the dielectric constant for these edge field components is thus $\Delta\varepsilon = 0.5\varepsilon_0\varepsilon_r + 0.5\varepsilon_0 - 0.25\varepsilon_0\varepsilon_r - 0.75\varepsilon_0 = 0.25\varepsilon_0\varepsilon_r - 0.25\varepsilon_0 = 0.25\varepsilon_0(\varepsilon_r - 1)$. The same effective dielectric change applies to the $E_y$ components at the intersection of the surface $z = z_{\text{max}}$ with the two surfaces $x = x_{\text{min}}$ and $x = x_{\text{max}}$. All other tangential $E_x$ and $E_y$ components on the surface $z = z_{\text{max}}$ of the discontinuity have a perturbed permittivity of $\varepsilon = \varepsilon_0\varepsilon_r$. The change in the permittivity for these field components is thus $\Delta\varepsilon = \varepsilon_0\varepsilon_r - 0.5\varepsilon_0\varepsilon_r - 0.5\varepsilon_0 = 0.5\varepsilon_0(\varepsilon_r - 1)$.

The assumed perturbation in the parameter $z_{\text{max}}$ does not affect only the tangential components to the surface $z = z_{\text{max}}$ of the discontinuity. It affects also the field components tangential to the plane $z = z_{\text{max}} + \Delta z$ bounded by the discontinuity lateral surfaces (see Figure 3.15(b)). Before perturbing the discontinuity length, all these components were surrounded by air with an effective permittivity of $\varepsilon = \varepsilon_0$. The tangential field component $E_x$ at the intersection of the surface $z = z_{\text{max}} + \Delta z$ and the two surfaces $y = y_{\text{min}}$ and $y = y_{\text{max}}$ will have the perturbed effective permittivity $\varepsilon = 0.25\varepsilon_0\varepsilon_r + 0.75\varepsilon_0$. The same perturbation applies to the tangential field component $E_y$ at the intersection of the surface $z = z_{\text{max}} + \Delta z$ and the two surfaces, $x = x_{\text{min}}$ and $x = x_{\text{max}}$. The perturbation in the dielectric constant for all tangential edge components at $z = z_{\text{max}} + \Delta z$ is thus $\Delta\varepsilon = 0.25\varepsilon_0\varepsilon_r + 0.75\varepsilon_0 - \varepsilon_0 = 0.25\varepsilon_0\varepsilon_r - 0.25\varepsilon_0 = 0.25\varepsilon_0(\varepsilon_r - 1)$. All other tangential components at the surface $z = z_{\text{max}} + \Delta z$ of the perturbed discontinuity will undergo a change of $\Delta\varepsilon = 0.5\varepsilon_0\varepsilon_r + 0.5\varepsilon_0 - \varepsilon_0 = 0.5\varepsilon_0\varepsilon_r - 0.5\varepsilon_0 = 0.5\varepsilon_0(\varepsilon_r - 1)$.

In addition to affecting the tangential field components at the planes $z = z_{\text{max}}$ and $z = z_{\text{max}} + \Delta z$, the normal $E_z$ components between the two planes will also see a change in their effective permittivity (see Figure 3.15(b)). In the nonperturbed structure, all these components had an effective permittivity of $\varepsilon = \varepsilon_0$. The perturbation changes the effective permittivity of all $E_z$ components at the planes $x = x_{\text{min}}$ and $x = x_{\text{max}}$ and $y = y_{\text{min}}$ and $y = y_{\text{max}}$ (except for the corner ones) to $\varepsilon = 0.5\varepsilon_0\varepsilon_r + 0.5\varepsilon_0$. The corner ones have the perturbed permittivity $\varepsilon = 0.25\varepsilon_0\varepsilon_r + 0.75\varepsilon_0$ while all other interior $E_z$ components have the perturbed permittivity $\varepsilon = \varepsilon_0\varepsilon_r$. The perturbations in the permittivity for the surface $E_z$ components, corner $E_z$ components, and interior $E_z$ components are $0.5\varepsilon_0(\varepsilon_r - 1)$, $0.25\varepsilon_0(\varepsilon_r - 1)$, and $\varepsilon_0(\varepsilon_r - 1)$, respectively. The values of these perturbations are used for determining the residual vectors. For the parameter discontinuity shown in Figure 3.15 with $4 \times 4$ cells in the cross section, the $E_x$ stamp will have 40 components, the $E_y$ stamp will have 40 components, and the $E_z$ stamp will have 25 components.
In order to evaluate the inner product in (3.18) for each parameter, the residue (3.32) is divided by the perturbation in the $i$th parameter $\Delta p_i$. If $p_i$ is a material-related parameter such as the dielectric constant, and similar to the 2D case, we have

$$\frac{\Delta R}{\Delta \varepsilon} \approx \frac{dR}{d\varepsilon} = \left[ \frac{\partial \varepsilon}{\partial \varepsilon} 0 \right] \dot{E} + \left[ 0 0 \right] \dot{H},$$

(3.48)

where the diagonal matrix $\frac{\partial \varepsilon}{\partial \varepsilon}$ contains components belonging to the set $\{0, \varepsilon_0, 0.5\varepsilon_0, \text{and } 0.25\varepsilon_0\}$, as explained earlier, depending on the node and polarization of the associated field component. Applying a similar approach to the electric conductivity, we have

$$\frac{\Delta R}{\Delta \sigma} \approx \frac{dR}{d\sigma} = \left[ 0 0 \right] \dot{E} + \left[ \Gamma 0 \right] \dot{H},$$

(3.49)

where the diagonal matrix $\Gamma$ contains components belonging to the set $\{0, 1, 0.5, 0.25\}$ depending on the node and the orientation of the associated field component.

For shape parameters, as explained earlier for the 2D TMz case, the smallest possible perturbation is limited in a numerical solver by the utilized spatial discretization. For example, for one of the shape parameters associated with a nonmagnetic discontinuity, we have

$$\frac{\Delta R_i}{\Delta p_i} \approx \frac{dR_i}{dp_i} = \left[ \frac{\Delta \varepsilon_i}{\Delta p_i} 0 \right] \dot{E} + \left[ \Delta \sigma_i/\Delta p_i 0 \right] \dot{H}.$$  

(3.50)

In this case, both the $N$ and $K$ matrices may have nonzero stamps. The stamps associated with all material and shape parameters can be determined using (3.48)–(3.50).

The main steps of the AVM algorithm for the 3D FDTD case are the same as the 2D TMz case. The only difference is that there are more field components and more stamps to store. We illustrate this approach through an example.

**Example 3.3** Consider the structure shown in Figure 3.16. A dielectric block discontinuity is placed in the shown computational domain. It extends between corners with the coordinates $(0.151, 0.151, 0.13)$ m to $(0.16, 0.16, 0.16)$ m. The domain is excited with a line electric current density oriented parallel to the $z$-direction extending from $(0.112, 0.112, 0.112)$ m to $(0.112, 0.112, 0.115)$ m. The temporal dependence of this source is a Gaussian-modulated cosine with a center frequency of 2.0 GHz and a bandwidth of 1.0 GHz (see Figure 3.17). The energy received at another node of the domain located at $(0.172, 0.172, 0.172)$ m is the considered objective function. The target is to determine the derivative of the received energy relative to all the parameters of the dielectric discontinuity $p = [\varepsilon, \sigma, x_{\min}, x_{\max}, y_{\min}, y_{\max}, z_{\min}, z_{\max}]^T$ where $\varepsilon = 1.5$ and $\sigma = 0.01$ S/m.
Figure 3.16 The structure of the 3D problem; a dielectric discontinuity is excited by an electric current filament with the domain terminated by an air buffer and CPML layer.

Figure 3.17 The temporal variation of the excitation current filament.
A full 3D code that addresses this problem is utilized. Similar to most AVM codes throughout this book, the MATLAB script `TestAVM` invokes the original simulation followed by the adjoint simulation. The residues and the adjoint excitation are determined during the original simulation. The adjoint simulation is then invoked using the sources determined during the original simulation. As there are significant similarities between the 2D AVM code explained earlier and the 3D code discussed here, we will thus focus on the main differences between them. The script `initialize_residue_stamps` determines the stamps for all field components. Because the discontinuity is nonmagnetic and the objective function does not depend on the magnetic field, we need only to store the stamps for the three electric field components at the perturbation regions. As explained earlier, the field component may be a tangential component or a normal component depending on the direction of the perturbation of the shape parameter. We showed in Figure 3.14 the case where a discontinuity is perturbed in the z-direction. We noticed that because the perturbation is in the z-direction, there are two layers of affected tangential field components ($E_x$ and $E_y$), whereas there is only one layer of affected normal field components:

```
% MATLAB listing M3.17

% parameters associated with every block starting from the second one (the
% main air block is assumed invariant)
% notice that we do not store the polarization as we assume that for a
% general 3D problem all 6 field components should be stored for the
% affected node
for j=1:numberOfBricks
  for i=1:8 % repeat for all parameters related to the current brick
    parameter_index=(j-2)*8+i; % start index of parameters
    % get start of the block associated with the current block
    is=reserve_sampled_field(parameter_index).min_x;
    js=reserve_sampled_field(parameter_index).min_y;
    ks=reserve_sampled_field(parameter_index).min_z;
    ts=reserve_sampled_field(parameter_index).max_x;
    te=reserve_sampled_field(parameter_index).max_y;
    ke=reserve_sampled_field(parameter_index).max_z;
    material_epsilon=material_types(bricks(j).material_type).epsilon_REAL;
    material_permeability=material_types(bricks(j).material_type).mu_REAL;
    % each field component must be handled differently
    % so we now loop over all E field components
    % first consider Ex component
    build_Ex_stamp; % build stamp for Ex component
    N_x, N_y, N_z are the number of stored components for each component
    N_s=reserve_sampled_field(parameter_index).N_stamp_Ex;
    N_s=reserve_sampled_field(parameter_index).N_stamp_Ex;
    K_s=reserve_sampled_field(parameter_index).K_stamp_Ex;
    K_s=reserve_sampled_field(parameter_index).K_stamp_Ex;
    % last consider Ez component
    build_Ez_stamp; % build stamp for Ez component
    N_s=reserve_sampled_field(parameter_index).N_stamp_Ez;
    N_s=reserve_sampled_field(parameter_index).N_stamp_Ez;
    K_s=reserve_sampled_field(parameter_index).K_stamp_Ez;
    K_s=reserve_sampled_field(parameter_index).K_stamp_Ez;
  end
```

A full 3D code that addresses this problem is utilized. Similar to most AVM codes throughout this book, the MATLAB script `TestAVM` invokes the original simulation followed by the adjoint simulation. The residues and the adjoint excitation are determined during the original simulation. The adjoint simulation is then invoked using the sources determined during the original simulation. As there are significant similarities between the 2D AVM code explained earlier and the 3D code discussed here, we will thus focus on the main differences between them. The script `initialize_residue_stamps` determines the stamps for all field components. Because the discontinuity is nonmagnetic and the objective function does not depend on the magnetic field, we need only to store the stamps for the three electric field components at the perturbation regions. As explained earlier, the field component may be a tangential component or a normal component depending on the direction of the perturbation of the shape parameter. We showed in Figure 3.14 the case where a discontinuity is perturbed in the z-direction. We noticed that because the perturbation is in the z-direction, there are two layers of affected tangential field components ($E_x$ and $E_y$), whereas there is only one layer of affected normal field components.
The script M3.17 creates the stamps for all these components. It starts by retrieving the indices of the corners of the perturbation region affected by the parameter perturbation. For every discontinuity brick, we have eight parameters, two material parameters, and six shape parameters.

The scripts `build_Ex_stamp`, `build_Ey_stamp`, and `build_Ez_stamp` create the weights (stamps) associated with the respective field component for each parameter. The script `build_Ex_stamp` is shown in MATLAB Listing M3.18.
This script starts by determining the number of \( E_x \) components that will be stored in each direction \( N_x, N_y, \) and \( N_z \). If the considered parameter is a material parameter \((i = 1 \text{ or } 2)\), the stamp size is equal to the number of \( E_x \) components in the discontinuity. The stamp component belong to the set \( \{0, \varepsilon_0, 0.5\varepsilon_0, 0.25\varepsilon_0\} \) as explained in (3.48) for the permittivity and the set \( \{0, 1, 0.5, 0.25\} \) for the electric conductivity. The assumed value depends on the location of the component and whether it is outside the discontinuity, an interior component, a surface component, or a corner component.

For the shape parameters, the stamp of the \( E_x \) component depends on whether it is normal or tangential to the perturbed surface. If the considered parameters are \( x_{\text{min}} \) or \( x_{\text{max}} \) (the parameter \( i \) assumes a value of 1 or 2), then the function `get_normal_stamp()` is invoked. This function returns the stamp for only one layer of \( E_x \) components. For the other four parameters \((y_{\text{min}}, y_{\text{max}}, z_{\text{min}}, \text{or } z_{\text{max}})\), the \( E_x \) component is a tangential component. Two parallel layers of \( E_x \) components are affected by the minimum on-grid perturbation for these parameters. The values for the stamps are determined by whether the component is an edge component or an interior surface component. The function `get_tangential_stamp()` returns the stamp values in these cases. This function receives as input the number of components in each direction, the relative perturbation in the material parameters, the direction of the perturbation, and the direction of the considered field component. Both functions return the stamp for both the \( N \) and \( K \) matrices. An initial background value is given to all components, which is the value assumed by the interior components. The parameter `perturbation` is the perturbation in the respective material property caused by the parameter perturbation. For example, if the parameter \( z_{\text{min}} \) is increased by \( 1\Delta z \), the parameter `perturbation` assumes a value of \( \varepsilon_0 - \varepsilon_0\varepsilon_r \) for the permittivity and a value of \( -\sigma_r \) for the conductivity. If the parameter \( z_{\text{max}} \) is increased by \( 1\Delta z \), the parameter `perturbation` assumes

MATLAB Listing M3.19

```matlab
% MATLAB Listing M3.19
function stamp = get_normal_stamp(N_y, N_z, perturbation)
% initially, we give all components the same interior perturbation
stamp = perturbation * ones(N_y, N_z); % one layer of Ex components
% surface components other than corners one change by half perturbation
if(N_y>3)
    for m=2:(N_y-1)
        stamp(m,1) = 0.5*perturbation;
        stamp(m,N_z) = 0.5*perturbation;
    end
end
if(N_z>=3)
    for n=2:(N_z-1)
        stamp(1,n) = 0.5*perturbation;
        stamp(N_y,n) = 0.5*perturbation;
    end
end
% corner components change by quarter perturbation
stamp(1,1) = 0.25*perturbation;
stamp(N_y,1) = 0.25*perturbation;
stamp(N_y,N_z) = 0.25*perturbation;
stamp(1,N_z) = 0.25*perturbation;
end
```
a value of $\varepsilon_0\varepsilon_r - \varepsilon_0$ for the permittivity and a value of $\sigma^e$ for the conductivity. The same applies for all the other shape parameters.

Once the residue stamps have been determined, the 3D AVM code proceeds in a way similar to the 2D TMz case. The residues are determined at odd multiples of $0.5\Delta t$ using the residue stamps and original fields. Also, the adjoint sources are determined during the main time marching loop of the original problem. The algorithm then proceeds to run the adjoint simulation. The script `initialize_adjoint_storage` determined the volumes for which the adjoint fields are stored in the adjoint simulation. It is given in MATLAB Listing M3.21.

This script is similar to the 2D case. There are eight parameters associated with every dielectric brick. The adjoint storage for the first two parameters (the permittivity and conductivity) covers the whole volume of the discontinuity. For all shape parameters, the perturbed volume is just one layer of cells that are either being converted from air to dielectric for the parameters $x_{\text{max}}$, $y_{\text{max}}$, and $z_{\text{max}}$ or is perturbed from dielectric to air for the parameters $x_{\text{min}}$, $y_{\text{min}}$, and $z_{\text{min}}$. Comparing the storages in this script to the similar script for the original problem, one notices a shift in the stored volume by $1\Delta x$ for the parameters $x_{\text{min}}$ and $x_{\text{max}}$, a shift in the stored volume by $1\Delta y$ for the parameters $y_{\text{min}}$ and $y_{\text{max}}$, and a shift in the stored volume by $1\Delta z$ for the parameters $z_{\text{min}}$ and $z_{\text{max}}$. The reason for this shift is the one-to-one mapping implemented between the exact adjoint simulation and the parameter-independent adjoint simulation.
MATLAB Listing M3.21

for k=2:numeroBricks
    parameter_start_index=(k-2)*8+1; % start index of parameters
    % first parameter is relative permittivity. Storage covers all brick
    adjoint_sampled_field(parameter_start_index).min_x = n1.is;
    adjoint_sampled_field(parameter_start_index).min_y = n1.js;
    adjoint_sampled_field(parameter_start_index).min_z = n1.ks;
    adjoint_sampled_field(parameter_start_index).max_x = (n1.ie)-1;
    adjoint_sampled_field(parameter_start_index).max_y = (n1.je)-1;
    adjoint_sampled_field(parameter_start_index).max_z = (n1.ke)-1;
    % second parameter is conductivity. Storage covers all brick
    adjoint_sampled_field(parameter_start_index+1).min_x = n1.is;
    adjoint_sampled_field(parameter_start_index+1).min_y = n1.js;
    adjoint_sampled_field(parameter_start_index+1).min_z = n1.ks;
    adjoint_sampled_field(parameter_start_index+1).max_x = (n1.ie)-1;
    adjoint_sampled_field(parameter_start_index+1).max_y = (n1.je)-1;
    adjoint_sampled_field(parameter_start_index+1).max_z = (n1.ke)-1;
    % third parameter is min_x of the brick
    adjoint_sampled_field(parameter_start_index+2).min_x = (n1.ie)-1; % shift
    adjoint_sampled_field(parameter_start_index+2).min_y = n1.js;
    adjoint_sampled_field(parameter_start_index+2).min_z = n1.ks;
    adjoint_sampled_field(parameter_start_index+2).max_x = (n1.ie)-1;
    adjoint_sampled_field(parameter_start_index+2).max_y = (n1.je)-1;
    adjoint_sampled_field(parameter_start_index+2).max_z = (n1.ke)-1;
    % fourth parameter is max_x of the brick
    adjoint_sampled_field(parameter_start_index+3).min_x = (n1.ie)-1; % shift
    adjoint_sampled_field(parameter_start_index+3).min_y = n1.js;
    adjoint_sampled_field(parameter_start_index+3).min_z = n1.ks;
    adjoint_sampled_field(parameter_start_index+3).max_x = (n1.ie)-1;
    adjoint_sampled_field(parameter_start_index+3).max_y = (n1.je)-1;
    adjoint_sampled_field(parameter_start_index+3).max_z = (n1.ke)-1;
    % fifth parameter is min_y of the associated brick.
    adjoint_sampled_field(parameter_start_index+4).min_x = n1.is;
    adjoint_sampled_field(parameter_start_index+4).min_y = (n1.js)-1; % shift
    adjoint_sampled_field(parameter_start_index+4).min_z = n1.ks;
    adjoint_sampled_field(parameter_start_index+4).max_x = (n1.is)-1;
    adjoint_sampled_field(parameter_start_index+4).max_y = (n1.js)-1;
    adjoint_sampled_field(parameter_start_index+4).max_z = (n1.ks)-1;
    % sixth parameter is max_y of the associated brick. Storage covers only
    % two layers of cells
    adjoint_sampled_field(parameter_start_index+5).min_x = n1.is;
    adjoint_sampled_field(parameter_start_index+5).min_y = (n1.js)-1;
    adjoint_sampled_field(parameter_start_index+5).min_z = n1.ks;
    adjoint_sampled_field(parameter_start_index+5).max_x = (n1.is)-1;
    adjoint_sampled_field(parameter_start_index+5).max_y = (n1.js)-1;
    adjoint_sampled_field(parameter_start_index+5).max_z = (n1.ks)-1;
    adjoint_sampled_field(parameter_start_index+6).min_x = n1.is;
    adjoint_sampled_field(parameter_start_index+6).min_y = n1.js;
    adjoint_sampled_field(parameter_start_index+6).min_z = (n1.ks)-1;
    adjoint_sampled_field(parameter_start_index+6).max_x = (n1.is)-1;
    adjoint_sampled_field(parameter_start_index+6).max_y = (n1.js)-1;
    adjoint_sampled_field(parameter_start_index+6).max_z = (n1.ks)-1;
    adjoint_sampled_field(parameter_start_index+7).min_x = n1.is;
    adjoint_sampled_field(parameter_start_index+7).min_y = n1.js;
    adjoint_sampled_field(parameter_start_index+7).min_z = (n1.ks)-1;
    adjoint_sampled_field(parameter_start_index+7).max_x = (n1.is)-1;
    adjoint_sampled_field(parameter_start_index+7).max_y = (n1.js)-1;
    adjoint_sampled_field(parameter_start_index+7).max_z = (n1.ks)-1;
end
The rest of the 3D AVM implementation is very similar to the 2D TMz case. Running the adjoint sensitivity code, we get the following sensitivities:

\[
\text{sensitivities} = 
1.0e-16 *
\]

\[
\begin{align*}
\text{Columns 1 through 4} & \\
0.002795628091345 & -0.002974295274035 \\
-0.090270118162698 & 0.145967884800818 \\
\end{align*}
\]

\[
\begin{align*}
\text{Columns 5 through 8} & \\
-0.090270118162698 & 0.145967884800818 \\
-0.082786892971836 & -0.029235000485191 \\
\end{align*}
\]

We then compare these adjoint sensitivities against forward finite differences (FFD). Although the adjoint sensitivities require only one extra simulation, the FFD requires eight extra simulations. We utilize the parameter perturbations \( \Delta p = [0.05 \ 0.001 \ 0.003 \ 0.003 \ 0.003 \ 0.003 \ 0.003 \ 0.003]^T \). The FFD sensitivities achieved are as follows:

\[
\text{FFD} = 
1.0e-16 *
\]

\[
\begin{align*}
\text{Columns 1 through 4} & \\
0.002826614467590 & -0.003009756962765 \\
-0.103164167198110 & 0.174510423563151 \\
\end{align*}
\]

\[
\begin{align*}
\text{Columns 5 through 8} & \\
-0.103164167198110 & 0.174510423563213 \\
-0.088624733353228 & -0.041282200555188 \\
\end{align*}
\]

Comparing the FFD sensitivities to the adjoint ones, we note that some components are better matched than others. Our experience shows that this indicates a highly nonlinear objective function where the FFD, BFD, and CFD show significant differences. The BFD sensitivities are estimated using the same perturbations and are given by

\[
\text{BFD} = 
1.0e-16 *
\]

\[
\begin{align*}
\text{Columns 1 through 4} & \\
0.002766753747236 & -0.002914471891599 \\
-0.090573610318721 & 0.142061649000000 \\
\end{align*}
\]
Comparing all three results, we note that the BFD and FFD estimates also differ by as much as 20%. This is a sign of a highly nonlinear function. We note also that adjoint sensitivity estimates are in between the BFD and FFD estimates. This is also a characteristic that has been reported about adjoint sensitivities. Utilizing a finer mesh will reduce the difference between estimates at the expense of an increased computational time. In this example, the AVM required only one extra simulation, whereas the FFD required eight extra simulations.

References


Sensitivity analysis for frequency-dependent objective functions

In Chapter 3, we discussed the adjoint-based sensitivity approach for estimating the response sensitivities of simple energy functions [1–3] through time-domain electromagnetic simulations. The resultant adjoint sensitivities are real-valued. In this chapter, we address the practical case of frequency-dependent objective functions such as the $S$-parameters, the $Z$-parameters, and the $Y$-parameters. These parameters are usually estimated over a band of frequencies. The time-domain solver returns a vector of complex numbers representing the values of these responses at different frequencies. The AVM approach estimates the sensitivities of these responses over the desired frequency band using at most one extra simulation.

The wideband adjoint treatment of the scattering-parameters was initially proposed in [4,5] for the transmission-line matrix [6] method. In the subsequent work reported in [7–9], this adjoint treatment was adapted to other grid-based time-domain solvers, such as the finite-difference time-domain method [10]. Moreover, it was also noted in [5] that if all media in the simulation are reciprocal (which is the case for the majority of practical problems), then the adjoint simulation is redundant and thus not required. This implies that the adjoint sensitivities of the complex frequency-domain responses relative to all parameters are estimated as a by-product of the original simulation with no extra simulations required.

4.1 The monochromatic case

We first handle the monochromatic case, where there is only one excited frequency. Our target is to evaluate the spectrum of the fields at the observation domain. This spectrum is given by

$$
\tilde{F}(\omega_o) = \int_0^{T_w} F(V(t))e^{-j\omega_o t} dt = \int_0^{T_w} \psi(V)dt,
$$

where $j = \sqrt{-1}$. The spectrum is evaluated by multiplying the time domain observation function $F(V)$ by $\exp(-j\omega_o t)$ and integrating over time. In a time
domain solver, where the values of the fields are only evaluated at discrete instants of time, the integral (4.1) may be approximated by the discrete summation:

\[
\tilde{F}(\omega_0) \approx \Delta t \sum_k F(V(k\Delta t))e^{-j\omega_0 k\Delta t}.
\] (4.2)

The summation (4.2) is a scaled expression of the commonly used discrete Fourier transform. The vector of state variables \( V \) is a function of the parameters (dimensions and material properties of different discontinuities). The sensitivities of (4.2) with respect to all parameters can be evaluated using finite differences by perturbing the parameters one at a time. The vector \( V \) is evaluated at every time step using the time-domain system (3.8) for the 1D case, using the system (3.27) for the 2D TMz case, or using the system (3.42) for the 3D case.

To evaluate the adjoint sensitivities of (4.1) with respect to the vector of parameters \( p \), we first split the complex integral (4.1) into a real part and an imaginary part. These two parts are given by

\[
\text{Re}\left(\tilde{F}(\omega_0)\right) = \int_0^{T_m} F(V(t))\cos(\omega_0 t)dt, \tag{4.3}
\]
\[
\text{Im}\left(\tilde{F}(\omega_0)\right) = -\int_0^{T_m} F(V(t))\sin(\omega_0 t)dt. \tag{4.4}
\]

Each one of these two functions is real-valued and thus the approach explained in Chapter 3 for estimating the adjoint sensitivities of real-valued objective functions is applicable. We first consider the real part of the spectrum given by (4.3). Using the expression (3.19), the adjoint excitation is the derivative of the kernel of the integral (4.3) with respect to the system variables \( V \)

\[
Q_r = \frac{\partial F(V(t))}{\partial V} \cos(\omega_0 t). \tag{4.5}
\]

Here, the subscript \( r \) refers to the adjoint excitation corresponding to the real part of the spectrum (4.3). It follows from (4.5) that the adjoint excitation’s temporal profile is a cosine function multiplied at every time step by the derivative of the function \( F \) with respect to the system variables \( V \). In many cases, such as in the calculation of the \( S \)-parameters, the function \( F \) is a linear function of the electric and magnetic fields. It follows that the vector \( \partial F(V(t))/\partial V \) is a constant vector that does not depend on time with nonzero components only in the observation domain. The adjoint excitation (4.5) can thus be written as

\[
Q_r = a \cos(\omega_0 t). \tag{4.6}
\]

Here \( a \) is a constant time-independent vector. Because the adjoint excitation (4.6) is independent of the original response, the adjoint simulation with the excitation (4.6) can be run in parallel with the original simulation.
As explained in Chapter 3, the adjoint excitation (4.6) has to be reversed in time before being applied to the adjoint simulation. The reversed adjoint excitation is given by

$$Q_r(\tau) = a \cos(\omega_o(T_m - \tau)). \quad (4.7)$$

The adjoint excitation (4.7) corresponding to the real part of the spectrum results in an adjoint response $\lambda_r(\tau)$ throughout the computational domain.

For the imaginary part of the spectrum given by (4.4), and using similar steps to (4.5)–(4.7), we obtain the following adjoint excitation corresponding to the imaginary part of the spectrum

$$Q_i(\tau) = -a \sin(\omega_o(T_m - \tau)). \quad (4.8)$$

This adjoint excitation results in the adjoint response $\lambda_i(\tau)$.

It appears from the previous discussion that two separate adjoint simulations are needed with the excitations (4.7) and (4.8). This is not the case. The phase shift between the adjoint excitation of the real part and the adjoint excitation of the imaginary part is $\pi/2$. This implies that the adjoint response $\lambda_i(\tau)$ is a shifted replica of $\lambda_r(\tau)$ by the same shift for a linear time-invariant simulation. This shift in angle is translated into a shift in the sampled field values using the formula [4]:

$$\omega_o m \Delta t = \frac{\pi}{2} \Rightarrow 2\pi f_o m \Delta t = \frac{\pi}{2} \Rightarrow m = \text{round}\left(\frac{1}{4f_o \Delta t}\right). \quad (4.9)$$

where $m$ is the shift in FDTD samples between the real and imaginary responses. It follows that by using only one adjoint simulation, the adjoint responses corresponding to both the real and imaginary parts of the spectrum are determined. The adjoint sensitivities of the real and imaginary parts with respect to all parameters are thus estimated using only one adjoint simulation. Figure 4.1 illustrates the phase shift between the two adjoint responses.

One may notice that the adjoint excitation (4.7) has a nonzero start value at $\tau = 0$ with $Q_r(0) = a \cos(\omega_o T_m)$. This sharp start at the time $\tau = 0$ may excite high-order frequencies and is undesirable in a time-domain solver. To avoid this scenario, we ramp up the adjoint excitation by multiplying it by a Gaussian function to obtain

$$Q_r(\tau) = \begin{cases} a \cos(\omega_o(T_m - \tau)) & \tau \leq \tau_o \\ a \cos(\omega_o(T_m - \tau))e^{-(\tau-\tau_o)^2} & \tau \geq \tau_o. \end{cases} \quad (4.10)$$

This ramping of the adjoint excitation introduces very little error in the adjoint sensitivity calculations because it was assumed in deriving the adjoint variable approach that the energy of the original simulation is already dissipated by the end of the simulation. The ramping up parts of the adjoint response, thus intersects with near zero field values in the original simulation. This fact is illustrated in Figure 4.2.
Figure 4.1  An illustration of the phase shift concept in the adjoint simulation: the steady-state response from a ramped sine wave (solid lagging graph) is a shifted version of the steady-state response due to a ramped cosine wave (dotted leading graph).

Figure 4.2  An illustration of the ramping of the adjoint excitation: the original response (Gaussian-modulated sinusoid) is effectively zero at the time at which the sinusoidal adjoint excitation is ramped. Note that the original simulation runs from left to right (in the direction of $t$), while the adjoint simulation runs from right to left (in the direction of $\tau$).
Example 4.1 Consider the 1D FDTD example of Section 3.2. The same discretization, number of cells, excitation, and boundaries are utilized here. It is required to evaluate the sensitivity of the output spectrum

\[
\tilde{E}(\omega_0) = \int_0^{T_m} E(t)e^{-j\omega_0 t} dt,
\]

where \(\omega_0 = 2\pi f_o\) with \(f_o = 5.0\) GHz.

The configuration of this problem is shown in Figure 4.3 again for convenience. The main difference between this problem and Example 3.1 is that the objective function is not a real-valued energy function but rather the complex spectrum component of the electric field at a specific frequency.

The original problem is the same for both examples. The only difference is in the adjoint excitation. We apply the ramped cosine excitation shown in the dotted plot of Figure 4.2. This creates the adjoint response corresponding to the real part of the spectrum. This response is then shifted by the number of steps (4.9) to estimate the adjoint response corresponding to the imaginary part of the spectrum. The implementation of this approach is shown in MATLAB® Listing M4.1.

The code starts first by the parameterization part. For the parameters \(p = [L \quad \varepsilon_r]^T\), we determine the perturbation domains affected by forward and backward perturbations in each parameter. Because the dielectric discontinuity extends over cells with indices \(\text{slab}_{\text{start}}\) to \(\text{slab}_{\text{stop}}\), there is only one cell affected by perturbing the parameter \(L\) in the forward direction (with index \(\text{slab}_{\text{stop}} + 1\)) and also one cell is affected by perturbing it in the backward direction (with index \(\text{slab}_{\text{stop}}\)). Perturbing the parameter \(\varepsilon_r\) either in the forward direction (by increasing it) or in the backward direction (by decreasing it) affects all cells of the dielectric discontinuity (cells with indices \(\text{slab}_{\text{start}}\) to \(\text{slab}_{\text{stop}}\)).

The original simulation is first carried out, and the original responses at the perturbation regions are stored. The parameter \text{frequency} determines the frequency at which the spectral adjoint sensitivities are calculated. The value of the complex spectrum at the observation point is estimated using the command:

\[
\text{spectrum} = \sum(\text{dft_exponents} \times \text{td_original}(\text{:,response_domain})),
\]

This command multiplies every time-domain value by the corresponding complex exponent and then carries out the summation as given by (4.2).
The adjoint excitation of the real part of the spectrum is a ramped cosine function and is given by the command

\[
\text{real\_adjoint\_excitation = -1.0*cos(omega\_0*(number\_of\_time\_steps-time\_indices)*dt).*smoothing\_function,}
\]

where the \text{smoothing\_function} vector is set in the settings script and has a ramped Gaussian profile. This smoothing function avoids creating a discontinuity at \( t = 0 \) as explained earlier. The negative sign is needed to utilize the adjoint excitation as an electric current density source as explained in Chapter 3.
The adjoint simulation with the ramped cosine excitation is carried out. Similar to theory explained in Chapter 3, the corresponding adjoint responses are used to estimate the sensitivities of the real part of the spectrum relative to the two parameters $p_1$ and $p_2$. These sensitivities are expressed by the parameters $\text{real\_sensitivity\_p1}$ and $\text{real\_sensitivity\_p2}$.

The code then deduces the adjoint response corresponding to the imaginary part of the spectrum using the adjoint response corresponding to the real part. The command,

$$\text{adjoint\_p1\_f\_imag((shift+1):end)} = -1*\text{adjoint\_p1\_f\_real(1:end-shift)},$$

copies the adjoint response at the perturbed region of the first parameter corresponding to the real part of the spectrum with a shift using the $\text{shift}$ parameter. This shift value is given by (4.9). The same copy and shift approach is also implemented for the second parameter.

Once the adjoint responses corresponding to the imaginary parts are determined, they are used to evaluate the sensitivities of the imaginary part of the spectrum with respect to both parameters using the commands

$$\begin{align*}
\text{imag\_sensitivity\_p1} &= -\text{sum}\left(\text{sum}(\text{residue\_p1\_f} \cdot \text{adjoint\_p1\_f\_imag(2:number\_of\_time\_steps,1)})) \right) \cdot \frac{\text{dt}}{\text{dp1}} \\
\text{imag\_sensitivity\_p2} &= -\text{sum}\left(\text{sum}(\text{residue\_p2\_f} \cdot \text{adjoint\_p2\_f\_imag(2:number\_of\_time\_steps,:)})) \right) \cdot \frac{\text{dt}}{\text{dp2}}.
\end{align*}$$

These commands execute the typical summation of the product between original residues and adjoint responses. Executing this code, MATLAB creates the numerical output

- $\text{real\_sensitivity\_p1} = -1.7827e-10$
- $\text{real\_sensitivity\_p2} = -2.2763e-13$
- $\text{imag\_sensitivity\_p1} = 5.1769e-11$
- $\text{imag\_sensitivity\_p2} = 6.6283e-14$

To verify these results, we utilize a finite difference code that evaluates the spectrum for perturbed values of the parameters $p_1$ and $p_2$. This code is given in MATLAB Listing M4.2. This code perturbs every parameter in the forward and backward simulation and then evaluates forward difference approximations,
backward difference approximations, and central difference approximations. The CFD approximations are

\[
\text{sensitivity}_p_1_{\text{cfd}} = -1.7806e-10 + 5.1624e-11i
\]

\[
\text{sensitivity}_p_2_{\text{cfd}} = -2.2727e-13 + 6.7442e-14i.
\]
It should be clear to the reader that these results are presented in their complex form. A comparison between the AVM spectral sensitivities and the CFD sensitivities shows good agreement. The AVM sensitivities are estimated for different frequencies by changing the value of the parameter \texttt{frequency} in MATLAB Listings M4.1 and M4.2. This sweep is shown in Figure 4.4. Good match is observed between both approaches. The AVM approach requires only one extra simulation regardless of the number of parameters.

4.2 The wideband case

The approach presented in the previous section handles the case when it is desired to estimate the sensitivities of a spectrum at only one frequency. This, however, is not the general case for a time-domain solver. Usually, a wide frequency band is handled simultaneously. To estimate the wideband sensitivities, we assume that the adjoint problem is excited with a wideband temporal excitation $h(\tau)$ and a spatial distribution $a$. The discretized Fourier transform expression (4.2) is used to
estimate the spectrum of the temporal adjoint excitation $h(t)$ at the desired frequencies. This spectrum is given by

$$
\tilde{H}(f_m) \approx \Delta t \sum_{n=1}^{N_T} h(\tau_n) \exp(-2\pi j f_m n\Delta \tau).
$$

(4.12)

In theory, any frequency $f_m$ can be utilized in (4.12). However, because we are using only $N_T$ time samples, the spectrum becomes periodic, and we have only $N_T$-independent spectral values. These frequencies are usually selected as uniformly distributed with $f_m = m\Delta f$, $\Delta f = 1/(N_T\Delta \tau)$, and $\tau_n = n\Delta \tau = n\Delta t$ [11]. Using this selection of frequencies, the spectrum can be written as

$$
\tilde{H}(f_m) \approx \Delta t \sum_{n=1}^{N_T} h(\tau_n) \exp\left(-2\pi j \left(\frac{m}{N_T}\right)\right), \quad m = 1, 2, \ldots, N_T.
$$

(4.13)

In practice, only a small subset of the frequencies in (4.13) is sufficient to cover the band of the excited source. The inverse discrete Fourier transform (IDFT) evaluates the time-domain signal values using the evaluated frequency domain spectrum given by (4.12). In this book, we utilized the scaled IDFT

$$
h(\tau_n) \approx \Delta f \sum_{m=1}^{N_T} \tilde{H}(f_m) \exp(2\pi j f_m n\Delta \tau).
$$

(4.14)

For a real band-limited excitation, the spectrum (4.12) exhibits the property $\tilde{H}(f_m) = \tilde{H}(f_{N_T-m})^*$ where $(\cdot)^*$ denotes complex conjugacy. It follows that for a wideband real adjoint excitation $h(t)$, we have

$$
h(\tau_n) \approx 2\Delta f \sum_m \left|\tilde{H}(f_m)\right| \cos(2\pi f_m \tau_n + \angle\tilde{H}(f_m)),
$$

(4.15)

where $\tilde{H}(f_m) = |\tilde{H}(f_m)| \exp(i\angle\tilde{H}(f_m))$. A general wideband adjoint excitation $h(t)$ can thus be written as a sum of cosines with different amplitudes and phases. The contribution of the $m$th frequency to the expansion (4.15) is given by

$$
\text{mth contribution} \approx 2\Delta f \left|\tilde{H}(f_m)\right| \cos(2\pi f_m \tau + \angle\tilde{H}(f_m)).
$$

(4.16)

It follows from (4.15) that we know the steady-state phasor of each of the excited frequencies. The wideband excitation $h(t)$ results in the wideband adjoint excitation $l(t)$ over the whole computational domain.

The question that stands out is as follows: knowing the wideband adjoint response $l_h(t)$ for all time steps, can we predict the sinusoidal adjoint responses $l_r(t)$ and $l_i(t)$ corresponding to the adjoint excitations (4.7) and (4.8) at any frequency of interest. The answer is “yes”. The adjoint response $l_h(t)$ can be decomposed to its spectral components using (4.2) to obtain $\tilde{l}_{h_m}(f_m)$, $\forall m$ as given by (4.13). These phasors determine the amplitude and phase of the steady-state sinusoidal responses due to the different frequency components within the signal $h(t)$. If we know the
spectral response of the $j$th component $\tilde{\lambda}_{h,j}(f_m)$ of the spectrum $\tilde{\lambda}_h(f_m)$ due to the sinusoidal excitation $a(2\Delta f|H(f_m)|\cos[2\pi f_m\tau_n + \angle H(f_m)])$, we can predict the amplitude and phase of the adjoint response corresponding to the excitation (4.7) through the formula [4]:

$$
\lambda_{r,j}(f_m,n\Delta \tau) = 2\Delta f \frac{|\tilde{\lambda}_{h,j}(f_m)|}{2\Delta f|H(f_m)|} \cos\left(2\pi f_m n\Delta \tau + \angle \tilde{\lambda}_{h,j}(f_m) - \angle \tilde{H}(f_m) - 2\pi f_m T_m - \frac{\pi}{2}\right).
$$

(4.17)

Note that as per (4.15), the conversion from the phasor to the time-domain quantity requires a multiplication by $2\Delta f$, which cancels out with the $2\Delta f$ in the amplitude of the $m$th frequency contribution. All the entries in (4.17) are known and thus we can predict with accuracy the steady-state sinusoidal response corresponding to any frequency $f_m$ and thus evaluate the sensitivity of the real and imaginary parts of the spectrum frequencies, we have

$$
\Delta f \Delta \tau = \frac{1}{N_f}.
$$

Expressions (4.17) and (4.18) can be further simplified by noticing that $2\pi f_m T_m = 2\pi (m\Delta f)(N_f\Delta \tau) = 2mf_n$. It follows that the constant phase shift $2\pi f_m T_m$ inside the cosine argument in (4.17) and (4.18) can be ignored. Also, because of our choice of the spectrum frequencies, we have $\Delta f \Delta \tau = (1/N_f)$. Expressions (4.17) and (4.18) are used to estimate all the desired adjoint responses, $\forall j$.

Expressions (4.17) and (4.18) obtain the time-domain expression of the predicted adjoint responses $\lambda_{r,j}(n\Delta \tau)$ and $\lambda_{i,j}(n\Delta \tau)$. Alternatively, we can first obtain their phasors and convert them back to the time domain. These phasors are given by

$$
\tilde{\lambda}_{r,j}(f_m) = \frac{\tilde{\lambda}_{h,j}(f_m)}{2\Delta f|H(f_m)|} = |\tilde{\lambda}_{r,j}(f_m)|e^{j\tilde{\lambda}_{r,j}(f_m)},
$$

(4.19)

$$
\tilde{\lambda}_{i,j}(f_m) = \frac{\tilde{\lambda}_{h,j}(f_m)}{2\Delta f|H(f_m)|\tilde{s}} = |\tilde{\lambda}_{i,j}(f_m)|e^{j\tilde{\lambda}_{i,j}(f_m)},
$$

(4.20)

where $\tilde{s}$ is a constant complex phasor with value $\tilde{s} = e^{j(\pi/2)}$. Once the two phasors (4.19) and (4.20) are determined for all frequency components, the corresponding steady-state sinusoidal functions are given by

$$
\lambda_{r,j}(n\Delta \tau) = 2\Delta f |\tilde{\lambda}_{r,j}(f_m)| \cos\left(2\pi f_m n\Delta \tau + \angle \tilde{\lambda}_{r,j}(f_m)\right),
$$

(4.21)

$$
\lambda_{i,j}(n\Delta \tau) = 2\Delta f |\tilde{\lambda}_{i,j}(f_m)| \cos\left(2\pi f_m n\Delta \tau + \angle \tilde{\lambda}_{i,j}(f_m)\right).
$$

(4.22)
The algorithm for implementing the wideband adjoint sensitivity analysis is thus given by

**Step 1. Parameterization**

For each parameter $p_i$, we identify its perturbation domains, both forward and backward $S_{pi}$. This parameterization step is the same as the one used in Section 3.2.

**Step 2. Original simulation**

Carry out the original simulation and store the residues $\Delta R_i(k\Delta t)$ associated with each parameter, $\forall i$. There is no need to store the adjoint excitation as it is independent of the original simulation.

**Step 3. Sensitivity analysis**

Select a wideband adjoint excitation $h(t)$. Using (4.2), determine the corresponding spectral components $\tilde{H}(f_m) = \tilde{H}(f_m) \exp(j \cdot \tilde{H}(f_m))$, $\forall m$. Execute the adjoint simulation and store the temporal adjoint response $\lambda_h(t)$ in the perturbation domains. Carry out the discrete transformation (4.2) at these cells to obtain $\hat{\lambda}_{i,j}(f_m)$, $\forall j$, and $\forall m$. Using formulae (4.19) and (4.20), determine the corresponding phasors for the real and imaginary parts of the spectrum. Determine the steady-state temporal profiles $\lambda_{r,j}(n\Delta t)$ and $\lambda_{i,j}(n\Delta t)$, $\forall n$. Apply formula (3.37) to estimate the derivatives of the real and imaginary part of the spectrum.

The following example illustrates the 2D case.

---

**Example 4.2** Consider the 2D structure shown in Figure 4.5. A discontinuity made of a good conductor is illuminated by an incident plane wave. We would like to determine the sensitivity of the output spectrum

$$
\tilde{E}(\omega) = \int_0^{T_w} \left( \int_{\Omega} E(t, \Omega) \cdot E_m(\Omega) d\Omega \right) e^{-j\omega t} dt = \int_0^{T_w} A(t) e^{-j\omega t} dt 
$$

with respect to the discontinuity parameters $p = [\varepsilon_r \sigma \varepsilon x_{\min} x_{\max} y_{\min} y_{\max}]^T$ at the values $p_o = [1 \ 10^{10} \ -5.0 \text{ mm} \ 6.0 \text{ mm} \ 28.0 \text{ mm} \ 41.0 \text{ mm}]^T$. $A(t)$ is the temporal modal amplitude, and $\Omega$ is the output port extending from $x = -30 \text{ mm}$ to $x = 30 \text{ mm}$. $E_m(\Omega)$ is the normalized modal profile across the output port that is a constant for this problem.

The integral,

$$
A(t) = \int_{\Omega} E(t, \Omega) \cdot E_m(\Omega) d\Omega,
$$

evaluates the amplitude of the mode at time $t$. Taking the Fourier transform of this modal amplitude in (4.23) determines the spectrum of the mode at the output port.

For the wideband case, the original and adjoint simulations are decoupled. The excitation of the adjoint problem can be any wideband waveform with the desired bandwidth. The only change in the original simulation is to evaluate the complex
output spectrum instead of calculating a scalar objective function. The modified script \texttt{update\_objective\_function} is shown in MATLAB Listing M4.3.

The modal amplitude $A(t)$ is approximated by the discrete space summation

$$A(t) \approx \Delta\Omega \sum_{l} E(t,\Omega_l) \cdot E_m(\Omega_l) = \Delta x \sum_{l} E(t,x_l) \cdot E_m(x_l),$$

where $l$ is the index of the nodes in the observation domain. This expression is implemented by the command

\begin{verbatim}
modal_amplitude = dx*electric_field_average*electric_field_modal_distribution.
\end{verbatim}
The product between the electric field values and the modal values is represented by a product between the row vector of electric field values across the output port and the column modal values across the output port. The matrix `current_dft_exponents` includes all the exponents \( \exp(-j\omega_m (k+0.5)\Delta t) \), for all considered frequencies and for all time steps. Because these exponents are used often in the wideband case, they are stored in this matrix beforehand and initialized in the original simulation. The integral (4.23) is approximated using the expression

\[
\tilde{E}(\omega_m) = \Delta t \sum_k A((k + 0.5)\Delta t)E^{-j\omega_m (k+0.5)\Delta t}.
\]

This expression is implemented for all frequencies \( \omega_m \), \( \forall m \) using the command

\[
\text{spectrum}=\text{spectrum}+\text{dt} \cdot \text{modal_amplitude} \cdot \text{current_dft_exponents};
\]

where the variable `spectrum` contains 12 components for the 12 considered frequencies.

As explained earlier, the temporal profile of the adjoint simulation is independent of the original simulation. Any wideband temporal profile that includes the frequencies of interest will suffice. The spatial profile of the adjoint excitation, however, has to observe the modal distribution. To show this dependence, we rewrite the spectrum (4.22) in the form

\[
\tilde{E}(\omega) = \int_0^{T_m} \left( \int_\Omega E(t,\Omega) \cdot E_m(\Omega) d\Omega \right) e^{-j\omega t} dt
\]

\[
= \Delta x \int_0^{T_m} \sum_l E(t,x_l) \cdot E_m(x_l) e^{-j\omega t} dt.
\]

(4.25)

The adjoint excitation related to the real part of the spectrum at the point \( x_l \) of the observation line is the derivative of (4.25) with respect to field values and is given by

\[
Q_r(t,x_l) = \Delta x E_m(x_l) \cos (\omega t) \Rightarrow Q_r(t,x_l) = \Delta x E_m(x_l) \cos (\omega (T_m - \tau)).
\]

(4.26)

Similarly, the adjoint excitation related to the imaginary part of the spectrum at the point \( x_l \) the observation line is given by

\[
Q_i(t,x_l) = -\Delta x E_m(x_l) \sin (\omega t) \Rightarrow Q_i(t,x_l) = \Delta x E_m(x_l) \sin (\omega (\tau - T_m)).
\]

(4.27)

Using (4.26) and the discussion on wideband adjoint sensitivity analysis of frequency-dependent responses, the adjoint excitation used in this problem is selected as

\[
Q(\tau) = \Delta x E_m h(\tau),
\]

(4.28)

where \( E_m \) is the vector of modal profile at all cells in the observation line, and \( h(\tau) \) is a wideband temporal function. In this example, we use a Gaussian-modulated sinusoidal signal for \( h(\tau) \) with a bandwidth that covers the selected frequencies.

The adjoint code is slightly different from the previously addressed examples where we had only a single objective function. The adjoint simulator is given by the script `fdtd_solve_adjoint` and is shown in MATLAB Listing M4.4. This code still keeps that same geometry and the same boundaries as the original simulation. The adjoint source is, however, a Gaussian-modulated sinusoid.
In addition to the temporal value of the adjoint excitation, we also need to determine its spectrum as per (4.12).

The adjoint simulator starts first by determining the adjoint excitation temporal and frequency profiles. The adjoint excitation time profile and its spectrum are initialized in the script `initialize_adjoint_excitation_time_profile` shown in MATLAB Listing M4.5. This code initializes all the values of the temporal function \( h(k\Delta r) \), \( \forall k \). It also evaluates the spectrum \( H(f_m) \) given by (4.12) at the desired frequencies. These frequencies are given by the vector `dft_frequencies` and their number is given by the variable `number_of_dft_frequencies`. Both variables are set in the original simulation.
Once the temporal profile of the adjoint excitation is determined, the algorithm moves to determine the current sources to be used as adjoint excitations. They have the temporal profile $h(t)$, but at the same time, they have the spatial modal profile determined by the modal excitation vector $E_m$ as given by (4.28). As per formula (3.20), the line of electric current sources corresponding to (4.28) is given by

$$J_z(x_l, t) = -\Delta x E_m(x_l) h(t), \forall l.$$

The sources in (4.29) are polarized in the $z$-direction normal to the 2D plane. The adjoint current sources are initialized in the script `initialize_adjoint_domain` shown in MATLAB Listing M4.6. This script starts by first disabling the plane wave source utilized in the original simulation. It then determines the number of cells across the observation domain (extending from $i_0$ to $i_e$). This sets the

The utilized time-profile and its corresponding spectrum are shown in Figure 4.6 (a) and (b).

Figure 4.6 The adjoint excitation utilized for the 2D metallic discontinuity example: (a) the time-domain profile of the adjoint excitation and (b) the absolute value of the spectrum at the selected set of 12 frequencies
Once the number of adjoint electric sources is determined, the script initializes all the information of each current source including its time profile. The command

```matlab
impressed_J(ind).waveform =-1*dx*
electric_field_modal_distribution(ind)*
adjoint_excitation_time_profile;
```

implements (4.29) where the variable `adjoint_excitation_time_profile` contains the time domain values of the Gaussian-modulated sinusoid. In this example, because a plane wave excitation is utilized, the modal distribution is a constant. However, for structures with rectangular or other closed waveguides, the modal distribution may have other spatial profiles.

The next step in the adjoint simulation is to set the adjoint storage needed for each variable. When the objective function is a scalar, we showed that we have to store the adjoint excitation \( \lambda \) at all cells associated with the parameters for all time steps. Here, however, we adopt a different implementation. Because we need only to know the adjoint response phasor \( \lambda_{h,j}(f_m) \) for all associated nodes, we then store for each cell its spectrum at all considered frequencies. The spectrum requires much less storage than the temporal response. For example, if we run the adjoint problem for 2,000 time steps, we previously had to store 2,000 double values for
any cell within the perturbation domain of any of the parameters. However, in our current implementation, as we consider 12 frequencies, only 24 doubles are stored for every node within the perturbation domains.

The script that initializes the adjoint storage is `initialize_adjoint_storage`. We show a subset of this script in MATLAB Listing M4.7. As in the

```matlab
for k=2:numberOfRectangles
    parameter_start_index=(k-2)*6+1; % start index of parameters
    % first parameter is relative permittivity. Storage covers all brick
    ni = get_node_indices(rectangles(k), edtd_domain);
    adjacent_sampled_field(parameter_start_index).min_x = ni.is;
    adjacent_sampled_field(parameter_start_index).min_y = ni.js;
    adjacent_sampled_field(parameter_start_index).max_x = (ni.ie)-1;
    adjacent_sampled_field(parameter_start_index).max_y = (ni.js)-1;
    adjacent_sampled_field(parameter_start_index).direction=m';
    adjacent_sampled_field(parameter_start_index).storeflag=true;
    adjacent_sampled_field(parameter_start_index).display_plot = false;
    adjacent_sampled_field(parameter_start_index).storage_size=ni.ie-ni.is+1;*(ni.js-ni.js+1);
    adjacent_sampled_field(parameter_start_index).spectrum=zeros((ni.ie-ni.is+1)*(ni.js-ni.js+1),number_of_dft_frequencies);
    % second parameter is conductivity. Storage covers all brick
    adjacent_sampled_field(parameter_start_index+1).min_x = ni.is;
    adjacent_sampled_field(parameter_start_index+1).min_y = ni.js;
    adjacent_sampled_field(parameter_start_index+1).max_x = (ni.ie)-1;
    adjacent_sampled_field(parameter_start_index+1).max_y = (ni.js)-1;
    adjacent_sampled_field(parameter_start_index+1).direction=m';
    adjacent_sampled_field(parameter_start_index+1).storeflag=true;
    adjacent_sampled_field(parameter_start_index+1).display_plot = false;
    adjacent_sampled_field(parameter_start_index+1).storage_size=ni.ie-ni.is+1;*(ni.js-ni.js+1);
    adjacent_sampled_field(parameter_start_index+1).spectrum=zeros((ni.ie-ni.is+1)*(ni.js-ni.js+1),number_of_dft_frequencies);
    % third parameter is min_x of the brick
    adjacent_sampled_field(parameter_start_index+2).min_x = (ni.is)-1;
    adjacent_sampled_field(parameter_start_index+2).min_y = ni.js;
    adjacent_sampled_field(parameter_start_index+2).max_x = (ni.is)-1;
    adjacent_sampled_field(parameter_start_index+2).max_y = (ni.js)-1;
    adjacent_sampled_field(parameter_start_index+2).direction=xn';
    adjacent_sampled_field(parameter_start_index+2).storeflag=true;
    adjacent_sampled_field(parameter_start_index+2).display_plot = false;
    adjacent_sampled_field(parameter_start_index+2).storage_size=2*(ni.is-ni.js+1);
    adjacent_sampled_field(parameter_start_index+2).spectrum=zeros(2*(ni.js-ni.js+1),number_of_dft_frequencies);
    % fourth parameter is max_x of the brick.
    adjacent_sampled_field(parameter_start_index+3).min_x = (ni.ie)-1;
    adjacent_sampled_field(parameter_start_index+3).min_y = ni.js;
    adjacent_sampled_field(parameter_start_index+3).max_x = (ni.ie)-1;
    adjacent_sampled_field(parameter_start_index+3).max_y = (ni.js)-1;
    adjacent_sampled_field(parameter_start_index+3).storeflag=true;
    adjacent_sampled_field(parameter_start_index+3).display_plot = false;
    adjacent_sampled_field(parameter_start_index+3).storage_size=2*(ni.ie-ni.js+1);
    adjacent_sampled_field(parameter_start_index+3).spectrum=zeros(2*(ni.js-ni.js+1),number_of_dft_frequencies);
end
```
other 2D examples addressed so far, we assume that every discontinuity has a rectangular shape. There are six parameters associated with every rectangular region \( p = [\varepsilon_r, \sigma, x_{\min}, x_{\max}, y_{\min}, y_{\max}]^T \). The code determines the number of cells associated with the perturbation region of each parameter. The command,

\[
\text{adjoint_sampled_field}(\text{parameter_start_index}).\text{storage_size} = (\text{ni.ie}-\text{ni.is}+1)(\text{ni.je}-\text{ni.js}+1),
\]

determines the size of the needed storage as the product of the number of field components in the \( x \)-direction and the number of field components in the \( y \)-direction within the perturbation region of the current parameter (with index \text{parameter_start_index}). The storage size required for all other parameters is set in the same way. For every component, we store only one complex number per frequency corresponding to the phasor of the field component at this point. These phasors are calculated during the adjoint simulation at all desired frequencies. The command,

\[
\text{adjoint_sampled_field}(\text{parameter_start_index}).\text{spectrum} = \text{zeros}((\text{ni.ie}-\text{ni.is}+1)(\text{ni.je}-\text{ni.js}+1),\text{number_of_dft_frequencies});
\]

stores one complex number per field component per frequency for the current parameter within its perturbation domain.

The script \text{run_adjoint_fdtd_time_marching_loop_2d} carries out the adjoint simulation. A main difference between this implementation and the implementation discussed in Chapter 3 is that we do not carry out the product between the temporal adjoint fields and the original residues within the adjoint simulation loop. The code has first to estimate the spectra of all adjoint field components within the perturbation regions of all parameters at every frequency. These spectra are available only at the end of the adjoint simulation. It follows that an on-the-fly approach is not applicable in this implementation.

The script that updates the spectra at all field components is \text{update_adjoint_spectrum}. It is invoked at every time step within the adjoint simulation to update the adjoint spectra using the new adjoint field values as they become available. This script is shown in MATLAB Listing M4.8. The command,

\[
\text{adjoint_sampled_field}(\text{parameter_index}).\text{spectrum}(::,\text{ind}) = \text{adjoint_sampled_field}(\text{parameter_index}).\text{spectrum}(::,\text{ind}) + \text{dt} \times \text{average_Ez_adjoint'} \times \text{dft_exponents}(:,\text{ind},\text{time_step});
\]

updates the vector of spectra for all nodes associated with the parameter with index \text{parameter_index} at the frequency with index \text{ind} at the time step \text{time_step}. This command is repeated for all the frequencies considered for Fourier transform. By the end of the adjoint simulation, the storage \text{adjoint_sampled_field}
(parameter_index).spectrum contains the Fourier transform for all the nodes associated with the parameter parameter_index for all frequencies. To evaluate the corrected spectra (4.19) and (4.20), the code proceeds to correct the adjoint spectra in both amplitude and phase. The script calculate_parameter_sensitivities is invoked twice to correct these spectra. It also calculates, using these spectra, the corresponding time-domain cosine and sine waveforms. These cosine and sine functions are then used along with the original residues to estimate the derivatives of the modal spectrum with respect to all parameters at all frequencies. The parameter fixed_shift is the constant phase shift phasor \( \tilde{s} \) utilized in (4.19) and (4.20). Note that in (4.19), this phasor causes a zero-phase shift. The script calculate_parameter_sensitivities is shown in MATLAB Listing M4.9. The command,

```
relative_domain_spectrum(ind,:) =
transpose(domain_spectrum(ind,:))./
(2*df*(adjoint_excitation_spectrum.*fixed_shift));
```
computes the corrected spectrum at a frequency with an index \( \text{ind} \) as per (4.19) and (4.20). Once the corrected spectra are available, they are then used to calculate the time-domain cosines through the command

\[
\text{current\_cosine\_vector} = 2 \cdot \text{df} \cdot \text{real}(\text{current\_phasor\_vector} \cdot \text{current\_dft\_exponents});
\]

This command calculates the time-domain values by multiplying the vector of nodal phasors \( \text{current\_phasor\_vector} \) by the vector \( \text{current\_dft\_exponents} \) containing the values of \( e^{j\omega t} \) for all time at the considered frequency. The code loops to consider all frequencies.
The last step in the adjoint part is to carry out an inner product between the corrected temporal sinusoids and the residues of the original simulation. This part is also shown in MATLAB Listing M4.9. The command,

```matlab
sensitivities(1,dft_frequency_index) = sensitivities(1,dft_frequency_index) - dt*(residue_Ez*current_cosine_vector(:,time_step));
```

![Graphs showing sensitivities](image)

**Figure 4.7** The sensitivities of the real part of the spectrum of the 2D metal discontinuity example with respect to all parameters: the forward finite difference estimates (–) as compared to the adjoint sensitivity estimates (○) over the 12 considered frequencies.
updates the sensitivities of the spectrum at the frequency with index \texttt{dft\_frequency\_index} with respect to the current parameter at the time step \texttt{time\_step}. This command is repeated for all parameters, for all frequencies, and for all time steps to evaluate all sensitivities.

The sensitivities of the real part and imaginary part of the spectrum at the considered frequencies are shown in Figures 4.7 and 4.8. Good agreement is achieved with forward finite difference approximations. As expected, the spectrum shows negligible little sensitivity with respect to the dielectric constant and the electric conductivity because the discontinuity’s material is a good conductor.

\textbf{Figure 4.8} The sensitivities of the imaginary part of the spectrum of the 2D metal discontinuity example with respect to all parameters: the forward finite difference estimates (--) as compared to the adjoint sensitivity estimates (\cdot) over the 12 considered frequencies.
4.3 The self-adjoint case

In the previous section, we showed that the sensitivities of wideband frequency domain spectra can be evaluated with respect to all parameters using only one adjoint simulation with a wideband excitation. We show in this section that, for certain types of problems and responses, the adjoint problem can be eliminated altogether. The required adjoint simulation field values can be deduced from the original simulation. Such an approach is referred to as self-adjoint sensitivity analysis (SASA) [5,12]. The $2n$ extra simulations required for accurately estimating the gradient of an objective function using central finite differences (CFDs) are thus reduced to 0 using the SASA approach.

Earlier in Chapter 3, we showed that the considered original simulation has the form

$$N \dot{V} + KV = G(t),$$  \hspace{1cm} (4.30)

where $G(t)$ indicates the excitation utilized in the original simulation. The derivatives in (4.30) are all relative to the original time parameter $t$. The adjoint simulation corresponding to (4.30) was shown to be

$$N^T \frac{d\lambda}{dt} + K^T \lambda = Q(\tau) = \left( \frac{\partial \psi(T_m - \tau)}{\partial V} \right).$$  \hspace{1cm} (4.31)

All derivatives in (4.31) are with respect to the adjoint time variable $\tau$. If the system matrices are symmetric, as in most problems with symmetric permittivity and permeability tensors, the adjoint system can be further simplified to

$$N \dot{\lambda} + K \lambda = Q(\tau).$$  \hspace{1cm} (4.32)

The two systems (4.30) and (4.32) have significant similarity. The system matrices are identical. One simulation is running forward in time with respect to $t$, while the other one is running backward in time with respect to $t$ (forward in time with respect to $\tau$). The vectors of original field values $V(t)$ and adjoint field values $\lambda(\tau)$ are different because the original and adjoint excitations $G(t)$ and $Q(\tau)$ are not related. In some problems, because of the nature of the objective function, the adjoint excitation is a scaled and shifted version of the original excitation. This adjoint excitation can be expressed as

$$Q(\tau) = \alpha G(t - \xi),$$  \hspace{1cm} (4.33)

where $\alpha$ and $\xi$ are the scale factor and the time shift, respectively. If this condition is satisfied, this implies, because of the linearity of the adjoint system, that the adjoint fields are also scaled and shifted replica of the original fields, that is,

$$\lambda(\tau) = a V(t - \xi).$$  \hspace{1cm} (4.34)

It follows from (4.34) that there is no need to carry out the adjoint simulation (4.32). The adjoint fields are deduced at all time steps using the original fields using proper scaling and time shifting. This approach does not require an adjoint simulation and is thus labeled as a self-adjoint approach.
Example 4.3 Consider the structure shown in Figure 4.9. A parallelepiped metallic discontinuity is placed inside a rectangular waveguide of dimensions $2.8 \times 1.8$ cm. The waveguide has a length of 20 cm. It is discretized using a cubic cell of size $\Delta x = \Delta y = \Delta z = 1.0$ mm. The waveguide is excited with the dominant TE$_{10}$ mode through a layer of current density source located at $z = 5 \Delta z$. The waveguide is terminated at both ends with an absorbing boundary. The incident spectrum and transmitted spectrum are observed at $z = 10 \Delta z$ and $z = 180 \Delta z$, respectively. We show in this example that the wideband sensitivities of the scattering parameter $S_{11}$ are estimated with respect to all parameters of the discontinuity using no adjoint simulation.

To evaluate the $S$-parameters, we first run an original simulation with a hollow waveguide to determine the reference spectrum. This reference simulation is given by

$$ N_{\text{ref}} \dot{V} + K_{\text{ref}} V = \begin{bmatrix} AE_m(r)h(t) \end{bmatrix}, $$

where $N_{\text{ref}}$ and $K_{\text{ref}}$ are the system matrices of the reference simulation. The vector $E_m(r)$ is the spatial modal profile distribution of the electric current density source. In this example, the spatial modal profile is the half-sine profile of the dominant TE$_{10}$ mode. The temporal profile $h(t)$ describes the time dependency of the excitation, and $A$ is the amplitude of the excitation. The excitation is located at $z = 5 \Delta z$. In this example, a Gaussian-modulated sinusoidal signal with a center frequency of 8 GHz is utilized. The reference spectrum is given by

$$ \tilde{E}_{\text{ref}}(\omega) = \int_{0}^{T_m} \left( \int_{\Omega} E(t, r) \cdot E_m(r) d\Omega \right) e^{-j\omega t} dt, $$

where $\Omega$ is the rectangular waveguide cross section at $z = 10 \Delta z$. The reference spectrum (4.36) is used to scale the total and transmitted spectra to estimate the $S$-parameters $S_{11}$ and $S_{21}$.

![Figure 4.9](image)

*Figure 4.9* A metallic discontinuity in a rectangular waveguide. The parameters of the discontinuity are $p = [\varepsilon, \sigma, x_{\min}, x_{\max}, y_{\min}, y_{\max}, z_{\min}, z_{\max}]^T$. 
In addition to the reference simulation of a hollow waveguide, the structure is simulated with the metallic discontinuity present. This original simulation is given by

\[
N \dot{V} + KV = \begin{bmatrix} AE_m(r)h(t) \\ 0 \end{bmatrix},
\]

where \( N \) and \( K \) are the system matrices with the metallic discontinuity present. Using the results of the original simulation (4.37), the total spectrum is calculated using

\[
\tilde{E}_{\text{total}}(\omega) = \int_0^{T_w} \left( \int_{\Omega} E(t,r) \cdot E_m(r) d\Omega \right) e^{-j\omega t} dt
\]

\[
\approx \int_0^{T_w} \Delta x \Delta y \left( \sum_l E(t,r_l) \cdot E_m(r_l) \right) e^{-j\omega t} dt,
\]

where the summation is carried out over all the cells of the cross section of the rectangular waveguide at the observation plane \( z = 10 \Delta z \). Using the total and reference spectra, the \( S \)-parameter \( S_{11} \) is given by

\[
S_{11}(\omega) = \frac{\tilde{E}_{\text{total}}(\omega) - \tilde{E}_{\text{ref}}(\omega)}{\tilde{E}_{\text{ref}}(\omega)}.
\]

The sensitivities of \( S_{11} \) can be evaluated using finite differences by perturbing the parameters of the discontinuity, one by one, and then evaluating (4.37)–(4.39) for the perturbed parameter values. To apply adjoint analysis, we determine the adjoint excitations of the real and imaginary parts, from (4.38) to get

\[
Q_r = \begin{bmatrix} \Delta x \Delta y E_m(r) \cos(\omega t) \\ 0 \end{bmatrix},
\]

\[
Q_i = \begin{bmatrix} -\Delta x \Delta y E_m(r) \sin(\omega t) \\ 0 \end{bmatrix}.
\]

These two excitations are to be applied at the observation domain at \( z = 10 \Delta z \). Comparing the original simulation (4.37) with the adjoint simulation (4.32) but with the adjoint excitation (4.40), we see that the two simulations have the same modal profile. One of them is wideband, while the other one is monochromatic. The original excitation is at \( z = 5 \Delta z \), while the adjoint excitation is at \( z = 10 \Delta z \). It follows that we can adapt formulae (4.17) and (4.18) to predict the adjoint impulses corresponding to the adjoint excitations (4.40) and (4.41). We only have to add a delay corresponding to the \( 5 \Delta z \) shift in the location of the excitations along the waveguide. There is no need for an adjoint simulation in this case.

The approach explained in (4.35)–(4.41) is implemented in MATLAB Listing M4.10. This code shows the first part of the original simulation that is also the adjoint simulation. Within the time-marching loop \texttt{run_fdtd_original_time_marching_loop}, both the original residues and the spectrum of the adjoint spectra
are estimated. Similar to the wideband case, the spectra of the stored fields need to be adjusted in amplitude and phase to predict the desired adjoint fields. The command 
\[ \text{adjustment\_shift} = 5 \]
defines the shift between the \( z \)-coordinate of the original excitation and the \( z \)-coordinate of the adjoint excitation. The second part of the code is shown in MATLAB Listing M4.11.

MATLAB Listing M4.11 is very similar to the second part of MATLAB Listing M4.4. Two main differences exist. First, there is extra phase shift added to the phasors due to the \( 5\Delta z \) spatial shift of the original and adjoint excitation. The command,

\[
\text{fixed\_shift} = \text{transpose}(\exp(\sqrt{-1} \times 2\pi \times \text{dft\_frequencies} \times \text{dt} \times (\text{number\_of\_time\_steps} - \text{adjustment\_shift})))
\]

are estimated. Similar to the wideband case, the spectra of the stored fields need to be adjusted in amplitude and phase to predict the desired adjoint fields. The command 
\[ \text{adjustment\_shift} = 5 \]
defines the shift between the \( z \)-coordinate of the original excitation and the \( z \)-coordinate of the adjoint excitation. The second part of the code is shown in MATLAB Listing M4.11.

MATLAB Listing M4.11 is very similar to the second part of MATLAB Listing M4.4. Two main differences exist. First, there is extra phase shift added to the phasors due to the \( 5\Delta z \) spatial shift of the original and adjoint excitation. The command,
adds the extra phase shift for the real part of the spectrum, while the command,
\[
\text{fixed_shift=transpose(exp(sqrt(-1)*(2*pi*dft_frequencies*dt* (number_of_time_steps-adjustment_shift))+0.5*pi))},
\]
adds the extra phase shift for the imaginary part of the spectrum. The second difference is that MATLAB Listing M4.11 handles a 3D problem. At every stored
MATLAB Listing M4.12

MATLAB Listing M4.12

// MATLAB Listing M4.12
% now we evaluate the sensitivities of S11 using the sensitivities of the
% incident spectrum
for brick_index=5:number_of_bricks
    for local_parameter_index=1:6
        global_parameter_index=(brick_index-5)*6+local_parameter_index;
        real_adjoint_sensitivities(global_parameter_index,:)=
            real_spectrum_sensitivities(global_parameter_index,:).*real(reference_spectrum)+
            imag_spectrum_sensitivities(global_parameter_index,:).*imag(reference_spectrum)./abs(reference_spectrum).^2;

        imag_adjoint_sensitivities(global_parameter_index,:)=
            (imag_spectrum_sensitivities(global_parameter_index,:).*real(reference_spectrum)-
            real_spectrum_sensitivities(global_parameter_index,:).*imag(reference_spectrum))./abs(reference_spectrum).^2;
    end
end
save('real_adjoint_sensitivities.mat', 'real_adjoint_sensitivities');
save('imag_adjoint_sensitivities.mat', 'imag_adjoint_sensitivities');


node, there are three components for the residues and three components of the adjoint fields. The phasor amplitude and phase adjustments have to be applied to all field components. The contributions to the sensitivities from different components are then added. It should be also clear that for the 3D case, each discontinuity brick is associated with eight parameters, while for the 2D case only six parameters are considered.

The MATLAB Listing M4.11 evaluates the sensitivities of the real and imaginary parts of the incident spectrum at $z = 10 \Delta z$ relative to all parameters. The scattering parameter $S_{11}$ is given by (4.39). It follows that the sensitivity of $S_{11}$ relative to the parameter $p_i$ is given by

$$\frac{\partial S_{11}}{\partial p_i} = \frac{1}{E_{\text{ref}}} \frac{\partial E_{\text{total}}}{\partial p_i} \cdot \frac{\partial E_{\text{ref}}}{\partial p_i} + \frac{\partial \text{Re}(E_{\text{ref}})}{\partial p_i} \cdot \frac{\partial \text{Re}(E_{\text{total}})}{\partial p_i}$$

$$+ j \frac{\partial \text{Im}(E_{\text{ref}})}{\partial p_i} \cdot \frac{\partial \text{Im}(E_{\text{total}})}{\partial p_i}$$

(4.42)

The last part of the main script `fdtd_solve_original`, shown in MATLAB Listing M4.12, implements this relationship to determine the sensitivities of the real and imaginary parts of $S_{11}$ using the estimated sensitivities of the total spectrum. The reference spectrum of the hollow waveguide is stored in the file `reference_spectrum.mat` that is loaded during the program to the workspace.

Sensitivity analysis for frequency-dependent objective functions
The results for this example are shown in Figures 4.10–4.15. The original excitation time profile is shown in Figure 4.10. The reference spectrum of the hollow waveguide is shown in Figure 4.11. The estimated $S$-parameters, $S_{11}$ and $S_{21}$, at the nominal parameter values are shown in Figure 4.12. The adjoint
Figure 4.12  The scattering parameters at the nominal value of the rectangular waveguide discontinuity example: (a) $|S_{11}|$ and (b) $|S_{21}|$

Figure 4.13  The adjoint sensitivities of $S_{11}$ with respect to the material parameters $(o)$ as compared to forward finite difference approximations $(\sim)$: (a) sensitivities of $\text{Re}(S_{11})$ with respect to $\varepsilon_r$, (b) sensitivities of $\text{imag}(S_{11})$ with respect to $\varepsilon_r$, (c) sensitivities of $\text{Re}(S_{11})$ with respect to $\sigma_e$, and (d) sensitivities of $\text{imag}(S_{11})$ with respect to $\sigma_e$
sensitivities of the real and imaginary parts of $S_{11}$ with respect to the material parameters are shown in Figure 4.13. The sensitivities of the real and imaginary parts of $S_{11}$ with respect to all shape parameters at all frequencies are shown in Figures 4.14 and 4.15. Figures 4.13–4.15 show, as predicted, that $S_{11}$ has negligible sensitivities with respect to the material parameters. All sensitivities are plotted on the same scale to show which parameters strongly affect the $S$-parameters.

Figure 4.14  The adjoint sensitivities of $\text{Re}(S_{11})$ with respect to the material parameters (•) as compared to forward finite difference approximations (–) over the considered frequency band.
Good agreement is achieved with the forward finite difference approximations. At some frequencies, there is better match than others. This is expected as the response at some frequencies may be more nonlinear than at other frequencies. At such frequencies, it can be shown that the forward, backward, and CFDs have significant differences. The adjoint sensitivity estimates, in such cases, are within the range spanned by all three estimates.

Figure 4.15 The adjoint sensitivities of $\text{Imag}(S_{11})$ with respect to the material parameters ($\cdot$) as compared to forward finite difference approximations ($-$) over the considered frequency band.
References


In Chapter 3, we addressed the case of a scalar objective function. The gradient of this scalar objective function with respect to all parameters is estimated using one extra adjoint simulation regardless of the number of parameters. In Chapter 4, we discussed the frequency-dependent responses. We showed that over the band of interest, the sensitivities of a frequency-dependent response with respect to all parameters are obtained using at most one extra wideband adjoint simulation. We showed also that for certain frequency domain responses, the adjoint simulation is not needed. In this case, the adjoint responses are deduced from the original simulation. It follows that only the original simulation is needed to estimate all sensitivities over the band of interest.

In this chapter, we address the transient time domain adjoint sensitivity analysis problem. Without loss of generalization, we will consider the desired transient response to be an electric field component at a point \( r_o \) in the computational domain \( E(r_o, k\Delta t) \). We show in this chapter that we can evaluate the derivatives \( \frac{\partial E(r_o, k\Delta t)}{\partial p_i} \) for all parameters, and for all time steps, using only one extra simulation. In other words, we predict how the complete transient response at the probe changes due to incremental changes in any of the parameters.

We start by first formulating the scalar case where we are interested in estimating the gradient of the response at a specific time step \( k_o \). We show that an impulsive adjoint excitation is needed to estimate the gradient \( \frac{\partial E(r_o, k_o\Delta t)}{\partial p_i} \). We also show that in order to evaluate the derivatives of the complete transient response, we only need one adjoint simulation with an impulsive excitation. Two different approaches for implementing the adjoint transient technique are presented. The theory presented in this chapter is a variation of the work published in [1,2].

### 5.1 The single time-response case

We start by first formulating the single time-step case. We would like to find the sensitivities of the observed electric field at a given point in space \( r_o \) at a specific time step \( k_o \) with respect to all parameters. The electric field at that time step is \( E(r_o, t_o) = E(r_o, k_o\Delta t) \). This electric field may be polarized along the direction of...
any of the coordinates. It can be put in the integral form

$$E(r_o, t_o) = E(r_o, k_o \Delta t) = \int_0^{T_m} E(r_o, t) S(t - t_o) dt,$$

(5.1)

where $S(t - t_o)$ is the selection function shown in Figure 5.1. This rectangular function has an area of 1 and a width of $\Delta t = \Delta \tau$. To determine the adjoint excitation corresponding to (5.1), we apply (3.20) to differentiate the kernel of the integral (5.1):

$$\frac{\partial \psi}{\partial \mathbf{V}} = \frac{\partial (E(r_o, t) S(t - t_o))}{\partial \mathbf{V}} = a S(t - t_o),$$

(5.2)

where $a$ is a vector with a value of 1 in only one component corresponding to the probe node and polarization. The selection function $S$ is not dependent on the vector of state variables $\mathbf{V}$. This vector of state variables, as mentioned earlier, includes all electric and magnetic field components at all nodes.

Because the selection function $S(t - t_o)$ has a value of $\frac{1}{\Delta t}$ at $t = t_o = k_o \Delta t$ and is zero for all other time steps, the adjoint excitation (5.2) is impulsive in nature. It follows that the corresponding adjoint simulation has the adjoint electric current density source at the observation point $r_o$:

$$J = \frac{-1}{\Delta t} \delta(t - t_o),$$

(5.3)

where $\delta(t - t_o)$ is the discrete-time direct delta function located at the observation point $r_o$. The current density source in (5.3) has the same polarization of the observed electric field. As explained in Chapter 3, the negative sign in (5.3) is needed because of the way electric current density sources appear in our formulation.

The adjoint simulation (3.20) is then executed with the excitation (5.3). Using the original residues recorded during the original simulation and the adjoint responses, the sensitivities of the response $E(t_o)$ with respect to all parameters are estimated.

The following example illustrates the single time-response approach.
Example 5.1 Consider the 1D domain shown in Figure 5.2. This domain has two dielectric discontinuities with relative permittivities \( \varepsilon_{r1} = 3 \) and \( \varepsilon_{r2} = 2 \). It is required to find the sensitivities of the electric field at the 580th node at time step \( k_0 = 6,000 \) with respect to the lengths and relative permittivities of these discontinuities.

This example is a variation of the 1D example considered in previous chapters. The settings of this problem are defined in the file `settings.m` shown in MATLAB® Listing M5.1. The first discontinuity starts at `slab1_start` with

![Figure 5.2](image)

**Figure 5.2** The domain of the two-dielectric slabs example. A current source is excited at \( z = 20\Delta z \) and the transient response is observed at \( z = 580\Delta z \).
a length of \( \text{slab1\_length} \). The second discontinuity starts at \( \text{slab2\_start} \) with a length of \( \text{slab2\_length} \). The commands

\[
\begin{align*}
\text{slab1\_stop} &= (\text{slab1\_start} + \text{slab1\_length} - 1); \\
\text{slab2\_stop} &= (\text{slab2\_start} + \text{slab2\_length} - 1); \\
\text{epsilon}r(\text{slab1\_start:slab1\_stop}) &= \text{epsilon}\_\text{slab1}; \\
\text{epsilon}r(\text{slab2\_start:slab2\_stop}) &= \text{epsilon}\_\text{slab2};
\end{align*}
\]

define the end index of each discontinuity and assign the proper relative permittivity value to the domain cells. The rest of the domain is set to air. The domain has 600 cells with the excitation at the 20th cell. The excitation is a Gaussian-modulated sinusoid as described in MATLAB Listing M5.1. The first discontinuity has a length of 50 cells, while the second one has a length of 30 cells. We consider four parameters in this example \( \mathbf{p} = [L_1 \ \epsilon_r_1 \ L_2 \ \epsilon_r_2]^T \). These parameters are the lengths and material properties of the two discontinuities. The main MATLAB code for this implementation is the script in MATLAB Listing M5.2.

This code follows the typical steps of an AVM code. It first starts by determining the perturbation domain associated with the perturbation of each parameter. The reader should notice that the length of discontinuity is perturbed by perturbing the right end of this discontinuity. This is why in the parameterization part, the two commands

\[
\begin{align*}
\text{slab1\_p1\_f} &= [\text{slab1\_stop} + 1]'; \\
\text{slab2\_p3\_f} &= [\text{slab2\_stop} + 1]';
\end{align*}
\]

select the two nodes just outside the right edges of the two discontinuities. The command \( \text{probe\_time\_step}=6000 \) sets the time step of interest. The reader should notice that if the time index of interest is \( k_o \) then the time index of the impulsive excitation in the adjoint simulation should be \( N_T - k_o + 1 \). The adjoint simulation is running backward in time (forward with respect to its time parameter \( \tau \)) and this time reversal must be taken into account. The commands

\[
\begin{align*}
\text{adjoint\_excitation} &= \text{zeros}(1, \text{number\_of\_time\_steps}); \\
\text{adjoint\_excitation}(1,\text{number\_of\_time\_steps}\_\text{-probe\_time\_step+1}) &= -1/dt;
\end{align*}
\]

set the adjoint excitation to zero at all time steps except for the time step \( \text{number\_of\_time\_steps}\_\text{-probe\_time\_step+1} \) where a discrete impulse of value \(-1/\Delta t \) is used. This takes care of the time reversal in the adjoint simulation. The adjoint excitation has the same geometry but with impulsive excitation. Using the adjoint responses and the original residues, the sensitivities of the response \( E_z(k_o\Delta t) \) at the 580th node are estimated. The output of this code is

\[
\text{sensitivity\_p1} = -0.133773553107762
\]
MATLAB Listing M5.2

sensitivity_p2 =
-1.7090584772779e-04

sensitivity_p3 =
-0.084246701442826

sensitivity_p4 =
-1.244762021373259e-04
To confirm these results, the FDTD code is also used to estimate FFD estimates of the sensitivities using the perturbations \( \Delta p = [\Delta \varepsilon, \Delta \varepsilon_r, \Delta \varepsilon_r] \) with \( \Delta \varepsilon = 0.05e^{-3} \) and \( \Delta \varepsilon_r = 0.01 \). The FFD estimates are

\[
\begin{align*}
\text{Gradient}_{\text{FFD}} &= -0.133805727002978 \\
-1.706799143716100e-04 \\
-0.084506015619546 \\
-1.244082819456497e-04
\end{align*}
\]

This result shows good match between the adjoint sensitivities and the forward finite difference estimates. The FFD estimates require four extra simulations, while the AVM approach requires only one extra adjoint simulation with an impulsive excitation.

5.2 The complete transient response case

In the previous section, we addressed the single time-response case. We showed that to estimate the sensitivities of a single transient field component \( E(r_o, t_1) \) response, which represents a specific field component at one node with location \( r_o \) at time \( t_1 \), we utilize a discrete-time impulsive excitation of amplitude \( -1/\Delta t \) at time \( t = t_1 \). Using the formula (3.45), the corresponding adjoint simulation is given by

\[
N \dot{\lambda} + K \lambda = a \delta(t - t_1), \quad (5.4)
\]

where \( a \) is a vector with only one non zero component of value \( -1/\Delta t \) at the location corresponding to the probe node and selected field component. The impulsive adjoint excitation appears at the time \( \tau_1 = T_m - t_1 \). The resulting adjoint response in the whole domain due to this excitation is \( \lambda_1(\tau) \). If we also wish to evaluate the sensitivities of the response \( E(r_o, t_2) \) with \( t_2 < t_1 \), the corresponding adjoint system is

\[
N \dot{\lambda} + K \lambda = a \delta(t - \tau_2), \quad (5.5)
\]

The impulse adjoint excitation utilized in (5.4) appears at the time \( \tau_2 = T_m - t_2 \). The resulting adjoint response in the whole domain due to this excitation is \( \lambda_2(\tau) \).

Comparing the two adjoint systems (5.4) and (5.5), we notice that the two systems are identical except for a time shift in the excitation. Assuming that the computational domain includes only linear and time-invariant materials, it follows that we have

\[
\lambda_2(\tau) = \lambda_1(\tau + \tau_1 - \tau_2), \quad \tau \geq \tau_2. \quad (5.6)
\]

In other words, the response \( \lambda_2(\tau) \) is a shifted version of \( \lambda_1(\tau) \). There is thus no need to carry out the adjoint simulation (5.5). The adjoint response \( \lambda_2(\tau) \) is deducible from (5.4). Figure 5.3 illustrates the time shift concept.
The discussion given in (5.4)–(5.6) can be generalized to estimate the sensitivities of the complete transient response. Only one adjoint simulation with the adjoint excitation \( a\delta(t) \) appearing at \( t = 0 \) is carried out. This simulation yields the adjoint fields \( \lambda_0(t) \) in the whole computational domain. This response can be used to estimate the sensitivities of the field component \( E(r, T_m) \). The adjoint response corresponding to the observed field component \( E(r_o, t_k) = E(r_o, k\Delta t) \) is deducible by

\[
\lambda_k(\tau) = \begin{cases} 
\lambda_0(\tau - T_m + k\Delta\tau), & \tau \geq T_m - k\Delta\tau \\
0, & \tau < T_m - k\Delta\tau,
\end{cases}
\] (5.7)

where (5.7) applies for all values of \( k \) and \( \Delta\tau = \Delta t \). It follows that only one adjoint simulation with an impulsive excitation at time \( t = 0 \) is carried out. This simulation yields all the adjoint fields needed to estimate the sensitivities of the complete transient response with respect to all parameters. The following example illustrates this approach.

**Example 5.2** Here, we adapt example 5.1 to estimate the sensitivities of the complete transient response for all 12,000 time steps. The field \( E_z \) is observed at the 580th FDTD cell. It is required to estimate the sensitivities of this response with respect to all four parameters. The main script that implements this functionality is `avm_slab_waveform` which is shown in MATLAB Listing M5.3. The first part of this code is similar to MATLAB Listing M5.2. The code executes the original simulation and uses it to calculate the original residues. It also executes the adjoint simulation with an impulsive source at \( t = 0 \) (first time step) and stores the adjoint...
MATLAB Listing M5.3

% this file estimate the adjoint sensitivities of the transient response of example 5.2
clear all;

% initialize sensitivities of the complete waveform relative to all parameters
sensitivities=zeros(number_of_time_steps, number_of_parameters);

% Step 1: parameterization
slab1_p1_f = [slab1_stop + 1]'; slab1_p1_b = [slab1_stop]';
slab1_p2_f = [slab1_start;lab1_stop]'; slab1_p2_b = [slab1_start;lab1_stop]';
slab2_p3_f = [slab2_stop + 1]'; slab2_p3_b = [slab2_stop]';
slab2_p4_f = [slab2_start;lab2_stop]'; slab2_p4_b = [slab2_start;lab2_stop]';

% Step 2: original field simulation
td_original = fdtd'id(dx, dt, number_of_time_steps, epsilonr, source_signal, excitation_domain);
slab1_response_p1_f = td_original(:,slab1_p1_f); slab1_response_p1_b = td_original(:,slab1_p1_b);
slab1_response_p2_f = td_original(:,slab1_p2_f); slab1_response_p2_b = td_original(:,slab1_p2_b);
slab2_response_p3_f = td_original(:,slab2_p3_f); slab2_response_p3_b = td_original(:,slab2_p3_b);
slab2_response_p4_f = td_original(:,slab2_p4_f); slab2_response_p4_b = td_original(:,slab2_p4_b);

% adjoint excitation is a delta in time
adjoint_excitation = zeros(1, number_of_time_steps);
adjoint_excitation(1,1) = 1/dt;

% Step 3: adjoint simulation
td_adjoint = fdtd'id(dx, dt, number_of_time_steps, epsilonr, adjoint_excitation, excitation_domain);
slab1_adjoint_response_p1_f = flipud(td_adjoint(:,slab1_p1_b)); % store adjoint responses
slab1_adjoint_response_p2_f = flipud(td_adjoint(:,slab1_p2_f));
slab2_adjoint_response_p3_f = flipud(td_adjoint(:,slab2_p3_b)); % store adjoint responses
slab2_adjoint_response_p4_f = flipud(td_adjoint(:,slab2_p4_f));

% Evaluate original residues
slab1_residue_p1_f = (epsilon_slab1 - epsilon_air) * epsilon_0 * diff(slab1_response_p1_f) / dt;
slab1_residue_p2_f = diff(slab1_response_p2_f) * d_epsilonr * epsilon_0 / dt;
slab2_residue_p3_f = (epsilon_slab2 - epsilon_air) * epsilon_0 * diff(slab2_response_p3_f) / dt;
slab2_residue_p4_f = diff(slab2_response_p4_f) * d_epsilonr * epsilon_0 / dt;

% estimate sensitivities
we shift the adjoint response at every time step by 1dt
for time_step=1:number_of_time_steps-1
% shift the adjoint response and add leading zeroes
slab1_adjoint_response_p1_shifted = [slab1_adjoint_response_p1_f(time_step:);number_of_time_steps)];
slab1_adjoint_response_p2_shifted = [slab1_adjoint_response_p2_f(time_step:);number_of_time_steps)];
slab2_adjoint_response_p3_shifted = [slab2_adjoint_response_p3_f(time_step:);number_of_time_steps)];
slab2_adjoint_response_p4_shifted = [slab2_adjoint_response_p4_f(time_step:);number_of_time_steps)];
% evaluate the sensitivities for the current time step. Only non zero components are multiplied
sensitivity_p1 = -sum(sum(slab1_residue_p1_f(1:number_of_time_steps-time_step,1)) * slab1_adjoint_response_p1_shifted(2:(number_of_time_steps-time_step+1),1)) / dz;
fields. The code then implements the shift formula (5.7) for all time steps \( \text{k} \). This shift is implemented for all four parameters by the commands

\[
slab1\text{\_adjoint\_response\_p1\_shifted}=\left[\text{slab1\_adjoint\_response\_p1\_f(time\_step:number\_of\_time\_steps)}\right] \ast \text{dt} / \text{d}\_\text{epsilon1};
\]

\[
slab1\text{\_adjoint\_response\_p2\_shifted}=\left[\text{slab1\_adjoint\_response\_p2\_f(time\_step:number\_of\_time\_steps,:)}\right] \ast \text{dt} / \text{d}\_\text{epsilon2};
\]

\[
slab2\text{\_adjoint\_response\_p3\_shifted}=\left[\text{slab2\_adjoint\_response\_p3\_f(time\_step:number\_of\_time\_steps)}\right] \ast \text{dt} / \text{d}\_\text{epsilon3};
\]

\[
slab2\text{\_adjoint\_response\_p4\_shifted}=\left[\text{slab2\_adjoint\_response\_p4\_f(time\_step:number\_of\_time\_steps,:)}\right] \ast \text{dt} / \text{d}\_\text{epsilon4};
\]

These commands shift the adjoint responses produced by the factor \( \text{time\_step} \) while inserting leading zeroes. Notice that for parameters \( p_1 \) and \( p_3 \), we store the adjoint fields at only one cell, while the adjoint field is stored at several cells for parameters \( p_2 \) and \( p_4 \). The program loops the time shift parameter \( \text{time\_step} \) to produce the adjoint responses corresponding to every time step. The shifted adjoint responses are multiplied by the original residues to estimate the sensitivities. The commands

\[
sensitivities(number\_of\_time\_steps-time\_step+1,1)=\text{sensitivity}\_p1;
\]

\[
sensitivities(number\_of\_time\_steps-time\_step+1,2)=\text{sensitivity}\_p2;
\]

\[
sensitivities(number\_of\_time\_steps-time\_step+1,3)=\text{sensitivity}\_p3;
\]

\[
sensitivities(number\_of\_time\_steps-time\_step+1,4)=\text{sensitivity}\_p4;
\]

store the calculated sensitivities. Notice that the adjoint response shifted by \( k \) time steps is used to evaluate the sensitivities of the original response at \( T_m - k\Delta t \). This is why the sensitivities are stored in an opposite direction. The matrix \textit{sensitivities} is used to store all sensitivities. It has \( N_T \) rows corresponding to all time steps and four columns corresponding to the four parameters. The output from this adjoint code is compared with that obtained using the FFD
method with perturbations \( \Delta \mathbf{p} = [\Delta z \ \Delta \varepsilon_r \ \Delta z \ \Delta \varepsilon_r]^T \) where \( \Delta z = 0.05e^{-3} \) and \( \Delta \varepsilon_r = 0.01 \). This comparison is shown in Figure 5.4. The transient sensitivities estimated using the adjoint method and using FFD show excellent agreement.

The reader may notice that because there are effectively 12,000 responses in this example, the postprocessing adjoint sensitivity calculations may create significant computational cost. This is indeed the case. This approach, like all adjoint sensitivity techniques, yields good efficiency as long as the computational overhead of the AVM is much smaller than the cost of one simulation. So, there is a threshold of calculations beyond which the AVM method will not yield much gain.

### 5.3 An alternative formulation

In the previous section, we introduced one possible approach to estimate the sensitivities of the complete transient response with respect to all parameters. The approach utilized successive shifting of the complete adjoint response in the \(+\)ve \( \tau \) time
coordinate. It focuses on evaluating the sensitivities of the original response at one time step at a time. Another approach shifts the focus to the adjoint response. Every sample of the adjoint response contributes to the sensitivities of a number of original time domain samples. In this case, every sample of the adjoint response is multiplied by a set of different original residues to contribute its share to the different sensitivities.

To illustrate this concept, consider Figure 5.5. This figure shows the first few adjoint response samples $\lambda_0(\tau)$ resulting from the impulsive adjoint excitation.

Figure 5.5 An illustration of the alternative approach for estimating the transient sensitivities: (a) the fifth adjoint response in the transient response $\lambda_0(\tau)$ multiplies the residue $R(N_T - 5)$ to contribute to the derivatives $\partial E(N_T)/\partial p$, (b) the fifth adjoint response (now appearing at $6\Delta t$) multiplies the residue $R(N_T - 6)$ to contribute to the derivatives $\partial E(N_T - \Delta t)/\partial p$, and (c) the fifth adjoint response (now appearing at $7\Delta t$) multiplies the residue $R(N_T - 7)$ to contribute to the derivatives $\partial E(N_T - 2\Delta t)/\partial p$. This fifth impulse contributes to all sensitivities up $\partial E(5\Delta t)/\partial p$.
with $\tau_1 = 0$. It shows also the same response shifted in the +ve $\tau$-direction by $1\Delta\tau$ and $2\Delta\tau$. If we consider one of the responses of $\lambda_0(\tau)$, say $\lambda_0(5\Delta\tau)$, we see that this value is multiplied by a certain original residue value to contribute to the sensitivities $\partial E(T_m)/\partial p$. In Figure 5.5(b), this response is shifted by $1\Delta\tau$. This same response appears now at the time step $6\Delta\tau$ and is thus multiplied by a different original residue to contribute to the sensitivities $\partial E(T_m - \Delta\tau)/\partial p$. In Figure 5.5(c), this response is shifted by $2\Delta\tau$. This same response appears now at the time step $7\Delta\tau$ and thus multiplies a different original residue to contribute to the sensitivities $\partial E(T_m - 2\Delta\tau)/\partial p$. This specific response contributes to the sensitivities of all time domain sensitivities from $\partial E(T_m)/\partial p$ up to $\partial E(5\Delta\tau)/\partial p$. It follows that once this fifth adjoint response $\lambda_0(5\Delta\tau)$ is available at the fifth time step of the adjoint simulation, we can estimate all its contributions to all transient sensitivities. There is no need to store the complete adjoint response or successively shift it.

It follows from this discussion that we do not have to shift the whole adjoint response as was done in Section 5.3. Rather, we focus on the adjoint response samples, one by one, as they become available during the adjoint simulation and calculate their contributions to all sensitivities at once. This approach is exactly equivalent to the shift approach discussed earlier but it is simpler in its implementation. The following example illustrates this approach.

**Example 5.3** Consider the 3D structure shown in Figure 5.6. This structure shows two lossy dielectric discontinuities illuminated by an electric current density filament. The resultant field is observed at another point in space. The whole structure is surrounded by a PML absorbing boundary condition.

The domain for this problem has a size of $30\Delta l \times 30\Delta l \times 40\Delta l$, with $\Delta l = 3.0$ mm. The electric current source excites a wideband signal with a center

![Figure 5.6](image-url)
frequency of 2.0 GHz with a 1.0 GHz bandwidth. It is required in this example to evaluate the sensitivities of the response \( E_z(t) \) at the observation point with respect to the parameters \( p = [\varepsilon_1 \sigma_1 x_{\text{min}1} x_{\text{max}1} y_{\text{min}1} y_{\text{max}1} z_{\text{min}1} \\
\varepsilon_2 \sigma_2 x_{\text{min}2} x_{\text{max}2} y_{\text{min}2} y_{\text{max}2} z_{\text{min}2} z_{\text{max}2}]^T \). The adjoint sensitivities are estimated at the nominal parameter values \( p_o = [1.55 \ 0.01 \ \text{S/m} \ 0.151 \ \text{m} \ 0.160 \ \text{m} \ 0.151 \ \text{m} \ 0.130 \ \text{m} \ 0.160 \ \text{m} \ 0.151 \ \text{m} \ 0.160 \ \text{m} \ 0.205 \ \text{m}]^T \). There are thus 16 parameters that cover the material properties and dimensions of the two lossy dielectric discontinuities.

The set up files for this problem are similar to the ones used in Example 3.3 in Chapter 3. The TestAVM script invokes the original and adjoint simulations. During this original simulation, the temporal residues corresponding to all parameters are calculated and stored. The main adjoint script fDTD Solve adjoint is also identical to the one utilized in Chapter 3. A main difference exists in the script initialize_adjoint_domain, which initializes the adjoint sources. The commands

```plaintext
excitation_waveform=zeros(number_of_time_steps,1);
excitation_waveform(1,1)=-1/dt;
number_of_current_density_sources=1;
```

allocate memory for the adjoint excitation waveform and defines this single source located at the observation point to be impulsive with a value of \((-1/\Delta t)\) appearing at the first time step. This current density source is polarized in the \( z \)-direction. The adjoint simulation utilizes the same material distribution and boundary conditions as the original one. The second main difference exists in the way the adjoint responses are handled as they become available. The script update_adjoint_product, which is called at every time step of the adjoint simulation, is shown in MATLAB Listing M5.4. This script calculates all the contributions made to all the transient sensitivities by the current adjoint response and stores them in the proper storage. The loop

```plaintext
index=number_of_time_steps;
for residue_index=time_step:number_of_time_steps
    residue_time_index=number_of_time_steps-residue_index+1;
    residue_Ex=residue_sampled_field(parameter_index).
    residue_Ex(residue_time_index,:);
    residue_Ey=residue_sampled_field(parameter_index).
    residue_Ey(residue_time_index,:);
    residue_Ez=residue_sampled_field(parameter_index).
    residue_Ez(residue_time_index,:);
    sensitivities(index,parameter_index)=
    sensitivities(index,parameter_index)-
    dt*'(residue_Ex*current_Ex_adjoint')+
    (residue_Ey*current_Ey_adjoint')+
    (residue_Ez*current_Ez_adjoint');
    index=index-1;
end
```
MATLAB Listing M5.4

% MATLAB listing M5.4
% This script updates the contributions to all transient sensitivities
% using the adjoint response available at the current time step
for j=2:numberOfBricks % all bricks apart from the air background
    for i=1:8 % repeat for all 8 parameters associated with the jth brick
        parameter_index=(j-2)*8+1;
        % storage volume for current parameter
        min_x=adjoint_sampled_field(parameter_index).min_x; % x start
        max_x=adjoint_sampled_field(parameter_index).max_x; % x end
        min_y=adjoint_sampled_field(parameter_index).min_y; % y start
        max_y=adjoint_sampled_field(parameter_index).max_y; % y end
        min_z=adjoint_sampled_field(parameter_index).min_z; % z start
        max_z=adjoint_sampled_field(parameter_index).max_z; % z end
        Ex_max_x=max_x+adjoint_shifts(1,4); % get number of stored Ex components
        Ex_max_y=max_y+adjoint_shifts(1,8); % get number of stored Ex components
        Ey_max_x=max_x+adjoint_shifts(1,5); % get number of stored Ey components
        Ey_max_y=max_y+adjoint_shifts(1,9); % get number of stored Ey components
        Ez_max_x=max_x+adjoint_shifts(1,2); % get number of stored Ez components
        Ez_max_y=max_y+adjoint_shifts(1,6); % get number of stored Ez components
        vector_length_x=(max_x-min_x+1)*(Ex_max_x-min_y+1)*
        (Ex_max_z-min_z+1); % total number of stored components
        vector_length_y=(max_y-min_y+1)*(Ey_max_x-min_x+1)*
        (Ey_max_z-min_z+1); % total number of stored components
        vector_length_z=(max_z-min_z+1)*(Ez_max_x-min_x+1)*
        (Ez_max_y-min_y+1); % total number of stored components
        % get current value of adjoint fields
        current_Ex_adjoint=reshape(Ex(min_x:max_x, min_y:max_y, min_z:max_z),1,vector_length_x);
        current_Ey_adjoint=reshape(Ey(min_x:max_x, min_y:max_y, min_z:max_z),1,vector_length_y);
        current_Ez_adjoint=reshape(Ez(min_x:max_x, min_y:max_y, min_z:max_z),1,vector_length_z);
        % the adjoint response at the current time step contributes to the
        % sensitivities of many time responses.
        % we thus create a loop to create all relevant contributions
        index=number_of_time_steps;
        % repeat for all applicable residues
        for residue_index=time_step:number_of_time_steps
            residue_time_index=number_of_time_steps-residue_index+1;
            residue_Ex=adjoint_sampled_field(parameter_index, residue_time_index,:);
            residue_Ey=adjoint_sampled_field(parameter_index, residue_time_index,:);
            residue_Ez=adjoint_sampled_field(parameter_index, residue_time_index,:);
            % now carry out inner product and store the sensitivities in the
            % proper time index
            sensitivities(index,parameter_index)=
            dot((residue_Ex*current_Ex_adjoint)+(residue_Ey*current_Ey_adjoint)+(residue_Ez*current_Ez_adjoint));
            index=index+1;
        end
    end
end
multiplies the current adjoint response, defined by the three variables `current_Ex_adjoint`, `current_Ey_adjoint`, and `current_Ez_adjoint`, by relevant residues and store these contributions. The parameter `index` determines the index of the sensitivities in which the contribution is stored.

The results for all parameters are shown in Figures 5.7–5.9. Good match is observed for most parameters. Some difference is observed in some time steps for certain parameters. This is attributed to the nonlinearity of the observed probe field at certain time steps.

Figure 5.7 The sensitivities of the response of Example 5.3 with respect to the material parameters of the two discontinuities
Figure 5.8 The sensitivities of $E_z(t)$ at the probe node with respect to all the geometrical parameters of the first discontinuity: ($\times$) adjoint sensitivities versus FFD (−)
Figure 5.9  The sensitivities of $E_z(t)$ at the probe node with respect to all the geometrical parameters of the second discontinuity; ($\times$) adjoint sensitivities versus FFD (−)
References


Chapter 6

Adjoint sensitivity analysis with dispersive materials

In the previous chapters, we developed and demonstrated adjoint sensitivity analysis techniques that apply for different types of problems. We first handled the case of a simple scalar objective function that is routinely addressed in optimization algorithms. We then addressed frequency-dependent responses such as the $S$-parameters. In Chapter 5, we showed how to estimate the sensitivities of a complete transient response. In all these cases, we demonstrated that only one adjoint simulation is sufficient to estimate the sensitivities over the band of interest with respect to all parameters. We showed also that for certain objective functions, there is no need for this adjoint simulation.

In all the previously discussed cases, we assumed that the materials utilized in the FDTD simulations are nondispersive. This means that all material properties (permittivity, permeability, and conductivity) do not change with frequency. In many interesting applications, however, this is not the case. For example, in the emerging area of metamaterials, the effective permittivity and permeability show strong dependency on frequency [1]. In the area of plasmonics, all metals have dispersive properties [2–5]. The same applies to modeling materials in the THz and infrared frequency regimes [6,7]. It is thus of prime importance to be able to estimate the sensitivities of different responses with respect to all geometrical and material parameters of structures with dispersive material properties.

In this chapter, we develop a general theory for adjoint sensitivity analysis of high frequency dispersive structures. This formulation applies to materials with commonly used types of dispersion profiles such as Lorentz [8,9], Drude [10–12], and Debye [13,14]. We show that only one dispersive adjoint simulation is required to estimate the sensitivities of the desired response with respect to all parameters. We illustrate this approach through one example.

### 6.1 The general dispersive material case

In this section, we derive a general formulation that applies to most dispersive profiles. For an isotropic and dispersive medium, the governing Maxwell’s equations are given by

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} + \sigma \mathbf{E} + \mathbf{J}_\text{i}, \tag{6.1}$$
To simplify the derivation, and without loss of generality, we ignore the magnetic conductivity and the magnetic current density. Notice that in (6.1), the displacement vector $D$ is not linearly proportional to $E$ as is the case of nondispersive media. For a dispersive medium, the current value of $D$ depends on the history of $E$. We assume that the relationship between the vectors $D$ and $E$ is given by

$$D = \varepsilon E + \int_0^t S(t - \tau) E(\tau) d\tau.$$  

Because we still assume that the material is isotropic, the permittivity matrix $\varepsilon$ and the convolution matrix $S$ are both diagonal matrices. The components of the matrix $\varepsilon$ are time invariant, while the diagonal components of the matrix $S$ are time-dependent. We will later show that different dispersion profiles satisfy (6.3) using different definitions of the time-dependent matrix $S$.

There are several approaches to derive the adjoint system corresponding to (6.1)–(6.3). The FDTD-based approach suggested in [15] utilizes the wave equation. Here, we opt to utilize an alternative formulation that is based on Maxwell’s equations [16]. Substituting from (6.3) into (6.1), we have

$$\nabla \times H = \varepsilon \frac{\partial E}{\partial t} + \sigma^e E + \frac{\partial}{\partial t} (S(t) * E(t)) + J_i$$  

$$\nabla \times E = -\frac{\partial B}{\partial t} = -\mu \frac{\partial H}{\partial t},$$  

where the operator (*) denotes time convolution. Utilizing the properties of the convolution integral, we have [17]:

$$\frac{\partial}{\partial t} (S(t) * E(t)) = \frac{\partial S(t)}{\partial t} * E(t) = C(t) * E(t).$$  

It follows that Maxwell’s equations given by (6.4) and (6.5) can be written for the whole computational domain as

$$\begin{bmatrix} \varepsilon & 0 \\ 0 & \mu \end{bmatrix} \begin{bmatrix} \dot{E} \\ \dot{H} \end{bmatrix} + \begin{bmatrix} \sigma^e & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} E \\ H \end{bmatrix} + \begin{bmatrix} -\nabla \times H \\ \nabla \times E \end{bmatrix} + \begin{bmatrix} C(t) & 0 \\ 0 & 0 \end{bmatrix} * \begin{bmatrix} E \\ H \end{bmatrix} = -\begin{bmatrix} J_i \\ 0 \end{bmatrix}.$$  

The system (6.7) can be put in the standard form

$$N \dot{V} + K V + P \ast V = q,$$  

where $V = [E^T \ H^T]^T$. To derive the adjoint system corresponding to (6.8), we follow a similar approach to that adopted in Chapter 3 for the nondispersive case.
We perturb the $i$th parameter by a small perturbation $\Delta p_i$. This perturbation results in perturbations in all the system matrices in (6.8). The system solution $V$ is also perturbed. This results in the new perturbed original simulation:

$$(N + \Delta N_i)(\dot{V} + \Delta \dot{V}_i) + (K + \Delta K_i)(V + \Delta V) + (P + \Delta P_i) \ast (V + \Delta V) = q.$$  

(6.9)

Subtracting (6.8) from (6.9) and reorganizing, we have

$$(N + \Delta N_i)\Delta \dot{V}_i + (K + \Delta K_i)\Delta V + (P + \Delta P_i) \ast \Delta V = -\Delta N_i \dot{V} - \Delta K_i V - \Delta P_i \ast V.$$  

(6.10)

The right-hand side of (6.10) is known because the fields resulting from the original simulation (6.8) are known. The perturbations in the system matrices corresponding to the perturbation $\Delta p_i$ are also known. Without loss of generality, we assume that the parameter $p_i$ is a material parameter (permittivity, conductivity, or dispersion profile parameter). This implies that all the system matrices are analytical functions of the parameter. A similar derivation can be carried out for shape parameters. Dividing both sides of (6.10) by $\Delta p_i$ and taking the limit as $\Delta p_i \to 0$, one gets

$$N \frac{d\dot{V}}{dp_i} + K \frac{dV}{dp_i} + P \ast \frac{dV}{dp_i} = -R_i,$$  

(6.11)

where the residue $R_i$ corresponding to the $i$th parameter is given by

$$R_i = \frac{dN}{dp_i} \dot{V} + \frac{dK}{dp_i} V + \frac{dP}{dp_i} \ast V.$$  

(6.12)

This temporal residue is evaluated for every considered parameter at all-time steps during the original simulation. The derivation continues in a manner similar to that utilized in Chapter 3. We multiply both sides of (6.11) by the yet-to-be determined temporal adjoint vector $\lambda$ and integrate over the simulation time to get

$$\int_0^{T_m} \lambda^T \left( N \frac{d\dot{V}}{dp_i} + K \frac{dV}{dp_i} + P \ast \frac{dV}{dp_i} \right) dt = -\int_0^{T_m} \lambda^T R_i dt.$$  

(6.13)

The first and second terms on the left-hand side of (6.13) are handled in the same way as in Chapter 3. The first term is integrated by parts and the adjoint variable is set to have $\lambda(T_m) = 0$. The convolution term in the left-hand side of (6.13) requires more explanation. This term may be written as

$$\int_0^{T_m} \lambda^T P \ast \frac{dV}{dp_i} dt = \int_0^{T_m} \lambda^T (t) \left( \int_0^t P(t - \nu) \frac{dV}{dp_i} dv \right) dt.$$  

(6.14)
The causality of the system imposes the condition that the convolution matrix \( P \) has zero components for any negative argument. It follows that the upper limit in the integral of (6.14) can be changed to

\[
\int_0^{T_m} \mathbf{\lambda}^T \left( P \ast \frac{dV}{dp_i} \right) dt = \int_0^{T_m} \mathbf{\lambda}^T(t) \left( \int_0^{T_m} P(t - \nu) \frac{dV(\nu)}{dp_i} d\nu \right) dt. \tag{6.15}
\]

Switching the order of the two integrations in (6.15), we get

\[
\int_0^{T_m} \mathbf{\lambda}^T \left( P \ast \frac{dV}{dp_i} \right) dt = \int_0^{T_m} \frac{dV(\nu)^T}{dp_i} \left( \int_0^{T_m} P^T(t - \nu) \mathbf{\lambda}(t) dt \right) d\nu. \tag{6.16}
\]

As the system matrix \( P \) is both symmetric and causal, (6.16) can be written in the form

\[
\int_0^{T_m} \mathbf{\lambda}^T \left( P \ast \frac{dV}{dp_i} \right) dt = \int_0^{T_m} \frac{dV(\nu)^T}{dp_i} \left( \int_0^{T_m} P^T(t - \nu) \mathbf{\lambda}(t) dt \right) d\nu. \tag{6.17}
\]

The integral

\[
\int_0^{T_m} P(t - \nu) \mathbf{\lambda}(t) dt \tag{6.18}
\]

evaluates the convolution part of the adjoint system. The value of the integral (6.18) at a time \( \nu \) depends on all future values of \( \mathbf{\lambda} \) \(( \nu \leq t \leq T_m \)). This is expected because the adjoint system runs backward in time. Equation (6.18) represents the same convolution operation of the original simulation (6.8) but implemented in a reversed time. Combining (6.18) and (6.13) and comparing to the integral of the objective function (3.2), we obtain the adjoint system:

\[
N \dot{\mathbf{\lambda}} + K \mathbf{\lambda} + P \ast \mathbf{\lambda} = \frac{\partial F}{\partial V}, \tag{6.19}
\]

where the time derivative in (6.19) is with respect to the adjoint time variable \( \tau \). The convolution integral is also with respect to this time variable. Comparing (6.19) and (6.8), it is obvious that the adjoint system is governed by the same equations governing the original system. The only difference is in the excitation. Also, while (6.8) is running forward in time with respect to \( t \), (6.19) is running backward in time with respect to \( \tau \) (forward with respect to \( t \)). It follows that the same FDTD solver used to solve the original system with respect to \( t \) can also be used to solve the adjoint system with respect to \( \tau \). Once the adjoint fields \( \mathbf{\lambda} \) are determined by solving (6.19), the adjoint sensitivities are given by

\[
\frac{\partial F}{\partial p_i} = -\int_0^{T_m} \mathbf{\lambda}^T R_i dt \approx \Delta t \sum_k \mathbf{\lambda}^T(k\Delta t) R_i(k\Delta t). \tag{6.20}
\]

It follows from the previous discussion that regardless of the dispersion profile, the sensitivities of the objective function with respect to all parameters are estimated
using only the response of the original simulation (6.8) and the response of the
adjoint simulation (6.19).

The previous discussion is applicable for any dispersive material governed by
the relationship (6.3). We show in the following subsections how (6.3) applies to
most popular dispersion profiles. Because we assume in our derivation that the
material is isotropic, we will focus on deriving the dispersion expressions for only
one field component.

6.1.1 The Lorentz model

The Lorentz dispersion model relates the displacement vector with the electric field
through a frequency-dependent relationship. For only one field component, this
relationship is given by [8,9]:

\[
D(s) = \varepsilon E(s) + \sum_{m=1}^{N_p} \frac{\varepsilon_{p,m} \omega_{p,m}^2}{s^2 + 2 \zeta_m s + \omega_{p,m}^2} E(s),
\]

where \(s\) is the complex frequency. The parameter \(\varepsilon\) is the high-frequency permit-
tivity. The parameters \(\zeta_m, \omega_{p,m},\) and \(\varepsilon_{p,m}\) are the damping frequency, the resonant
frequency, and the relative strength of the \(m\)th Lorentz pole, respectively.

Assuming that \(\omega_{p,m} > \zeta_m\) and defining \(\omega_d^2 = \omega_{p,m}^2 - \zeta_m^2\), the dispersion relation-
ship (6.21) can be rewritten as

\[
D(s) = \varepsilon E(s) + \sum_{m=1}^{N_p} \frac{\varepsilon_{p,m} \omega_{p,m}^2}{(s + \zeta_m)^2 + \omega_d^2} E(s).
\]

(6.22)

Taking the inverse Laplace transform of (6.22), we have

\[
D(t) = \varepsilon E(t) + \left( \sum_{m=1}^{N_p} S_m(t) \right) \ast E(t) = \varepsilon E(t) + S(t)E(t),
\]

(6.23)

where the \(m\)th convolution function \(S_m(t)\) is given by

\[
S_m(t) = \frac{\varepsilon_{p,m} \omega_{p,m}^2}{\omega_d} \sin(\omega_d t) \exp(\zeta_m t).
\]

(6.24)

Comparing (6.23) and (6.24) to (6.3), we see that the convolution matrix \(S(t)\) is a
diagonal matrix with equal components given by the sum of all functions \(S_m(t)\).

The derivative of the convolution function is given by

\[
C(t) = \frac{dS(t)}{dt} = \frac{d}{dt} \sum_{m=1}^{N_p} S_m(t)
\]

\[
= \sum_{m=1}^{N_p} \varepsilon_{p,m} \omega_{p,m}^2 \left( \cos(\omega_d t) \exp(-\zeta_m t) - \frac{\zeta_m}{\omega_d} \sin(\omega_d t) \exp(-\zeta_m t) \right).
\]

(6.25)
The function $C(t)$ can be calculated at any time once the parameters of the Lorentz poles are defined. It can be shown that the Laplace transform of (6.25) is given by

$$\tilde{C}(s) = \sum_{m=1}^{N_p} \frac{\varepsilon_{p,m} \omega_{p,m}^2 s}{(s + \tau_m)^2 + \omega_{d,m}^2}.$$  \hspace{1cm} (6.26)

As we will see later, it is more efficient to evaluate the convolution (6.6) in the frequency domain and then apply inverse Fourier transform to transform the result back in the time domain.

### 6.1.2 The Drude model

Another useful dispersion model is the Drude model. This model finds applications in the modeling of plasmonic and metamaterial structures [10–12]. The governing equation of the Drude model for a single field component is given by

$$D(s) = \varepsilon E(s) + \sum_{m=1}^{N_p} \frac{\varepsilon_{0} \omega_{p,m}^2}{s^2 + s/\tau_{c,m}} E(s),$$  \hspace{1cm} (6.27)

where $\tau_{c,m}$ and $\omega_{p,m}$ are the collision time constant and the plasma frequency of the $m$th Drude term, respectively. Following a similar derivation to the one given by (6.23)–(6.26), we obtain the following equation relating the displacement vector $D$ and the electric field $E$:

$$D(t) = \varepsilon E(t) + \left( \sum_{m=1}^{N_p} S_m(t) \right) * E(t),$$  \hspace{1cm} (6.28)

where the $m$th time-dependent convolution term $S_m(t)$ is given by

$$S_m(t) = \varepsilon_{0} \tau_{c,m} \omega_{p,m}^2 \left( 1 - \exp \left( \frac{-t}{\tau_{c,m}} \right) \right) u(t).$$  \hspace{1cm} (6.29)

It follows that the matrix $S$ is a diagonal matrix with equal components that are all given by the sum of convolution terms in (6.28). The corresponding function $C(t)$ is given by

$$C(t) = \frac{dS(t)}{dt} = \frac{d}{dt} \sum_{m=1}^{N_p} S_m(t) = \sum_{m=1}^{N_p} \varepsilon_{0} \omega_{p,m}^2 e^{-t/\tau_{c,m}}.$$  \hspace{1cm} (6.30)

### 6.1.3 The Debye model

Electromagnetic behavior of biological tissues in the microwave frequency range follows the Debye dispersion model [13,14] given by the relationship:

$$D(s) = \varepsilon E(s) + \sum_{m=1}^{N_p} \frac{\varepsilon_{0} \Delta \chi_{c,m}}{1 + s \tau_{c,m}} E(s),$$  \hspace{1cm} (6.31)
where $\tau_{e,m}$ is the dielectric relaxation time and $\varepsilon$ is the high-frequency permittivity. $\Delta \chi_{e,m}$ is the difference between the DC and the high-frequency dielectric susceptibility. Following a similar derivation to that given in (6.23)–(6.26), we achieve the following time domain equation:

$$D(t) = \varepsilon E(t) + \sum_{m=1}^{N_p} \frac{\varepsilon_0 \Delta \chi_{e,m}}{\tau_{e,m}} e^{-t/\tau_{e,m}} E(t).$$

(6.32)

The corresponding derivative of the convolution function is given by

$$C(t) = \frac{dS(t)}{dt} = \frac{d}{dt} \sum_{m=1}^{N_p} S_m(t) = -\sum_{m=1}^{N_p} \frac{\varepsilon_0 \Delta \chi_{e,m}}{\tau_{e,m}^2} e^{-t/\tau_{e,m}}.$$  

(6.33)

### 6.2 Implementation

The implementation of the adjoint variable method (AVM) for structures with dispersive materials follows essentially the same steps of all AVM algorithms with some changes. The first step is to carry out the original simulation (6.8) to determine the original response $V = [E' \ H']^T$ for all time steps. The auxiliary differential equation (ADE) method discussed in Chapter 2 is utilized to solve this system. At the $k\text{th}$ step of the original simulation, we evaluate the residues corresponding to all parameters. The residue corresponding to the $i\text{th}$ parameter at the $k\text{th}$ time step is given by

$$R_i(k\Delta t) = \frac{1}{\Delta p_i} \left( \Delta N_i \dot{V}(k\Delta t) + \Delta K_i V(k\Delta t) + \Delta P_i * V \right), i = 1, 2, \ldots, N_T.$$  

(6.34)

If the considered parameter is a material parameter, then all matrices are analytical functions of this parameter. As explained earlier, the perturbation $\Delta p_i$ can be made arbitrarily small to obtain the residue expression

$$R_i(k\Delta t) = \left( \frac{\partial N_i}{\partial p_i} \dot{V}(k\Delta t) + \frac{\partial K_i}{\partial p_i} V(k\Delta t) + \frac{\partial P_i}{\partial p_i} * V \right), i = 1, 2, \ldots, N_T.$$  

(6.35)

The stamp matrices $\Delta N_i$ and $\Delta K_i$ are constructed similar to the isotropic case addressed in Chapter 3. The stamp matrix $\Delta P_i$ is defined by

$$\Delta P_i = P(p + \Delta p_i e_i) - P(p), i = 1, 2, \ldots, n,$$  

(6.36)

where $e_i$ is a vector of zeros with only 1 in the $i\text{th}$ component. It follows that the stamp corresponding to the convolution matrix involves perturbing the convolution functions with respect to all the parameters one by one using (6.36) and then convoluting the perturbed convolution function with the field using (6.34).

To understand how the stamp matrix $\Delta P_i$ is constructed, consider Figure 6.1. This figure shows an electromagnetic structure with a dispersive discontinuity. Every relevant field component within the discontinuity volume is convoluted with the appropriate convolution function. If any of the material parameters is
perturbed, the nonzero diagonal components of $\Delta P_i$ are affected. Because the matrix $P_i$ is an analytical function of the material parameters, we can write

$$
\lim_{\Delta p_i \to 0} \Delta P_i = \frac{\partial P}{\partial p_i}. 
$$

(6.37)

Because the convolution matrix $P$ is a diagonal matrix, differentiating it with respect to the material parameters is equivalent to differentiating its nonzero components. For example, for the Lorentz dispersion profile, the stamps of the convolution matrix are diagonal matrices whose nonzero components are given by

$$
\frac{\partial C(t)}{\partial \varepsilon_r} = \sum_{m=1}^{N_p} \varepsilon_o \omega_{p,m}^2 \left( \cos(\omega_{d,m} t) \exp(-\zeta_m t) - \frac{\zeta_m}{\omega_{d,m}} \sin(\omega_{d,m} t) \exp(-\zeta_m t) \right),
$$

(6.38)

$$
\frac{\partial C(t)}{\partial \sigma^e} = 0,
$$

(6.39)

$$
\frac{\partial C(t)}{\partial \varepsilon_{s,m}} = \sum_{m=1}^{N_p} \varepsilon_o \omega_{p,m}^2 \left( \cos(\omega_{d,m} t) \exp(-\zeta_m t) - \frac{\zeta_m}{\omega_{d,m}} \sin(\omega_{d,m} t) \exp(-\zeta_m t) \right),
$$

(6.40)

$$
\frac{\partial C(t)}{\partial \omega_{d,m}} = \sum_{m=1}^{N_p} \varepsilon_{p,m} \omega_{p,m}^2 \left( \frac{\zeta_m}{\omega_{d,m}^2} \sin(\omega_{d,m} t) \exp(-\zeta_m t) - \frac{\zeta_m}{\omega_{d,m}} \sin(\omega_{d,m} t) \exp(-\zeta_m t) \right)
- \zeta_m t \cos(\omega_{d,m} t) \exp(-\zeta_m t)
+ 2 \varepsilon_{p,m} \omega_{d,m} \left( \cos(\omega_{d,m} t) \exp(-\zeta_m t) - \frac{\zeta_m}{\omega_{d,m}} \sin(\omega_{d,m} t) \exp(-\zeta_m t) \right).
$$

(6.41)
\[
\frac{\partial C(t)}{\partial \zeta_{d,m}} = \sum_{m=1}^{N_o} \varepsilon_{p,m} \omega_{p,m}^2 \left( -t \cos(\omega_{d,m} t) \exp(-\zeta_{m} t) - \frac{1}{\omega_{d,m}} \sin(\omega_{d,m} t) \exp(-\zeta_{m} t) \right) \\
+ \frac{\zeta_{m}}{\omega_{d,m}} t \sin(\omega_{d,m} t) \exp(-\zeta_{m} t) \\
+ 2\varepsilon_{p,m} \zeta_{m} \left( \cos(\omega_{d,m} t) \exp(-\zeta_{m} t) - \frac{\zeta_{m}}{\omega_{d,m}} \sin(\omega_{d,m} t) \exp(-\zeta_{m} t) \right),
\]

(6.42)

where in deriving all these expressions we utilized the relationships \( \varepsilon_{p,m} = \varepsilon_o (\varepsilon_{s,m} - \varepsilon_r) \) and \( \omega_{p,m}^2 = \omega_{d,m}^2 + \zeta_{m}^2 \). Notice that in differentiating \( P \) with respect to material parameters, the zero diagonal components of \( P \) remains zero. Only the field components within a dispersive discontinuity are affected.

For shape parameters, the situation is different. Figure 6.2 illustrates this difference. For the original structures, only field components indicated with an \( x \) are associated with dispersive material properties. It follows that an integral of the form (6.6) is carried out for all these field components. When the parameter \( L \) is perturbed by one cell, a new column set of fields marked with an “\( o \)” is within the perturbed discontinuity. In the original structure, these components were surrounded by air and thus their corresponding convolution function \( C(t) \) was zero. For the perturbed structure, these field components are a part of the dispersive discontinuity and their convolution function is nonzero. This convolution function follows the dispersion profile of the discontinuity. It follows that the perturbation in the corresponding convolution matrix for these components is given by

\[
\Delta C_i = C(p + \Delta p_i e_i) - 0 = C(t).
\]

(6.43)

This convolution function is then used to estimate the residue (6.34). A similar expression can be derived for all other shape-related parameters.

---

Figure 6.2 The field components inside a dispersive continuity; a perturbation in the length by \( 1\Delta l \) affects only the convolution functions of the field components marked with \( o \). The convolution function for the field components inside the original discontinuity are not affected.
The algorithm for estimating adjoint sensitivities of 3D structures with dispersive materials follows the same steps as the one explained in Section 3.4. The main difference is the evaluation of the residue terms as it involves a convolution integral as stated in (6.34). There are two possible approaches to evaluate these convolution integrals. The first one is to carry out the convolution in the time domain. For example, for a perturbation \( \Delta p_i \) in the \( i \)th parameter, a typical component of the vector convolution may be evaluated through the summation:

\[
\Delta C_i(t) * E(t) = \int_0^{t'} \Delta C_i(t - \tau) E(\tau) d\tau \approx \Delta t \sum_{k=1}^{k'} \Delta C_i(k' \Delta t - k\Delta t) E(k\Delta t) \tag{6.44}
\]

where this convolution integral is evaluated at the time instant \( t' = k\Delta t' \). Evaluating this integral at every time step for all affected field components involves significant computational overhead. The number of field components affected by changing the dielectric constant of a discontinuity may be large. Evaluating (6.44) for each one of these links would be prohibitive and would kill the benefit of utilizing adjoint sensitivities. An easier and more computationally efficient approach is to evaluate the convolution (6.44) in the frequency domain. The Fourier transform converts the convolution integral to a multiplication. The Fourier transform of (6.44) is thus given by

\[
\tilde{I}(f) = \Im(\Delta C_i(t) * E(t)) = \tilde{\Delta C_i}(f) \tilde{E}(f) \tag{6.45}
\]

where \( \tilde{\Delta C_i}(f) \) and \( \tilde{E}(f) \) are the Fourier transforms of \( \Delta C_i(t) \) and \( E(t) \), respectively. The Fourier transforms of the convolution functions are available analytically. The Fourier transform of the considered field component is evaluated numerically. Usually, we need to evaluate the product (6.45) for only a small subset of frequencies \( \{f_m, m = 1, 2, \ldots, N_f\} \). The convolution in (6.34) is evaluated by taking the inverse Fourier transform of (6.45) to get

\[
I(t) = \int_{-\infty}^{\infty} \tilde{I}(f)e^{2\pi if} df \approx 2\Delta f \sum_{m=1}^{N_f} |\tilde{I}(f_m)| \cos(2\pi f_m t + \angle \tilde{I}(f_m)), \tag{6.46}
\]

where \( \tilde{I}(f_m) = |\tilde{I}(f_m)| \angle \tilde{I}(f_m) \). It follows that we only need to evaluate the spectrum of the utilized fields and the convolution functions at a small set of frequencies. The inverse DFT (6.46) can thus be evaluated more efficiently than evaluating (6.44). Notice that (6.44) is to be updated at every time step, while (6.46) is evaluated only once after all spectra become available after the original simulation is carried out.

**Example 6.1** We verify the dispersive AVM approach through the dispersive slab in a parallel-plate waveguide shown in Figure 6.3. In this example, the thin layer is a dispersive 3-pole Lorentz medium. The values of the parameters of these poles are shown in Table 6.1. It is required in this 2D example to evaluate the sensitivities of the transmission \( S_{21} \) over the frequency band (20–80 GHz) with
respect to the parameters \( p = [\varepsilon_r, \sigma^e, \varepsilon_{s,1}, \omega_{d,1}, \zeta_1, \varepsilon_{s,2}, \omega_{d,2}, \zeta_2, \varepsilon_{s,3}, \omega_{d,3}, \zeta_3, x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}]^T \). There are thus 15 parameters in this example. The parameters \( x_{\text{min}} \) and \( x_{\text{max}} \) are redundant for this example as the discontinuity extends throughout the whole cross section of the waveguide. The waveguide has a width of 3.75 mm with a square cell size of \( \Delta l = 75.00 \) \( \mu \text{m} \). The domain size is 50 \( \Delta l \times 100 \Delta l \) in addition to two PML absorbing boundaries at the start and end of the waveguide. The structure is excited with a wide-band Gaussian pulse centered at 50 GHz with an excited spectrum extending between 20 and 80 GHz. The dispersive Lorentzian slab has a width \( W = y_{\text{max}} - y_{\text{min}} = 60 \Delta l - 40 \Delta l = 1.5 \) mm.

The implementation of the dispersive AVM approach follows the same structure as the examples addressed in previous chapters. The TestAVM script invokes both the original and the adjoint simulations. The script ftdtd_solve_original, shown in MATLAB\textsuperscript{\textregistered} Listing M6.1, runs the original simulation and prepares the residues of all parameters. The script define_problem_
space_parameters_2d defines the materials of the problem. In particular, the parameters of the dispersive material are defined by the commands

\begin{verbatim}
% lorentz material
material_types(4).eps_r = 1.5;  % eps_inf
material_types(4).mu_r = 1;
material_types(4).sigma_e = 0;
material_types(4).sigma_m = 0;
material_types(4).color = [0 0 1];
material_types(4).lorentz_dispersive = true;
material_types(4).number_of_lorentz_poles = 3;
material_types(4).lorentz(1).eps_s = 2.10;
material_types(4).lorentz(1).A = 1;
material_types(4).lorentz(1).omega = 2*pi*30.0e9;
material_types(4).lorentz(1).delta = 2*pi*3.0e9;
material_types(4).lorentz(2).eps_s = 1.8;
material_types(4).lorentz(2).A = 1;
material_types(4).lorentz(2).omega = 2*pi*40e9;
material_types(4).lorentz(2).delta = 2*pi*8e9;
material_types(4).lorentz(3).eps_s = 2.4;
\end{verbatim}
The flag `material_types(4).lorentz_dispersive` identifies the material to be dispersive or nondispersive. For each dispersive material, we store $\varepsilon_s$, $\omega_p$, and $\zeta$ for every pole. The field $A$ defines the amplitude of the pole and is set to one for all poles in our implementation.

The MATLAB script `define_geometry_2d` defines the geometry of the problem. Two rectangles are defined. The first rectangle represents the surrounding air, while the second one represents the dispersive material. The left and right boundaries are terminated by perfect magnetic walls. The two remaining boundaries are terminated by a PML.

The script `define_sources_2d` defines the sources in the domain. Because we want to excite the dominant mode, which has a constant field profile in the cross section, a line of current density sources of equal amplitude is utilized across the cross section of the waveguide. This line of sources is at $y = 5 \Delta l$. These sources have a Gaussian-modulated temporal dependence. The script `define_output_parameters_2d` defines the output parameters. Because we want to determine the $S$-parameters, the field is stored along two lines at $y = 10 \Delta l$ and $y = 90 \Delta l$. The first line of field values is used to determine the reflected spectrum, while the second line of field values is used to determine the transmitted spectrum. The field values along these lines are multiplied by the modal profile to determine the modal amplitude at every time step. Discrete Fourier Transform is then used to determine the reflected and the transmitted spectra. Figure 6.3 shows the excitation line, the input observation line, and the output observation line. Figure 6.4 shows the absolute value of spectrum of the excited wave.

![Figure 6.4 The spectrum of the excitation of the Lorentzian slab example](image)
The transmission coefficient ($|S_{21}|$) is shown in Figure 6.5 over the frequency range from 20 to 80 GHz.

The script `initialize_updating_coefficients_2d`, as in previous chapters, initializes the coefficients utilized in updating the different field components. In addition to the update coefficients, this script also initializes the coefficients utilized in integrating the dispersion storage as explained in Chapter 2. This is carried out by calling the script `initialize_dispersive_coefficients` shown in MATLAB Listing M6.2. This script loops over all Lorentz materials. Using the indices of the field components within the dispersive media (with indices `lorentz_material_index.Ex_indices`, `lorentz_material_index.Ey_indices`, `lorentz_material_index.Ex_indices`), the corresponding storages are located. As explained in Chapter 2, for every field component and for every pole, we need to store two extra components $Q_k$ and $Q_{k+1}$ at every time step. Because in general, there are three field components at any node, this script allocates six matrices for this storage. In addition, the code initializes the update coefficients used to evaluate $Q_{k+1}$ using $Q_k$, $Q_{k-1}$, and $E_k$ as explained in Chapter 2. These coefficients are stored in the variable $Cqq$, $Cqqm$, and $Cqe$.

The script `initialize_residue_storage` is the one that determines the perturbation domains associated with each parameter. These domains, as explained in previous chapters, are needed for calculating the residues. The implementation of this script for dispersive media is different from the non-dispersive case. MATLAB Listing M6.3 shows a part of this script to illustrate the main differences. This script loops over all rectangular discontinuities in the domain. Every dispersive discontinuity is associated with $6+3*N_p$ parameters, where $N_p$ is the number of Lorentz poles. For the current example, we have for the 3-pole material a total of 15 parameters. For each parameter, the flag `residue_sampled_field(parameter_index).convolution_flag` determines

![Figure 6.5](image_url)
if calculating the parameter’s residue involves a convolution integral of the form (6.6). All parameters except for the electrical conductivity of the discontinuity have this flag set to true. The matrix residue_sampled_field(parameter_index).convolution_map determines which field components within the perturbation region has a convolution part. For example, as illustrated in Figure 6.2, only one line of fields has a perturbation in the convolution part despite the fact that two layers of fields are affected by this perturbation. In the current code implementation, we do not utilize interpolation for dispersion parameters which is only utilized for the parameters $\varepsilon_r$ and $\varepsilon_i$.

The storage residue_sampled_field(parameter_index).spectrum is used to store the spectra of all the field components associated with the perturbation region of the parameter with index parameter_index. At every time step of the original simulation, these spectra are updated using a DFT formula. After the end of the original simulation, formula (6.45) is used to evaluate the contribution of the convolution part in the frequency domain. Inverse DFT is then applied as in (6.46) to adjust the residues to include the convolution part.

MATLAB Listing M6.2

```matlab
% MATLAB Listing M6.2
for lorentz_material_index = 1:number_of_lorentz_materials % repeat for all Lorentz materials
    % get index of current material
    material = material_types(lorentz(lorentz_material_index).material);
    number_of_lorentz_terms = size(material.lorentz, 2); % number of Lorentz terms
    lorentz(lorentz_material_index).number_of_lorentz_terms = number_of_lorentz_terms;

    number_of_cells_with_this_material
    number_of_elements = size(lorentz(lorentz_material_index).Ex_indices, 2);
    lorentz(lorentz_material_index).Qxp = zeros(number_of_lorentz_terms, number_of_elements);
    lorentz(lorentz_material_index).Qx = zeros(number_of_lorentz_terms, number_of_elements);
    number_of_elements = size(lorentz(lorentz_material_index).Ey_indices, 2);
    lorentz(lorentz_material_index).Qyp = zeros(number_of_lorentz_terms, number_of_elements);
    lorentz(lorentz_material_index).Qy = zeros(number_of_lorentz_terms, number_of_elements);
    % repeat the same for the Ez components of the current Lorentz material
    number_of_elements = size(lorentz(lorentz_material_index).Ez_indices, 2);
    lorentz(lorentz_material_index).Qzp = zeros(number_of_lorentz_terms, number_of_elements);
    lorentz(lorentz_material_index).Qz = zeros(number_of_lorentz_terms, number_of_elements);

    % repeat for all terms of current lorentz material
    for lorentz_term_index = 1:number_of_lorentz_terms
        psi = eps_0 * material.lorentz(lorentz_term_index).A ... % (material.lorentz(lorentz_term_index).eps_s = material.eps_r) ...
        material.lorentz(lorentz_term_index).omega^2; % This is psi coefficient
        denominator = material.lorentz(lorentz_term_index).delta * dt + 1;
        lorentz(lorentz_material_index).Cqp(lorentz_term_index) = (2 - dt^2); % (material.lorentz(lorentz_term_index).omega/2)^2)/denominator;
        lorentz(lorentz_material_index).Cqp = ... % coefficient of Qn in calculating Qn+1
        (material.lorentz(lorentz_term_index).delta * dt - 1)/denominator;
        lorentz(lorentz_material_index).Cqp(lorentz_term_index) = ... % coefficient of Qn-1 in calculating Qn+1
        dt^2 * psi / denominator;
    end
end
```
Once the perturbation region of each parameter has been determined, the code determines the stamps associated with the $N$, $K$, and $P$ matrices. As evident from (6.34), the stamps for the diagonal matrices $N$ and $K$ are still the same as in the nondispersive case. For the matrix $P$, we store the Fourier transform of the convolution function at the selected frequencies. The script that initializes the stamps is initialize_residue_stamps. This script invokes a number of scripts that build the stamps for all 15 parameters. For example, the script build_Ez_stamp_1psilonr is shown in MATLAB Listing M6.4. This script shows
how the stamps of the matrices $N$, $K$, and $P$ are constructed when the dielectric constant of the discontinuity is changed. This script invokes the function `get_convolution_function_derivative_epsilon_r` to evaluate the Fourier transform of the derivative of (6.38) at the frequencies $j_{dft} \omega_{\text{omegas}}$ whose number is `number_of_dft_frequencies`. Notice that the Fourier transform is evaluated numerically at the frequencies considered for the DFT transform. These frequencies are sufficient to express the excited signal. The script `store_stamp` stores the calculated stamps for later use in evaluating the residues.

Once all residue stamps have been determined, the code proceeds to carry out the original FDTD simulation. The script `run_original_fdtd_time_marching_loop_2d` carries out this simulation. At every time step, the magnetic fields are first updated as in the nondispersive case. The electric field is updated and then the dispersion contribution is added using the dispersion storage. The ADE formulation used in this part was introduced in Chapter 2. One major change exists in the way the residues are evaluated at every time step. The script `store_original_fields`, shown in MATLAB Listing M6.5, evaluates only the residue part contributed by the stamps of the $N$ and $K$ matrices for all parameters. It also updates the spectrum of all the field components associated with every parameter. After the end of the original simulation, these spectra are multiplied by the $P$ matrix frequency stamp to evaluate the convolution in the frequency domain as in (6.45). The script `adjust_residues` adjusts the temporal residues by adding the contribution from the $P$-matrix stamp. This script loops through all
parameters (15 parameters for the current example). It loads the spectra of all field components related to the current parameter (Ez_spectrum). It also loads the $P$-matrix stamp corresponding to this parameter ($P_{stamp\_Ez}$). The code then multiplies the spectrum of each component of Ez_spectrum by the spectrum $P_{stamp\_Ez}$. Inverse Fourier transform is applied to convert the product spectrum (current_convolution_contribution) to the time domain. The matrix dft_conjugate_exponents contains the values of $\exp(j2\pi ft)$ for all considered frequencies at all time steps. This function evaluates the complete temporal waveform of each frequency. The command

```
temporal_component =
2*df*real(current_phasor*dft_conjugate_exponents)
(dft_frequency_index,:));
```

evaluates one cosine waveform of the expression (6.46). Notice that the convolution contributions are added to the residues only if the flag conversion_map (ind) is true. These components are the ones whose $P$-matrix is affected by the perturbation in the current parameter. After this script is executed, the residues for

![Graphs showing sensitivities of S21](image)

**Figure 6.6** The $S_{21}$ sensitivities of the Lorentzian slab example: the adjoint sensitivities (○) as compared to forward finite difference approximation (—) over wide frequency band
Figure 6.7  The $S_{21}$ sensitivities of the Lorentzian slab example with respect to the parameters of the first Lorentzian pole: the adjoint sensitivities (○) as compared to forward finite difference approximation (—) over wide frequency band.
all parameters, whether those including a convolution term or not become available. The script `fdtd_solve_original` ends by evaluating the $S$-parameters using the calculated incident and transmitted spectra. Both the incident and transmitted spectra are divided frequency-by-frequency by a reference spectrum. This spectrum is evaluated by running a separate simulation of the waveguide with the dispersive discontinuity removed.

Figure 6.8 The $S_{21}$ sensitivities of the Lorentzian slab example with respect to the parameters of the second Lorentzian pole: the adjoint sensitivities ($\circ$) as compared to forward finite difference approximation ($\cdots$) over wide frequency band.
The adjoint simulation of this code follows the same line as the examples of Chapter 4. The main difference is that the simulation is dispersive and thus the ADE is also included in the field calculations. All other calculations are essentially the same as for frequency-dependent responses.

The adjoint sensitivity estimates as compared to forward finite differences are shown in Figures 6.6–6.10. Good match is observed between both estimates over most of the considered frequency band. A slight deviation is observed at both edges.
of the considered frequency band. This is mainly due to the reduced accuracy of finite difference estimates at these frequencies. As evident from Figure 6.4, the excited spectrum is weak at these frequencies and this creates more errors in calculating the $S$-parameters at these frequencies. At the edge frequencies, the adjoint-based sensitivities are believed to be more accurate than the ones estimated using finite differences.

The perturbations used for the parameters are

$$
\begin{bmatrix}
\Delta \varepsilon_r & \Delta \sigma^{\varepsilon} & \Delta \varepsilon_{s,1} & \Delta \omega_{d,1} \\
\Delta \varepsilon_{s,1} & \Delta \varepsilon_{s,2} & \Delta \omega_{d,2} & \Delta \varepsilon_{s,3} \\
\Delta \omega_{d,2} & \Delta \varepsilon_{s,3} & \Delta \omega_{d,3} & \Delta \xi \xi \xi \\
\Delta \omega_{d,3} & \Delta \xi \xi \xi & \Delta y_{\min} & \Delta y_{\max}
\end{bmatrix}
= \begin{bmatrix}
0.05 & 0.0001 & 0.05 & 2*\pi*0.1005e9 & 2*\pi*0.05e9 & 0.05 & 2*\pi*0.1005e9 & 2*\pi*0.05e9 & 0.05 & 2*\pi*0.1005e9 & 2*\pi*0.05e9 & 75e-6 & 75e-6
\end{bmatrix}^T.
$$

Note that the parameters $x_{\min}$ and $x_{\max}$ are redundant and thus we do not estimate their corresponding finite difference sensitivities. The forward finite difference approximations require 13 extra simulations. The adjoint variable method requires only one extra simulation to estimate all sensitivities over the desired band of interest.

**Figure 6.10** The $S_{21}$ sensitivities of the Lorentzian slab example with respect to the parameters $y_{\min}$ and $y_{\max}$: the adjoint sensitivities (○) as compared to forward finite difference approximation (—) over wide frequency band.
References


174  Adjoint sensitivity analysis of high frequency structures with MATLAB®


Chapter 7

Adjoint sensitivity analysis of anisotropic structures

In previous chapters, we discussed several adjoint sensitivity analysis approaches that apply for linear and isotropic media. In many applications, however, the material considered is anisotropic in nature [1–5]. The anisotropy may show in the electrical conductivity, magnetic resistivity, electric permittivity, magnetic permeability, or electric-magnetic coupling. The emerging area of transformation electromagnetics [6] results usually in electromagnetic structures with anisotropic properties. One of the main applications of anisotropic materials is the area of cloaking [7–10]. An object is rendered “invisible” by surrounding it by a cloak of a metamaterial that usually has anisotropic properties.

In this chapter, we present an algorithm for adjoint sensitivity analysis of anisotropic materials. We show that using only one extra adjoint simulation, the sensitivities of the objective function with respect to all parameters are estimated. The material property tensors of the adjoint problem are the transpose of those of the original problem. The computational cost of the adjoint problem is the same as that of the original simulation.

The derivation considered in this chapter addresses the nondispersive case. The considered tensors are assumed to be independent of time. The formulation presented in this chapter is adapted from [11]. The anisotropic and dispersive case is still a subject of research.

7.1 AVM for anisotropic materials

We aim at deriving an adjoint approach for estimating the sensitivities of an objective function of the form (3.1), given here for convenience:

\[ F = \int_0^{T_w} \psi(p, V) dt, \]  

(7.1)

where \( p \in \mathbb{R}^n \) is the design parameter vector whose components may include geometrical and material properties. The vector \( V = [E^T \quad H^T]^T \) is the vector of electric and magnetic field values over the whole computational domain. We limit ourselves in the discussion that follows to considering sensitivities relative to material properties.
For the 3D anisotropic case, the material properties are all tensors that may have nondiagonal components. In this case, the six Maxwell’s equations given in (3.41) are adapted to give

\[
\begin{bmatrix}
\varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\
\varepsilon_{yx} & \varepsilon_{yy} & \varepsilon_{yz} \\
\varepsilon_{zx} & \varepsilon_{zy} & \varepsilon_{zz}
\end{bmatrix}
\begin{bmatrix}
-\mu_{xx} & -\mu_{xy} & -\mu_{xz} \\
-\mu_{yx} & -\mu_{yy} & -\mu_{yz} \\
-\mu_{zx} & -\mu_{zy} & -\mu_{zz}
\end{bmatrix}
\begin{bmatrix}
E_x \\
E_y \\
E_z
\end{bmatrix}
=
\begin{bmatrix}
\dot{E}_x \\
\dot{E}_y \\
\dot{H}_x
\end{bmatrix}
\]

\[
\begin{bmatrix}
\sigma^e_{xx} & \sigma^e_{xy} & \sigma^e_{xz} \\
\sigma^e_{yx} & \sigma^e_{yy} & \sigma^e_{yz} \\
\sigma^e_{zx} & \sigma^e_{zy} & \sigma^e_{zz}
\end{bmatrix}
\begin{bmatrix}
E_x \\
E_y \\
E_z
\end{bmatrix}
+
\begin{bmatrix}
-\sigma^m_{xx} & -\sigma^m_{xy} & -\sigma^m_{xz} \\
-\sigma^m_{yx} & -\sigma^m_{yy} & -\sigma^m_{yz} \\
-\sigma^m_{zx} & -\sigma^m_{zy} & -\sigma^m_{zz}
\end{bmatrix}
\begin{bmatrix}
H_x \\
H_y \\
H_z
\end{bmatrix}
\]

\[
\begin{bmatrix}
\partial / \partial z & -\partial / \partial y & \partial / \partial x \\
-\partial / \partial z & \partial / \partial x & -\partial / \partial y \\
\partial / \partial z & -\partial / \partial y & \partial / \partial x \\
-\partial / \partial z & \partial / \partial x & -\partial / \partial y \\
\partial / \partial y & \partial / \partial z & -\partial / \partial x \\
\partial / \partial y & \partial / \partial z & -\partial / \partial x
\end{bmatrix}
\begin{bmatrix}
E_x \\
E_y \\
E_z
\end{bmatrix}
=
\begin{bmatrix}
-J_{ix} \\
-J_{iy} \\
-J_{iz}
\end{bmatrix}
\]

(7.2)

In (7.2), we assume that the permittivity, the permeability, the electric conductivity, and the magnetic conductivity are full tensors each with nine components. Utilizing Yee’s discretization and writing this equation for all cells in the computational domain, we have

\[
\begin{bmatrix}
\varepsilon & 0 \\
0 & -\mu
\end{bmatrix}
\begin{bmatrix}
\dot{E} \\
\dot{H}
\end{bmatrix}
+
\begin{bmatrix}
\sigma^e & 0 \\
0 & -\sigma^m
\end{bmatrix}
\begin{bmatrix}
E \\
H
\end{bmatrix}
+\mathbf{K}\mathbf{c}
\begin{bmatrix}
E \\
H
\end{bmatrix}
=\begin{bmatrix}
-J_i \\
M_i
\end{bmatrix},
\]

(7.3)

where \(\varepsilon, \mu, \sigma^e,\) and \(\sigma^m\) are block diagonal matrices representing the material properties of the computational domain. Each diagonal block represents the material tensor of the respective spatial cell. This system has the form (3.42). Following a similar derivation to that given in Chapter 3, the corresponding adjoint system is given by

\[
-\begin{bmatrix}
\varepsilon^T & 0 \\
0 & -\mu^T
\end{bmatrix}
\begin{bmatrix}
\lambda_E \\
\lambda_H
\end{bmatrix}
+\begin{bmatrix}
\sigma^e^T & 0 \\
0 & -\sigma^m^T
\end{bmatrix}
\begin{bmatrix}
\lambda_E \\
\lambda_H
\end{bmatrix}
+\mathbf{K}\mathbf{c}
\begin{bmatrix}
\lambda_E \\
\lambda_H
\end{bmatrix}
= \frac{\partial \psi}{\partial V}.
\]

(7.4)
Changing the time variable in (7.4) from the original simulation time variable \( t \) to the adjoint time variable \( \tau \), where \( \tau = T_m - t \), the system (7.4) is cast in the form

\[
\begin{bmatrix}
\varepsilon^T & 0 \\
0 & -\mu^T
\end{bmatrix}
\begin{bmatrix}
\dot{\lambda}_E \\
\dot{\lambda}_H
\end{bmatrix}
+ \begin{bmatrix}
\sigma^T & 0 \\
0 & -\sigma^m T
\end{bmatrix}
\begin{bmatrix}
\lambda_E \\
\lambda_H
\end{bmatrix}
+ K_c \begin{bmatrix}
\dot{\lambda}_E \\
\dot{\lambda}_H
\end{bmatrix} = \left( \frac{\partial \psi}{\partial V} (T_m - \tau) \right),
\]

(7.5)

where all derivatives in (7.5) are with respect to the adjoint time variable \( \tau \). The right-hand side of (7.5) represents the adjoint excitation. It is evaluated during the original simulation and then applied backward in time. The adjoint system (7.5) is similar to the original system (7.3) with two main differences. The material tensors utilized in (7.5) are the transpose of the material tensors in (7.3). For an isotropic medium, the tensor is a diagonal matrix and thus is symmetric. For a general anisotropic medium, this may not be the case. The adjoint system (7.5) has an adjoint excitation that is determined during the original simulation and is applied backward in time.

Once the original simulation (7.3) and the adjoint simulation (7.5) are carried out, the sensitivities of the objective function relative to all parameters are given by

\[
\frac{\partial F}{\partial p_i} = \int_0^{T_m} \lambda^T \left( \frac{-\Delta R_i}{\Delta p_i} \right) dt \approx -\frac{1}{\Delta p_i} \sum_k \lambda_k^T \Delta R_i, \quad i = 1, 2, \ldots, n,
\]

(7.6)

where \( \Delta R_i = \Delta N_i \dot{V} + \Delta K_i V \) is the temporal residue vector associated with the \( i \)th parameter as explained in Chapter 3.

### 7.2 Implementation

The derivatives of the system matrices with respect to the components of the anisotropy tensors are available analytically. We illustrate these derivatives for the first column of the permittivity tensor matrix (with components \( \varepsilon_{r,xx}, \varepsilon_{r,yy}, \) and \( \varepsilon_{r,zz} \)). Similar to the formula (3.48), the residues in this case are given by

\[
\frac{\Delta R}{\Delta \varepsilon_{r,ix}} \approx \frac{\partial R}{\partial \varepsilon_{r,ix}} = \begin{bmatrix}
\partial \varepsilon / \partial \varepsilon_{r,ix} & 0 \\
0 & 0
\end{bmatrix} \begin{bmatrix}
\dot{E} \\
\dot{H}
\end{bmatrix}
+ \begin{bmatrix}
0 & 0 \\
0 & 0
\end{bmatrix} \begin{bmatrix}
E \\
H
\end{bmatrix},
\]

(7.7)

where \( i \in \{x, y, z\} \). Because the tensor component \( \varepsilon_{r,xx} \) multiplies the electric field in the \( x \)-direction, the matrix \( \partial \varepsilon / \partial \varepsilon_{r,xx} \) is a diagonal matrix with nonzero components multiplying the \( \dot{E}_x \) components. Also, and as explained in Chapter 3, the components of this matrix belong to the set \( \{0, 0.25 \varepsilon_o, 0.5 \varepsilon_o, \) and \( \varepsilon_o\} \) depending on the position of this field component within the anisotropic discontinuity. This is a result of the interpolation scheme utilized within the finite difference time domain (FDTD) algorithm. It can be shown that the tensor matrix derivatives \( \partial \varepsilon / \partial \varepsilon_{r,yy} \) and \( \partial \varepsilon / \partial \varepsilon_{r,zz} \) are both nondiagonal matrices with nonzero components also multiplying the \( \dot{E}_x \) components. These nonzero components also belong to the set \( \{0, 0.25 \varepsilon_o, 0.5 \varepsilon_o, \) and \( \varepsilon_o\} \). It can also be shown that the components of the derivatives of the second column of the permittivity tensor matrix multiply the \( \dot{E}_y \) components, and...
the components of the derivatives of the third column of the permittivity tensor matrix multiply the $\hat{E}_z$ components. These derivatives contain components corresponding to the set \{0, 0.25\$\varepsilon_0$, 0.5\$\varepsilon_0$, and $\varepsilon_0$\}. The residues (7.7) are evaluated with respect to all parameters during the original simulation. Their calculations include adding weighted derivatives of the original fields. The adjoint excitation to be utilized in (7.5) is also determined during the original simulation.

The adjoint simulation is then carried out to evaluate the adjoint variables using the transposed material tensors. As the adjoint fields become available, the inner product (7.6) is updated for all parameters. For the parameter $\varepsilon_{r,xx}$, and using the discussion on the calculation of the residues corresponding to material properties, this inner product is given by

$$\frac{\partial F}{\partial \varepsilon_{r,xx}} = -\int_0^{T_m} \lambda^T \left( \frac{\partial R}{\partial \varepsilon_{r,xx}} \right) dt \approx -\Delta t \sum_k \lambda_k^T \left( \frac{\partial R(k\Delta t)}{\partial \varepsilon_{r,xx}} \right)$$

$$= -\Delta t \sum_k \lambda_k^T \begin{bmatrix} \partial \varepsilon / \partial \varepsilon_{r,xx} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{E} \\ \hat{H} \end{bmatrix} = -\Delta t \sum_k \sum_j a_{xx,j} \bar{\lambda}_{E_x,j} \dot{E}_{x,j}, \quad (7.8)$$

where $j$ is the index of all nodes affected by the change in this material parameter. The adjoint field $\bar{\lambda}_{E_x,j}$ is the $x$-polarized adjoint electric field at the $j$th node at the $k$th time step. The bar indicates that this component must be evaluated at the same position at which the original field time derivative is sampled. The quantity $\dot{E}_{x,j}$ denotes the time derivative of the $x$-polarized original electric field of the $j$th node at the $k$th time step. The weight $a_{xx,j}$ denotes the stamp associated with the $j$th node for the parameter $\varepsilon_{r,xx}$. Its values, as indicated earlier, are position dependent and assume values from the set \{0, 0.25\$\varepsilon_0$, 0.5\$\varepsilon_0$, and $\varepsilon_0$\}. Expressions similar to (7.8) can be derived for all other permittivity tensor components. These expressions are given by

$$\frac{\partial F}{\partial \varepsilon_{r,xy}} \approx -\Delta t \sum_k \sum_j a_{xy,j} \bar{\lambda}_{E_y,j} \dot{E}_{x,j}, \quad (7.9)$$

$$\frac{\partial F}{\partial \varepsilon_{r,xz}} \approx -\Delta t \sum_k \sum_j a_{xz,j} \bar{\lambda}_{E_z,j} \dot{E}_{x,j}, \quad (7.10)$$

$$\frac{\partial F}{\partial \varepsilon_{r,yy}} = -\Delta t \sum_k \sum_j a_{yy,j} \bar{\lambda}_{E_y,j} \dot{E}_{y,j}, \quad (7.11)$$

$$\frac{\partial F}{\partial \varepsilon_{r,yz}} \approx -\Delta t \sum_k \sum_j a_{yz,j} \bar{\lambda}_{E_z,j} \dot{E}_{y,j}, \quad (7.12)$$

$$\frac{\partial F}{\partial \varepsilon_{r,zy}} \approx -\Delta t \sum_k \sum_j a_{zy,j} \bar{\lambda}_{E_y,j} \dot{E}_{z,j}, \quad (7.13)$$
Similar expressions can also be derived for the sensitivities with respect to the components of the magnetic permeability tensor, the electric conductivity tensor, and the magnetic conductivity tensor.

The expressions (7.8)–(7.16) require multiplying an adjoint field component by the weighted time derivatives of different original field components and summing the product over all time steps. A complication that arises from Yee’s cell is that the different polarizations of the electric field of the same node are not collocated. Figure 7.1 shows how the adjoint electric fields polarized in the y-direction are located with respect to the x-component of one of the cells. Four y-polarized adjoint electric field components surround the x-polarized original component. The electric field to be utilized in (7.9) at the jth node is approximated as the average of all these adjoint components:

$$\bar{E}_y = 0.25(\lambda_{E_x}(i,j) + \lambda_{E_x}(i,j - 1) + \lambda_{E_x}(i + 1,j) + \lambda_{E_x}(i + 1,j - 1)).$$  

Four components of each adjoint field component are thus averaged at all-time steps to supply the components required in (7.8)–(7.16). A similar approach is utilized within the numerical technique adopted in this book for solving the anisotropic Maxwell’s equations [12] as explained in Chapter 2. It should be clear that in (7.8), (7.11), and (7.14), averaging of adjoint field components is not needed. Both the adjoint field component and the residue have the same polarization and are thus collocated. For the case of isotropic media, the steps presented in this chapter reduce to the ones discussed in Chapter 3 for isotropic media.
Example 7.1 To illustrate the anisotropic adjoint sensitivity algorithm, we consider the structure shown in Figure 7.2. The computational domain in this 3D problem has dimensions $25\Delta l \times 25\Delta l \times 25\Delta l$ with $\Delta l = 4.0 \text{ mm}$. An anisotropic discontinuity is at the center of this air-filled computational domain. This discontinuity has dimensions $5\Delta l \times 5\Delta l \times 5\Delta l$. Its permittivity tensor is

$$
\mathbf{\varepsilon} = \varepsilon_0 \begin{bmatrix} 2.0 & 0.2 & 0.4 \\ 0.2 & 2 & 0.5 \\ 0.4 & 0.5 & 3.1 \end{bmatrix}.
$$

The domain is terminated by absorbing boundaries in all directions. The excitation is an incident plane wave propagating in the $z$-direction with its electric field polarized in the $x$-direction. The objective function is defined by the integral

$$
F = 0.5 \int_0^{T_w} \overline{E}_x^2 dt,
$$

where $\overline{E}_x$ is an $x$-polarized space-averaged electric field defined by

$$
\overline{E}_x = 0.25(E_x(20\Delta l, 20\Delta l, 20\Delta l) + E_x(20\Delta l, 21\Delta l, 20\Delta l) + E_x(20\Delta l, 21\Delta l, 21\Delta l)) + E_x(20\Delta l, 20\Delta l, 21\Delta l) + E_x(20\Delta l, 21\Delta l, 21\Delta l)).
$$

This averaged field represents a cell-centered rather than edge field component.
As in all the codes presented throughout this book, this example starts by calling the script `TestAVM`, shown in MATLAB® Listing M7.1. This code invokes the original solver script `fdtd_solve_original` followed by the adjoint solver script `fdtd_solve_adjoint`. It finally prints out the estimated adjoint sensitivities given by the vector `sensitivities`.

The original solver starts by setting up the material properties, the excitations, the boundaries, and the output. It also sets up the storage needed for the residues and the values of these residues at every time step. The script `define_problem_space_parameters` defines the materials and discretizations utilized in the problem. The commands

```matlab
material_types(5).eps_r = [2.0 0.2 0.4; 0.2 2.0 0.5; 0.4 0.5 3.1];
material_types(5).mu_r = 1;
material_types(5).sigma_e = 0.0;
material_types(5).sigma_m = 0.0;
```

define the anisotropic and lossless material utilized in this example. The script `define_output_parameters` (shown in MATLAB Listing M7.2) defines the four probes used to evaluate the average field $\bar{E}_x$. The x-polarized electric field at these probes is stored throughout the simulation time to be utilized in determining the adjoint sources. The script `initialize_residue_storage` (shown in MATLAB Listing M7.3) determines the perturbation region of the problem parameters. In this example, we choose $p = \left[ \varepsilon_{r,xx}, \varepsilon_{r,xy}, \varepsilon_{r,xz}, \varepsilon_{r,yy}, \varepsilon_{r,yz}, \varepsilon_{r,zy}, \varepsilon_{r,xz}, \varepsilon_{r,zz} \right]^T$ for the anisotropic discontinuity. Because we consider here only material parameters, a change in any of these parameters affects the whole volume of the discontinuity. This explains why the perturbation domains in MATLAB Listing 7.3 are all identical. The residues corresponding to all parameters are determined and stored within these volumes. The parameters `ni.is`, `ni.js`, and `ni.ks` are the starting indices (in terms of $\Delta l$) of the discontinuity. The parameters `(ni.ie)-1`, `(ni.je)-1`, and `(ni.ke)-1` are the end indices of the discontinuity.

Once the perturbation volumes have been set, the code determines the stamps associated with these volumes. This is carried out in the script `initialize_residue_stamps` shown in MATLAB Listing M7.4. As explained in (7.8)–(7.16), the stamps multiply derivatives of the different polarizations of the electric field. The stamps of the parameters $p_1$, $p_4$, and $p_7$ ($\varepsilon_{r,xx}, \varepsilon_{r,yy}, \varepsilon_{r,zz}$) multiply the time derivative of the $E_x$ field within the discontinuity. The number of these stamps and their values are determined by the number of $E_x$ components and their positions within the discontinuity as explained in Chapter 3. Similarly, the stamps of the parameters $p_2$, $p_5$, and $p_8$ ($\varepsilon_{r,xy}, \varepsilon_{r,yz}, \varepsilon_{r,zy}$) multiply the time derivative of the $E_y$.
field within the discontinuity. The number of these stamps depends on the number of $E_z$ components and their positions within the discontinuity. Finally, the stamps of the parameters $p_3, p_6,$ and $p_9$ ($\varepsilon_{r,xx}, \varepsilon_{r,yz},$ and $\varepsilon_{r,zz}$) are determined by the number and locations of the $E_z$ field components within the discontinuity. The stamps have values of $\varepsilon_0$, $0.5\varepsilon_0$, and $0.25\varepsilon_0$ depending on whether the field component is an interior component, a surface component, or an edge component, respectively, as explained in Chapter 3.

The script `initialize_updating_coefficients` evaluates the coefficients used in the FDTD update equations. For isotropic nodes, the typical FDTD update sequence is utilized. The H-field components at time $(k + 0.5) \Delta t$ are updated and then the electric field components at time $(k + 1) \Delta t$ are updated. For these cells, the script `initialize_updating_coefficients` evaluates the update matrices $C_{exe}, C_{exhz}, C_{exhy}, C_{eye}, C_{eyhx}, C_{eyhz}, C_{ez}$,
MATLAB Listing M7.3

% MATLAB Listing M7.3
% 9 permittivity material parameters associated with every block starting from
% the second one (the
% main air block is assumed invariant
number_of_parameters=0;
for k=2:number_of_blocks
    % anisotropic tensor parameters per block
    parameter_index=(k-2)*9+1; % current parameter
    % repeat for all parameters
    ni = get_node_indices(bricks(k), ftd_domain); % get extent of kth block
    notice that the structure ni end values have to be subtracted by one
    residue_sampled_field(parameter_index).min_x = ni.is;
    residue_sampled_field(parameter_index).min_y = ni.js;
    residue_sampled_field(parameter_index).min_z = ni.ks;
    residue_sampled_field(parameter_index).max_x = (ni.ie)-1;
    residue_sampled_field(parameter_index).max_y = (ni.js)-1;
    residue_sampled_field(parameter_index).max_z = (ni.ks)-1;
    residue_sampled_field(parameter_index).direction="n"; % material type parameter
    residue_sampled_field(parameter_index).storeflag=true;
    residue_sampled_field(parameter_index).display_plot = false;
    number_of_parameters=number_of_parameters+1;
end
end

MATLAB Listing M7.4

% MATLAB Listing M7.4
% 9 parameters associated with every block starting from the second one (the
% main air block is assumed invariant
for k=2:number_of_blocks
    % repeat for all parameters related to the current brick
    parameter_index=(k-2)*9+1; % parameter index
    % get start of the block associated with the current block
    is=residue_sampled_field(parameter_index).min_x;
    js=residue_sampled_field(parameter_index).min_y;
    ks=residue_sampled_field(parameter_index).min_z;
    max_x=residue_sampled_field(parameter_index).max_x;
    max_y=residue_sampled_field(parameter_index).max_y;
    max_z=residue_sampled_field(parameter_index).max_z;
    % each field component must be handled differently
    % we first consider Ex component
    if((i==1)||(i==4)||(i==7)) % epsilonxx or epsilonxy or epsilonxzy
        build Ex stamp; % build stamp for Ex components related to current parameter
        % Nx, Ny, Nz are the number of stored components for each parameter
        residue_sampled_field(parameter_index).K_stmEx=reshape(K_stm, 1, N_x*N_y*N_z);
        residue_sampled_field(parameter_index).K_stmEx=reshape(K_stm, 1, N_x*N_y*N_z);
    end
    if((i==2)||(i==5)||(i==8)) % epsilonxy or epsilonxy or epsilonxzy
        build By stamp; % build stamp for By components related to current parameter
        residue_sampled_field(parameter_index).K_stmBy=reshape(K_stm, 1, N_x*N_y*N_z);
        residue_sampled_field(parameter_index).K_stmBy=reshape(K_stm, 1, N_x*N_y*N_z);
    end
    if((i==3)||(i==6)||(i==9)) % epsilonxy or epsilonxy or epsilonxzy
        build Ez stamp; % build stamp for Ez components related to current parameter
        residue_sampled_field(parameter_index).K_stmEz=reshape(K_stm, 1, N_x*N_y*N_z);
        residue_sampled_field(parameter_index).K_stmEz=reshape(K_stm, 1, N_x*N_y*N_z);
    end
end
end
Cezhy, Cezhx, Chxh, Chxez, Chxey, Chy, Chyex, Chyez, Chzh, Chzey, and Chzex for the whole domain. For anisotropic cells, the update formulas are different. The H-field components are evaluated using the electric field components. The components of the electric displacement vector $D$ are then evaluated using the updated magnetic field components. The electric field components are then evaluated from the $D$ components using the analytical inverse of the permittivity tensor [12] as explained in Chapter 2. New sets of coefficients are thus needed for updating the components of the electric displacement vector and for evaluating the electric field components using the $D$ components. The script $\text{initialize anisotropic coefficients}$ evaluates these coefficients for the anisotropic cells only. The reader may consult [12] for more details on the FDTD-based approach adopted in this book for solving anisotropic structures.

The script $\text{run_fdtd_original_time_marching_loop}$ executes the time-marching loop of the FDTD algorithm. This script invokes other scripts for updating the $H$-field components and updating the $E$ components for the whole domain. It also invokes scripts that update the $D$-field components for anisotropic cells. Using the electric displacement values, the electric field components are updated. At every time step, the residues related to all anisotropy parameters are calculated using the formula (7.7). The different stamps multiply derivatives of different field components as explained earlier. The evaluation of the residues at every time step is carried out using the script $\text{store_original_fields}$ shown in MATLAB Listing M7.5. This script evaluates the first and second right-hand contributions in (7.7). Even though the contribution from the $K$ stamp is zero for components of the permittivity tensor, the code still takes them into account. This contribution will be nonzero when calculating residues corresponding to the components of the electric or magnetic conductivity tensors.

The script $\text{store_original_fields}$ utilizes the stamps of the $N$ matrix to multiply the derivative of the proper component of the electric field. It also utilizes the stamp of the $K$ matrix to multiply the proper component of the electric field. The script evaluates the size of the stamp in all three coordinates. This number varies from one field component to another. The electric field derivative over the perturbation region is then estimated as the difference between the current field values and the previous field values. The current field values over the perturbation region are given by the vectors $\text{current_Ex}$, $\text{current_Ey}$, and $\text{current_Ez}$. The previous field values are stored in the variables $\text{previous_Ex}$, $\text{previous_Ey}$, and $\text{previous_Ez}$. The code retrieves the proper field information based on the parameter index. The code also stores the current field values to be used in the next time step as previous field values. Zero-field derivatives are assumed at time $t = 0$. The commands

```matlab
residue_sampled_field(parameter_index).residue_Ex(time_step,:)=
  ((1/dt)*(N_stamp_Ex.*(current_Ex-previous_Ex)))+
  (0.5*K_stamp_Ex.*(current_Ex+previous_Ex));
residue_sampled_field(parameter_index).residue_Ey(time_step,:)=
  ((1/dt)*(N_stamp_Ey.*(current_Ey-previous_Ey)))+
  (0.5*K_stamp_Ey.*(current_Ey+previous_Ey));
residue_sampled_field(parameter_index).residue_Ez(time_step,:)=
  ((1/dt)*(N_stamp_Ez.*(current_Ez-previous_Ez)))+
  (0.5*K_stamp_Ez.*(current_Ez+previous_Ez));
```
implement (7.7) for the current parameter (with index `parameter_index`) using the derivative of the appropriate field component.

Once the original simulation is done, the script `fdtd_solve_adjoint` is invoked. Unlike adjoint codes discussed in previous chapters, the material of the adjoint simulation is different from the original simulation. All material tensors are transposed in the adjoint simulation. The script `define_adjoint_problem_space_parameters` defines the permittivity tensor of the adjoint problem. It includes the single command

```matlab
data_types(5).eps_r = (data_types(5).eps_r)';
```
which takes the transpose of the permittivity of the fifth defined material (the discontinuity material). The script `define_adjoint_sources_and_lumped_elements` determines the locations and time profiles of the adjoint current density sources. This script is shown in MATLAB Listing M7.6. Because the original problem has four electric field probes, the adjoint problem has four adjoint electric current density sources. The command

```matlab
excitation_value_start=
-0.125*(sampled_electric_fields(1).sampled_value+
          sampled_electric_fields(2).sampled_value+
          sampled_electric_fields(3).sampled_value+
          sampled_electric_fields(4).sampled_value);
```

evaluates the derivative of the kernel of the objective function integral $\psi$ at all-time steps. The negative sign is needed to convert the derivative of the kernel function to an adjoint current density source as explained in Chapter 3. Because the current density source in the FDTD algorithm is applied at $(k + 0.5)$ time step, the values of temporal vector `excitation_value_start` are averaged over every two consecutive time steps to give the vector `excitation_value_center`. The command

```matlab
current_density_sources(i).current_density_values=
    excitation_value_center(end:-1:1)
```

then sets the temporal time profile of the $i$th adjoint electric current density source. A number of scripts are also invoked within the adjoint simulation to initialize different matrices. In particular, the script `initialize_adjoint_storage` determines the perturbation regions over which the adjoint fields are to
be stored. Because the discussed implementation evaluates sensitivities with respect to material parameters, these adjoint regions are identical to the perturbation regions of the original problem.

The script `run_fdtd_adjoint_time_marching_loop` executes the adjoint simulation time-marching loop. The same original update steps are applied to update the adjoint magnetic and electric fields. The script `update_adjoint_product` updates the product (7.6) at every time step for each
parameter. This script retrieves the adjoint fields and multiplies them element-by-element with the corresponding residue. If the adjoint fields have different polarization, the four adjoint fields surrounding the location at which the original residue is calculated are averaged as illustrated in Figure 7.1.

The script `TestAVM` creates the following output:

```
sensitivities =
  1.0e-16 *
Columns 1 through 4
  0.363380270761769  -0.011184912310042
                   -0.020955460962291  0.114095818897454
Columns 5 through 8
  -0.001150224776113  -0.006436938795988
                    0.086913267669490  -0.002493543284067
Column 9
  -0.002759564282213
```

![Graph](image.png)

**Figure 7.3** The objective function values for a sweep of the permittivity tensor component $\varepsilon_{r,xx}$
Figure 7.4  The adjoint sensitivities of the objective function of Example 7.1 as compared with forward finite difference approximations for a sweep of the parameter $\varepsilon_{r,xx}$. 
To verify these results, the forward finite difference approximations are evaluated using the perturbations $\Delta p = [0.05 \ 0.005 \ 0.01 \ 0.005 \ 0.05 \ 0.0125 \ 0.01 \ 0.0125 \ 0.05]^T$. The calculated finite difference sensitivities are

$$\text{gradient} = 1.0e-16 *$$

Columns 1 through 4

0.367949915297321 -0.015667207719949 -0.028507643947685
0.102595647850101

Columns 5 through 8

-0.002404251532404 -0.007223248106839 0.056841802677180
-0.002133614547810

Column 9

-0.002720396453607

We see that in both estimates the sensitivity with respect to the permittivity parameter $\varepsilon_{r,xx}$ dominates. Most of the components show reasonable match with the finite difference approximation. The value of the objective function at the given permittivity tensor is $2.828562413354588e-17$. The sensitivities are relatively large with respect to the value of the objective function indicating a fast changing objective function.

The permittivity tensor parameter $\varepsilon_{r,xx}$ is swept between the two values 2.0 and 3.0. The value of the objective function for different value of $\varepsilon_{r,xx}$ is shown in Figure 7.3. The objective function shows a steep change. The adjoint sensitivities are compared in Figure 7.4 in comparison with the forward finite differences. Only the sensitivities with respect to the upper half of the sensitivity tensors are shown. A reasonable match is achieved for all permittivity tensor components.

References


[3] E. Doumanis, G. Goussetis, J. L. Gomez-Tornero, R. Cahill, and V. Fusco, “Anisotropic impedance surfaces for linear to circular polarization...


Chapter 8
Nonlinear adjoint sensitivity analysis

Some of the materials known to humans exhibit nonlinearity with respect to the applied electric field. In this case, the permittivity and/or the conductivity may be dependent on the applied electric field [1–3]. This case results in a displacement vector $D = \varepsilon E$ and a current density vector $J = \sigma E$ that are nonlinear functions of the electric field. This nonlinearity results in broadening the bandwidth of the excited signal which can be used to realize many interesting functionalities.

In previous chapters, we derived algorithms for estimating the sensitivities of arbitrary objective functions or responses with respect to all parameters. We considered only the linear case for isotropic, nondispersive, anisotropic, and dispersive materials. In this chapter, we focus on the nonlinear case. We show that if the relative permittivity, conductivity, or both are polynomial functions of the electric field, we can extract the sensitivities of the considered objective function or response with respect to all nonlinearity parameters using one extra simulation [4,5]. This approach is illustrated for the isotropic and nondispersive case. However, the theory can be extended to nonlinear materials with arbitrary anisotropy and dispersion profiles. The theory of adjoint sensitivity analysis of nonlinear material in computational electromagnetics is still at its initial stages. We report here on some initial results using the finite difference time domain (FDTD) method.

Because this theory is more complex than the linear case, we illustrate it first through a simple circuit example. We also utilize 1D and 2D examples to show that this approach works for FDTD-based problems.

8.1 Nonlinear AVM

We aim in this chapter to derive an adjoint approach for estimating the sensitivities of an objective function of the form [6]:

$$F = \int_{0}^{T_{m}} \psi(p, V) dt,$$

(8.1)

for the case where the computational domain contains nonlinear materials. As in previous chapters, $T_{m}$ is the simulation time and $p \in \mathbb{R}^{n}$ is the design parameters vector that may include geometrical and material properties. The vector $V = [E^T \ H^T]^T$ is the vector of electric- and magnetic-field values over the whole computational domain.
The FDTD discretization of Maxwell’s equations for a problem with nonlinear material properties is given by

\[
\begin{bmatrix}
\varepsilon & 0 \\
0 & -\mu
\end{bmatrix}
\dot{V} + K_c V + \begin{bmatrix}
\sigma^e & 0 \\
0 & 0
\end{bmatrix} V = G(t).
\] (8.2)

In this book, without loss of generality, we consider the electric case where only the permittivity and electric conductivity are functions of the electric field. In this case, we have for a typical nonlinear material inside the computational domain:

\[
\varepsilon_r = C_0^e + C_1^e E + C_2^e E^2.
\] (8.3)

Similarly, the conductivity of a nonlinear material is given by

\[
\sigma^e = C_0^s + C_1^s E + C_2^s E^2.
\] (8.4)

Note that definitions (8.3) and (8.4) both apply also to linear materials with \( C_1^e = C_2^e = C_1^s = C_2^s = 0 \). We define, without loss of generality, the vector of parameters \( p = [C_0^e \ C_1^e \ C_2^e \ C_0^s \ C_1^s \ C_2^s] \) as the vector of all nonlinearity coefficients. A perturbation in any shape parameters is a perturbation in the nonlinearity coefficients.

As the permittivity and the electric conductivity can be both functions of the electric field and the nonlinearity coefficients, we can write (8.2) as

\[
N(p, V) \dot{V} + K_C V + K(p, V) V = G(t),
\] (8.5)

where \( N(p, V) = \begin{bmatrix}
\varepsilon & 0 \\
0 & -\mu
\end{bmatrix} \) and \( K(p, V) = \begin{bmatrix}
\sigma^e & 0 \\
0 & 0
\end{bmatrix} \). It is clear that \( K \) is a symmetric matrix, and \( K_C \) is a constant symmetric matrix. It should also be clear that the matrices \( N(p, V) \) and \( K(p, V) \) are diagonal matrices with the \( j \)th component dependent on only the local field values. We assume that the material properties at one point are not dependent on the fields at other points in space.

The derivation of the nonlinear adjoint variable method (AVM) approach starts by differentiating both sides of (8.5) with respect to the parameter \( p_i \) to get

\[
\frac{\partial N}{\partial p_i} \dot{V} + \frac{\partial (N V)}{\partial V} \frac{\partial V}{\partial p_i} + N \frac{\partial^2 V}{\partial p_i \partial t} + K_c \frac{\partial V}{\partial p_i} + \frac{\partial K}{\partial p_i} V + \frac{\partial (K V)}{\partial V} \frac{\partial V}{\partial p_i} + K \frac{\partial V}{\partial p_i} = 0,
\] (8.6)

where \( \overline{V} \) and \( \overline{V} \) are the nominal values of the vectors \( \partial V / \partial t \) and \( V \), respectively. They are treated as constants during the differentiation operations in (8.6). Note that because we limit our derivation to the nonlinearity coefficients, the derivatives of all matrices are analytically available. Note also that the matrices \( N \) and \( K \) have both explicit dependence on the parameters \( p \) and also an implicit dependence
through the vector of fields $V$. Reorganizing (8.6) to move the known terms to the right-hand side, we get

$$\begin{align*}
N \frac{\partial^2 V}{\partial p_i \partial t} + \frac{\partial (N V)}{\partial V^T} \frac{\partial V}{\partial p_i} + K_c \frac{\partial V}{\partial p_i} + \frac{\partial (K V)}{\partial V^T} \frac{\partial V}{\partial p_i} + K \frac{\partial V}{\partial p} &= - \frac{\partial \psi \dot{V}}{\partial p_i} - \frac{\partial \psi}{\partial p_i} V.
\end{align*}$$

(8.7)

Similar to all our previous AVM derivations, we multiply both sides by the temporal adjoint variable vector $\lambda^T$ and integrate over the simulation time to get

$$\begin{align*}
\int_0^{T_m} \lambda^T \left( N \frac{\partial^2 V}{\partial p_i \partial t} + \frac{\partial (N V)}{\partial V^T} \frac{\partial V}{\partial p_i} + K_c + \frac{\partial (K V)}{\partial V^T} + K \right) \frac{\partial V}{\partial p_i} \, dt &= - \int_0^{T_m} \lambda^T \left( \frac{\partial \psi}{\partial p_i} \dot{V} + \frac{\partial \psi}{\partial p} V \right) \, dt.
\end{align*}$$

(8.8)

Integrating the first term in (8.8) by parts yields

$$\begin{align*}
\lambda^T N \frac{\partial V}{\partial p_i} \bigg|_0^{T_m} - \int_0^{T_m} \left( \lambda^T N + \lambda^T \dot{N} \right) \frac{\partial V}{\partial p_i} \, dt \\
+ \int_0^{T_m} \lambda^T \left( \frac{\partial (N V)}{\partial V^T} + K_c + \frac{\partial (K V)}{\partial V^T} + K \right) \frac{\partial V}{\partial p_i} \, dt \\
= - \int_0^{T_m} \lambda^T \left( \frac{\partial \psi}{\partial p_i} \dot{V} + \frac{\partial \psi}{\partial p} V \right) \, dt.
\end{align*}$$

(8.9)

To nullify the first term, we choose $\lambda(T_m) = 0$. Also, the vector $V$ is assumed to have zero initial condition. Equation (8.9) is thus simplified to

$$\begin{align*}
\int_0^{T_m} \left( - \dot{\lambda}^T N - \lambda^T \dot{N} + \lambda^T \left( \frac{\partial (N V)}{\partial V^T} + K_c + \frac{\partial (K V)}{\partial V^T} + K \right) \right) \frac{\partial V}{\partial x_i} \, dt \\
= - \int_0^{T_m} \lambda^T \left( \frac{\partial \psi}{\partial p_i} \dot{V} + \frac{\partial \psi}{\partial p} V \right) \, dt.
\end{align*}$$

(8.10)

Equating the second term in the objective function (3.1) with the left-hand side of (8.10), we have

$$\begin{align*}
- \dot{\lambda}^T N - \lambda^T \dot{N} + \lambda^T \left( \frac{\partial (N V)}{\partial V^T} + K_c + \frac{\partial (K V)}{\partial V^T} + K \right) &= \left( \frac{\partial \psi}{\partial V} \right)^T.
\end{align*}$$

(8.11)
Reorganizing (8.11), we have

$$-N^T \frac{d\lambda}{dt} - N^T \lambda + \left( K_c + K + \frac{\partial (N \overline{V})^T}{\partial \overline{V}} + \frac{\partial (K \overline{V})^T}{\partial \overline{V}} \right) \lambda = \frac{\partial \psi}{\partial \overline{V}}.$$  

(8.12)

Changing the time variable in (8.12) from the original time variable $t$ to the adjoint time variable $\tau$ with $\tau = T_m - t$, to have

$$N^T \frac{d\lambda}{d\tau} + \left( -\dot{N}^T + K_c + K + \frac{\partial (N \overline{V})^T}{\partial \overline{V}} + \frac{\partial (K \overline{V})^T}{\partial \overline{V}} \right) \lambda = \frac{\partial \psi}{\partial \overline{V}}.$$  

(8.13)

Defining $K^{\lambda} = -\dot{N}^T + K + \left( \frac{\partial (N \overline{V})^T}{\partial \overline{V}} \right) + \left( \frac{\partial (K \overline{V})^T}{\partial \overline{V}} \right)$ as the conductivity matrix of the adjoint problem, the adjoint system is thus given by

$$N \dot{\lambda} + K_c \lambda + K^{\lambda} \lambda = \frac{\partial \psi}{\partial \overline{V}}.$$  

(8.14)

This adjoint system has the same form as the original system (8.5) with two differences. The conductivity matrix is time-dependent and its temporal dependence has to be determined during the original simulation. Also, the matrix $N$ utilized in the adjoint simulation is also time-dependent and is determined during the original simulation.

Once the original response $\overline{V}$ is evaluated using (8.5) and the adjoint response $\lambda$ is evaluated using (8.14) at every time step, the adjoint sensitivity of the objective function with respect to the $i$th parameter, $\forall i$, is given by

$$\frac{\partial F}{\partial \pi_i} = \int_0^{T_m} \frac{\partial \psi}{\partial \pi_i} dt - \int_0^{T_m} \dot{\lambda}^T \left( \frac{\partial \psi}{\partial \pi_i} \right) \overline{V} + \frac{\partial \psi}{\partial \pi_i} \overline{V} dt, \ i = 1, 2, \ldots, n.$$  

(8.15)

### 8.2 Implementation

Equation (8.14) describes the adjoint system used for calculating the sensitivities of the objective function. The matrix $N$ is the same for both the original and adjoint simulation. We should note that though the original simulation is running in the forward direction with respect to the parameter $t$, the adjoint simulation is running in the forward direction with respect to the adjoint time parameter $\tau$. This makes a difference for the matrix $N$ because, unlike the linear case, it is dependent on the temporal field values. The components of $N$ determined during the original simulation are applied backward in time in the adjoint simulation.

The matrix $K_c$ representing the discretization of the curl operator is the same for both the original and adjoint simulations. The matrix $K^{\lambda}$ representing the electric conductivity of the adjoint problem requires special attention. First, this matrix
is time dependent and thus the conductivity at every node is also time dependent.
The first component matrix of $K^l$ is $-\tilde{N}^T$. This matrix is symmetric because $N$ is a diagonal matrix. The matrix $-\tilde{N}$ is given by

$$-\tilde{N} = -\begin{bmatrix} \dot{\varepsilon} & 0_{N\times N} \\ 0_{N\times N} & 0_{N\times N} \end{bmatrix},$$

(8.16)

with the diagonal matrix $\dot{\varepsilon}$ representing the derivatives of the local permittivities with respect to time. Because of the assumed nonlinearity relationship (8.3), the permittivity is an implicit function of time through the local electric field and we can thus write

$$-\dot{\varepsilon} = -\frac{\partial \varepsilon}{\partial t} = -\begin{bmatrix} \frac{\partial \varepsilon_1}{\partial t} & \cdots & \frac{\partial \varepsilon_N}{\partial t} \\ \vdots & \ddots & \vdots \\ \frac{\partial \varepsilon_N}{\partial t} & \cdots & \frac{\partial \varepsilon_1}{\partial t} \end{bmatrix} = -\begin{bmatrix} \frac{\partial \varepsilon_1}{\partial E_1} & \cdots & \frac{\partial \varepsilon_N}{\partial E_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial \varepsilon_1}{\partial E_N} & \cdots & \frac{\partial \varepsilon_N}{\partial E_N} \end{bmatrix},$$

(8.17)

where $\varepsilon_j$ is the permittivity associated with the $j$th component of the matrix $\varepsilon$, and $E_j$ is the corresponding local electric field. Note that we made the assumption that each permittivity component is a function of the electric field component it multiplies. The matrix (8.17) is a diagonal matrix.

The second component matrix of $K^l$ is $K$. This is the same diagonal matrix utilized in the original simulation. This matrix is assumed as per (8.4) to be a nonlinear function of the electric field and it has the form

$$K = \begin{bmatrix} \sigma^e & 0_{N\times N} \\ 0_{N\times N} & 0_{N\times N} \end{bmatrix},$$

(8.18)

where the diagonal conductivity matrix is given by

$$\sigma^e = \begin{bmatrix} \sigma_1^e(E_1) \\ \vdots \\ \sigma_N^e(E_N) \end{bmatrix}.$$ 

(8.19)

In (8.19), we made the assumption that the electric conductivity is a nonlinear function of its corresponding electric field component. Other general dependency forms can also be integrated with the AVM formulation.

The third component matrix of $K^l$ is $\partial(N\tilde{V})^T/\partial V$. Calculating this matrix requires some clarification. First, the product $N\tilde{V}$ is given by

$$N\tilde{V} = \begin{bmatrix} \varepsilon & 0_{N\times N} \\ 0_{N\times N} & \mu \end{bmatrix} \tilde{V} = \begin{bmatrix} \varepsilon & 0_{N\times N} \\ 0_{N\times N} & \mu \end{bmatrix} \begin{bmatrix} \tilde{E} \\ \tilde{H} \end{bmatrix} = \begin{bmatrix} \varepsilon \tilde{E} \\ \mu \tilde{H} \end{bmatrix},$$

(8.20)
where the vector $\vec{eE}$ is expanded as

$$
\vec{eE} = \begin{bmatrix}
\varepsilon_1 \vec{E}_1 \\
\vdots \\
\varepsilon_N \vec{E}_N
\end{bmatrix} = 
\begin{bmatrix}
\varepsilon_1 \\
\ddots \\
\varepsilon_N
\end{bmatrix} \begin{bmatrix}
\vec{E}_1 \\
\vdots \\
\vec{E}_N
\end{bmatrix}.
$$

(8.21)

Note that the bar on top of the variables indicates that these quantities are treated as constants during any differentiation operation. The permeability matrix in (8.20) is assumed a constant and thus its derivative with respect to any variable is zero.

The matrix $\partial(\vec{N}\vec{V})^T / \partial \vec{V}$ is thus given by

$$
\frac{\partial (\vec{N}\vec{V})^T}{\partial \vec{V}} = \frac{\partial \begin{bmatrix} \varepsilon_1 \vec{E} \\ \vdots \\ \varepsilon_N \vec{E} \\ \mu_1 \vec{H} \\ \vdots \\ \mu_N \vec{H} \end{bmatrix}}{\partial \begin{bmatrix} E_1 \\ \ldots \\ E_N \\ H_1 \\ \ldots \\ H_N \end{bmatrix}} = \begin{bmatrix} \vec{D} & 0_{N \times N} \\ 0_{N \times N} & 0_{N \times N} \end{bmatrix},
$$

(8.22)

where the matrix $\vec{D}$ is given by

$$
\begin{bmatrix}
\frac{\partial \varepsilon_1}{\partial E_1} \vec{E}_1 \\
\ddots \\
\frac{\partial \varepsilon_N}{\partial E_N} \vec{E}_N
\end{bmatrix}.
$$

(8.23)

Note that the matrix $\vec{D}$ is identical to the matrix (8.17) but with a negative sign. These two matrices thus cancel one another, and there is no need to include them in our calculations. The last component matrix of $\vec{K}$ is $\partial(\vec{K}\vec{V})^T / \partial \vec{V}$. Evaluating this matrix follows similar steps to (8.20) – (8.23). First, the product $\vec{K}\vec{V}$ is given by

$$
\vec{K}\vec{V} = \begin{bmatrix} \sigma^e & 0_{N \times N} \\ 0_{N \times N} & 0_{N \times N} \end{bmatrix} \vec{V} = \begin{bmatrix} \sigma^e & 0_{N \times N} \\ 0_{N \times N} & 0_{N \times N} \end{bmatrix} \begin{bmatrix} \vec{E} \\ \vec{H} \end{bmatrix} = \begin{bmatrix} \sigma^e \vec{E} \\ 0_{N \times 1} \end{bmatrix}.
$$

(8.24)

where the vector $\sigma^e \vec{E}$ is expanded as

$$
\sigma^e \vec{E} = \begin{bmatrix} \sigma^e_1 \vec{E}_1 \\
\vdots \\
\sigma^e_N \vec{E}_N \end{bmatrix} = 
\begin{bmatrix}
\sigma^e_1 \\
\ddots \\
\sigma^e_N
\end{bmatrix} \begin{bmatrix}
\vec{E}_1 \\
\vdots \\
\vec{E}_N
\end{bmatrix}.
$$

(8.25)
The bar over the field quantities indicates that these quantities are treated as constants in any differentiation operation. The matrix $\partial(KV)^T/\partial V$ is thus given by

$$
\frac{\partial (KV)^T}{\partial V} = \frac{\partial}{\partial \begin{bmatrix} \sigma^e E \\ 0_{N\times1} \end{bmatrix}} = \begin{bmatrix}
\partial \sigma^e_1 E_1 \\
\vdots \\
\partial \sigma^e_N E_N \\
0_{N\times1}
\end{bmatrix}
\begin{bmatrix}
E_1 & \ldots & E_N & H_1 & \ldots & H_N
\end{bmatrix} = \begin{bmatrix}
J & 0_{N\times N} \\
0_{N\times N} & 0_{N\times N}
\end{bmatrix},
$$

(8.26)

where the matrix $J$ is given by

$$
J = \begin{bmatrix}
\frac{\partial \sigma^e_1}{\partial E_1} E_1 \\
\vdots \\
\frac{\partial \sigma^e_N}{\partial E_N} E_N
\end{bmatrix}.
$$

(8.27)

The derivatives of the relative permittivity and electric conductivity utilized in all the formulae can be deduced from (8.3) and (8.4) as

$$
\frac{\partial \varepsilon_j}{\partial E_j} = \varepsilon_0 \left( C^e_1 + 2C^e_2 E_j \right), \quad \forall j.
$$

(8.28)

$$
\frac{\partial \sigma^e_j}{\partial E_j} = \varepsilon_0 \left( C^e_1 + 2C^e_2 E_j \right), \quad \forall j.
$$

(8.29)

As all the matrices comprising $K^d$ are diagonal, the matrix $K^d$ is also a diagonal matrix. Its diagonal components are the effective electric conductivities of the adjoint problem. The $j$th component of this matrix, $j = 1, 2, \ldots, N$, is thus given by

$$
(K^d)_j = \sigma^e_j + \frac{\partial \sigma^e_j}{\partial E_j} E_j.
$$

(8.30)

These components are calculated and stored at every step of the original simulation and then utilized in the adjoint simulation.

**Example 8.1** To illustrate the basic nonlinear AVM concept, we first consider the circuit shown in Figure 8.1. The capacitance is a nonlinear function of its terminal voltage $V_c$. This nonlinear dependency is given by

$$
C = C_0 + C_2 V_c^2,
$$

where $C_0 = 15 \mu F$ and $C_2 = 2 \mu F/V^2$. The supply voltage is given by

$$
V_s(t) = e^{-\left((t-t_0)/(2\pi)\right)}.
$$
with $t_0 = 0.5$ s and $\nu = 0.1$. The target is to evaluate the sensitivities of the energy dissipated in the resistor $R_1$ with respect to the parameters $p = [C_0 \ C_2 \ R_1 \ R_2]^T$. This energy is given by the integral

$$F = \int_0^{T_m} \frac{(V_s - V_c)^2}{R_1} dt.$$ 

To estimate the adjoint sensitivities, we first put the governing equations of this circuit in a form similar to (8.5). Using loop analysis gives

$$V_s = i_s R_1 + V_c$$

(8.31)

Writing Kirchhoff’s current law at the only node yields

$$i_s = \frac{V_c}{R_2} + \frac{d(CV_c)}{dt}.$$ 

As the capacitor is nonlinear, the capacitance is not a constant value and the current through the capacitor is $i_C = \frac{d(CV_c)}{dt}$. Eliminating $i_s$ using (8.31) gives

$$\frac{(V_s - V_c)}{R_1} = \frac{V_c}{R_2} + \frac{d(CV_c)}{dt}.$$ 

(8.32)

Reorganizing (8.32) gives

$$R_1 \frac{d(CV_c)}{dt} + \left( 1 + \frac{R_1}{R_2} \right) V_c = V_s.$$ 

(8.33)

Because $C = C_0 + C_2 V_c^2$, the term $\frac{d(CV_c)}{dt}$ is given by

$$\frac{d(CV_c)}{dt} = \frac{d(C_0V_c + C_2 V_c^3)}{dt} = \left( C_0 + 3C_2 V_c^2 \right) \frac{dV_c}{dt}.$$ 

(8.34)

Substituting from (8.34) in (8.33), we have

$$R_1 \left( C_0 + 3C_2 V_c^2 \right) \frac{dV_c}{dt} + \left( 1 + \frac{R_1}{R_2} \right) V_c = V_s.$$ 

(8.35)
Comparing (8.35) with (8.5) gives $N = R_1(C_0 + 3C_2V_c^2)$, $K = (1 + R_1/R_2)$, $K_c = 0$, and $G(t) = V_s(t)$. Note that the system (8.35) has only one state variable, and thus all the system matrices are scalars. As per our derivation, the corresponding adjoint system is given by

$$N \frac{d\lambda}{dt} + \left( K + \frac{\partial(KV_c)}{\partial V_c} \right) \lambda = \frac{\partial \psi}{\partial V_c}$$

(8.36)

where these terms are

$$\frac{\partial(KV_c)}{\partial V_c} = V_c \frac{\partial K}{\partial V_c} = 0,$$

$$\frac{\partial \psi}{\partial V_c} = \frac{2(V_s - V_c)}{R_1}.$$

All these quantities are calculated during the original simulation and stored. Once the original and adjoint simulations are carried out, the formula (8.15) is then invoked to estimate the sensitivities. The matrix derivatives in (8.15) are given by

$$\frac{\partial^e N}{\partial C_0} = R_1, \quad \frac{\partial^e N}{\partial C_2} = 3R_1V_c^2, \quad \frac{\partial^e N}{\partial R_1} = (C_0 + 3C_2V_c^2), \quad \frac{\partial^e N}{\partial R_2} = 0,$$

(8.37)

and

$$\frac{\partial^e K}{\partial C_0} = 0, \quad \frac{\partial^e K}{\partial C_2} = 0, \quad \frac{\partial^e K}{\partial R_1} = \frac{1}{R_2}, \quad \frac{\partial^e K}{\partial R_2} = -\frac{R_1}{R_2^2}.$$

(8.38)

It should be clear in (8.37) that these explicit derivatives may be time dependent, and their values are determined as a post-processing of the original simulation.

MATLAB® Listing M8.1 shows the first part of the implementation of the nonlinear AVM approach to this nonlinear circuit. The second part, which involves postprocessing to estimate the sensitivities, is shown in MATLAB Listing M8.2. This code solves the original system through finite differences to get the forward time marching scheme:

$$R_1\left(C_0 + 3C_2V_c^2\right) \frac{dV_c}{dt} + \left(1 + \frac{R_1}{R_2}\right)V_c = V_s,$$

$$R_1\left(C_0 + 3C_2\left(V_c^k\right)^2\right) \frac{V_c^{k+1} - V_c^k}{\Delta t} + \left(1 + \frac{R_1}{R_2}\right)V_c^k = V_s^k,$$

(8.39)

where $V_c^j = V_c(k\Delta t)$ and $V_s^j = V_s(k\Delta t)$. The finite difference formula can be organized into the time marching scheme:

$$V_c^{k+1} = V_c^k - \Delta t \left(\frac{1}{R_2} + \frac{1}{R_2}\right) \frac{V_c^k}{C_0 + 3C_2\left(V_c^k\right)^2} + \Delta t \frac{V_s^k}{R_1\left(C_0 + 3C_2\left(V_c^k\right)^2\right)}.$$

(8.40)
MATLAB Listing M8.1

```
% MATLAB Listing M8.1
% This circuit simulates example 8.1
close all
clear all
tic

global dt time Vs;
dt=1E-4;
time=0:dt:; % run the simulation for one second
number_of_time_steps=length(time); % number of time steps
Vs=exp(-tome-0.5).*2/(2*(0.1)^2)); % exp(-tome-0.5)/2*sigma^2)
FD_step=0.02; % perturbation percentage
C0=15E-6; % value of C0
dC0=p0*FD_step; % perturbation in C0
R1=1000; % value of resistance R1
dR1=R1*FD_step; % perturbation in R1
R2=1000; % value of R2
dR2=R2*FD_step; % perturbation of R2
C2=2E-6; % value of C2
dc2=C2*FD_step; % perturbation of C2

% Nonlinear Original system simulation:-
Vc=zeros(1,number_of_time_steps); % values of the capacitor voltages
dVc_dt=zeros(1,number_of_time_steps); % derivative of capacitor voltage
depsi_dVc=zeros(1,number_of_time_steps); % derivative of objective function

depsi_dVc(1)=2*(Vc(1)-Vs(1))/R1;
for ind=1:(number_of_time_steps-1)
    carry out finite difference scheme
    C_prime=(C0+C2+Vc(ind)*Vc(ind));
    Vc(ind+1)=((1-(dt*(R1+R2)/(R1*R2*C_prime)))*Vc(ind)+dt*Vc(ind)/(R1*R2*C_prime));
    % update excitation of adjoint problem
    depsi_dVc(ind)=2*(Vc(ind+1)-Vc(ind+1))/R1; % store adjoint excitation
    dVc_dt(ind+1)=(Vc(ind+1)-Vc(ind))/dt; % derivative of voltage
end

% Nonlinear Adjoint System:
lambda=zeros(1,number_of_time_steps);
for ind=1:(number_of_time_steps-1)
    C_prime=(C0+C2+Vc(number_of_time_steps-ind+1));
    % notice time reversal
    K=(R1+R2)/(R1*R2)); % K matrix
    lambda(ind+1)=((1-(dt*K/C_prime))*lambda(ind));
    % sensitivity evaluations
    dVc_dt(ind+1)=(Vc(ind+1)-Vc(ind))/dt;
end
```
This update formula is implemented in MATLAB Listing M8.1 for the original simulation using the command

\[ V_c(\text{ind}+1) = ((1 - (dt*(R1+R2)/(R1*R2*C_{\text{prime}}))) \times V_c(\text{ind})) + (dt*V_s(\text{ind})/(R1*C_{\text{prime}})); \]

where the variable \( C_{\text{prime}} \) denotes the quantity \( (C_0 + 3C_2(V_c^k)^2) \). The same update formula (8.40) is also utilized for the adjoint problem with a different source and a different \( K \) matrix. The following commands implement this update for the adjoint problem

\[
K = ((R1+R2)/(R1*R2)); \quad \% \text{K matrix}
\]

\[
\lambda(\text{ind}+1) = ((1 - (dt*K/C_{\text{prime}})) \times \lambda(\text{ind})) + dt*depsi_dVc(\text{number_of_time_steps-ind+1})/(R1*C_{\text{prime}}).
\]

One important point to note is that the adjoint problem is running backward in time with respect to the original problem time \( t \). This explains why all quantities evaluated during the original simulation are used in a reverse order. The original field quantities utilized at the time instant with an index \( \text{ind} \) of the adjoint simulation has a time index \( \text{number_of_time_steps-ind+1} \).

Once the original and adjoint simulations are done, the code carries out the postprocessing part shown in MATLAB Listing M8.2. The commands

\[
\text{dNe_dC0} = R1*\text{ones(1,number_of_time_steps)}; \quad \text{dNe_dC2} = 3*R1*(V_c.^2);
\]

\[
\text{dNe_dR1} = (C0+3*C2*V_c.^2); \quad \text{dNe_dR2} = \text{zeros(1,number_of_time_steps)};
\]

evaluate the explicit system matrix derivatives given by (8.37). Because all these explicit matrices are diagonal ones, we store their diagonal components in vectors. The same applies to the \( K \) matrix of the original problem. The commands

\[
\text{dF_dC0_AVM} = -dt*((\text{dNe_dC0}.*dVc_dt+dKe_dC0.*Vc)*\text{flipud}(\lambda'));
\]

\[
\text{dF_dC2_AVM} = -dt*((\text{dNe_dC2}.*dVc_dt+dKe_dC2.*Vc)*\text{flipud}(\lambda'));
\]

\[
\text{dF_dR1_AVM} = -dt*((\text{dNe_dR1}.*dVc_dt+dKe_dR1.*Vc)*\text{flipud}(\lambda'));
\]

\[
\text{dF_dR2_AVM} = -dt*((\text{dNe_dR2}.*dVc_dt+dKe_dR2.*Vc)*\text{flipud}(\lambda'));
\]

implement the inner product in (8.15) for all four parameters. Because the adjoint voltage \( \lambda \) is evaluated in the forward direction with respect to the adjoint time variable \( \tau \), its values have to be reversed using the command \text{flipud}(). Because the resistance \( R_1 \) appears explicitly in the energy integral of the objective function \( F \), it has an explicit derivative part. This explicit dependence is given by

\[
\frac{\partial F}{\partial R_1} = \int_0^{T_m} \left( \frac{(V_s - V_c)^2}{R_1^2} \right) dt.
\]
This explicit derivative is evaluated and added to the implicit part through the commands

\[
depsi_{dR1} = -(Vs-Vc)^2/R1^2; \\
dF_{dR1}_{AVM} = dF_{dR1}_{AVM} + (dt*\text{sum}(depsi_{dR1})).
\]

When the AVM code is executed, the output is given by

\[
\begin{align*}
dF_{dC0}_{AVM} &= 0.050758752053209 \\
dF_{dC2}_{AVM} &= 0.013601140945411 \\
dF_{dR1}_{AVM} &= -1.587568722731625e-10 \\
dF_{dR2}_{AVM} &= -4.375938998662403e-08.
\end{align*}
\]

Estimating the sensitivities using central finite differences (CFD) with perturbations of 2% of the nominal parameter values gives the sensitivities

\[
\begin{align*}
dF_{dC0}_{CFD} &= 0.050544300392052 \\
dF_{dC2}_{CFD} &= 0.013543964713203 \\
dF_{dR1}_{CFD} &= -1.579927673481658e-10 \\
dF_{dR2}_{CFD} &= -4.376812465975869e-08.
\end{align*}
\]

We note that there is a good match between the two estimates. Figure 8.2 shows a comparison between the AVM sensitivities and the CFD sensitivities for a sweep of the parameter $C_2$. This parameter controls the nonlinearity level of the problem. It is obvious that the two estimates have good match. The AVM approach requires only one extra adjoint simulation per gradient evaluation whereas the CFD requires eight extra simulations.
Example 8.2 To further illustrate the nonlinear AVM approach, we consider the one-dimensional nonlinear dielectric slab example shown in Figure 8.3. The dielectric constant at every cell in the slab is given by $\varepsilon_r = C_0 + C_1 E + C_2 E^2$ where $E$ is the local electric field. The considered parameters for this example are the length of the discontinuity in the $z$-direction and the permittivity coefficients

$$J_x$$

Mur’s BC

$20\Delta z$

$100\Delta z$

$L$

$\varepsilon_r = C_0 + C_1 E + C_2 E^2$

$580\Delta z$

Mur’s BC

Figure 8.2 The sensitivities of the objective function of Example 8.1: nonlinear adjoint sensitivity analysis ($\circ$) as compared to CFD (—)

Figure 8.3 The domain of the dielectric slab example: a current source is excited at $z = 20\Delta z$ and the objective function is observed at $z = 580\Delta z$
of the discontinuity material \( p = [L \ C_0 \ C_1 \ C_2]^T \). The slab is assumed to be lossy with a constant electric conductivity \( \sigma = 0.04 \). The nonlinearity coefficients considered in this example are \( C_0 = 3 \), \( C_1 = 50 \), and \( C_2 = 100 \). The length \( L \) is allowed to vary between \( 0.1 \lambda \) and \( \lambda \), where \( \lambda \) is the wavelength inside the corresponding linear slab. The utilized discretization is \( \Delta z = 0.5 \) mm. The domain is excited with a Gaussian-modulated electric current density source at \( z = 20\Delta z \). The domain has a length of \( 600\Delta z \). The considered objective function is given by the energy function (3.22) where the electric field in (3.22) is measured at \( z = 580\Delta z \).

The MATLAB implementation for this problem starts by calling a settings file containing all the information about the problem. This settings file is given in MATLAB Listing M8.3. This settings file determines the values of all the parameters. The perturbations utilized in evaluating the system derivatives and in
estimating the central finite difference approximations are defined by the commands

```matlab
dp1 = d_z;  \% the amount of perturbation in p1
dp2 = 0.02*C0_eps; \% perturbation in C0
dp3 = 0.02*C1_eps; \% perturbation in C1
dp4 = 0.02*C2_eps; \% perturbation in C2.
```

The dielectric slab starts at $100\Delta z$ and has a length of $L$. The main script of the code is the script `avm_slab_nonlinear`. This script is the one that invokes the settings file, the original simulation, and the adjoint simulation. This script is shown in MATLAB Listing M8.4. The matrix `permittivity_coefficients_matrix` contains the nonlinearity coefficient for all nodes in the domain. It has three rows and 600 columns. For all linear cells, the second and third rows of this matrix are zeros. The commands

```matlab
permittivity_coefficients_matrix(1, slab_start:slab_stop) = C0_eps;
permittivity_coefficients_matrix(2, slab_start:slab_stop) = C1_eps;
permittivity_coefficients_matrix(3, slab_start:slab_stop) = C2_eps;
```

set the nonlinearity coefficients for the FDTD cells within the discontinuity. The matrix `conductivity_coefficient_matrix` holds the nonlinearity coefficients related to the conductivity. Because we assume that the conductivity is linear in this example; this matrix has zero second and third rows.

Unlike the 1D AVM codes addressed earlier for isotropic problems, the original and adjoint simulations are different from one another. The original simulation is invoked through the command

```matlab
td_original = fdtd_1d_nonlinear_lossy(d_z, d_t, number_of_time_steps, epsilon, sigma_e, permittivity_coefficients_matrix, conductivity_coefficient_matrix, source_pulse, port_in).
```

The permittivity utilized within this code and the updating coefficients utilized in the FDTD simulator are adjusted at every time step according to the local field value at the previous time step. We allowed a delay of one time step between the change in the local electric field and the change in the corresponding material properties. This delay of $1\Delta t$ is physical as most materials take some time to respond to a change in the electric field [6]. Other approaches for including the nonlinearities have been addressed, for example, [2,3,7]. The commands

```matlab
eps_r_new = eps_r + permittivity_coefficients(2,:).*Ey + permittivity_coefficients(3,:).*Ey.^2;
sigma_e_new = sigma_e + conductivity_coefficients(2,:).*Ey + conductivity_coefficients(3,:).*Ey.^2
```
evaluate the new relative permittivity and conductivity for the whole domain at the next time step using the domain electric field at the current time step. These values are then used to update the FDTD coefficients $k_d$, $k_h$, $k_e$, and $k_j$ used in the FDTD time marching loop. The code for the original simulation is shown in MATLAB Listing M8.4

```matlab
% MATLAB Listing M8.4
%
% A subset of the code of the avm_slab_nonlinear script
% td_original = ftdtd_1d_nonlinear_lossy(d_z, d_t, number_of_time_steps, epsilon, sigma, permittivity_coefficients_matrix, conductivity_coefficients_matrix, source_pulse, port_in); % original simulation
% store original fields in perturbation domains
dir_pl_f = td_original(:,psi_pl_f);
dir_pl_b = td_original(:,psi_pl_b);
dir_p2_f = td_original(:,psi_p2_f);
dir_p3_f = td_original(:,psi_p3_f);
dir_p4_f = td_original(:,psi_p4_f);
% determine the adjoint source (i.e. d(epsay) dV)
adjoint_source = -1.0 * flipud(td_original(:,port_out)); % notice
% time reversal of excitation
% execute the adjoint simulation with the nonlinear matrices
% original fields are used reversed in time
% td_adjoint = ftdtd_1d_nonlinear_lossy_adjoint(d_z, d_t, number_of_time_steps, epsilon, sigma, permittivity_coefficients_matrix, conductivity_coefficients_matrix, flipud(td_original),adjoint_source, port_out);
% store adjoint responses within each of the perturbation volumes
% notice time reversal to make all fields forward with respect to t
adj_pl_f = flipud(td_adjoint(:,psi_pl_f));
adj_pl_b = flipud(td_adjoint(:,psi_pl_b));
adj_p2_f = flipud(td_adjoint(:,psi_p2_f));
adj_p3_f = flipud(td_adjoint(:,psi_p3_f));
adj_p4_f = flipud(td_adjoint(:,psi_p4_f));
% compute the residual vectors:
dN_dp1_f = EPS0 * ((epsilon_slab - epsilon_air) +
(permittivity_coefficients_matrix(2,slab_stop)*dir_pl_f) +
(permittivity_coefficients_matrix(3,slab_stop)*dir_pl_f.^2)) / dpl;
dK_dp1_f = (sigma_e_slab - sigma_e_slab) / dpl;
residue_pl = (dN_dp1_f.*diff(dir_p1_f,0:dir_p1_f(end,:))/d_t) + (dK_dp1_f*dir_p1_f);
% second parameter
dN_dp2 = EPS0; dK_dp2 = 0;
residue_p2 = (dN_dp2.*diff(dir_p2_f,0:dir_p2_f(end,:))/d_t) + (dK_dp2*dir_p2_f);
% third parameter
dN_dp3 = EPS0 * dir_p3_f; dK_dp3 = 0;
residue_p3 = (dN_dp3.*diff(dir_p3_f,0:dir_p3_f(end,:))/d_t) + (dK_dp3*dir_p3_f);
% fourth parameter
dN_dp4 = EPS0 * dir_p4_f.^2; dK_dp4 = 0;
residue_p4 = (dN_dp4.*diff(dir_p4_f,0:dir_p4_f(end,:))/d_t) + (dK_dp4*dir_p4_f);
% compute the adjoint sensitivities
sens_pl_favm = -sum(adj_pl_f.*residue_pl)*d_t;
sens_p2_favm = -sum(adj_p2_f.*residue_p2)*d_t;
sens_p3_favm = -sum(adj_p3_f.*residue_p3)*d_t;
sens_p4_favm = -sum(adj_p4_f.*residue_p4)*d_t;
```
MATLAB Listing M8.5

```matlab
function td = ffdtd_1d_nonlinear_lossy_adjoint(d_z, d_t, number_of_time_steps, 
    eps_r, sigma_e, permittivity_coefficients, conductivity_coefficients, 
    source_signal, source_position)
    % define physical constants
    M00 = 4 * pi * le-7;  C = 3.0E8; %299792458.0;
    EPS0 = 1.0 / (C * C * M00);
    % initialize the fields and outputs
    td = zeros(number_of_time_steps, length(eps_r));
    Ey = zeros(1, length(eps_r));
    Hx = zeros(1, length(eps_r) - 1);
    % compute the FDTD update coefficients
    k_h = d_t / (d_z * M00);
    templ = d_t * sigma_e ./ (2 * EPS0 * eps_r);
    temp2 = d_t ./ (d_z * EPS0 * eps_r);
    k_d = (1 - templ) ./ (1 + templ);
    k_e = temp2 ./ (1 + templ);
    k_j = d_t / (EPS0 * eps_r(source_position));
    % compute the constants for the 1st order MUT's
    m00_l = C / sqrt(eps_r(1));
    m00_r = C / sqrt(eps_r(end));
    mur_l = (m00_l * d_z - d_z) / (m00_l * d_t + d_z);
    mur_r = (m00_r * d_z - d_z) / (m00_r * d_t - d_z);
    % start the FDTD algorithm
    for time_index = 1:number_of_time_steps
        % store previous fields for Mur's ABCs
        prev_left = Ey(2);
        prev_right = Ey(end - 1);
        % apply FDTD equations
        Hx = Hx - diff(Ey) .* k_h;
        Ey(2:end-1) = (k_d(2:end-1) .* Ey(2:end-1)) - 
                        (k_e(2:end-1) .* diff(Hx));
        Ey(source_position) = Ey(source_position) - k_j .* 
                                source_signal(time_index);
        % impose Mur's ABCs
        Ey(1) = prev_left + mur_l .* (Ey(2) - Ey(1));
        Ey(end) = prev_right + mur_r .* (Ey(end - 1) - Ey(end));
        % update FDTD coefficients for the next time step
        eps_r_new = eps_r + permittivity_coefficients(2, :) .* Ey
                                + permittivity_coefficients(3, :) .* Ey
                                + conductivity_coefficients(3, :) .* Ey
                                + conductivity_coefficients(3, :) .* Ey
                                + conductivity_coefficients(2, :);  
        templ = d_t * sigma_e_new ./ (2 * EPS0 * eps_r_new);
        temp2 = d_t ./ (d_z * EPS0 * eps_r_new);
        k_d = (1 - templ) ./ (1 + templ);
        k_e = temp2 ./ (1 + templ);
        k_j = d_t / (EPS0 * eps_r_new(source_position));
        % store the electric field
        td(time_index,:) = Ey();
    end
end
```

MATLAB Listing M8.5. This listing returns the matrix `td_original` that stores the direct (original) fields at all cells for all time steps.

Once the original simulation is executed, and the original fields in the perturbation regions are stored; the code invokes the adjoint simulation by executing the function

```matlab
td_adjoint = ffdtd_1d_nonlinear_lossy_adjoint(d_z, d_t, 
number_of_time_steps, epsilon, sigma_e, permittivity_coefficients_matrix, 
conductivity_coefficient_matrix, 
flipud(td_original),adjoint_source, port_out).
```
Unlike the original simulation, the adjoint simulation receives as an input the original fields. These fields are reversed in time through the command flipud(). As per (8.14), the matrices $N$ and $K$ utilized in the adjoint simulation are time-dependent matrices. Their values depend on the temporal values of the original fields not the adjoint fields. This explains why the temporal original field values are sent as inputs to the adjoint simulation.

The adjoint simulator is identical to the code shown in MATLAB Listing M8.6 except for the following commands implementing the update equations of nonlinear permittivity and conductivity

```matlab
eps_r_new = eps_r + permittivity_coefficients(2,:) .* reversed_original_field(time_index,:) +
            permittivity_coefficients(3,:) .* reversed_original_field(time_index,:).^2;
sigma_e_new = sigma_e + conductivity_coefficients(2,:) .* reversed_original_field(time_index,:) +
            conductivity_coefficients(3,:) .* reversed_original_field(time_index,:).^2.
```

These equations use the reversed original field values to update the adjoint permittivity and conductivity.
Once the original and adjoint simulations have been carried out, the code moves to evaluate the original simulation residues. The commands

\[ \frac{dN}{dp_1} = \frac{\epsilon_0 \left( (\epsilon_{\text{slab}} - \epsilon_{\text{air}}) + (\text{permittivity\_coefficients\_matrix}(2,\text{slab\_stop}) \times \text{dir\_p1}_f) + (\text{permittivity\_coefficients\_matrix}(3,\text{slab\_stop}) \times \text{dir\_p1}_f^2) \right)}{dp_1}; \]

\[ \frac{dK}{dp_1} = \frac{(\sigma_{\epsilon_{\text{slab}}} - \sigma_{\epsilon_{\text{air}}})}{dp_1}; \]

\[ \text{residue\_p1} = \left( \frac{dN}{dp_1} \right) \times \left[ \text{diff}(\text{dir\_p1}_f); 0-\text{dir\_p1}_f(\text{end}) \right] / d_t + \left( \frac{dK}{dp_1} \times \text{dir\_p1}_f \right) \]

evaluate the residues corresponding to the first parameter \( L \). Note that perturbing this parameter in the forward direction by \( 1\Delta z \) affects the permittivity and permeability of only one cell. For this cell, the material parameters change from that of air to that of the dielectric. It should be also noted that the dielectric is nonlinear and its permittivity is a function of the electric field. The field at the cell just outside the discontinuity is used to evaluate the new value of the permittivity. The derivative of the matrix \( N \) yields a vector because only one cell is perturbed.

The commands

\[ \frac{dN}{dp_2} = \epsilon_0; \frac{dK}{dp_2} = 0; \]

\[ \text{residue\_p2} = \left( \frac{dN}{dp_2} \right) \times \left[ \text{diff}(\text{dir\_p2}_f); 0-\text{dir\_p2}_f(\text{end},:) \right] / d_t + (\frac{dK}{dp_2} \times \text{dir\_p2}_f); \]

\[ \frac{dN}{dp_3} = \epsilon_0 \times \text{dir\_p3}_f; \frac{dK}{dp_3} = 0; \]

\[ \text{residue\_p3} = \left( \frac{dN}{dp_3} \right) \times \left[ \text{diff}(\text{dir\_p3}_f); 0-\text{dir\_p3}_f(\text{end},:) \right] / d_t + (\frac{dK}{dp_3} \times \text{dir\_p3}_f); \]

\[ \frac{dN}{dp_4} = \epsilon_0 \times \text{dir\_p4}_f^2; \frac{dK}{dp_4} = 0; \]

\[ \text{residue\_p4} = \left( \frac{dN}{dp_4} \right) \times \left[ \text{diff}(\text{dir\_p4}_f); 0-\text{dir\_p4}_f(\text{end},:) \right] / d_t + (\frac{dK}{dp_4} \times \text{dir\_p4}_f); \]

evaluate the residues for the next three parameters \( C_0, C_1, \) and \( C_2 \). Note that \( C_1 \) and \( C_2 \) are multiplied by powers of the electric field. All cells of the slab are affected by the change in these parameters. The residues associated with these parameters are thus matrices and not vectors as for the length \( L \). These residues are then multiplied by the adjoint fields at every time step and the result is then integrated over time.

For a length of \( L = 0.1\lambda \), the adjoint sensitivities are given by

\[ \text{sens\_p1\_favn} = \]
\[ -3.256861383607839e-13 \]

\[ \text{sens\_p2\_favn} = \]
\[ -2.73621307779329e-16 \]

\[ \text{sens\_p3\_favn} = \]
\[ -6.674932779497943e-21 \]

\[ \text{sens\_p4\_favn} = \]
\[ -3.326236613116380e-21 \]
Evaluating the sensitivities for the same set of parameters using CFD yields the result

\[
\begin{align*}
sens_{p1\_cfd} &= -3.226849657911287e-13 \\
sens_{p2\_cfd} &= -2.828236673117062e-16 \\
sens_{p3\_cfd} &= -6.590829772214661e-21 \\
sens_{p4\_cfd} &= -3.443825601836987e-21
\end{align*}
\]

which matches well the AVM sensitivities. The AVM sensitivities require one extra simulation whereas the CFD sensitivities require eight extra simulations. Figure 8.4
shows the sensitivities for a sweep of the parameter $L$. The AVM sensitivities are reasonably matched with the CFD ones. For some parameters, the match is better than the other parameters depending on the nonlinearity of the objective function with respect to these parameters.

**Example 8.3** We consider the 2D structure shown in Figure 8.5. An electric current filament excites an electromagnetic wave at point $Q$. The field is probed at point $P$. A nonlinear discontinuity with nonlinear permittivity exists with parameters $C_0 = 1.5$, $C_1 = 20$, and $C_2 = 5.0$. Our target is to find the sensitivity of the frequency domain transfer function:

$$S_{PQ}(f) = \frac{\tilde{E}_P(f)}{\tilde{E}_{\text{ref}}(f)}$$

where $\tilde{E}_P(f)$ is the measured spectrum at the point $P$ at frequency $f$ with the discontinuity present and $\tilde{E}_{\text{ref}}(f)$ is the reference spectrum measured at the same point at frequency $f$ with the discontinuity not present. The sensitivities of the real and imaginary parts of $S_{PQ}(f)$ are to be evaluated over the frequency band 3.5 to 6.5 GHz.

The code implementing the AVM approach starts, similar to most previously shown algorithms, by the script `TestAVM`. This script invokes both the original and adjoint simulations. The original simulation defines the materials, the discontinuities, the boundaries, and the sources. The nonlinear material is defined through the commands

```matlab
material_types(4).eps_r = 1.5;
material_types(4).mu_r = 1;
```

![Figure 8.5](image) **Figure 8.5** The geometry of Example 8.3: a nonlinear discontinuity is placed between the excitation at point $Q$ and the observation point at point $P$. 
material_types(4).sigma_e = 0;
material_types(4).sigma_m = 0;
material_types(4).color = [0 0 1];
material_types(4).nonlinear = true;
material_types(4).C0 = material_types(4).eps_r;
material_types(4).C1 = 20.0;
material_types(4).C2 = 5.0;

where the nonlinearity coefficients are given by the parameters $C_0$, $C_1$, and $C_2$. The parameter $\text{eps}_r$ represents the relative permittivity value when the local electric field is zero. The same functions are used to create all the necessary storage. In this example, we consider only the four parameters $p = [C_0 \; C_1 \; C_2 \; \sigma^c]^T$.

The script \texttt{run_original_fdtd_time_marching_loop_2d} carries out the original simulation. This script is shown in MATLAB Listing M8.6. There are a number of differences between this loop and the similar loop for the isotropic and linear case. First, the stamps representing the derivative of the matrices $N$ and $K$ with respect to the parameters are functions of time. The script \texttt{update_stamps} is invoked at every time step to calculate the derivative of the system matrices. This script is shown in MATLAB Listing M8.7. In this script, the controlling fields in the nonlinear discontinuity are retrieved through the commands

\[
i = \text{get_node_indices}(\text{rectangles(rectangle_index), fdtd_domain});
\]
\[
is = ni.is; js = ni.js; ie = ni.ie; je = ni.je;
\]
\[
\text{controlling_field} = \text{Ez(is:ie+1,js:je+1)}.
\]

The matrix of controlling fields is then used to set the stamps associated with the different parameters.

For the parameter $C_0$, which represents the relative dielectric constant for the linear case, the stamp calculation is the same as in linear media. For the parameter $C_1$, we first calculate the derivative of the permittivity with respect to this parameter. The non-interpolated derivative for this parameter is given by the command

\[
\text{initial_stamp} = \text{eps}_0 \times \text{controlling_field}(1: \text{N}_x, 1: \text{N}_y).
\]

As explained in previous chapter, this stamp is adjusted for circumference field components to take into account the interpolation between the dielectric constants of the discontinuity and the surrounding air. The function \texttt{get_material_stamp()} creates the stamp from the non-interpolated derivative. Similarly, for the parameter $C_2$, the non-interpolated derivative is given by

\[
\text{initial_stamp} = \text{eps}_0 \times \text{controlling_field}(1: \text{N}_x, 1: \text{N}_y) \times \text{controlling_field}(1: \text{N}_x, 1: \text{N}_y).
\]
After the stamps have been determined, the script `store_original_fields` evaluates the residues at the current time step using the current stamps and stores them. The script `update_nonlinear_permittivities` (shown in MATLAB Listing M8.8) then evaluates the permittivities for the next time step using the updated electric fields. The current permittivities are stored for the adjoint simulation. This is achieved through the command

\[ \text{original} \_\text{epsilon} \_r(\text{nonlinear} \_\text{index}, \text{number} \_\text{of} \_\text{time} \_\text{steps}-\text{time} \_\text{step}+1, :, :) = \text{epsilon} \_r. \]
The reader should note that these relative permittivities are stored in reverse order. They will be used to construct the matrices of the adjoint problem that is running backward in time. This explains why the relative permittivities calculated at the time step \texttt{time\_step} are stored at the location \texttt{number\_of\_time\_steps-time\_step+1}.

MATLAB Listing M8.8

```
% This file updates the rectangles with nonlinear permittivities at every time step

nonlinear_index=0; % index of nonlinear materials
for rectangle_index = 2:number_of_rectangles % repeat for % all rectangles
    material_index = rectangles(rectangle_index).material_type;
    % get material type of current rectangle
    material = material_types(material_index); % get material type
    if material.nonlinear % is this a nonlinear material?
        % convert rectangles end coordinates to node indices
        ni = get_node_indices(rectangles(rectangle_index), ftdt_domain);
        is = ni.is; js = ni.js;
        ie = ni.ie; je = ni.je;
        controlling_field=Ez(is(js,je)); % get controlling fields
        % evaluate new permittivity distribution
        epsilon_r=material.C1+material.C2*controlling_field;
        nonlinear_index=nonlinear_index+1; % increment index
        original_epsilon_r(nonlinear_index, number_of_time_steps-
            time_step+1, :, :) = epsilon_r; % store for adjoint simulation
        eps_r_z(is(js,je)) = epsilon_r; % copy all values
        % interpolation
        eps_r_z(is(js+1,je)) = 0.5*eps_r_z(is(js+1,je))+0.5;
        eps_r_z(is(js,je+1)) = 0.5*eps_r_z(is(js,je+1))+0.5;
        eps_r_z(is(js+1,je+1)) = 0.5*eps_r_z(is(js+1,je+1))+0.5;
        % now the 4 corners
        eps_r_z(is(js)) = 0.25*eps_r_z(is(js),je)+0.75;
        eps_r_z(is(js),je) = 0.25*eps_r_z(is(js),je)+0.75;
        eps_r_z(is(js),je) = 0.25*eps_r_z(is(js),je)+0.75;
        eps_r_z(is(js),je) = 0.25*eps_r_z(is(js),je)+0.75;
        % update coefficients
        Ceze(is(js,je)) = (2*eps_r_z(is(js,je)))*eps_0 -
            dt*sigma_e_z(is(js,je))/...;
        Cezh(is(js,je)) = (2*eps_r_z(is(js,je)))*eps_0 +
            dt*sigma_e_z(is(js,je))/;
        Cezh(is(js,je)) = (2*eps_r_z(is(js,je)))*eps_0 +
            dt*sigma_e_z(is(js,je))/;
        Cezh(is(js,je)) = (2*eps_r_z(is(js,je)))*eps_0 +
            dt*sigma_e_z(is(js,je))/;
    end
end
```
Once the original simulation is done; the adjoint simulation is then invoked. The script `fdtd_solve_adjoint` carries out the adjoint simulation. The adjoint source is a single current filament at the observation point. Similar to the S-parameter case of Chapter 4, the adjoint problem is excited with a wideband signal. The main difference between this adjoint simulation and the linear case is that the nonlinear permittivities of the adjoint problems are recovered from the stored value in the adjoint simulation and then used to update the coefficients of the FDTD scheme. The script `update_nonlinear_adjoint_permittivities` executes this function. It is invoked at the start of each time step. The rest of the code is identical to the isotropic case. Using the adjoint fields and the stored original residues, the sensitivities of the parameter $S_{PQ}$ are estimated relative to all parameters.

The output from this implementation is shown in Figures 8.6 and 8.7. We compare in these two figures the AVM sensitivities to the forward finite difference approximations.

![Figure 8.6](image.png)

Figure 8.6 The sensitivities of the real part of the response $S_{PQ}$ with respect to the material parameters as compared to forward finite difference approximations

Once the original simulation is done; the adjoint simulation is then invoked. The script `fdtd_solve_adjoint` carries out the adjoint simulation. The adjoint source is a single current filament at the observation point. Similar to the S-parameter case of Chapter 4, the adjoint problem is excited with a wideband signal. The main difference between this adjoint simulation and the linear case is that the nonlinear permittivities of the adjoint problems are recovered from the stored value in the adjoint simulation and then used to update the coefficients of the FDTD scheme. The script `update_nonlinear_adjoint_permittivities` executes this function. It is invoked at the start of each time step. The rest of the code is identical to the isotropic case. Using the adjoint fields and the stored original residues, the sensitivities of the parameter $S_{PQ}$ are estimated relative to all parameters.

The output from this implementation is shown in Figures 8.6 and 8.7. We compare in these two figures the AVM sensitivities to the forward finite differences. A reasonable match is achieved for most parameters.
Figure 8.7  The sensitivities of the imaginary part of $S_{PQ}$ with respect to the material parameters as compared to forward finite difference approximations

References


Gradient-estimation of objective functions and responses is important in modeling and optimization of high-frequency structures. We dedicated eight chapters to explain how to estimate the adjoint-based gradient information for problems with different types of materials. Most modeling and optimization techniques of high-frequency structures usually utilize only first-order derivatives because the cost of evaluating higher order sensitivities can be prohibitive [1]. Having these sensitivities at a reasonable computational cost can result in improving the accuracy of modeling approaches and accelerating the convergence of optimization algorithms [2,3].

We show in this chapter, two different approaches for efficiently estimating the second-order derivatives (Hessian matrix) of a given objective function. The cost of evaluating the Hessian using classical finite difference approach is $O(n^2)$ where $n$ is the number of parameters. The first adjoint approach reduces the cost of estimating all components of the Hessian matrix to only $2n$ extra simulations. This approach is simple, and it uses the algorithms developed in previous chapters.

A second approach for estimating the complete Hessian is also presented. This approach is more complex than the first approach and requires extra memory storage. This approach requires only $n + 1$ extra simulations per Hessian evaluation. It follows that the computational cost is approximately one half of the first adjoint approach. This saving comes at the cost of a more complex algorithm and more extensive storage. This approach is a simpler variation of the work presented in [4,5].

### 9.1 Hessian finite difference evaluation

The Hessian matrix is the matrix including all second derivatives of an objective function or a scalar response $F(p)$. It is given by

$$
H = \begin{bmatrix}
\frac{\partial^2 F}{\partial p_1^2} & \frac{\partial^2 F}{\partial p_1 \partial p_2} & \cdots & \frac{\partial^2 F}{\partial p_1 \partial p_n} \\
\frac{\partial^2 F}{\partial p_2 \partial p_1} & \frac{\partial^2 F}{\partial p_2^2} & \cdots & \frac{\partial^2 F}{\partial p_2 \partial p_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 F}{\partial p_n \partial p_1} & \frac{\partial^2 F}{\partial p_n \partial p_2} & \cdots & \frac{\partial^2 F}{\partial p_n^2}
\end{bmatrix}.
$$

(9.1)
The Hessian matrix in (9.1) is symmetric, and thus evaluating the upper part of the matrix is sufficient. If the function is known as an analytical function of the parameters, then the gradient and the Hessian can be both evaluated analytically. This is, however, a special case. The more general case is that the function or response of interest is evaluated numerically. In this case, finite difference approximations can be used to estimate the Hessian matrix. In this book, we make use of the following forward finite difference formula to estimate the Hessian components [6]:

$$H_{ij} \approx \frac{F(p_i^+, p_j^+) - F(p_i^+, p_j) - F(p_i, p_j^+) + F(p_i, p_j)}{\Delta p_i \Delta p_j}, \forall i, \forall j, i \neq j, \quad (9.2)$$

where $p_i^+ = p_i + \Delta p_i$ and $p_j^+ = p_j + \Delta p_j$. It can be shown that $O(n^2)$ function evaluations are required to evaluate all components of the Hessian matrix. If $i = j$, (9.2) is reduced to the well-known formula:

$$H_{ii} \approx \frac{F(p_i^+) - 2F(p_i) + F(p_i^-)}{\Delta p_i^2}, \forall i. \quad (9.3)$$

The following example illustrates the estimation of the Hessian matrix for a simple analytic function.

**Example 9.1** It is required to analytically evaluate the Hessian of the function $F(p) = p_1^3 + 2p_1 p_2^2 - 3p_2 p_3^2 + p_4^4$ at the point $p = [1 \ -1 \ 1]^T$. Using the definition of the Hessian matrix (9.1), we have

$$\frac{\partial F}{\partial p_1} = 3p_1^2 + 2p_2^2.$$ 

This component of the gradient vector is then differentiated relative to all three parameters to give the first row (or first column) of the Hessian matrix:

$$\frac{\partial^2 F}{\partial p_1^2} = 6p_1, \quad \frac{\partial^2 F}{\partial p_1 \partial p_2} = 4p_2, \quad \frac{\partial^2 F}{\partial p_1 \partial p_3} = 0.$$ 

Similarly, the second row (or second column) of the Hessian matrix is evaluated by differentiating $\partial F/\partial p_2$ with respect to all three parameters to have

$$\frac{\partial^2 F}{\partial p_2 \partial p_1} = 4p_2, \quad \frac{\partial^2 F}{\partial p_2^2} = 4p_1, \quad \frac{\partial^2 F}{\partial p_2 \partial p_3} = -6p_3.$$
The third row (or column) of the Hessian matrix is obtained by differentiating $\partial F / \partial p_3$ with respect to all parameters to get

$$\frac{\partial^2 F}{\partial p_3 \partial p_1} = 0, \quad \frac{\partial^2 F}{\partial p_3 \partial p_2} = -6p_3, \quad \frac{\partial^2 F}{\partial p_3^2} = -6p_2 + 12p_3^2.$$ 

It follows that the Hessian matrix at an arbitrary point $p$ is given by

$$H(p) = \begin{bmatrix} 6p_1 & 4p_2 & 0 \\ 4p_2 & 4p_1 & -6p_3 \\ 0 & -6p_3 & -6p_2 + 12p_3^2 \end{bmatrix}$$

At the set of parameter values $p_0 = [1 \ -1 \ 1]^T$, the Hessian is given by

$$H(p_0) = \begin{bmatrix} 6 & -4 & 0 \\ -4 & 4 & -6 \\ 0 & -6 & 18 \end{bmatrix}$$

To verify this analytical answer, we implement the finite difference formula (9.3) in a MATLAB® code. This code is shown in MATLAB Listing M9.1. In this code, the function get_function_value() evaluates the function value for the given set of parameters. It is repeatedly invoked for perturbed parameter values. The code starts by defining the perturbations to be utilized for all parameters through the command perturbations = [0.03 0.03 0.03]. An identity matrix is used to select which parameter to perturb every time. By selecting the $i$th column of this matrix, the perturbation is applied only to the $i$th parameter, $i = 1, 2, \ldots, 3$. We first perturb the parameters, one at a time, and store the corresponding function values in the vector perturbed_objective_function_values. The code then perturbs the parameters in pairs through the two commands:

```matlab
parameters=nominal_parameters+
perturbations(i)*I(i,:)+perturbations(j)*I(j,:);
objective_function=get_function_value(parameters).
```

The code then implements formula (9.2) through the command

$$Hessian(i,j) = \frac{(value_{ij}-value_i-value_j+nominal_objective_function)/}{(perturbations(i)*perturbations(j))}$$

The output from this code is

$$\begin{bmatrix} 6.1800 & -3.9400 & -0.0000 \\ -3.9400 & 4.0000 & -6.0900 \\ -0.0000 & -6.0900 & 18.7326 \end{bmatrix}$$
The numerical results reasonably matches the analytical result. Using smaller perturbations result in a better match between the numerical and analytical values.

9.2 A hybrid adjoint technique

There are several reported adjoint-based approaches to obtain an estimate of the Hessian matrix of a scalar objective function or response of high-frequency structures, for example, [7, 8]. We present a simple approach in this section. We label this approach as a hybrid adjoint technique (HAT) for reasons that will become clearer soon. This approach does not require any further theoretical derivations. It makes repeated calls to an adjoint-based gradient solver. This approach requires $2n$ extra simulations per Hessian estimation.

Assume that we want to estimate the Hessian matrix $H(p)$ at a specific point $p_0$. We make use of the fact that the $i$th column (or row) of the Hessian matrix is the derivative of the gradient $g(p) = \nabla f(p)$ with respect to the $i$th parameter at the point...
of interest. It follows that we have

\[
H(p_0) = H(p_0) = \frac{\partial g(p)}{\partial p_i} \bigg|_{p_0} \approx g(p_0 + \Delta p_i e_i) - g(p_0) \frac{\Delta p_i}{\Delta p_i},
\]

\[i = 1, 2, \ldots, n,\] (9.4)

where \(e_i\) is the \(i\)th column of the identity matrix. Equation (9.4) offers the core concept of the HAT. The gradients \(g(p_0)\) and \(g(p_0 + \Delta p_i e_i)\), \(i = 1, 2, \ldots, n\) are all estimated using adjoint sensitivity analysis. Each one of these gradient evaluations requires an original simulation and an adjoint simulation. Note that the gradients are evaluated at different perturbed parameter values. It follows that we need \(2n\) extra simulations. The original responses at \(p_0\) are already available through the original simulation. The hybrid approach thus combines adjoint sensitivity analysis and finite differences. The finite difference formula (9.4) is used to estimate the \(i\)th column of the Hessian using the available adjoint-based gradients.

The following algorithm illustrates the main steps of the Hybrid approach:

Step 1: Given \(p_0\) and a set of perturbation \(\Delta p = [\Delta p_1, \Delta p_2, \ldots, \Delta p_n]^T\)

Step 2: Execute the original simulation at \(p_0\) and the corresponding adjoint simulation at the same point to get \(g(p_0)\)

Step 3: \(i = 1\)

Step 4: Execute the original and adjoint simulations for the parameter values \(p_0 + \Delta p_i e_i\) to estimate \(g(p_0 + \Delta p_i e_i)\)

Step 5: Utilize the formula (9.4) to evaluate the \(i\)th column of the Hessian matrix \(H\)

Step 6: \(i = i + 1\). If \(i <= n\), go to step 4

Step 7: Output \(H\).
The following example illustrates the hybrid AVM approach:

**Example 9.2** Consider the 2D FDTD discontinuity shown in Figure 9.1. We would like to write a MATLAB code to estimate the Hessian of the corresponding energy function

\[
F = \int_0^{T_m} \int_{\Omega} E_z^2 d\Omega dt,
\]

where \(\Omega\) is the observation line shown in Figure 9.1. The Hessian is evaluated relative to the parameters \(p = [\epsilon_r \sigma_x x_{\text{min}} x_{\text{max}} y_{\text{min}} y_{\text{max}}]^T\) at the point \(p_0 = [3 \ 0 \ -5e-3 \ 6e-3 \ 28e-3 \ 40e-3]^T\). We utilize the perturbations \(\Delta p = [0.05 \ 1.0e-3 \ 1.0e-3 \ 1.0e-3 \ 1.0e-3 \ 1.0e-3]\). MATLAB Listing M9.2 shows the script `evaluate_Hessian` that implements the hybrid algorithm. The function `get_response_and_derivatives()` evaluates the objective function value and its gradient at the given point through adjoint sensitivity analysis. This function, which is shown in MATLAB Listing M9.3, invokes the original simulation followed by the adjoint simulation. Using the stored residues and the adjoint response, the gradient is estimated. The implementation of the adjoint simulation has

---

**Figure 9.1** The structure of Example 9.2. The parameters of this problem are \(p = [\epsilon_r \sigma_x x_{\text{min}} x_{\text{max}} y_{\text{min}} y_{\text{max}}]^T\). The energy objective function is estimated over the observation domain (line)
been addressed in previous chapters. Using this hybrid approach, the estimated Hessian is given by

\[
\text{Hessian} = 1.0e-05 * \\
\begin{bmatrix}
0.0000 & -0.0000 & -0.0014 & 0.0014 & -0.0012 & 0.0013 \\
-0.0000 & -0.0000 & 0.0377 & -0.0377 & 0.0331 & -0.0366 \\
-0.0014 & 0.0375 & -0.0019 & -0.0056 & 0.2088 & -0.2388 \\
0.0014 & -0.0379 & -0.0051 & -0.0039 & -0.2084 & 0.2379 \\
-0.0012 & 0.0328 & 0.2134 & -0.2134 & -0.0195 & -0.0045 \\
0.0014 & -0.0366 & -0.2451 & 0.2451 & -0.0042 & 0.0368
\end{bmatrix}
\]

The hybrid approach requires \(2n = 12\) extra simulations to evaluate all components of the Hessian matrix. The MATLAB script `evaluate_Hessian_finite_differences`, which is a variation of MATLAB Listing M9.1, is used to evaluate the Hessian directly using the finite-difference formula (9.2) with no adjoint simulation. It requires \(n(n + 1)/2 = 21\) extra simulations to evaluate the

![Graphs showing Hessian components](image.png)

Figure 9.2 Some of the Hessian components evaluated using the hybrid approach; the sensitivities evaluated using finite differences (—) and the sensitivities evaluated using the hybrid approach (○)
upper part of the Hessian matrix. The Hessian as evaluated using finite-differences is given by

\[
\text{Hessian} = 1.0 \times 10^{-5} \times \\
\begin{pmatrix}
0.0000 & -0.0000 & -0.0014 & 0.0014 & -0.0012 & 0.0014 \\
-0.0000 & -0.0000 & 0.0375 & -0.0379 & 0.0328 & -0.0366 \\
-0.0014 & 0.0375 & -0.0001 & -0.0053 & 0.2140 & -0.2457 \\
0.0014 & -0.0379 & -0.0053 & -0.0058 & -0.2135 & 0.2449 \\
-0.0012 & 0.0328 & 0.2140 & -0.2135 & -0.0199 & -0.0046 \\
0.0014 & -0.0366 & -0.2457 & 0.2449 & -0.0046 & 0.0381 \\
\end{pmatrix}
\]

Comparing the two Hessians, one sees that the dominant components agree well. There are some differences in the weaker components. This is expected because of the finite accuracy imposed by the utilized discretization. Our experience

\[\begin{array}{c}
\text{(a)} H_{35} \\
\text{(b)} H_{36} \\
\text{(c)} H_{45} \\
\text{(d)} H_{46}
\end{array}\]

Figures 9.3 Some of the Hessian components evaluated using the hybrid approach; the sensitivities evaluated using finite differences (—) and the sensitivities evaluated using the hybrid approach (○)
shows that a finer grid may be required to improve the match for all components. In most practical cases, however, the achieved accuracy is sufficient for both modeling and optimization purposes. It should be clear that in this implementation, we did not apply any scaling to the parameters. This results in a Hessian matrix whose components have orders of magnitude difference. Integrating scaling in both adjoint gradient and Hessian calculations should be straightforward.

Figures 9.2–9.4 show some of the key components of the Hessian matrix for a sweep of the dielectric constant. We see in all these figures that a good match is achieved between the hybrid approach and the finite-difference approximation. Due to space limitations, we show only this comparison for a subset of the Hessian matrix components. Good match is achieved though for all significant Hessian components.

Figure 9.4  Some of the Hessian components evaluated using the hybrid approach; the sensitivities evaluated using finite differences (—) and the sensitivities evaluated using the hybrid approach (○)
9.3 The fully adjoint approach

We present in this section a fully adjoint approach that requires only \( n + 1 \) extra simulations to evaluate all the components of the Hessian matrix. This reduction in the computational cost comes at the expense of a more sophisticated theory with more memory storage. The algorithm we adopt here is a variation of the algorithms presented in [4,5] that does not use a wave equation and acts directly on Maxwell’s equations. We show the derivation for isotropic and non-dispersive materials. The approach is, however, general and can be applied to electromagnetic problems with arbitrary types of materials.

As explained in Chapter 3, Maxwell’s equations for linear, isotropic, and non-dispersive materials can be written as

\[
N(p) \dot{V} + K(p) V = q, \quad (9.5)
\]

where the system matrices \( N \) and \( K \) are both functions of the parameters \( p \). The vector of parameters \( p \) may include both material and shape parameters. Our target is to obtain an estimate of the Hessian of the objective function:

\[
F = \int_0^T \psi(p, V) dt \quad (9.6)
\]

where the kernel function \( \psi \) may have both explicit and implicit dependence on \( p \). Differentiating both sides with respect to the \( i \)th parameter \( p_i \), we have

\[
\frac{\partial F}{\partial p_i} = \frac{\partial F}{\partial p_i} + \int_0^T \frac{\partial \psi}{\partial V} \frac{\partial V}{\partial p_i} dt, \quad \forall i. \quad (9.7)
\]

To evaluate the component \( H_{ij} \) of the Hessian matrix, we differentiate both sides of (9.7) with respect to the \( j \)th parameter \( p_j \) to get

\[
\frac{\partial^2 F}{\partial p_i \partial p_j} = \frac{\partial^2 F}{\partial p_i \partial p_j} + \int_0^T \left( \frac{\partial V}{\partial p_j} \right)^T \frac{\partial^2 \psi}{\partial V^2} \frac{\partial V}{\partial p_i} dt + \int_0^T \frac{\partial \psi}{\partial V} \frac{\partial^2 V}{\partial p_i \partial p_j} dt. \quad (9.8)
\]

The first term in (9.8) is the explicit second-order derivative. In most practical cases, the objective function is only implicitly dependent on the parameters, and this term is zero. The second term requires evaluating the Hessian of the kernel function with respect to the vector \( V \):

\[
H_\psi = \frac{\partial^2 \psi}{\partial V^2}. \quad (9.9)
\]

As the objective function is evaluated over the observation domain only, the derivatives \( \frac{\partial V}{\partial p_j} \) and \( \frac{\partial V}{\partial p_i} \) are stored only within the observation domain. Evaluating the second term directly without using adjoint sensitivities requires \( n \) extra simulations. The last term in (9.8) requires evaluating the derivative \( \frac{\partial^2 V}{\partial p_i \partial p_j} \) over the observation domain for every parameter pair \((i, j)\). This last term would thus require \( n^2 \) extra simulations.
Instead of calculating (9.8) directly, a more efficient approach can be achieved by using adjoint sensitivities. The derivation of this approach starts by differentiating (9.5) with respect to the $i$th parameter to get

$$\frac{\partial N}{\partial p_i} \dot{\mathbf{V}} + N \frac{\partial \dot{\mathbf{V}}}{\partial p_i} + \frac{\partial K}{\partial p_i} \mathbf{V} + K \frac{\partial \mathbf{V}}{\partial p_i} = \mathbf{0}, \quad \forall i. \tag{9.10}$$

Note that in (9.10), we assume that the system matrices are analytical functions of the parameters. This is true for the material parameters only. We will show later, how to approximate these derivatives for shape parameters. Differentiating (9.10) with respect to the $j$th parameter, we have

$$\frac{\partial N}{\partial p_i} \frac{\partial \dot{\mathbf{V}}}{\partial p_j} + N \frac{\partial^2 \mathbf{V}}{\partial p_j \partial p_i} + \frac{\partial K}{\partial p_i} \frac{\partial \mathbf{V}}{\partial p_j} + K \frac{\partial^2 \mathbf{V}}{\partial p_i \partial p_j} + \frac{\partial K}{\partial p_j} \frac{\partial \mathbf{V}}{\partial p_i} = \mathbf{0}, \quad \forall i, \forall j. \tag{9.11}$$

Reassembling the terms in (9.11), we have

$$N \frac{\partial^2 \mathbf{V}}{\partial p_i \partial p_j} + K \frac{\partial^2 \mathbf{V}}{\partial p_i \partial p_j} = -R_{ij} \tag{9.12}$$

where $R_{ij}$ represents the residue related to the parameters $p_i$ and $p_j$. This residue can be evaluated for all of the combinations $(i,j)$ using $n$ extra simulations. The vectors $\mathbf{V}$ and $\dot{\mathbf{V}}$ are already available from the original simulation at every point in the computational domain. The first- and second-order derivatives of the system matrices can be calculated for all parameters. The vectors $\frac{\partial \dot{\mathbf{V}}}{\partial p_i} \mathbf{V} \frac{\partial \mathbf{V}}{\partial p_i}$ can be estimated by perturbing the parameters one by one and utilizing finite differences. Note that these vectors are also needed in the second term of (9.8). We will assume for the rest of the discussion that the residues $R_{ij}$ are known for all combinations $(i,j)$ at the computational cost of $n$ extra simulations.

The derivation follows by multiplying both sides of (9.12) by the unknown adjoint vector $\lambda$ to get

$$\int_0^{T_m} \lambda^T \left( N \frac{\partial^2 \mathbf{V}}{\partial p_i \partial p_j} + K \frac{\partial^2 \mathbf{V}}{\partial p_i \partial p_j} \right) dt = - \int_0^{T_m} \lambda^T R_{ij} dt. \tag{9.13}$$

Similarly, the derivation of the first-order adjoint sensitivity, the first term in the left-hand side is integrated by parts. We then impose the condition that $\lambda(T_m = 0)$ to get

$$\int_0^{T_m} \left( -\dot{\lambda}^T N + \lambda^T K \right) \frac{\partial^2 \mathbf{V}}{\partial p_i \partial p_j} dt = - \int_0^{T_m} \lambda^T R_{ij} dt. \tag{9.14}$$
Equating the left-hand side of (9.14) to the last term of the right-hand side of (9.8), we have

$$-\dot{\lambda}^T N + \lambda^T K = \frac{\partial \psi}{\partial V_T}. \quad (9.15)$$

Taking the transpose of both sides and changing the variable in (9.15) from $t$ to $\tau$, we have

$$N \dot{\lambda} + K \lambda = \frac{\partial \psi}{\partial V}. \quad (9.16)$$

This is the same adjoint system needed to evaluate the first-order derivatives. Using the $(n + 1)$ original simulations, the first and second right-hand side terms of (9.12) are available. Using the original simulation (9.5) and the adjoint system (9.16), the third term of (9.8) is given by

$$\int_0^{T_w} \frac{\partial \psi}{\partial V} \frac{\partial^2 V}{\partial p_i \partial p_j} dt = - \int_0^{T_w} \lambda^T R_{ij} dt \approx -\Delta \tau \sum_{k=1}^{N_T} \lambda_k^T R_{ik}.$$

(9.17)

It follows that by using the original simulation, $n$ extra perturbed original simulations, and one adjoint simulation, all components of the Hessian matrix are estimated. This approach requires $(n + 1)$ extra simulations. The approach discussed in the last section requires $2n$ simulation per Hessian evaluation.

9.3.1 Implementation

The discussion given by (9.5)–(9.17) describes the approach for estimating the second-order sensitivities using the fully adjoint approach. A number of implementation details need more explanation. There are three main contributions to any Hessian component. The first one is the observation domain contribution given by

$$H_{ij}^{(1)} = \int_0^{T_w} \left( \frac{\partial V}{\partial p_j} \right)^T \frac{\partial^2 \psi}{\partial V^2} \frac{\partial V}{\partial p_i} dt = \int_0^{T_w} \left( \frac{\partial V}{\partial p_j} \right)^T H_{ij} \frac{\partial V}{\partial p_i} dt. \quad (9.18)$$

To evaluate this part, one should notice that the matrix $H_{ij}$ is a diagonal matrix with nonzero components corresponding only to nodes in the observation domain. This implies that the vectors $\frac{\partial V}{\partial p_j}$ and $\frac{\partial V}{\partial p_i}$ need to be stored only in the observation domain. The easiest way to evaluate these vectors is to store $V(p)$ calculated in the original simulation over the observation domain. The $i$th parameter $p_i$ is then perturbed and the original simulation is invoked again to calculate $V(p + \Delta p_i e_i)$. This perturbed response is also stored over the observation domain. The field derivatives needed in (9.18) are then approximated by

$$\frac{\partial V}{\partial p_i} \approx \frac{V(p + \Delta p_i e_i) - V(p)}{\Delta p_i}. \quad (9.19)$$
We will show later that we will use these same perturbed fields in other contributions as well. Notice that the contribution (9.18) does not require any inner product with the adjoint response. Also, if the kernel function is only a linear function of the field, as in the calculation of the scattering parameters, the matrix $H_y$ vanishes and thus the contribution (9.18) is zero.

From (9.13), the second and third contributions to the Hessian component $H_{ij}$ are given by

$$H_{ij}^{(2)} + H_{ij}^{(3)} = -\int_0^{T_w} \lambda^T R_{ij} dt \approx \Delta t \sum_k \lambda^T_k R_{ij,k}, \quad (9.20)$$

where

$$H_{ij}^{(2)} = -\int_0^{T_w} \lambda^T (\frac{\partial N}{\partial p_i} \frac{\partial \dot{V}}{\partial p_j} + \frac{\partial N}{\partial p_j} \frac{\partial \dot{V}}{\partial p_i} + \frac{\partial K}{\partial p_i} \frac{\partial V}{\partial p_j} + \frac{\partial K}{\partial p_j} \frac{\partial V}{\partial p_i}) dt \quad (9.21)$$

and

$$H_{ij}^{(3)} = -\int_0^{T_w} \lambda^T (\frac{\partial^2 N}{\partial p_i \partial p_j} \dot{V} + \frac{\partial^2 K}{\partial p_i \partial p_j} V) dt. \quad (9.22)$$

These two contributions need different approaches for calculating them. The part of the residue corresponding to (9.21) is

$$R_{ij}^{(2)} = \frac{\partial N}{\partial p_i} \frac{\partial \dot{V}}{\partial p_j} + \frac{\partial N}{\partial p_j} \frac{\partial \dot{V}}{\partial p_i} + \frac{\partial K}{\partial p_i} \frac{\partial V}{\partial p_j} + \frac{\partial K}{\partial p_j} \frac{\partial V}{\partial p_i}$$

$$= \left( \frac{\partial N}{\partial p_i} \frac{\partial \dot{V}}{\partial p_j} + \frac{\partial K}{\partial p_i} \frac{\partial V}{\partial p_j} \right) + \left( \frac{\partial N}{\partial p_j} \frac{\partial \dot{V}}{\partial p_i} + \frac{\partial K}{\partial p_j} \frac{\partial V}{\partial p_i} \right). \quad (9.23)$$

Recall that the residue associated with the $i$th parameter, as derived in Chapter 3, is given by

$$R_i(p) = \left( \frac{\partial N}{\partial p_i} \dot{V} + \frac{\partial K}{\partial p_i} V \right). \quad (9.24)$$

Comparing (9.24) and any of the two brackets in (9.23), we note that in (9.23), the matrices multiply derivatives of the field quantities relative to parameters. It follows that we can approximate (9.23) using residues evaluated at perturbed parameter values using

$$\left( \frac{\partial N}{\partial p_i} \frac{\partial \dot{V}}{\partial p_j} + \frac{\partial K}{\partial p_i} \frac{\partial V}{\partial p_j} \right) \approx \left( \frac{\partial N}{\partial p_i} (\dot{V}(p + \Delta p_j e_j) - \dot{V}(p)) + \frac{\partial K}{\partial p_i} (V(p + \Delta p_j e_j) - V(p)) \right). \quad (9.25)$$
The expression (9.25) thus utilizes the difference between first-order residues identical to (9.24). The same approximation can be applied to the second bracket in (9.23). A fine point to note is that the system matrices derivatives (stamps) are all estimated for the nominal structure. It follows that the \( n \) extra original simulations utilized in evaluating (9.18) are also used to evaluate the contribution (9.25) using the stamps of the nonperturbed structure.

The last contribution to the Hessian matrix is the one given by (9.22). The residue corresponding to this contribution is calculated using only the fields of the original simulation. The second-order Hessian stamps \( \partial^2 N / \partial \rho_i \partial \rho_j \) and \( \partial^2 K / \partial \rho_i \partial \rho_j \) are first calculated for all pairs \((i,j)\) and stored. There are many interesting combinations that arise in calculating these Hessian stamps. If the parameters \( i \) and \( j \) are both material parameters, these stamps are zeros. The reason for this is that the matrix \( N \) is a diagonal matrix of permittivity. Its second derivatives relative to \( \epsilon_r \) or \( \sigma_e \) is thus zero. The matrix \( K \) is a diagonal matrix of conductivity values and thus has also zero second-order derivatives relative to material parameters.

The situation is different if either \( i \) or \( j \) or both are shape parameters. First, we consider the illustrative case in which the \( i \)th parameter represents the dielectric constant of a discontinuity whereas the \( j \)th parameter represents the parameter \( y_{\text{max}} \) as shown in Figure 9.5. As the matrix \( N \) represents the local permittivity, the

![Figure 9.5](image_url)

**Figure 9.5** The evaluation of the Hessian stamp for a material parameter and a shape parameter: (a) the components of the matrix \( N \) for the non-perturbed structure, (b) the derivative of the non-perturbed matrix \( N \) with respect to \( \epsilon_r \), (c) the derivative of the perturbed matrix \( N \) with respect to \( \epsilon_r \), and (d) the non-zero components of the mixed derivatives of the matrix \( N \) with respect to \( y_{\text{max}} \) and \( \epsilon_r \)
components of the matrix $N$ corresponding to each FDTD node is shown in Figure 9.5(a). The analytical derivative $\partial N / \partial \varepsilon_r$ is shown in Figure 9.5(b). The second-order matrix derivative $\partial^2 N / \partial \varepsilon_r \partial y_{\text{max}}$ is approximated by

$$\frac{\partial^2 N}{\partial \varepsilon_r \partial y_{\text{max}}} \approx \frac{\partial N / \partial \varepsilon_r (y_{\text{max}} + \Delta y) - \partial N / \partial \varepsilon_r (y_{\text{max}})}{\Delta y}. \quad (9.26)$$

Figure 9.5(c) and (d) shows the components of $\partial N / \partial \varepsilon_r (y_{\text{max}} + \Delta y)$ and the second-order derivatives. The same approach applies for the Hessian stamps of the $K$ matrix.

If the parameters $i$ and $j$ are both shape parameters, and because we can perturb these parameters using only on-grid perturbations, we approximate the Hessian stamps through the formula:

$$\frac{\partial^2 N}{\partial p_i \partial p_j} \approx \frac{N(p + \Delta p_i e_i + \Delta p_j e_j) - N(p + \Delta p_i e_i) - N(p + \Delta p_j e_j) + N(p)}{\Delta p_i \Delta p_j}. \quad (9.27)$$

Figure 9.6 illustrates this calculation for the parameters $x_{\text{min}}$ and $y_{\text{max}}$. Figure 9.6(a) shows the permittivity distribution associated with the nodes at the nominal parameters ($N(p)$). Figure 9.6(b) and (c) shows the permittivity distribution when the parameters are perturbed one at a time to give $N(p + \Delta x e_3)$ and $N(p + \Delta y e_6)$, respectively. Figure 9.6(d) shows the case when both parameters are perturbed simultaneously to give $N(p + \Delta x e_3 + \Delta y e_6)$. Finally, Figure 9.6(e) shows the approximation (9.27) used to yield the Hessian stamp.

If the parameters $i$ and $j$ are the same shape parameters, we utilize the approximation:

$$\frac{\partial^2 N}{\partial p_i^2} \approx \frac{N(p + \Delta p_i e_i) - 2N(p) + N(p - \Delta p_i e_i)}{\Delta p_i^2}. \quad (9.28)$$

Figure 9.7 illustrates this approximation for the parameter $y_{\text{max}}$. Once the stamps have been created for all possible pairs of parameters $(i,j)$, the original simulation is executed and the Hessian residues:

$$R^{(3)}_{ij} = \frac{\partial^2 N}{\partial p_i \partial p_j} \dot{V} + \frac{\partial^2 K}{\partial p_i \partial p_j} \dot{V}, \forall i, \forall j, \quad (9.29)$$

are stored at every time step of the original simulation.

### 9.3.2 The algorithm

The derivations given in the last two sections can be summarized in the following algorithm to evaluate the Hessian of an objective function using adjoint sensitivity analysis:

Step 0: Given the point $p$ and the perturbations $\Delta p$

Step 1: Carry out the original simulation. Store the residues $R^{(3)}_{ij}$ given by (9.29), $\forall i, \forall j$. Store the nominal stamps $\partial N / \partial p_i$ and $\partial K / \partial p_i$, $\forall i$. Store the nominal fields over the observation domain.
Step 2: $i = 1$
Step 3: Carry out the perturbed original simulation with parameters $p + \Delta p_i e_i$.
Store the perturbed fields and derivatives $V(p + \Delta p_i e_i)$ and $\dot{V}(p + \Delta p_i e_i)$ within the perturbation region of the parameter. Store also the perturbed fields over the observation domain.
Step 4: $i = i + 1$. If $i \leq n$, go to Step 3
Step 5: Evaluate $R^{(2)}_{ij}$ in (9.23) using the perturbed fields, their time derivatives, and the nominal stamps, $\forall i, \forall j$.

Figure 9.6: The evaluation of the Hessian stamp corresponding to the two parameters $x_{\text{min}}$ and $y_{\text{max}}$: (a) the components of the matrix $N$ for the nonperturbed structure, (b) the components of the matrix $N$ with $x_{\text{min}}$ perturbed by $\Delta x$, (c) the components of the matrix $N$ with $y_{\text{max}}$ perturbed by $\Delta y$, (d) the components of the matrix $N$ with $x_{\text{min}}$ perturbed by $\Delta x$ and $y_{\text{max}}$ perturbed by $\Delta y$, and (e) the nonzero components of the mixed derivatives of the matrix $N$ with respect to $x_{\text{min}}$ and $y_{\text{max}}$. 

236 Adjoint sensitivity analysis of high frequency structures with MATLAB®
Step 6: Carry out the adjoint simulation to determine the adjoint fields $\lambda$ at all-time steps. Use formulas (9.18), (9.21), and (9.22) to evaluate all three contributions to the Hessian matrix.

Step 6: output the Hessian matrix.

This algorithm is illustrated by the following 2D FDTD example.

<table>
<thead>
<tr>
<th>$N(y_{max})$</th>
<th>$N(y_{max}+\Delta y)$</th>
<th>$N(y_{max}-\Delta y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
</tr>
<tr>
<td>$\varepsilon_r$</td>
<td>$\varepsilon_r$</td>
<td>$\varepsilon_r$</td>
</tr>
<tr>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
</tr>
<tr>
<td>$\varepsilon_r$</td>
<td>$\varepsilon_r$</td>
<td>$\varepsilon_r$</td>
</tr>
<tr>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
</tr>
<tr>
<td>$\varepsilon_r$</td>
<td>$\varepsilon_r$</td>
<td>$\varepsilon_r$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$N(y_{max})$</th>
<th>$N(y_{max}+\Delta y)$</th>
<th>$N(y_{max}-\Delta y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
</tr>
<tr>
<td>$\varepsilon_r$</td>
<td>$\varepsilon_r$</td>
<td>$\varepsilon_r$</td>
</tr>
<tr>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
</tr>
<tr>
<td>$\varepsilon_r$</td>
<td>$\varepsilon_r$</td>
<td>$\varepsilon_r$</td>
</tr>
<tr>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
<td>$\frac{1}{2} \varepsilon_r + \frac{1}{2} \varepsilon_r + \frac{3}{4}$</td>
</tr>
<tr>
<td>$\varepsilon_r$</td>
<td>$\varepsilon_r$</td>
<td>$\varepsilon_r$</td>
</tr>
</tbody>
</table>

Figure 9.7  The evaluation of the Hessian stamp corresponding to $y_{max}$:  
(a) the components of the matrix $N$ for the nonperturbed structure,  
(b) the components of the matrix $N$ with $y_{max}$ perturbed by $\Delta y$,  
(c) the components of the matrix $N$ with $y_{max}$ perturbed by $-\Delta y$, and  
(d) the nonzero components of the second-order derivatives of the matrix $N$ with respect to $y_{max}$.
Example 9.3 We repeat the same problem discussed in Example 9.2. We estimate the second-order sensitivities of the energy function across the output port. The main script that evaluates the Hessian is `evaluate_Hessian` and is shown in MATLAB Listing M9.4. The values at which the sensitivities are calculated are nominal_parameters.

The perturbations utilized in estimating the Hessian components are given by the vector perturbations. The function `get_residues()` receives the parameter values and an index that can assume values from 0 to 6. If the index has a zero value, then the parameters are the nominal parameters. It is very important to distinguish the nominal parameters from the perturbed parameters because the first-order stamps $\partial N / \partial p_i$ and $\partial K / \partial p_i$, $\forall i$ and the Hessian stamps $\partial^2 N / \partial p_i \partial p_j$ and $\partial^2 K / \partial p_i \partial p_j$, $\forall i$ and $\forall j$ are calculated only for the nominal parameters. They are used later with perturbed fields. The function `get_residues()` returns the first-order residues evaluated using the nominal stamps and the fields over the observation domain for the given parameter values. Because we have six parameters, every call to this function returns six residues. The function `get_residues()` is shown in MATLAB Listing M9.5. This function invokes the original simulation code shown in MATLAB Listing M9.6. This code is essentially similar to the original codes we used in other chapters with few minor changes. First, the statements

MATLAB Listing M9.4

```matlab
% This script evaluates second order sensitivities using adjoint
%sensitivities
clear all; close all;
number_of_parameters=6; %total number of parameters
nominal_parameters=[3 0 -5e-3 6e-3 28e-3 40e-3];
%these are the nominal parameters
perturbations=[0.05 1.0e-3 1.0e-3 1.0e-3 1.0e-3 1.0e-3];
%utilized perturbations for parameters
parameters=nominal_parameters; %get the current parameter values
[residues, observation_field]=get_residues(parameters,0);
%nominal residues and observation fields
residue_sampled_field(1,:)=residues; %store residues
observation_fields=[observation_field; %store observation fields
I=eye(6); %Identity matrix
Hessian=zeros(number_of_parameters, number_of_parameters);
for parameter_counter=1: number_of_parameters
parameters=nominal_parameters;
perturbations=parameter_counter'*I(parameter_counter,...);
%perturb the ith parameter
[residues, observation_field]=get_residues(parameters,
parameter_counter); %perturbed residues and observation fields
residue_sampled_field(parameter_counter,:)=residues;
observation_fields(parameter_counter,:)=observation_field;
end
save('residue_sampled_field.mat','residue_sampled_field')
save('observation_fields.mat','observation_fields');
[gradient, Hessian]=call_adjoint(nominal_parameters,perturbations);
%call the adjoint system at the nominal parameters
save('Hessian_adjoint_full.mat','Hessian');
```
MATLAB Listing M9.5

```matlab
function [residue_sampled_field, electric_field_on_a_line]=
    ftdt_solve_original;
    get_residues(parameters, index)
```

MATLAB Listing M9.6

```matlab
%MATLAB Listing M9.6
define_problem_space_parameters_2d;
define_geometry_2d;
define_sources_2d;
define_output_parameters_2d;
initialize_ftdt_material_grid_2d;
initialize_ftdt_parameters_and_arrays_2d;
initialize_sources_2d;
initialize_updating_coefficients_2d;
initialize_boundary_conditions_2d;
initialize_output_parameters_2d;
initialize_display_parameters_2d;
initialize_adjoint_parameters_and_arrays;
if(index==0) %we use the stamps of the nominal case only
    initialize_residue_storage;
    initialize_Hessian_residue_storage;
    initialize_residue_stamps;
    initialize_Hessian_residue_stamps;
    %save nominal stamps
    save('nominal_residue_stamps.mat','residue_sampled_field');
else
    load nominal_residue_stamps.mat; %load nominal stamp
end
%initialize objective function
objective_function=0;
% FDTD time marching loop
run_original_ftdt_time_marching_loop_2d;
if(index==0) %we use the stamps of the nominal case only
    save('nominal_Hessian_residues.mat','Hessian_residue_sampled_field');
    save('original_electric_field_on_a_line.mat','electric_field_on_a_line');
end
if(index==0) %we use the stamps of the nominal case only
    initialize_residue_storage;
    initialize_Hessian_residue_storage;
    initialize_residue_stamps;
    initialize_Hessian_residue_stamps;
    %save nominal stamps
    save('nominal_residue_stamps.mat','residue_sampled_field');
else
    load nominal_residue_stamps.mat; %load nominal stamp
end
create the nominal first-order residue stamps and Hessian stamps only for the nominal simulation (index=0). For perturbed parameter values (index >0), the script loads the nominal first-order stamps to be used in calculating residues.
The scripts `initialize_residue_stamps` and `initialize_Hessian_residue_stamps` create the first-order and second-order stamps. The main FDTD loop is executed by the script `run_original_fdtd_time_marching_loop_2d`. This script is identical to similar scripts discussed in earlier chapters with the minor change that the Hessian residues are calculated at every time step. These residues are calculated through the script `store_Hessian_fields`, shown in MATLAB Listing M9.7. This script carries out a double loop over all parameters \((i,j)\). The script uploads the stamps representing...
the second-order system matrices derivatives \( \frac{\partial^2 N}{\partial p_i \partial p_j} \) and \( \frac{\partial^2 K}{\partial p_i \partial p_j} \). These stamps are given by the parameters \( N_{\text{stamp} \_Ez} \) and \( K_{\text{stamp} \_Ez} \).

The command

\[
\text{Hessian\_residue\_sampled\_field}(k,i,j)\_\text{residue\_Ez}(\text{time\_step},:) = ((1/dt)*(N_{\text{stamp} \_Ez}.*(\text{current\_Ez}-\text{previous\_Ez}))+
0.5*K_{\text{stamp} \_Ez}.*(\text{current\_Ez}+\text{previous\_Ez}));
\]

executes (9.29) for evaluating Hessian residues. These residues are then stored to be utilized during the adjoint simulation.

Once all \( (n + 1) \) original simulations have been executed, MATLAB Listing M9.4 invokes the adjoint simulation by executing the function \text{call\_adjoint}. This function receives as inputs the parameters and the associated perturbations. It loads all the stored first-order residues calculated using the \( (n + 1) \) original simulations. It also loads the Hessian residues evaluated during the original simulation. The main task of this function, which is given in MATLAB Listing M9.8, is to evaluate all three contributions to the Hessian matrix. The two contributions given by (9.20) are evaluated within the main loop of the adjoint simulation executed by the script \text{fdtd\_solve\_adjoint}. The contribution of the observation domain given by (9.18) does not require any adjoint field information and is evaluated within the separate script \text{add\_observation\_domain\_contribution}. Within the two scripts, the code evaluates only the upper subset of the Hessian matrix.

The script \text{fdtd\_solve\_adjoint} is similar to other adjoint simulations discussed in previous chapters. The main difference is that, within the main FDTD loop, the code invokes the script \text{update\_Hessian\_adjoint\_product}. This script evaluates the inner product between the adjoint field and both the first-order residues (9.23) and the second-order residues (9.29). A subset of the code of this script is shown in MATLAB Listing M9.9. This part evaluates (9.21). The storage \( \text{residue\_Ez\_0i} \) represents the residue related to the \( i \)th parameter evaluated using the original simulation. The storage \( \text{residue\_Ez\_ji} \) is the residue corresponding to the \( i \)th parameter evaluated when the \( j \)th parameter is perturbed by \( \Delta p_j \). Both residues are evaluated using original stamps as explained earlier. The commands

\[
\text{for } i=1:6
\text{for } j=1:i-1
\quad \text{Hessian\_adjoint}(i,j)=\text{Hessian}(j,i);
\text{end}
\text{end}
\]

enforces the symmetry of the Hessian matrix.

Another part of this code implements the same using the residues \( \text{residue\_Ez\_0j} \) and \( \text{residue\_Ez\_ij} \) corresponding to the \( j \)th parameter. It should be clear to the reader that there are two brackets in (9.23), and this is why there is an \( ij \) and \( ji \) contribution to the Hessian matrix element \( H_{ij} \).
MATLAB Listing M9.9

```matlab
% MATLAB Listing M9.9
for k=1:6 % repeat for all parameters
    for j=1:6
        % first, add the j'th contribution
        % x start
        min_x= residue_sampled_field(1,i).
        max_x= residue_sampled_field(1,i).
        % y start
        min_y= residue_sampled_field(1,i).
        max_y= residue_sampled_field(1,i).
        % get number of stored Ez components
        Ez_max_x= max_x+1;
        Ez_max_y= max_y+1;
        % get current value of adjoint fields
        current_Ez_adjoint= reshape(Ez(max_x+1,min_y),
                                    min_y,Ez_max_y,1);
        residue_Ez_0i= residue_sampled_field(1,i).
        residue_Ez= residue_sampled_field(j+1,i).
        if(time_step==1) % get previous field values
            previous_Ez_adjoint= current_Ez_adjoint;
            % zero derivative
            else % recover previous field value
                previous_Ez_adjoint= residue_sampled_field(j+1,i).
            end
            % calculate average adjoint field @half time step
            average_Ez_adjoint= 0.5*(current_Ez_adjoint+
                                    previous_Ez_adjoint);
            % calculate Hessian
            Hessian(i,j)= Hessian(i,j)-(dt/perturbations(j))*
                                ((residue_Ez_ji-residue_Ez_0i)*average_Ez_adjoint');
    end
end
```

MATLAB Listing M9.10

```matlab
% MATLAB Listing M9.10
flag=Hessian_residue_sampled_field(k,i,j).store_flag;
if (flag==1) % add 2nd order contribution only if flag is 1
    min_x= Hessian_residue_sampled_field(k,i,j).
    max_x= Hessian_residue_sampled_field(k,i,j).
    % y start
    min_y= Hessian_residue_sampled_field(k,i,j).
    max_y= Hessian_residue_sampled_field(k,i,j).
    % get number of stored Ez components
    Ez_max_x= max_x;
    Ez_max_y= max_y;
    % get current value of adjoint fields
    current_Ez_adjoint= reshape(Ez(max_x+1,min_y),
                                min_y,Ez_max_y,1);
    if(time_step==1) % get previous field values
        previous_Ez_adjoint= current_Ez_adjoint;
        % zero derivative
        else % not flat time step
            previous_Ez_adjoint= Hessian_residue_sampled_field(k,i,j).
        end
        % get stored residues. Notice time reversal
        residue_Ez= Hessian_residue_sampled_field(k,i,j).
        % calculate average adjoint field @half time step
        average_Ez_adjoint= 0.5*(current_Ez_adjoint+
                                previous_Ez_adjoint);
        % store previous adjoint field values for the next time step
        Hessian_residue_sampled_field(k,i,j).
        %print Hessian
        previous_Ez_adjoint= current_Ez_adjoint;
end
```
Figure 9.8 A subset of the Hessian components of the 2D example; the finite difference approximation (—) as compared to the adjoint sensitivities estimated using the fully adjoint approach (○)

Second-order adjoint sensitivities
MATLAB Listing M9.10 shows another part of the function `update_Hessian_adjoint_product` that evaluates the contribution from the Hessian residues. The command

\[
\text{Hessian}(i,j) = \text{Hessian}(i,j) - dt \cdot (\text{residue}_Ez \cdot \text{average}_Ez_{\text{adjoint}}')
\]

executes the inner product given by (9.22). Note that the second-order contribution does not require any differential residues and that the residues are all evaluated during the original simulation.

Executing the MATLAB code `evaluate_Hessian` gives the following Hessian estimate:

\[
\text{Hessian} = \\
1.0e-05 \times \\
\begin{bmatrix}
0.0000 & -0.0000 & -0.0014 & 0.0014 & -0.0012 & 0.0014 \\
-0.0000 & -0.0000 & 0.0378 & -0.0376 & 0.0331 & -0.0364 \\
-0.0014 & 0.0378 & -0.0020 & -0.0053 & 0.2155 & -0.2457 \\
0.0014 & -0.0376 & -0.0053 & -0.0037 & -0.2134 & 0.2431 \\
-0.0012 & 0.0331 & 0.2155 & -0.2134 & -0.0196 & -0.0046 \\
0.0014 & -0.0364 & -0.2457 & 0.2431 & -0.0046 & 0.0375 \\
\end{bmatrix}
\]

![Graphs showing the Hessian components](image_url)

**Figure 9.9** A subset of the Hessian components of the 2D example; the finite difference approximation (—) as compared to the adjoint sensitivities estimated using the fully adjoint approach (○)
The finite difference approximation, copied here for convenience, is given by

\[
\text{Hessian} = 1.0e-05 *
\begin{bmatrix}
0.0000 & -0.0000 & -0.0014 & 0.0014 & -0.0012 & 0.0014 \\
-0.0000 & 0.0000 & 0.0375 & -0.0379 & 0.0328 & -0.0366 \\
-0.0014 & 0.0375 & -0.0001 & -0.0053 & 0.2140 & -0.2457 \\
0.0014 & -0.0379 & -0.0053 & -0.0058 & -0.2135 & 0.2449 \\
-0.0012 & 0.0328 & 0.2140 & -0.2135 & -0.0199 & -0.0046 \\
0.0014 & -0.0366 & -0.2457 & 0.2449 & -0.0046 & 0.0381
\end{bmatrix}
\]

We could see from these results that all the significant Hessian components are reasonably matched. The fully adjoint approach requires only seven extra simulations (six perturbed original simulations plus one adjoint simulation). The finite difference approach requires 21 extra simulations. Figures 9.8–9.10 show a comparison between the estimated adjoint Hessian components and the corresponding finite difference estimates for a sweep of the relative permittivity. Good match is achieved for all responses.

**Figure 9.10** A subset of the Hessian components of the 2D example; the finite difference approximation (---) as compared to the adjoint sensitivities estimated using the fully adjoint approach (*)
References


Throughout the previous chapters, we utilized finite difference time domain (FDTD) as a tool for implementing AVM algorithms that apply to different materials and responses. The reason for this choice is that FDTD is the most widely used numerical technique. Many researchers and industry experts are familiar with the FDTD technique. We demonstrated that, for different types of responses, media, and boundary conditions, the sensitivities of the objective function or response with respect to all material and shape parameters are estimated using at most one extra adjoint simulation. Using the stored original and adjoint fields at the perturbation regions, we are able to estimate all sensitivities.

The reader may have noticed that for evaluating the sensitivities relative to material parameters (dielectric constant, conductivity, dispersion parameters, non-linearity parameters, etc.), the electromagnetic fields are stored all over the associated discontinuity. This may require extensive storage. For a cubic discontinuity with a size of $50\Delta l$ simulated for 20,000 time steps, the required storage of the original residues is approximately $51 \times 51 \times 51 \times 20,000 = 2.666 \times 10^8$ storage. This storage can grow very easily to prohibitive values for a discontinuity with a larger size or for a simulation for a larger number of time steps. We discuss in this chapter approaches that can significantly reduce the memory storage associated with the AVM method. We show that the methods of spatial and spectral sampling [1,2] can be applied to make the AVM approach more efficient.

We also discuss how the AVM approach can be implemented using numerical techniques other than the FDTD method. We discuss how the approach applies to the transmission line modeling (TLM) method [3]. The TLM method is another time domain technique that can be shown to be related to the FDTD [4]. We also briefly discuss how the AVM technique can be applied using frequency domain solvers such as the finite element method (FEM) [5] and the method of moments (MoM) [6]. We conclude this chapter by briefly illustrating some applications of adjoint sensitivity analysis.

10.1 AVM improvements

The algorithms discussed in the previous chapters are all basic algorithms. Their memory storages, computational time, and implementations can be improved to make them even more efficient. We have seen in Chapter 4, how the adjoint
simulation can be eliminated altogether for some objective functions. This self-
adjoint approach [7,8] reduces the number of required simulations to only one half.
The one-to-one mapping used in evaluating the adjoint responses can be eliminated in
some applications as reported in [9]. We focus in this section on two types of
sampling techniques that significantly reduce the memory requirement of the AVM
approach for practical problems. These approaches are the coarse space sampling
approach and the spectral sampling approach.

10.1.1 Coarse spatial sampling
The storage required for the AVM approach, as demonstrated in the previous
chapters, depends on the number of FDTD cells affected by the assumed pertur-
bation in each parameter. If the perturbation associated with a parameter affects the
material properties of a large subset of the computational domain, the required
storage may be prohibitive. Figure 10.1 illustrates the case where the sensitivities of
the response with respect to the relative permittivity of a region are estimated. In
this case, the required storage includes all field components within the region for all
time steps.

The sensitivities of the considered objective function with respect to the \( i \)-th
parameter, as demonstrated in Chapter 3, are given by the integrals:

\[
\frac{\partial F}{\partial p_i} = - \int_0^{T_w} \lambda^T R_i dt, \quad i = 1, 2, \ldots, n.
\] (10.1)

The \( i \)-th residue vector \( R_i \) is a temporal vector with a length equal to the number of
field components in the entire computational domain. Its nonzero components

![Figure 10.1 The sampled electric field components for a 2D FDTD simulation. The field is sampled at all nodes within the considered discontinuity](image-url)
correspond to the FDTD cells affected by the perturbation in the $i$th parameter. The integral (10.1) can thus be simplified into the expression:

$$\frac{\partial F}{\partial p_i} \approx \sum_k \sum_j A_j^T (k\Delta t) R_i (k\Delta t) \Delta t \Delta \Omega,$$

(10.2)

where $j$ is the index of the nodes in the perturbation domain associated with the $i$th parameter. For a material parameter, this index sweeps all cells within the associated discontinuity. For 1D FDTD problem with propagation in the $z$-direction, the spatial integration step $\Delta \Omega$ is given by $\Delta \Omega = \Delta z$. This sampling assumes that the field at every computational cell is utilized in estimating the adjoint-based sensitivities. For 2D FDTD problems in the $xz$ plane, the spatial integration step is $\Delta \Omega = \Delta x \Delta z$. This step assumes that we sample all the field components within the discontinuity as shown in Figure 10.1. For a 3D FDTD simulation, the spatial integration step is given by $\Delta \Omega = \Delta x \Delta y \Delta z$.

Throughout this book, we utilized a fine field sampling approach. All field components within the perturbation regions are utilized in estimating the adjoint sensitivities. It was shown, however, that this fine spatial sampling may not be required [1]. The spatial step $\Delta \Omega$ can be made larger without affecting the accuracy of the AVM estimates. This is illustrated in Figure 10.2 for the 2D FDTD case. The circles show the case where $\Delta \Omega = 4\Delta x \Delta z$. Half the number of field components is utilized in each dimension. The stars show the case where $\Delta \Omega = 16\Delta x \Delta z$. Here, the fields are sampled once every four cells along each dimension. Finally, the

Figure 10.2 Coarse sampling of the electric field components for a 2D FDTD simulation. The field is sampled at every two cells (●), every four cells (★), and every eight cells (♦). The same scale is used in both dimensions.
diamonds in Figure 10.2 indicate using only one field sample in each dimension every eight spatial cells. The same approach can be also applied to the full 3D case.

It was reported in [1] that this sampling approach significantly reduces the required memory storage while maintaining the AVM accuracy. Both the original and adjoint simulations are carried out as explained in the previous chapters, but the residues and the corresponding adjoint impulses are stored at a reduced spatial rate. It was shown that the accuracy of the AVM deteriorates only after the sampling rate becomes too low (below quarter wavelength of the highest excited frequency). It follows that the user can determine the lowest sampling rate possible without causing loss of accuracy. Reducing the sampling rate by one half in a 3D problem reduces the required memory storage to only one eighth of its original value. It follows that problems with larger discontinuities can be handled with a more reasonable memory requirement. Figure 10.3 illustrates the spatial sampling approach through a 2D example [10]. The example represents a dielectric discontinuity with $\varepsilon_r = 30$ and $\sigma^e = 6.0$ S/m placed inside a waveguide with $\varepsilon_r = 6$ and $\sigma^e = 0.2$ S/m. One-eighth of the fine sampling rate is utilized without significantly reducing the accuracy of the adjoint sensitivity estimates.

10.1.2 Spectral sampling

In Chapter 4, we addressed the case where the response of interest is a frequency-domain spectrum of a real response. A complex objective function of the form

$$\tilde{f}(\omega_0) = \int_0^{T_m} f(V(t))e^{-j\omega_0 t} dt$$

(10.3)

Figure 10.3 Sensitivity analysis of the S-parameter $|S_{11}|$ with respect to the conductivity of the 2D discontinuity. The sensitivities are estimated for the sampling factors 2, 4, 8, and 20
is considered. The adjoint sensitivity analysis of this problem are estimated by first applying a wideband excitation $h_o(t)$ to the original problem. The wideband time domain residues resulting from this excitation are $R_i(t)$, $i = 1, 2, \ldots, n$. These wideband residues are stored as temporal quantities over the complete simulation time. An adjoint problem is then set up with a wideband excitation $h_a(t)$. Instead of storing the temporal adjoint field $\lambda(t)$ at every time step, we instead store its phasors $\tilde{\lambda}_m = \tilde{\lambda}_m(f_m)$, $\forall m$ using discrete Fourier transform (DFT). The amplitudes and phases of these phasors are then adjusted and used to derive sinusoidal waveforms for both the real and the imaginary parts $\lambda_{m,\text{real}}(t)$ and $\lambda_{m,\text{imag}}(t)$ where

$$
\lambda_{m,\text{real}}(t) = \text{Re}(\tilde{\lambda}_m \exp(-2\pi j f_m t)),
\lambda_{m,\text{imag}}(t) = \text{Im}(\tilde{\lambda}_m \exp(-2\pi j f_m t)).
$$

(10.4)

To explain the spectral approach, we evaluate the adjoint-based sensitivities of the real part of the spectrum (10.3) at the frequency $f_m$. These sensitivities are given by the integral:

$$
\frac{\partial \text{Re}(\tilde{f}(f_m))}{\partial p_i} = -\text{Re} \left( \tilde{\lambda}_m^T \int_0^{T_m} \lambda_{m,\text{real}}(t) R_i(t) dt, \quad i = 1, 2, \ldots, n. \right)
$$

(10.5)

Substituting from (10.4) into (10.5) we have

$$
\frac{\partial \text{Re}(\tilde{f}(f_m))}{\partial p_i} = -\text{Re} \left( \tilde{\lambda}_m^T \int_0^{T_m} R_i(t) \exp(-2\pi j f_m t) dt, \quad i = 1, 2, \ldots, n, \right)
$$

(10.6)

where we utilized the fact that the residues $R_i(t)$ are purely real temporal vectors and that the phasors $\tilde{\lambda}_m$ are complex quantities that are independent of time. Using the definition of phasors, the adjoint-based sensitivities (10.6) are written as

$$
\frac{\partial \text{Re}(\tilde{f}(f_m))}{\partial p_i} = -\text{Re} \left( \tilde{\lambda}_m^T \int_0^{T_m} \exp(-2\pi j f_m t) R_i(t) dt, \quad i = 1, 2, \ldots, n, \right)
$$

(10.7)

where $\tilde{\lambda}_{i,m}$ is the vector of phasors corresponding to the temporal vector $R_i(t)$ at the frequency $f_m$. Similarly, the adjoint-based sensitivities of the imaginary part of the spectrum (10.3) at a frequency $f_m$ are given by

$$
\frac{\partial \text{Im}(\tilde{f}(f_m))}{\partial p_i} = -\text{Im} \left( \tilde{\lambda}_m^T \tilde{R}_{i,m} \right).
$$

(10.8)

The expressions (10.7) and (10.8) show how the storage in the original simulations can be drastically reduced. Instead of storing the complete temporal residues $R_i(t)$, $\forall i$ at one of the considered perturbation nodes, we only store their complex phasors $\tilde{R}_{i,m}$ at the frequencies of interest. The calculation of the sensitivities of the spectrum at a frequency with respect to the $i$th parameters is thus reduced to the inner product of two complex vectors of phasors at the frequency of interest. It should be clear that evaluating the vector of phasors during the original simulation using the
DFT slows down the original simulation. The more components the residue vectors $R_i$ have, the more the required number of computations. Figure 10.4 shows an electromagnetic structure used to illustrate the spectral sampling approach [2]. The field is excited at point Q and is observed at the two points P and Q. The sensitivities of the spectrum of the observed field are estimated using adjoint sensitivities relative to all parameters of the dielectric discontinuity. A comparison between the regular memory-intensive approach and the more efficient spectral approach is shown in Figure 10.5.

![Figure 10.4 An example for the spectral sampling method [2]](image)

![Figure 10.5 An illustration of spectral self adjoint sensitivity analysis [2]; time-domain AVM (−) vs. spectral AVM (Δ)](image)
10.2 AVM for other numerical techniques

The main focus in this book is to illustrate adjoint sensitivity analysis using the popular FDTD method. We showed through simple examples how the AVM approach can be implemented using the FDTD techniques for problems with different types of materials and responses. The AVM approach was also implemented though in the literature using several other numerical techniques. We aim in this subsection to show the reader how these implementations work and how they correspond to the FDTD case addressed throughout this book.

10.2.1 The TLM method

The TLM is based on Huygens’ principle which states that every point on a wave front acts as a secondary source [11]. The TLM method uses a mesh of connected transmission lines to model the propagation of electromagnetic waves. The electric and magnetic fields in the computational domain are modeled through transmission line voltages and currents. The parameters of the transmission lines are analogous to the properties of the modeled medium. Similar to the FDTD method, the fields are available at multiples of the time step $\Delta t$. However, all field quantities in the TLM method are collocated in both time and space. The staggering that Yee’s cell introduces in the FDTD method is not present in the TLM method.

Figure 10.6 shows the basic steps of the TLM approach for the 2D case. A voltage impulse of 1.0 V is incident on one of the four transmission lines. This impulse scatters at the center of the node and give rise to scattering voltage impulses on all transmission lines. The scattered impulses then travel to neighboring cells where they get scattered themselves at the next time step. In general, at any time step and for any TLM cell, there are four incident impulses on all four transmission lines giving rise to four scattered impulses in the homogenous 2D case. The scattered impulses are related to the incident impulses through the nodal

Figure 10.6 The scattering and connection steps of the TLM technique
scattering matrix $S^j$. The components of this matrix are functions of the local material properties and the utilized discretization. This process repeats itself causing the electromagnetic energy to spread all over the domain.

Figure 10.7 shows two TLM nodes; the 2D shunt node [3] and the symmetric condensed node (SCN) [4]. For the 2D shunt node, there are four voltage impulses incident at any node at the $k$th time step $V_k = [V_1 \ V_2 \ V_3 \ V_4]^T$. For the 3D SCN, the number of incident impulses is 12 for the homogenous case. Parallel or series extra transmission line stubs can also be integrated at the center of the node to enable modeling structures with non-uniform permittivity and permeability [12].

We assume that the computational domain is discretized into a total of $N$ nodes with a total number of TLM links of $N_L$. For the $j$th node, the scattering step is given by the formula:

$$V_{r,j}^k = S^j V_j^k,$$

where $V_j^k$ is the vector of incident impulses on the $j$th node at the $k$th time step. The vector $V_{r,j}^k$ is the vector of reflected (scattered) impulses of the $j$th node, and $S^j$ is the scattering matrix of the $j$th node. The scattered impulses from the $j$th node connect to all neighboring nodes. For a TLM problem with non-dispersive media, the update relationship for the whole computational domain is given by [13]:

$$V_{k+1} = CSV_k + V_s^k,$$

where $V_k$ is the vector of incident impulses at all nodes at the $k$th time step. $S$ is the block diagonal scattering matrix of the whole computational domain. The $j$th diagonal block of this matrix is $S^j$. $C$ is the connection matrix that describes how transmission lines exchange voltage impulses. $V_s^k$ is the vector of source excitation.
Using the update equation (10.10), we can derive a differential system similar to (3.9). By utilizing first-order Taylor expansion, one can write

\[ V_k + \left( \frac{\partial V}{\partial t} \right)_k \Delta t \approx C S V_k + V^s_k. \]  

(10.11)

Reorganizing the terms in (10.11), we have

\[ \frac{\partial V}{\partial t} + K(p)V = \frac{V^s}{\Delta t}, \]  

(10.12)

where the system matrix \( K \) is given by \( K(p) = (I - C(p)S(p))/\Delta t \) and \( I \) is the identity matrix. Comparing (10.12) with (3.9), we see that the system (10.12) is identical to the differential system used in all chapters for nondispersive systems with \( N = I, \quad K = \frac{I - C S}{\Delta t}. \)  

(10.13)

It follows that the approaches explained in the previous chapters are applicable to the TLM method. Instead of storing the electric and magnetic fields in the perturbation domains, we store the voltage impulses incident on the transmission lines.

As an example, we consider the microstrip stub discontinuity [14] shown in Figure 10.8. The substrate has a relative dielectric constant \( \varepsilon_r = 2.2 \) and a thickness of 6.0 mm. The microstrip line is 12.00 mm wide and 1.00 mm thick. The utilized TLM cell size is 1.00 mm. The excitation is a Gaussian-modulated sinusoid with a center frequency of 900 MHz and a bandwidth of 800 MHz. The objective function of interest is the energy function:

\[ F(p, V) = \int_0^{T_w} \int_\Omega e_y \cdot h_x dx dy dt, \]  

(10.14)

where \( e_y \) is the \( y \) component of the electric field, and \( h_x \) is the \( x \) component of the magnetic field. Figure 10.9 shows a comparison between the AVM sensitivities and those estimated using finite differences. Good agreement is observed between the two approaches.
10.2.2 Frequency domain methods

In frequency-domain numerical techniques method, the electromagnetic problem is solved for a single frequency at a time. If the desired response is wideband, the structure is simulated repeatedly for the different frequencies within the band of interest. The most used frequency domain techniques are the finite element method (FEM) [5] and the method of moments (MoM) [6]. The two methods are different in nature. The FEM utilizes volumetric tetrahedral elements to discretize the computational domain (see Figure 10.10). It solves for the fields over the

![Figure 10.9](image1)

**Figure 10.9** Sensitivities of the microstrip discontinuity for different values of (a) \( w \), and (b) \( L \) in mm; \( \partial F / \partial L \) estimated using AVM (\( \circ \)), \( \partial F / \partial L \) estimated using central difference (\( -- \)), \( \partial F / \partial w \) estimated using AVM (\( \square \)) and \( \partial F / \partial w \) estimated using central difference (\( -- \)) [14]

![Figure 10.10](image2)

**Figure 10.10** Volumetric discretization of the computational domain; the volume is divided into small tetrahedra where the unknown fields are expressed in terms of volumetric basis functions

10.2.2 Frequency domain methods

In frequency-domain numerical techniques method, the electromagnetic problem is solved for a single frequency at a time. If the desired response is wideband, the structure is simulated repeatedly for the different frequencies within the band of interest. The most used frequency domain techniques are the finite element method (FEM) [5] and the method of moments (MoM) [6]. The two methods are different in nature. The FEM utilizes volumetric tetrahedral elements to discretize the computational domain (see Figure 10.10). It solves for the fields over the
discretized volume. The MoM, on the other hand, discretizes only surfaces using surface elements (see Figure 10.11). It solves for the currents flowing on these surfaces and then uses these currents to calculate the fields everywhere. Each one of these two techniques is more suitable for certain types of applications.

Despite the difference in the way these two methods discretize a problem, both of them lead to a complex system of equations of the form

$$Z(p)E = Q \quad (10.15)$$

where $Z$ is the complex system matrix. This matrix is a function of the material properties and dimensions of the different discontinuities. $E$ is the vector of unknowns. For the FEM, this vector represents the volumetric fields. It represents the surface currents for the MoM. $Q$ is the excitation vector of the considered problem. Because these methods solve for one frequency at a time, the solutions obtained using (10.15) are complex and represent the phasors of the corresponding quantities.

The target of the AVM method is to efficiently estimate the derivatives of a complex objective function $f(E)$ with respect to all the parameters $p$. This objective function may represent, for example, the input impedance of an antenna or the $S$-parameters of a waveguide filter. Deriving the AVM sensitivities starts by

![Figure 10.11 Surface discretization of an antenna; the surface is discretized into triangles on which the surface currents are expressed in terms of basic functions](image)
differentiating both sides of (10.15) with respect to the \( i \)th parameter \( p_i \) to get

\[
\frac{\partial Z}{\partial p_i} E + Z \frac{\partial E}{\partial p_i} = 0,
\]  

(10.16)

where we assume in (10.16) that the excitation is independent of the parameters. The expression (10.16) leads to the result

\[
\frac{\partial E}{\partial p_i} = -Z^{-1} \frac{\partial Z}{\partial p_i} E.
\]  

(10.17)

The relationship (10.17) is then utilized to derive the adjoint system. The objective function \( f(E) \) is assumed to be implicitly dependent on the parameters \( p \) through the vector \( E \). The derivative of the objective function with respect to the \( i \)th parameter \( p_i \) is thus given by

\[
\frac{\partial f}{\partial p_i} = \left( \frac{\partial f}{\partial E} \right)^T \frac{\partial E}{\partial p_i}.
\]  

(10.18)

Substituting from (10.17) into (10.18), we have

\[
\frac{\partial f}{\partial p_i} = -\left( \frac{\partial f}{\partial E} \right)^T Z^{-1} \frac{\partial Z}{\partial p_i} E = -\left( \frac{\partial f}{\partial E} (Z^T)^{-1} \right)^T \frac{\partial Z}{\partial p_i} E.
\]  

(10.19)

The adjoint solution \( \lambda \) is defined using (10.19) as

\[
\frac{\partial f}{\partial E} (Z^T)^{-1} = \lambda \quad \Rightarrow \quad Z^T \lambda = \frac{\partial f}{\partial E}.
\]  

(10.20)

Equation (10.20) shows the essence of adjoint sensitivity analysis in the frequency domain. The adjoint response \( \lambda \) is calculated by solving the system of equations (10.20) whose matrix is the transpose of the original system matrix. The excitation of the adjoint system is the derivative of the objective function relative to the original responses \( E \). Once the adjoint system (10.20) is solved, the sensitivities of the objective function with respect to all parameters are given by

\[
\frac{\partial f}{\partial p_i} = -\lambda^T \frac{\partial Z}{\partial p_i} E, \quad i = 1, 2, \ldots, n.
\]  

(10.21)

Similar to the time-domain case, the sensitivities of the system matrix \( Z \) with respect to all parameters are assumed to be known either analytically or through finite differences applied in the frequency domain. It follows that using only the original simulation responses \( E \) and the adjoint responses \( \lambda \), the sensitivities of the objective function with respect to all parameters are estimated.

Similar to the time-domain case, in some applications, the adjoint responses can be deduced from the original response. In that case, there is no need to carry out the adjoint simulation (10.20). Only the original simulation is sufficient to evaluate the objective function \( f \) and its derivatives [7].
To illustrate the frequency domain case, we consider the patch antenna shown in Figure 10.12. This structure is simulated using HFSS [17] which is based on the FEM solution in the frequency domain. Figure 10.13 shows the AVM sensitivities of the real and imaginary parts of the $S_{11}$ as compared to the central finite difference approximations. Good agreement is observed between both approaches.

**Figure 10.12** A simple patch antenna simulated using HFSS

**Figure 10.13** The sensitivities of the real and imaginary parts of $S_{11}$ with respect to the patch length as compared to finite difference approximations
10.3 Applications

Over the past decade, adjoint-based techniques have been applied in different modeling and optimization approaches of high-frequency structures. They have been applied to many areas including the design of microwave structures [18], modeling of passive devices using artificial neural networks (ANNs) [19], wideband microwave imaging [20,21], sensitivity analysis of transmission line networks [22], the design of optical devices [23], and plasmonic structures [24]. The area of adjoint sensitivity is still emerging and many other applications will appear as adjoint algorithms become available in all commercial high-frequency solvers.

References


Index

adjoint-based gradients 221, 225
adjoint-based sensitivity approach 99
adjoint electric current density 77–8, 134
adjoint excitation 9, 11–12, 59, 66, 77, 87, 101, 124, 145
adjoint fields, storage of 54
adjoint SA approaches 2
adjoint simulator 112–13, 210
adjoint transient sensitivities 142
adjoint variable method (AVM) 7, 27, 49, 57, 68–9
for anisotropic materials 175–7
AVM algorithms, implementation of 157
AVM sensitivities vs. FFD sensitivities 83
derivations 195
improvements 247
applications 260
coarse spatial sampling 248–50
frequency domain methods 256–9
spectral sampling 250–2
transmission line modeling (TLM) method 253–6
sensitivities 212–13, 217
adjoint variable sensitivities 24
adjoint variable vector 49
Ampere’s law 30
anisotropic adjoint sensitivity algorithm 180
anisotropic discontinuity 177, 180
anisotropic material modeling 44–7
anisotropic structures, adjoint sensitivity analysis of 175–90
AVM for anisotropic materials implementation 177–90
anisotropy 175
anisotropy tensors 177
antenna surface discretization 257
artificial neural networks (ANNs) 260
auxiliary differential equation (ADE) 42, 157
backward finite differences (BFDs) 2, 3–4, 61, 83
basic equations 27–9
basic nonlinear AVM concept 199
biological tissues, electromagnetic behavior of 156
Central AVM (CAVM) 60–2
central finite differences (CFD) 2, 3–4, 24, 59, 122, 204
CFD sensitivities 212
coarse spatial sampling 248–50
Cole–Cole model 42
composite geometries 27
computational electromagnetics (CEM) 27
conductivity matrix 196
conjugate gradient techniques 1
convolution function 157–60
convolution integral 154, 160
convolution matrix 155, 158
Debye model 42, 156–7
derivative-based optimization techniques 1
discrete Fourier transform (DFT) 165, 251
dispersion profiles, types of 151
dispersive materials 151–72
general dispersive material case 151–7
Debye model 156–7
Drude model 156
Lorentz model 155–6
implementation 157–72
modeling 41–4
Drude models 42, 44, 156
electric conductivity 31, 176, 179, 197, 199
electric current density filament 144
electric field components 51, 88, 184, 249
excitation current filament, temporal variation of 91
Faraday’s law 30
finite difference approximations 2–6, 245
finite-difference time-domain (FDTD) 27–47, 99, 193, 247, 253
anisotropic material modeling 44–7
basic equations, 27–9
dispersive material modeling 41–4
modeling Drude medium using ADE technique 44
modeling Lorentz medium using ADE technique 42–4
FDTD algorithm 27, 29, 31, 177, 184, 186
FDTD computational space 30
FDTD simulations 53–4, 151
FDTD simulator 207
time-marching loop 47, 74
time marching scheme 54
updating equations for one-dimensional space 40–1
for three-dimensional space 29–36
for two-dimensional space 37–9
finite element method (FEM) 247
first-order sensitivity information 1
forward finite differences (FFD) 2, 3–4, 61, 97, 138
Fourier transform 27, 160, 167
frequency-dependent objective functions, sensitivity analysis for 99
monochromatic case 99–107
self-adjoint case 122–31
wideband base 107–21
frequency-dependent responses 133, 151
frequency domain methods 256–9
frequency domain spectrum 108
frequency independent constitutive parameters, adjoint variable method for 49–98
1D case 50–62
2D TM case 62–84
3D AVM algorithm 85–98
fully adjoint approach 230
algorithm 235–45
implementation 232–5
Gaussian-modulated excitation 56
Gaussian-modulated sinusoid 102, 112, 115, 123, 255
gradient-estimation 221
grid-based time-domain solvers 99
Hessian component 222, 227–9, 232–3, 238, 243–5
Hessian finite difference evaluation 221–4
hfss 259
high-frequency electromagnetic simulations 2
hybrid adjoint technique (HAT) 224–9
impulsive adjoint excitation 133, 138, 143
inverse discrete Fourier transform (IDFT) 108
isotropic media 44, 175, 179
Kirchhoff’s current law 200
K-matrix stamp 72, 74, 76
linear systems, adjoint sensitivity analysis of 6–24
linear time-invariant simulation 101
Lorentz dispersion profile 158
Lorentzian slab parameters of 161
spectrum of excitation of 163
transmission coefficient for 164
Lorentz models 41–2, 155–6
magnetic conductivity tensor 179, 184
magnetic field components 29, 31, 33, 36–7, 40, 43–4, 51–2, 77, 85, 134, 184
magnetic permeability tensor 179
material parameters 29, 31–2, 41, 64–6, 68–9, 71, 73–4, 80, 87, 94, 153, 157–8, 187, 211, 217–18, 231, 234, 247, 249
Maxwell’s equations 27–9, 31, 44, 49, 62, 85, 151–2, 176, 179, 194, 230
method of moments (MoM) 247
microstrip discontinuity, sensitivities of 256
microstrip rectangular stub 255
microstrip stub discontinuity 255
Monte Carlo approach 2
N-matrix stamp 72
nonlinear adjoint sensitivity analysis implementation 196–218
nonlinear AVM 193–6
nonlinear AVM 193–6, 205
nonlinear discontinuity 213–14
nonlinearity coefficients 194, 206–7
nonlinear material 193–4, 213
objective function, adjoint sensitivities of 189
1D AVM algorithm 66, 207
1D FDTD case 55
one-to-one mapping 54–5, 86–7, 248
for material parameters 66
for shape parameters 65
optical devices, design of 260
parameter-independent adjoint system 64, 86, 95
parameterization 54, 68, 103, 110
Perfectly Matched Layer (PML) 63, 111
permeability 31, 41–2, 151, 211, 254
permeability matrix 198
permittivity 31, 41, 42, 64, 67, 194, 197, 199, 211
permittivity tensor 44, 177–8, 180, 184–5, 188, 190
perturbation (parameter) 3–4, 52–4, 64–8, 71, 73–4, 88–90, 92–4, 153, 157, 160, 164–6, 172, 181, 194
perturbed permittivity 89
plasmonic structures 260
real-valued objective functions 100
rectangular waveguide
  metallic discontinuity in 123
  reference spectrum of 128
  scattering parameters 129
recursive convolution 42
RLC circuit 16
scalar objective function 111, 133, 151, 224
second-order adjoint sensitivities 221–45
  fully adjoint approach 230
  algorithm 235–45
  implementation 232–5
  Hessian finite difference evaluation 221–4
  hybrid adjoint technique (HAT) 224–9
second-order derivatives 221, 223, 230–1
second-order sensitivity information 1
    self-adjoint sensitivity analysis (SASA) 122
    Sensitivities 7, 9–10, 53–4, 141
    sensitivity-based optimization algorithm 1
    sensitivity information, obtaining 2
    shape parameters 65, 73–4, 79, 90, 92–5, 159
    $S$-parameters 3, 99–100, 123–4, 128, 130, 151, 163, 170, 172, 217, 250, 257
    spectral sampling method 250–2
    spectral self adjoint sensitivity analysis 252
    steepest descent method 1
    system matrices derivatives 8, 234, 241
tangential field components 88–9, 93
temporal residue vectors 49, 52
3D anisotropic case 176
3D AVM algorithm 85–98
3D FDTD case, AVM algorithm for 90–8
3D FDTD simulation 249
3D symmetric condensed node (SCN) 254
time-domain CEM models 27
time-domain electromagnetic simulations 99
time-domain Maxwell’s equations 27, 49
time-domain signal values 108
time-intensive simulations 1–2
time reversal 78, 81, 136
transformation electromagnetics 175
transient adjoint sensitivity analysis 133
  alternative formulation 142–9
  complete transient response case 138–42
  single time-response case 133–8
  time shift property 139
transient sensitivities 142–5
transmission-line matrix method 99
transmission line modeling (TLM) method 247, 253–6
transmission line networks, sensitivity analysis of 260
transverse electric and magnetic to $x$ (TEMx) fields 40
2D AVM code 92
2D FDTD simulation 248–9
2D metallic discontinuity 111, 114
2D TEz 49
2D TLM shunt node 254
Index

wideband adjoint sensitivity analysis 49, 110, 112
wideband adjoint treatment 99
wide-band microwave imaging 260

Yee’s cell 30, 50–1, 62, 88, 179, 253
Yee’s discretization 85, 176

yield analysis 2
yield integral 2
Y-parameters 99

Z-parameters 99
Z-transform 42