A Small-Perturbation Automatic-Differentiation Method for Determining Uncertainty in Computation Electromagnetics

Michael S. Gilbert, Member, IEEE and Fernando L. Teixeira, Member, IEEE

Abstract—A new methodology for approximating uncertainties in computational electromagnetics (CEM) solutions is discussed. The proposed method involves the use of a Taylor series expansion to account for small-perturbations (uncertainty) in the input parameters and subsequent use of an automatic-differentiation technique throughout the numerical calculation chain. This approach is denoted as “small-perturbation automatic-differentiation” (SPAD) method and provides certain benefits over other CEM uncertainty methods available in the literature. The method is illustrated in a couple of examples involving a moment-method solution of the electric field integral equation.

Index Terms—Computational Electromagnetics, Uncertainty, Moment-Method, Integral Equations, Small-Perturbation Method.

I. INTRODUCTION

In real-world problems of interest, inputs to computational electromagnetics (CEM) simulations are often not known with full certainty (or precision). The reasons for this are many, including the fact that it is often not possible to manufacture geometry exactly to the required specifications, results of material measurement systems have errors and uncertainty, and the frequency of operation and angles of observation are not absolutely precise in measurement systems. In spite of this, virtually all CEM algorithms presume a single deterministic solution (often from assumed mean values of the uncertain input parameters).

The goal of this work is to devise a general approach for incorporating uncertainties directly into a given CEM code. There have been many advancements in this particular area recently. One such approach involves integrating “stochastic methods” (another term used to illustrate the probabilistic nature of uncertainty) into the finite-element method (1), (2), and (3)). These techniques do appear to work reasonably well, but require extensive modifications to the underlying algorithms and are computationally expensive. Another uncertainty approach involves use of “stochastic collocation”. These methods employ a specific set of (deterministically evaluated) output points from a computational method to compute an integration (often using Stroud cubature [4]) that provides an estimate of statistical properties of interest. Some references dealing with this approach include [5], which describes higher-order collocation methods for stochastic differential equations in general, and [6] and [7], which apply collocation methods in conjunction with CEM solvers. Finally, reference [8] provides a general overview of fast methods in the broader field of uncertainty quantification.

The approach proposed in this paper attacks this problem from a different angle—using a Taylor series expansion augmented by an automatic-differentiation (AD) solver. Note that the Taylor expansion is a classic solution to many problems, including statistical ones. It has, in fact, already been utilized as a method for solving probabilistic finite-element problems [9],[10]. These methods utilize the Taylor series to directly expand the resulting mass, damping, and stiffness matrices obtained from the finite-element discretization. The distinctive aspect of the approach described here is that (in essence) every operational step in the computational process is expanded rather than only certain higher-level quantities obtained via a particular solution technique; as such, the procedure can be more fully automated.

AD (sometimes referred to as algorithmic differentiation) is a technique originally envisioned as far back as the time of Newton and Leibniz [11]. In AD, the derivatives at each step of a computation chain are determined from the derivatives of the inputs to that step, thus producing any and all subsequent derivatives and culminating in the derivatives for any output of interest. References providing a good background on the AD methodology include [12], [13], [14], and [15]. Note that there are two basic approaches to automatic-differentiation: forward and backward. The discussion in this paper is based on the forward approach.

One advantage of the SPAD method proposed here is that it requires minimal changes to the underlying code (existing deterministic CEM solver), and it may be more efficient as compared to existing methods (based on chosen limits). Note that collocation methodologies [5] share the same advantage, but cannot support correlated input variables and do not provide a direct route for higher-order error analysis. Another SPAD advantage (also shared by collocation methods) is that, given the derivative values produced via the AD process, output statistics can be computed quickly for other assumed probability distributions without the need for any code rerun1. However, a limitation from this method is that it is only applicable to “small” perturbations, whereas some other methods can be less restrictive in terms of the input

1This statement assumes that the new probability distribution converges with less or same number of terms used in the original expansion.
parameter variations. Nevertheless, scenarios involving small deviations (such as tolerances in manufacturing and variability in measurement systems) do comprise the vast majority of CEM problems.

II. SPAD METHODOLOGY

A. Single-Variate Case

The Taylor series approximation for \( \Phi(x) \) expanded about the expected value of an input variable \( x = \langle x \rangle \), can be written as

\[
\Phi(x) \approx \sum_{n=0}^{N} \frac{(x - \langle x \rangle)^n}{n!} \frac{d^n \Phi(x)}{dx^n} \bigg|_{x=\langle x \rangle}
\]

where \( N \) is the number of terms in the expansion, \( \Phi(x) \) is a chosen output produced by a CEM calculation, and \( x \) is an input with uncertainty. Typically, \( \Phi(x) \) will correspond to quantities such as radar cross-section, antenna gain, or input impedance; while \( x \) may correspond to some geometrical parameter, frequency, or incidence angle. Because all of the derivatives of \( \Phi(x) \) are evaluated at deterministic values, \( x = \langle x \rangle \), it is possible to write a “separable” expression for the desired objective

\[
\langle \Phi(x) \rangle \approx \sum_{n=0}^{N} \frac{(x - \langle x \rangle)^n}{n!} \frac{d^n \Phi(x)}{dx^n} \bigg|_{x=\langle x \rangle}
\]

\[
\approx \sum_{n=0}^{N} A_n \frac{d^n \Phi(x)}{dx^n} \bigg|_{x=\langle x \rangle}
\]

where \( A_n = \langle (x - \langle x \rangle)^n \rangle \) are the \( n \)-th order statistical central moments of the input random variable \( x \). Note that Equation (2) requires the corresponding moments of \( A_n \) to converge. The latter is true for many commonly used probability density functions of interest, such as Gaussian and uniform distributions. For example, the central moments for a Gaussian random variable are given by

\[
A_n = \langle (x - \langle x \rangle)^n \rangle = \begin{cases} 0 & n \text{ odd} \\ \frac{(n-1)!!}{n} \langle \Delta x \rangle^n & n \text{ even} \end{cases}
\]

where \( \Delta x \) is the standard deviation of \( x \). Combining (2) and (3), the expression for the expected value of \( \Phi(x) \) for a Gaussian variable \( x \) can be written as

\[
\langle \Phi(x) \rangle \approx \sum_{n=0}^{N/2} \frac{(\Delta x)^{2n}}{(2n)!!} \frac{d^{2n} \Phi(x)}{dx^{2n^2}} \bigg|_{x=\langle x \rangle}
\]

For a uniform random variable \( x \), the central moments are given by

\[
A_n = \langle (x - \langle x \rangle)^n \rangle = \begin{cases} 0 & n \text{ odd} \\ \frac{\langle \Delta x \rangle^n}{n+1} & n \text{ even} \end{cases}
\]

Combining (2) and (4), the expected value of a function of uniform random variable \( x \) can be found via

\[
\langle \Phi(x) \rangle \approx \sum_{n=0}^{N/2} \frac{3^n (\Delta x)^{2n}}{(2n+1)!} \frac{d^{2n} \Phi(x)}{dx^{2n}} \bigg|_{x=\langle x \rangle}
\]

where the statistical parameters \( \langle x \rangle \) and \( \Delta x \) for the uniform distribution are related to the upper and lower limits \( b \) and \( a \) respectively over which that particular distribution is defined, i.e.

\[
\langle x \rangle = \frac{b + a}{2}, \quad \Delta x = \frac{b - a}{2\sqrt{3}}
\]

From the above, it can be observed that if the higher-order derivatives of \( \Phi(x) \) are known (and the series converges), then the expected value of \( \Phi(x) \) can be found. Of course, practically speaking, the series needs to be truncated, so at best, this method can only find a good approximation for the expected value. Since the convergence is faster for small \( \Delta x \), the term “small-perturbation” is apropos. This approach is “separable” in the sense that, for a given practical implementation, (2) needs to be computed only once. Once the derivatives are calculated for a specific expected value, it is rather fast to recompute it for any change in the probability distribution of the input random variable since those two parts of the resulting equation are separate. In other words, \( A_n \), is replaceable by (3), (4), or any set of known central moments for the input variables.

B. Derivative Computation

It is clear that, at this point, the problem has been reduced to determining all the derivatives up to a certain order of a (numerically computed) function \( \Phi(x) \), i.e. determining

\[
\frac{d^n \Phi(x)}{dx^n} \bigg|_{x=\langle x \rangle}
\]

One way to find these derivatives is via an “automatic-differentiation” procedure, which will involve only minimal modifications to the same deterministic CEM code that produced \( \Phi(x) \). Indeed, for an object-oriented CEM code, the only modification needed will be on the variable declarations because operators such as addition and multiplication can be overridden in such codes. Automatic-differentiation proceeds by decomposing the numerical operations used to compute \( \Phi(x) \) into elementary operations that can be differentiated individually and later combined using the chain rule. The details of automatic-differentiation procedure adopted here are provided in the Appendices. References [16] and [17] expound on automatic-differentiation from a general viewpoint.

Automatic-differentiation is an alternative approach to either numerical differentiation (i.e., finite-differencing of multiple solutions at nearby input data points) or symbolic differentiation (using symbolic mathematical packages). Numerical differentiation is problematic in this context because \( \Phi(x) \) is subject to “noise” due to numerical errors. It is well-known that numerical differentiation of such data is ill-posed: small errors in the data can produce large errors in the numerical approximation of the derivative [18]. This problem is compounded for the higher-order derivatives that are required here. With that said, finite-differencing might prove useful in certain situations and can be a potential avenue for future research. Symbolic differentiation, on the other hand, is problem-dependent and cannot be fully done in a practical fashion for a function \( \Phi(x) \) that is generated by codes of very high complexity, only to relatively simple functions. Still,
symbolic differentiation can be hybridized with the automatic-differentiation proposed here, i.e. with the application of the former to specific, limited parts of the computation chain. This could speed up the computation but at the expense of automation and generality and may be another avenue for future research.

C. Higher-order Statistical Moments

Given any one of these methods for obtaining the derivatives (and consequently the expected value), it is relatively straightforward to determine higher-order statistical moments as well. For example, the standard deviation can be computed by defining $\Lambda(x) = [\Phi(x)]^2$, calculating the higher-order derivatives of $\Lambda(x)$ via the routines as described in Appendix A, and using those values to determine the average $\langle \Lambda(x) \rangle$ (2). Finally the standard deviation for function $\Phi(x)$ can be calculated from

$$\Delta\Phi(x) = \sqrt{\langle \Lambda(x) \rangle - \langle \Phi(x) \rangle^2}$$

(8)

The same general idea can be applied to obtain any higher-order statistical moment of interest.

D. Multivariate Case

Single-variate uncertainty, while interesting, is not of primary interest itself since such computations can easily be achieved via Monte Carlo methods and simple integration techniques. It is however useful as a starting point for the more general problem; multivariate uncertainty, which will be discussed in detail in this section.

For this problem, the output function of interest $\Phi(x)$ is a function of multiple random input variables $x$, where $x = \{x_0, \ldots, x_d\}$ and $d$ is the number of random variables. Note that these variables are all potentially correlated. This relationship is described via covariance coefficients $C(x_l, x_m)$ between pairs of variables, the set of which form a symmetric $d \times d$ matrix. Note that covariance is related to correlation via

$$\rho(x_l, x_m) = \frac{\Delta x_l \Delta x_m \rho(x_l, x_m)}{\Delta x_l \Delta x_m}$$

(9)

where $\rho(x_l, x_m)$ is the correlation coefficient between variables $x_l$ and $x_m$ and $-1 \leq \rho(x_l, x_m) \leq 1$.

Again, the goal of this effort is to obtain an approximation for the expected value of any function interest. In this case, the same Taylor series expansion can be applied, although now in multiple dimensions

$$\langle \Phi(x) \rangle \approx \left\langle \sum_{n_0=0}^{N_0} \cdots \sum_{n_d=0}^{N_d} \left[ \prod_{k=0}^{d} \frac{(x_k - \langle x_k \rangle)^{n_k}}{n_k!} \frac{d^{n_k}}{dx_k^{n_k}} \right] \Phi(x) \right|_{x=(x)} \right\rangle$$

$$\approx \sum_{n_0=0}^{N_0} \cdots \sum_{n_d=0}^{N_d} A_n B_n \left[ \prod_{k=0}^{d} \frac{d^{n_k}}{dx_k^{n_k}} \right] \Phi(x) \right|_{x=(x)}$$

(10)

where $N_0, \ldots, N_d$ are the number of terms used in the expansion for each input variable (usually chosen to be the same number, $N = N_0 = \ldots = N_d$), $B_n$ is a simple combination of factorials and is given by

$$B_n = \prod_{k=0}^{d} \frac{1}{n_k!}$$

(11)

and

$$A_n = \prod_{k=0}^{d} (x_k - \langle x_k \rangle)^{n_k}$$

(12)

are the higher-order multivariate central moments of $x$. Similar to the single-variate case, these are all solely dependent on the covariances between the elements of $x$. As an example, if all of the elements of $x$ are jointly Gaussian, then the central moments are

$$A_n = \begin{cases} \sum_{l,m} \prod C(x_l, x_m) P_d \text{ even} \\ 0 \quad P_d \text{ odd} \end{cases}$$

(13)

where $\sum \prod$ symbolizes a sum over all $P_d/2$ products of the covariances between elements of $x$ [19] and

$$P_d = \sum_{k=0}^{d} n_k$$

(14)

For example, if $P_d = 4$ then $A_n$ is all of the combinations of pairs of covariances, and if $P_d = 6$ then $A_n$ is all of the combinations of triplets of covariances. There are $(P_d-1)!!$ of these terms/combinations in $A_n$ (for example there are 3 terms for $P_d = 4$ and 15 terms for $P_d = 6$). For other multivariate distributions (i.e. not jointly Gaussian), the form of $A_n$ is not quite so straightforward, but it can certainly be derived.

Finally with respect to (10),

$$\prod_{k=0}^{d} \frac{d^{n_k}}{dx_k^{n_k}} \Phi(x) \right|_{x=(x)}$$

are the higher-order multidimensional derivatives of $\Phi(x)$. Similar to the single-variate case, these are the values that need to updated throughout the computation. Conveniently, the foundations established in the previous single-variate section can be reused. However, the computational cost will be significantly greater since there are many more derivative values involved. Details of the multivariate automatic-differentiation approach are described in Appendix B.

Note that it also becomes possible to determine all of the higher-order moments of that function (in the same manner as described in the previous section for the single-variate case).

III. Examples

A. Induced Line Charge

According to [20] the deterministic electrostatic charge induced on a cylindrical wire can be obtained by solving the matrix equation

$$p = Z^{-1}v$$

(15)
where \( p_n \) are the induced line charges to be calculated, \( v_m = 4\pi\varepsilon_0 V_m \), and the elements of \( Z \) are given by

\[
Z_{mn} = \begin{cases} 
2\ln\left(\frac{\sqrt{s^2 + \Delta a^2} + |s/2|}{a_m}\right) & m = n = 0 \\
\ln\left(\frac{c_{mn} + \Delta a_{mn} + d_{mn}}{d_{mn} + \sqrt{d_{mn}^2 + a_m^2}}\right) & 0 < |m - n| \leq 2 \\
\ln\left(\frac{c_{mn}}{d_{mn}}\right) & |m - n| > 2
\end{cases}
\]  

(16)

where the wire is divided into \( N \) segments over length \( l \), \( V_m \) is the applied voltage, and \( a_m \) is the wire radius at segment \( m \). Note that \( c_{mn} = l_{mn} + s/2, d_{mn} = l_{mn} - s/2, l_{mn} = |y_m - y_n|, y_n = ns, s = l/N, n = \{0, ..., N\}, \) and \( m = \{0, ..., N\} \).

Using the fourth-order multivariate approach described in Appendix D, the radius \( a_m \) of each segment was chosen to be the same for each segment \( \langle a_m \rangle = 1 \) millimeter and the standard deviations were chosen to be \( \Delta a_m = \langle a_m \rangle / (10 + m) \). It is assumed that the radii are uncorrelated

\[
C(a_m, a_n) = \begin{cases} 
0 & m \neq n \\
\Delta a_m^2 & m = n
\end{cases}
\]  

(17)

in this case.

Using a deterministic line charge code based off of the above equations from [20], the approach described in Appendix B was utilized. The core of the approach was implemented using object oriented methods to override the mathematical operations (i.e. addition, multiplication, exponentiation, etc.) already in use by the deterministic implementation. Hence with this approach, an uncertainty-augmented code was easily written by recycling an existing deterministic code.

Note that the higher-order derivatives of the natural logarithm were needed to calculate the elements of the \( Z \) matrix. These are given by

\[
\frac{d^n}{du^n} \ln[u]_{u=g(x)} = \begin{cases} 
\ln[g'(x)] & n = 0 \\
\frac{d^{n-1}}{du^{n-1}}u^{-1} |_{u=g(x)} & n > 0
\end{cases}
\]  

(18)

where the derivatives of \( u^{-1} \) are given by the power derivatives described in Appendix A (38).

Also note that the matrix equation (15) was solved using Gaussian elimination, but any matrix solver, either direct or iterative, could have been used since the underlying mathematical operators are simply overridden.

Results for two scenarios for this example were calculated. In Figure 1, the line is divided into 5 segments and in Figure 2, the line is divided into 16 segments. As can be seen, the new method agrees very well with a Monte Carlo reference using 10^6 sample points.

### B. Scattering by Infinite Parallel Wires

While the previous electrostatics problem is revealing, the following example will demonstrate a more interesting electrodynamics scenario. For this case, the scattering from a set of infinite wires will be evaluated where the positions/placement of the wires are uncertain by a small amount.

Just like the line charge example, the goal is to solve the matrix equation

\[
\mathbf{p} = \mathbf{Z}^{-1}\mathbf{v}
\]  

(19)

where for infinite parallel wire scattering [21], the known elements are given by

\[
v_m = e^{-i(k_{zi}x_m + k_{yi}y_m)}
\]

\[
Z_{mn} = H_0^{(2)}(k_{\rho_{mn}})
\]  

(20)

and again the goal is to solve for the induced currents, i.e. the \( p_n \) values. Note that \( x_m \) and \( y_m \) are the coordinates of wire \( m \), \( a_m \) is the radius of wire \( m, m = \{0, ..., N\}, \) \( n = \{0, ..., N\} \), \( N \) is the number of wires, \( k = 2\pi/\lambda \) is the wavenumber, \( \lambda \) is the wavelength, \( k_{\rho_{mn}} = \sqrt{k^2 - k_{zi}^2}, k_{zi} = k \sin \theta_i \cos \phi_i, k_{yi} = k \sin \theta_i \sin \phi_i, k_{zi} = k \cos \theta_i, \) and \( \theta_i \) and \( \phi_i \) are the angles of the incident wave, and \( \rho_{mn} \) is given by

\[
\rho_{mn} = \begin{cases} 
a_m & m = n \\
\sqrt{(x_m - x_n)^2 + (y_m - y_n)^2} & m \neq n
\end{cases}
\]  

(21)

Once the induced currents \( p_n \) are found, the scattered field can be obtained by summing the radiated field of each
individual wire according to [21] as
\[ E(\phi_s) = \sum_{n=0}^{N-1} p_n e^{ik \phi_s (x_n \cos \phi_s + y_n \sin \phi_s)} \]  
(22)
where \( \phi_s \) is the observation angle.

Note that in order to apply the small-perturbation approach, the higher-order derivatives of the Hankel function are needed. These can be determined by applying the well-known recursion relation [22]
\[ \frac{d}{du} H^{(2)}_m(u) = \frac{1}{2} \left[ H^{(2)}_{m-1}(u) - H^{(2)}_{m+1}(u) \right] \]  
(23)
many times, thus producing the needed higher-order form, which is
\[ \frac{d^n}{du^n} H^{(2)}_m(u) \bigg|_{u=g(\chi)} = \frac{1}{2^n} \sum_{k=0}^{n} (-1)^k \binom{n}{k} H^{(2)}_{m+2k-n}(g(\chi)) \]  
(24)

A few scenarios were also calculated for this example. Figure 3 demonstrates the expected value and standard deviation of the scattering from 8 aligned uncorrelated parallel wires and Figure 4 presents the same for 12 aligned wires. As in the line charge example, it can be seen again that the new approach agrees very well with the Monte Carlo reference (again using \( 10^6 \) samples). Common deterministic inputs for these cases are: wavelength \( \lambda = \pi \), incident angles \( \theta_i = 90^\circ \) and \( \phi_i = 0^\circ \). In this case the uncertain parameter was chosen to be the \( y \)-placement of each of the wires. In Figures 3 and 4, these are assumed independent and given by
\[ C(y_n, y_m) = \begin{cases} 0 & m \neq n \\ \Delta y_n^2 & m = n \end{cases} \]  
(25)
where \( \Delta y_n = s/(n+10) \). Figure 5 demonstrates another case where the wire locations are in fact correlated. For this result, all parameters are chosen the same as for the previous cases except for the covariance matrix, which was chosen as
\[ C = \begin{bmatrix} \Delta y_0^2 & 0.85 \Delta y_0 \Delta y_1 & 0.86 \Delta y_0 \Delta y_2 \\ 0.85 \Delta y_1 \Delta y_0 & \Delta y_1^2 & 0.87 \Delta y_1 \Delta y_2 \\ 0.86 \Delta y_2 \Delta y_0 & 0.87 \Delta y_2 \Delta y_1 & \Delta y_2^2 \end{bmatrix} \]  
(26)
These correlation values were chosen arbitrarily for example purposes, and the number of wires was limited to three for readability/simplicity.

C. Discrete-Dipole Approximation (DDA) example

A final item of interest is to understand how the error scales versus the number of terms included in the truncated Taylor series. To illustrate this, a discrete-dipole problem [23],[24] was chosen. The example considered for this purpose involves two particles with an uncertain separation distance \( d \), where the uncertainty in \( d \) is given by a uniform probability distribution. The deterministic parameters for this problem are: frequency \( f = 10 \) GHz, particle radius \( a = 1 \) mm, and relative permittivity \( \varepsilon_r = 3 + 0.1i \). Results of the uncertainty in the backscattered RCS from these two particles are shown in Figure 6. This Figure shows the relative error on the RCS uncertainty w.r.t. (with respect to) Monte Carlo results and indicates the existence of two well-defined regimes (and a “radius of convergence”) that depend on the ratio \( \Delta d/\langle d \rangle \) where \( \Delta d \) is the standard deviation of \( d \). The relatively small number of terms required for convergence is likely due to the thin-tailed nature of the Gaussian distribution. Other, more fat-
tailed distributions could be more problematic and require a larger number of terms.

IV. CONCLUSIONS

It has been demonstrated that a small-perturbation automatic-differentiation (SPAD) method can be applied to determine the consequences of uncertainty in CEM calculations. The present approach offers a few advantages over other “stochastic methods” found in the existing literature. First, it can be applied to any existing object-oriented deterministic CEM code with minimal changes; only modification of variable declarations are needed. This is because operators such as addition and multiplication can be overridden in object-oriented languages. Second, the increased computational complexity is small when the number of terms chosen in the expansion is small, which is feasible when uncertainties of the input parameters are small. Finally, the result of the expansion applied is “separable” in the sense that once the derivatives are calculated for a specific expected value, it is rather fast to recompute it for any change in the probability distribution of the input variable(s) since those two parts of the resulting equation are separate. Stated another way, the probabilistic aspects of the problem have been separated from the core of the approach.

Since Stroud-based collocation techniques are considered to be perhaps the most computationally cost-effective at present, it is worth spending some time to discuss the relative merits of the SPAD technique with respect to this method. Interestingly, Stroud cubature only requires $O(d)$ operations to compute statistical properties of interest for both degree 2 and 3 versions of that theory, where $d$ is the number of random variables. As described in Appendix D, the SPAD approach can also produce results with the same numerical complexity; however, correlations between random variables would not be supported in such a configuration or via the Stroud methodology. At $O(d^2)$ cost (for which all results in this paper have been generated), the SPAD methodology in contrast does support correlation between input variables. Another difference is that (as discussed in [5]) collocation methods cannot produce error estimates on the result obtained; whereas such estimation is possible with the SPAD methodology.

One disadvantage of the SPAD methodology is that it does (as its name implies) require small perturbations and thus has a more limited range of applicability. However, as mentioned in the Introduction, most practical scenarios entail small perturbations and the well-known behavior of the Taylor series makes it tractable to understand these limits very well. Finally, it should be pointed out that the SPAD methodology is not limited to the method of moments as illustrated here and in principle can be applied to other CEM algorithms as well.

APPENDIX A

**SINGLE-VARIATE OPERATIONS**

With the goal of determining the higher-order derivatives of a function of interest, an exact automatic-differentiation approach is discussed in this appendix.

A CEM code is based on the execution of a sequence of arithmetic operations such as additions and multiplications, and evaluation of functions such as trigonometric functions, exponentials, special functions, etc. The fundamental idea of automatic-differentiation is that, through application of the chain rule to each arithmetic operation on a CEM code, it is possible to compute derivatives of arbitrary order exactly (to numerical accuracy).

It is convenient to first explain two approaches for obtaining the *initial* derivative values to be used as inputs for the chain rule computation. One approach would be to use a symbolic math package to obtain in closed-form the (analytical) derivatives of certain expressions along the numerical chain. The main advantage to this approach is that some chain rule computations will be avoided because the derivatives are simplified upfront; however, this has the drawback that the implementation will be tied to the particular problem at hand. Another approach is to utilize the fact that the derivatives of the input random variables are known and given by

$$
\frac{d^n}{dx^n}f(x)\bigg|_{x=(x)} = \begin{cases} 
\langle x \rangle & n = 0 \\
1 & n = 1 \\
0 & n \geq 2
\end{cases}
$$

This approach has the advantage in that it is generally applicable to any problem; however, since there are no simplifications upfront, the number of numerical calculations will be higher than for the symbolic approach.

Starting from the above known derivatives (27), the derivatives of any function compositions of the above can also be obtained via automatic-differentiation. For example, for an addition operation $f(x) = g(x) + h(x)$ (with higher-order derivatives of $g(x)$ and $h(x)$ already computed), the new derivatives are

$$
\frac{d^n}{dx^n}f(x)\bigg|_{x=(x)} = \frac{d^n}{dx^n}\left[g(x) + h(x)\right]_{x=(x)} = \frac{d^n}{dx^n}g(x)\bigg|_{x=(x)} + \frac{d^n}{dx^n}h(x)\bigg|_{x=(x)}
$$

**Fig. 6** Discrete-dipole approximation error analysis with respect to number of expansion terms and perturbation magnitude
where \( n = \{0, ..., N\} \) and again \( N \) is the number of terms used in the Taylor series expansion. For multiplication, the generalized Leibnitz rule (a widely known higher-order generalization of the product rule) can be applied. Given known derivatives for \( g(x) \) and \( h(x) \), the derivatives of \( f(x) = g(x)h(x) \) are

\[
\frac{d^n}{dx^n} f(x) \bigg|_{x=(x)} = \frac{d^n}{dx^n} \left[ g(x)h(x) \right] \bigg|_{x=(x)} = \sum_{m=0}^{\infty} \binom{n}{m} \frac{d^m}{dx^m} g(x) \bigg|_{x=(x)} \frac{d^{n-m}}{dx^{n-m}} h(x) \bigg|_{x=(x)}
\]

(29)

where

\[
\binom{n}{m} = \frac{n!}{m!(n-m)!}
\]

(30)

is the binomial coefficient, which gives the number of ways to choose \( m \) elements from a set of \( n \). Equation (29) is broadly known, but [25] puts it in a context useful for the approach pursued in this effort.

To obtain derivatives of all other operations, the Faà di Bruno formula (a higher-order generalization of the chain rule) can be applied. This rule is also defined in a useful context in [25]. However, in terms of an efficient numerical implementation it is far more ideal to make use of the Bell polynomial form as described in [26]. In this expression, lower-order polynomials are used recursively to determine higher-order ones. Using this form, the derivatives for \( f(x) = h(g(x)) \) can be found via

\[
\frac{d^n}{dx^n} f(x) \bigg|_{x=(x)} = \frac{d^n}{dx^n} h(g(x)) \bigg|_{x=(x)} = \sum_{m=0}^{n+1} B_{mn} \frac{d^{n+1}}{du^{n+1}} h(u) \bigg|_{u=g(x)}
\]

(31)

for \( n > 0 \) and

\[
\frac{d^n}{dx^n} f(x) \bigg|_{x=(x)} = h(g(\langle x \rangle))
\]

(32)

where the partial Bell polynomials \( B_{mn} \) are given by

\[
B_{mn} = T_{m,(n+1)} \frac{(m+1)!}{(n+1)!}
\]

(33)

where

\[
T_{mn} = \begin{cases} 
\frac{d^n}{dx^n} g(x) \bigg|_{x=(x)} & m = 0 \\
\sum_{k=1}^{n} \binom{n}{k} T_{nk} T_{(m-1),(n-k)} & m > 0, n > m \\
0 & n \leq m
\end{cases}
\]

(34)

For generality, the function \( h(u) \) is explicitly left undefined above since it can be validly replaced by any operation with known derivatives. For example, if the operation of interest is \( f(x) = e^{g(x)} \), then \( h(u) = e^{u} \), and its straightforward set of higher-order derivatives are

\[
\frac{d^n}{du^n} h(u) \bigg|_{u=g(x)} = e^{g(\langle x \rangle)}
\]

(35)

which can be inserted into the chain rule formula (31). Note that \( g(\langle x \rangle) \) is available as one of the known derivatives (the zeroth one) of the input function \( g(x) \).

Another interesting set of operations that require utilization of the chain rule are powers. The function of interest in this case is \( f(x) = (g(x))^m \) or \( h(u) = u^m \), which has higher-order derivatives given by

\[
\frac{d^n}{du^n} h(u) \bigg|_{u=g(x)} = \frac{m!}{(m-n)!} g(\langle x \rangle)^{m-n}
\]

or in terms of an efficient implementation

\[
\frac{d^n}{du^n} h(u) \bigg|_{u=g(x)} = g(\langle x \rangle)^m
\]

(36)

Note that various other mathematical operations can be obtained by reusing the above equations. For example, the higher-order derivatives of the square root operation can be determined with \( m = 0.5 \), and reciprocals (and consequently the division operation) can be obtained with \( m = -1 \).

All of these operations (addition, multiplication, and the chain rule) produce exact derivatives (to within machine precision) at every step of the computation. However, in terms of a real numerical implementation, machine precision can be an issue because rounding errors can accumulate along the numerical chain. A straightforward solution to this problem is to use decimal floating point values, rather than the default binary floating point used in most numerical computations today. This does cause a performance hit on hardware that does not support the IEEE 754-2008 standard [27], which includes most hardware available as of 2011.

**Appendix B**

**Multivariate Operations**

This Appendix discusses the automatic-differentiation procedure for the multivariate case. The scheme for initializing derivative values in this case is given by

\[
\frac{d^k}{dx^k} x_m \bigg|_{x=(x)} = \begin{cases} 
\langle x_m \rangle & m = l, k = 0 \\
1 & m = l, k = 1 \\
0 & \text{otherwise}
\end{cases}
\]

(39)

Via induction, the outcome of multivariate addition is

\[
\prod_{k=0}^{d} \frac{d^{nk}}{dx^{nk}} \Phi(x) \bigg|_{x=(x)} = \prod_{k=0}^{d} \frac{d^{nk}}{dx^{nk}} \left[ g(x) + h(x) \right] \bigg|_{x=(x)} = \prod_{k=0}^{d} \frac{d^{nk}}{dx^{nk}} g(x) \bigg|_{x=(x)} + \prod_{k=0}^{d} \frac{d^{nk}}{dx^{nk}} h(x) \bigg|_{x=(x)}
\]

(40)
Also, the multivariate multiplication rule [25] is given by
\[
\left[ \prod_{k=0}^{d} \frac{d^{m_k}}{dx_k^{n_k}} \right] \Phi(x) \bigg|_{x=(x)} = \left[ \prod_{k=0}^{d} \frac{d^{m_k}}{dx_k^{n_k}} \right] g(x) h(x) \bigg|_{x=(x)} = \left[ \prod_{k=0}^{d} \frac{d^{m_k}}{dx_k^{n_k}} \right] \left( \prod_{k=0}^{\infty} \frac{d^{m_k}}{dx_k^{n_k}} \right) g(x) \bigg|_{x=(x)}
\]
which can be derived by applying the single-variate multiplication rule (Faa di Bruno Formula) to progressively higher orders and extrapolating to express the general multivariate rule.

It is rather difficult to explicitly state the multivariate chain rule, and it does not appear to be stated explicitly in [25]. An approach for determining the indices is as follows
\[
\left[ \prod_{k=0}^{d} \frac{d^{m_k}}{dx_k^{n_k}} \right] \Phi(x) \bigg|_{x=(x)} = \left[ \prod_{k=0}^{d} \frac{d^{m_k}}{dx_k^{n_k}} \right] h(g(x)) \bigg|_{x=(x)} = \sum_{k=1}^{P_d} D(k, n) \frac{d^k}{du^k} h(u) \bigg|_{u=g(x)}
\]
for \( P_d > 0 \) where \( P_d \) is defined in Equation (14) and
\[
\left[ \prod_{k=0}^{d} \frac{d^{0}}{dx_k^{0}} \right] \Phi(x) \bigg|_{u=g(x)} = h(g(\langle x \rangle))
\]
for \( P_d = 0 \). Note that
\[
\frac{d^k}{du^k} h(u) \bigg|_{u=g(x)}
\]
are the same single dimensional higher-order derivatives of the outside function \( h(u) \) as described in detail in the previous section. \( D(k, n) \) are various combinations of the derivatives of the function \( g(x) \). These terms do not follow an easily definable pattern, so they must be generated by applying the chain rule and product rule until the term of interest is obtained.

For example, starting from \( f(x) = h(g(x)) \), apply the chain rule to get
\[
\frac{d}{dx_0} \Phi(x) \bigg|_{x=(x)} = \frac{dg(x)}{dx_0} \bigg|_{x=(x)} \frac{dh(u)}{du} \bigg|_{u=g(x)}
\]
which provides an element of \( D \)
\[
D(1, \{1, 0, \ldots, 0\}) = \frac{dg(x)}{dx_1} \bigg|_{x=(x)}
\]
Next apply the chain rule and product rule to (45) to obtain for example the next derivative
\[
\frac{d^2}{dx_0^2} \Phi(x) = \frac{d^2g(x)}{dx_0^2} \bigg|_{x=(x)} \frac{dh(u)}{du} \bigg|_{u=g(x)} + \left( \frac{dg(x)}{dx_0} \right)^2 \bigg|_{x=(x)} \frac{d^2h(u)}{du^2} \bigg|_{u=g(x)}
\]
which provides two more elements of \( D \)
\[
D(1, \{2, 0, \ldots, 0\}) = \frac{d^2g(x)}{dx_1^2} \bigg|_{x=(x)} \quad \text{(48)}
\]
\[
D(2, \{2, 0, \ldots, 0\}) = \left( \frac{dg(x)}{dx_1} \right)^2 \bigg|_{x=(x)} \quad \text{(49)}
\]

The same approach can be applied to (45) to obtain various other derivatives
\[
\frac{d}{dx_0} \frac{d}{dx_1} \Phi(x) = \frac{d}{dx_0} \frac{d}{dx_1} g(x) \bigg|_{x=(x)} \frac{dh(u)}{du} \bigg|_{u=g(x)}
\]
\[
= \frac{dg(x)}{dx_0} \bigg|_{x=(x)} \frac{dg(x)}{dx_1} \bigg|_{x=(x)} \frac{d^2h(u)}{du^2} \bigg|_{u=g(x)}
\]
which gives another two elements of \( D \)
\[
D(1, \{1, 1, 0, \ldots, 0\}) = \frac{d}{dx_0} \frac{d}{dx_1} g(x) \bigg|_{x=(x)} \quad \text{(51)}
\]
\[
D(2, \{1, 1, 0, \ldots, 0\}) = \frac{dg(x)}{dx_0} \bigg|_{x=(x)} \frac{dg(x)}{dx_1} \bigg|_{x=(x)} \quad \text{(52)}
\]
This approach can be applied further to obtain the required \( D(k, n) \) terms.

\section*{Appendix C \ Computational Complexity Estimates}

Since the SPAD methodology does add additional computational complexity as compared to the same problem done deterministically, it is prudent to understand the resulting costs. The additional complexity is best described by the number of operations required for the computation with uncertainty versus the number of operations for the same computation done deterministically. With respect to the single-variate case, addition with uncertainty (28) requires \( N \) operations for every deterministic addition, multiplication (29) requires \( 3N^2/2 \) operations for every deterministic multiplication, and all other operations (31) require \( N^2N^{-1} \) operations for every deterministic operation. In terms of memory usage, the algorithm with uncertainty requires \( N \) bytes for every deterministic byte used.

For the multivariate case, addition (40) requires \( N^d \) operations for every deterministic addition, multiplication (41) requires \( 3N^{2d}/2 \) operations for every deterministic multiplication, and all other operations (42) require \( N^dN^{-1} \) for every deterministic operation. In terms of storage, the algorithm with multivariate uncertainty requires \( N^d \) bytes for every deterministic byte. Again, \( N \) is the number of terms used in the expansions, and \( d \) is the number of random variables. For these conclusions, it has been assumed that the number of terms in each dimension are the same. At first glance, these costs appear to be severe. However, in practice it is possible to obtain good results with a small number of terms. For the results presented in this paper, an optimized fourth-order approach was used, which is discussed in the following appendix.

\section*{Appendix D \ An Optimized Fourth-Order Multivariate SPAD Implementation}

In order to develop an accurate implementation of this method while minimizing the computational overhead, a
fourth-order SPAD approach was applied to generate all the results in this paper.

Starting from (10) and keeping only derivatives up to the fourth-order, the expected value can be written as

\[
\langle \Phi(x) \rangle \approx \Phi(x) + \sum_{n=0}^{d} \sum_{m=n+1}^{d} \sum_{l+k \leq d} A_{lk}(x_n, m_l) \frac{d^l}{dx_n^l} \frac{d^k}{dx_m^k} \Phi(x) \bigg|_{x=(x)} 
\]

(53)

where \( d \) is the number of input random variables as described before, and \( A_{lk} \) (the central moments) are defined as

\[
A_{lk}(x_n, m_l) = \langle (x_n - \langle x_n \rangle)^l (m_l - \langle m_l \rangle)^k \rangle 
\]

(54)

As an example, the non-zero \( A_{lk} \) for a set of jointly Gaussian input random variables are

\[
A_{20}(x_n, 0) = C(x_n, x_n) 
\]

(55)

\[
A_{40}(x_n, 0) = 3C^2(x_n, x_n) 
\]

(56)

\[
A_{11}(x_n, x_m) = C(x_n, x_m) 
\]

(57)

\[
A_{13}(x_n, x_m) = 3C(x_n, x_n)C(x_n, x_m) 
\]

(58)

\[
A_{31}(x_m, x_n) = 3C(x_m, x_m)C(x_n, x_n) 
\]

(59)

\[
A_{22}(x_m, x_n) = 2C^2(x_m, x_n) + C(x_n, x_n)C(x_m, x_m) 
\]

(60)

Note that (53) is not completely fourth-order since terms involving more than two variables are intentionally excluded, e.g.

\[
\frac{d^2}{dx_n^2} \frac{d^1}{dx_m^1} \frac{d^1}{dx_l^1} \Phi(x) \bigg|_{x=(x)} \quad (n \neq m \neq l) 
\]

(61)

and

\[
\frac{d^1}{dx_n^1} \frac{d^1}{dx_m^1} \frac{d^1}{dx_k^1} \Phi(x) \bigg|_{x=(x)} \quad (n \neq m \neq l \neq k) 
\]

(62)

This is done to keep the scaling of the computational complexity manageable at \( O(d^2) \). The memory required for the (pseudo) fourth-order approximation above is \( 18d^2 \) versus \( N^d \) for the general multivariate case.

The complexity of the automatic-differentiation procedure is also much reduced. An example multiplication update equation is

\[
d^1 \frac{d^3}{dx_n^1 dx_m^1 dx_l^1} \Phi(x) \bigg|_{x=(x)} 
\]

\[
= \frac{d^0}{dx_n^0} \frac{d^0}{dx_m^0} \frac{d^0}{dx_l^0} g(x) \bigg|_{x=(x)} \frac{d^1}{dx_n^1} \frac{d^3}{dx_m^1 dx_l^1} h(x) \bigg|_{x=(x)} 
\]

\[
+ 3 \frac{d^0}{dx_n^0} \frac{d^1}{dx_m^1} \frac{d^2}{dx_l^2} g(x) \bigg|_{x=(x)} \frac{d^1}{dx_n^1} \frac{d^2}{dx_m^2} h(x) \bigg|_{x=(x)} 
\]

\[
+ 3 \frac{d^0}{dx_n^0} \frac{d^2}{dx_m^2} \frac{d^1}{dx_l^1} g(x) \bigg|_{x=(x)} \frac{d^1}{dx_n^1} \frac{d^2}{dx_m^2} h(x) \bigg|_{x=(x)} 
\]

\[
+ \frac{d^1}{dx_n^1} \frac{d^0}{dx_m^0} \frac{d^0}{dx_l^0} g(x) \bigg|_{x=(x)} \frac{d^1}{dx_n^1} \frac{d^2}{dx_m^2} h(x) \bigg|_{x=(x)} 
\]

\[
+ 3 \frac{d^1}{dx_n^1} \frac{d^1}{dx_m^1} \frac{d^0}{dx_l^0} g(x) \bigg|_{x=(x)} \frac{d^2}{dx_n^2} \frac{d^2}{dx_m^2} h(x) \bigg|_{x=(x)} 
\]

\[
+ 3 \frac{d^0}{dx_n^0} \frac{d^2}{dx_m^2} \frac{d^1}{dx_l^1} g(x) \bigg|_{x=(x)} \frac{d^2}{dx_n^2} \frac{d^2}{dx_m^2} h(x) \bigg|_{x=(x)} 
\]

\[
+ \frac{d^1}{dx_n^1} \frac{d^3}{dx_m^1 dx_l^1} \frac{d^0}{dx_k^0} g(x) \bigg|_{x=(x)} \frac{d^1}{dx_n^1} \frac{d^2}{dx_m^2} h(x) \bigg|_{x=(x)} 
\]

(64)

Other multiplication formulas can be derived in a rather straightforward manner using (41). An example of one of the chain rule update equations is

The other chain rule update equations can be derived using (42), but it was found to be a bit easier to manually start from the lowest order equations and derive the subsequent higher-order formulas by applying the chain and product rules as needed.
One final consideration is that the numerical complexity of (53) can be reduced to $O(d)$ by eliminating the third term in that equation. This approximation is only valid if correlations between input variables can be ignored (for example, if they are known to be uncorrelated).

REFERENCES


Michael S. Gilbert received B.S and M.S degrees in electrical engineering from the Ohio State University in 2002 and 2003 respectively. From 2003 to 2005, he was a research engineer at the Naval Surface Warfare Center in Dahlgren, Virginia. In 2005, he joined the Air Force Research Laboratory in Dayton, Ohio where he presently serves as a project engineer. His current interests include computational electromagnetics; in particular the uncertainties underlying such techniques.

Fernando L. Teixeira received the Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign in 1999. From 1999 to 2000, he was a Postdoctoral Associate with the Massachusetts Institute of Technology. In 2000, he joined the The Ohio State University, where he is now a Professor with the Department of Electrical and Computer Engineering and is also affiliated with the ElectroScience Laboratory. His current research interests include electromagnetic sensors for hydrocarbon exploration, timed-domain computational electromagnetics, metamaterials, and ultrawideband inverse scattering. Dr. Teixeira received the CAREER Award from the National Science Foundation, the triennial Booker Fellowship from the International Union of Radio Science (USNC/URSI), and the Outstanding Young Engineer Award from the IEEE Microwave Society (MTT-S). He has served as chair of the Joint IEEE AP/MTT-S Columbus Chapter and currently serves as an Associate Editor for the IEEE ANTENNAS AND WIRELESS PROPAGATION LETTERS.