Comput Geosci (2007) 11:283–296
DOI 10.1007/s10596-007-9052-z

ORIGINAL PAPER

Numerical tools for geoscience computations: Semiautomatic differentiation—SD

Abdulwahab A. Abokhodair

Received: 9 August 2006 / Accepted: 25 April 2007 / Published online: 5 June 2007

Abstract This paper presents a differentiation method [referred to here as semiautomatic differentiation (SD)] based on generalization and extension of the Squire and Trapp formula for complex differentiation of real-valued functions. The performance of the generalized formulas for first-order derivatives is tested and compared with manual, automatic (AD), and finite difference (FD) techniques. My results show that, in terms of accuracy, the SD technique is competitive with AD, and in terms of implementation simplicity, it is identical to the FD method with the added advantage of being step-size insensitive and, hence, free from the step-size dilemma that plagues FD. Using central differencing in the complex plane, I extend the SD method to second-order derivatives, thus enabling approximation of the Hessians. Performance of the extension formulas is evaluated and compared with AD and FD methods. The results indicate that the differencing operation reduces the accuracy of the extension formulas by four to five orders of magnitude below that of the original Squire and Trapp formula. Nonetheless, compared to FD schemes, the SD method is six to seven orders of magnitude more accurate in all tests conducted. In addition, the extension formulas exhibit step-size (h) insensitive behavior over the entire h-range of the tests ($1–10^{-30}$), indicating high numerical stability of the schemes. I show by examples that SD provides a complete differentiation system that is computationally stable, efficient, highly accurate, and easy to implement.

A. A. Abokhodair (✉)
Department of Earth Sciences,
King Fahd University of Petroleum & Minerals,
Dhahran 31261, Saudi Arabia
e-mail: akwahab@kfupm.edu.sa

Keywords Numerical derivatives · Complex differentiation · Optimization · Inversion · Sensitivity analysis

1 Introduction

Numerical differentiation plays a crucial role in computational geoscience. Problems such as curve fitting, parameter estimation, modeling of natural systems, and simulation of geoprocesses are essentially optimization problems that require inversion of linear and nonlinear model equations. Efficient numerical algorithms designed to solve such inverse problems require the user to supply first- and second-order derivative objects of the underlying model and associated constraint equations in the form of gradients, Jacobians and Hessians.

There are three different methods in common use for the computation of derivatives: manual differentiation (MD); finite difference approximation (FD); and, more recently, automatic differentiation (AD). Key issues in choosing a differentiation technique are its accuracy, computational expense, and ease of implementation.

The MD method, though most accurate, is seldom used in actual practice. It is characteristically time consuming and error prone regardless of whether differentiation is performed by hand or by means of a symbolic manipulation system. Furthermore, the method in some cases may not be applicable at all, particularly when the equations of interest are implicit and do not lend themselves easily to explicit closed-form differentiation [5].

The emerging technology of AD has unsurpassed accuracy, limited only by machine precision. Theoretically, AD produces exact results in infinite precision arithmetic. Several AD software packages have been introduced in

different languages, including C/C++, Fortran 77/90, and Matlab®, and with different features and limitations (for more details, visit http://www.autodiff.org). Most of these packages, however, are experimental "Beta" releases that have yet to reach development maturity. Consequently, there is little published information at present on the actual computational cost of AD in terms of CPU time and memory requirements. Perhaps this is one reason why many researchers are hesitant at present to incorporate AD tools in optimization software packages [13].

Despite its universal popularity, the classical FD method is neither efficient nor accurate regardless of the particular FD formulas employed. Ease of implementation is its main attractive feature that accounts for its widespread use. It is well known that FD-based Jacobians are often ill-conditioned [6]. The method suffers from the so-called step-size dilemma, which seriously limits its accuracy. This refers to the delicate balance between the size of the differencing interval and the magnitudes of the truncation and round-off errors. Choosing a small step size to minimize truncation error could cause round-off errors and subtractive cancellation to dominate, thus degrading the quality of the derivatives [10].

An alternative to FD with superior qualities is provided by a technique little known among geoscientists based on the theory of complex variables. Exploiting Cauchy integral theorem, Lyness [8] and Lyness and Moler [9] introduced several formulas for the computation of the nth derivative of a complex analytic function from its contour integral using numerical quadratures. These ideas were subsequently used by Squire and Trapp [17] to develop a simple expression for approximating first-order derivative of real-valued functions. These authors have shown that this complex differentiation method is robust, highly accurate, and easy to implement.

The primary objective of this paper is to introduce a generalized version of the Squire–Trapp differentiation method with extensions to second-order derivatives. The generalized formula together with its extensions form a differentiation system [hereafter referred to as semiautomatic differentiation (SD)] that allows approximation of derivatives including gradients, Jacobians, Hessians, and 2D and 3D partial and cross-partial spatial derivatives. It is shown that, in terms of performance, the proposed system is competitive with AD and superior to ordinary real-domain FD with the added advantage of being step-size insensitive. Moreover, it is as easy to implement as FD by means of simple and maintainable code.

The plan of the paper is as follows: the next section presents a heuristic example that illustrates the inner-workings of complex differentiation of real-valued functions; a brief review of mathematical basis of the SD method is also presented. Section 3 focuses on derivation and performance evaluation of the generalized formulas for first-order derivatives. Extension to second-order derivatives and performance evaluation of the extension formulas are presented in Section 4. Implementation of the approximation schemes for the gradient, Jacobian, Hessian, and 2D and 3D spatial differentiation is illustrated in Section 5 by examples of different geophysical computational problems. Section 6 concludes the paper and summarizes its findings. Throughout the paper, short Matlab® scripts and lines of code will be intermingled with the discussion to illustrate relevant points.

2 Mathematical background

2.1 A heuristic example

To illustrate how SD works, we consider the function $f(x) = \sin(\omega/x)$ and estimate its partial derivative with respect to x, i.e., $f_x = \partial f/\partial x$, as follows:

> Let $f(x) = \sin(\omega/x) = \sin(p)$ where $p = \omega/x$
> Then the analytical partial derivative is $f_x = p_x \cos(p)$
> where $p_x = -\omega/x^2$.

To approximate a given partial derivative using SD, the basic rule is to perturb the corresponding variable by a pure imaginary step ih, where $i = \sqrt{-1}$ and h is a small step size (e.g., $h=10^{-16}$ ~ eps). Thus, to compute f_x, we convert the real argument x into a complex argument: $z=x+ih$, so that the complex version of the original real-valued function becomes:

$$f(z) = \sin(\omega/z) = \sin(q), (q = \omega/z)$$
$$q = \frac{\omega}{z} = \frac{1}{|z|^2}(\omega x - i\omega h)$$

But for small $h \ll 1$, we may write:

$$|z|^2 = x^2 + h^2 \approx x^2 \Rightarrow q \approx (\omega/x) + ih\left(-\omega/x^2\right) = p + ihp_x,$$

Therefore,

$$\begin{aligned} f(z) &= \sin(\omega/z) = \sin(q) \approx \sin(p + ihp_x) \\ &= \sin(p)\cosh(hp_x) + i\ \cos(p)\sinh(hp_x) \\ &\approx \sin(p) + ih[p_x \cos(p)], h \ll 1; \\ &= u(x) + iv(x). \end{aligned}$$

This is Squire–Trapp formula in action. The real part of $f(z)$ is $O(h^2)$ approximation of $f(x)$and the imaginary part is $O(h^2)$ approximation of its partial derivative f_x. Of course, our heuristic approach here does not show the remainder terms of the different expansions which would indicate the $O(h^2)$-order of approximation.

It is important to note here that the derivative approximation f_x involves no subtractive cancellation, and hence,

h may take up extremely small values. Furthermore, from a single function evaluation in the complex plane, we obtain $O(h^2)$ approximations of both the function value and its partial derivative. This is the essence of the power of the SD differentiation method.

The partial derivative $f_\omega = \partial f/\partial \omega$ may be obtained by following the same steps above. The following Matlab script implements these ideas for the example above.

```
% Evaluate the function f(x)= sin(w/x) and its exact derivative
%          df/dx = f_x
    X =(-1:.005:1)'; w=2*pi;
  tf  = sin(w./x);                  % Exact function value
  tfx = (-w./x.^2).*cos(w./x);      % Exact derivative df/dx

% The partial wrt x
% Convert real x to complex z and evaluate f(z).
     h  = 1.0E-25;        % Very small step size
     z  = complex(x,h);   % z = x + ih
     fc = sin(w./z);      % Evaluate complex extension f(z)

% Extract approximations of f(x)=ef & f_x(x)=efx
     ef  = real(fz);      % Estimated f(x)
     efx = imag(fz)/h;    % Estimated derivative df/dx

% Plot and compare analytical and estimated results.
     figure;
     subplot(1,2,1),plot(x,tf,'--',x,ef,'ro'),grid
     subplot(1,2,2),plot(x,tfx,'--',x,efx,'ro'),grid
```

The results are shown in Fig. 1. Note the close agreement between the exact (analytical) and SD estimates of the derivative despite the very small step size used ($h=10^{-25}$), which indicates the stability of the SD scheme.

2.2 Informal derivation

A derivation of the Squire–Trapp formula based on the Cauchy–Riemann conditions for analytic functions of complex variables is given by Martins et al. [11]. A simpler approach, however, is via Taylor series expansion of $f(z)$. Under the assumption of analyticity, $f(z)$ may be expanded in the neighborhood of x as:

$$\left.\begin{aligned} f(z) = f(x+ih) &\cong f(x) + ihf'(x) - \frac{h^2}{2}f''(x) \frac{ih^3}{3!}f'''(x) + O(h^4) \\ &= \mathrm{Re}[f(z)] + i\mathrm{Im}[f(z)] \end{aligned}\right\}$$

where

$$\left.Re\,[f(z)] \cong f(x) - \frac{h^2}{2}f''(x) Im\,[f(z)] \cong h\left[f'(x) - \frac{h^2}{3!}f'''(x)\right]\right\}$$

Therefore,

$$\left.f(x) \cong Re\,[f(z)] + \frac{h^2}{2}f''(x)f'(x) \cong \frac{Im\,[f(z)]}{h} + \frac{h^2}{6}f'''(x)\right\} \tag{1}$$

Equation 1 shows the original Squire and Trapp [17] formulas. They clearly show that the approximation of the derivative $f'(x)$ is $O(h^2)$, an accuracy equivalent to that of ordinary central finite difference (CFD) on the real axis. Unlike CFD approximation, the approximation in Eq. 1 is free from any subtractive cancellation, a significant advantage over any FD formulation. Moreover, the real part of $f(z)$ also provides an $O(h^2)$ approximation of the function value $f(x)$, an added advantage of the SD method.

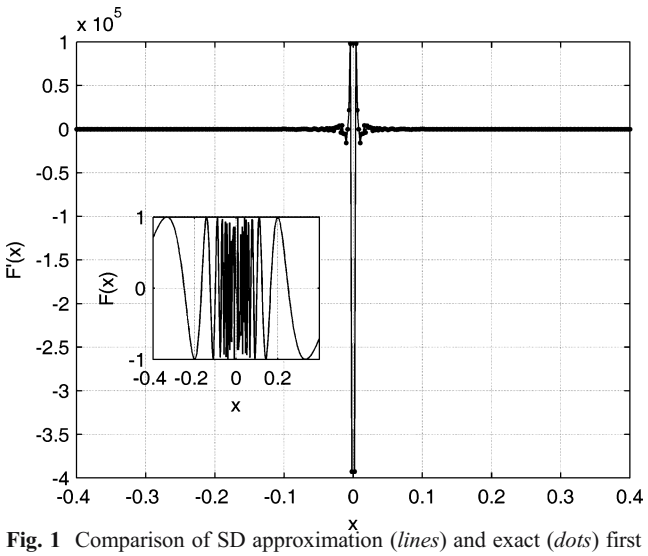

Fig. 1 Comparison of SD approximation (*lines*) and exact (*dots*) first derivative of the function $f(x) = \sin(\omega/x)$. The function is shown in the inset frame

3 Generalization

Equation 1 may be generalized to vector-valued functions $\mathbf{f}(\mathbf{p})$ by defining a complex argument vector: $\mathbf{z}=\mathbf{p}+ih\mathbf{e}$, where $\mathbf{e}$ is an arbitrary unit m-vector and h is a small step size ($h \ll 1$). Expanding $\mathbf{f}(\mathbf{z})=\mathbf{f}(\mathbf{p}+ih\mathbf{e})$ in the complex neighborhood of $\mathbf{p}$, we arrive at:

$$\begin{aligned} \mathbf{f}(\mathbf{p}) &= \mathrm{Re}[\mathbf{f}(\mathbf{z})] + h^2 R \\ \mathbf{e}^T\mathbf{f}'(\mathbf{p}) &= \tfrac{1}{h}\mathrm{Im}[\mathbf{f}(\mathbf{z})] + h^2 Q \end{aligned} \tag{2}$$

where R and Q are remainder terms and $\mathbf{f}'(\mathbf{p})=\nabla\mathbf{f}(\mathbf{p})$ is the gradient (see Appendix A for details).

Note that the imaginary part of Eq. 2 is an approximation of the directional derivative $\mathbf{e}^T\nabla\mathbf{f}(\mathbf{p})$ along $\mathbf{e}$, showing that the original Squire–Trapp formula in Eq. 1 is a special case corresponding to $\mathbf{e}=\mathbf{e}_k$, the kth standard basis vector (with one at the kth element and zero elsewhere).

3.1 Performance evaluation

The performance of the SD method for first-order derivatives was evaluated by comparison with real-domain

forward (FFD) and CFD formulas. The following analytic function was used for the evaluation test:

$$f(x) = e^{-\alpha x \sin(\pi x)} \tag{3}$$

For a performance index, we compute the absolute relative error of the different schemes as function of step size h in the range $[1 \geq h \geq 10^{-30}]$ according to:

$$E_r(h) = \frac{\left|f'_{\text{Est}}(h) - f'_{\text{Exact}}\right|}{\left|f'_{\text{Exact}}\right|} \tag{4}$$

where f'_{Exact} is the exact (analytical) derivative and $f'_{\text{Est}}(h)$ is its approximate value. The upper bound on the truncation error of $O(h^2)$ approximations was also computed from:

$$\tau(h) \leq \frac{h^2}{6}\left|\max\left[f'''(h)\right]\right| \tag{5}$$

Figure 2 shows the absolute relative error curves $E_r(h)$ where we see that the FD formulas define two sharp optimum values (h_{opt}=10^{-5} and 10^{-8} for CFD and FFD, respectively) at which they attain maximum accuracy (10^{-12} for CFD and 10^{-8} for FFD). Below these optimum h values, the quality of the FD approximations degrades gradually until it completely fails at $h \approx 10^{-16}$. In contrast, the SD formula converges quadratically to machine precision ($\simeq 10^{-16}$) at h$\approx 10^{-7}$, beyond which point, the scheme becomes insensitive to h. The upper bound on the truncation error $\tau(h)$ (Eq. 5) of the SD formula decrease to zero very rapidly, as may be seen in the inset of Fig. 2.

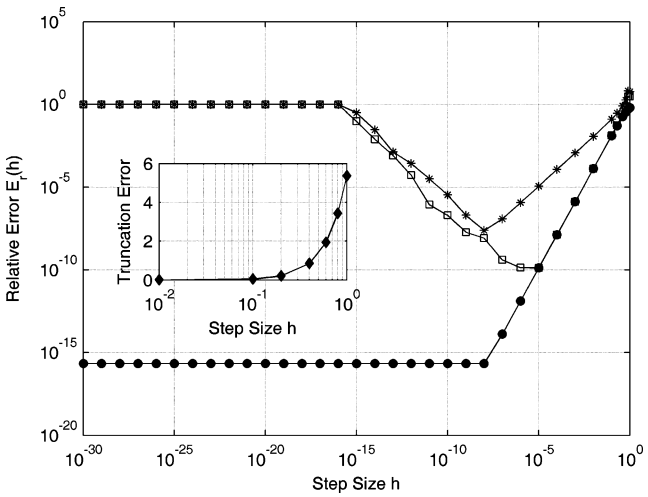

Fig. 2 Comparison of relative errors $E_r = \left|f'_{\text{est}} - f'_{\text{exact}}\right| / \left|f'_{\text{exact}}\right|$ of the SD (*circles*), CFD (*squares*), and FFD (*asterisks*) derivatives. FD formulas define an optimum h_{opt} h where they attain maximum accuracy. In contrast, the SD formula converges to minimum error value quadratically and maintains this performance throughout the h range. The truncation error in SD is measured as $\tau(h) \leq h^2\left|f'''(x_o)\right|/6$

3.2 Quality assessment

The quality of results of the SD scheme is assessed by comparison with AD and CFD. The AD package used for this comparison is ADMAT, an AD toolbox written for Matlab® by Coleman and Verma [4]. The object of comparison is the Jacobian of the model function given in Eq. 6, which is the familiar thin-plate approximation of the gravity anomaly due to a step-fault (e.g., Parasnis [15]).

$$\left.\begin{aligned}\mathbf{f}(\mathbf{p},\mathbf{x}) &= \text{At}\left[\tfrac{\pi}{2} + \tan^{-1}\left(\tfrac{x-x_0}{z_o+t}\right)\right] \\ \mathbf{p} &= [t \quad x_o \quad z_o]\end{aligned}\right\} \tag{6}$$

The Jacobian of $\mathbf{f}(\mathbf{p},\mathbf{x})$ was computed using AD and was approximated using SD (see Matlab function *cJac.m* in Appendix B) and ordinary central difference (CFD) schemes. The relative absolute residuals of the SD and CFD estimates with reference to AD were computed element-wise according to:

$$\varepsilon_r = \frac{|\mathbf{J}_{\text{AD}} - \mathbf{J}_{\text{Est}}|}{(1 + |\mathbf{J}_{\text{AD}}|)} \tag{7}$$

where $\mathbf{J}_{AD}$ is the AD-Jacobian and $\mathbf{J}_{Est}$ is its estimate.

The computed Jacobians and their residuals are shown in Figs. 3 and 4, respectively. The similarities of the Jacobians displayed in Fig. 3 are shown by Fig. 4 to be quite deceptive. The SD residuals (left panel) amount to little more than numerical noise, lying within the range of machine precision ($\approx 10^{-16}$). In contrast, the CFD residuals (right panel) are six to eight orders of magnitude greater and show distinct trends that appear to correlate with the corresponding Jacobian elements. These numerical errors may contribute to the ill-conditioning of FD-based Jacobian. In a typical optimization cycle, the Jacobians are repeatedly computed a large number of times, and residuals of the size indicated in Fig. 4 accumulate and amplify rapidly as computation progresses, leading ultimately to destabilization and divergence of the optimization algorithm.

4 Extension to second-order derivatives

The generalized Squire–Trapp formulas in Eq. 2 provide approximation schemes for first-order derivatives only. These formulas, therefore, are useful for computing gradients and Jacobians but not for higher-order partial and cross-partial derivatives. An extension of these formulas to second-order derivatives is accomplished by computing a FD approximation of $\mathbf{f}''(\mathbf{z})$ in the complex plane. As

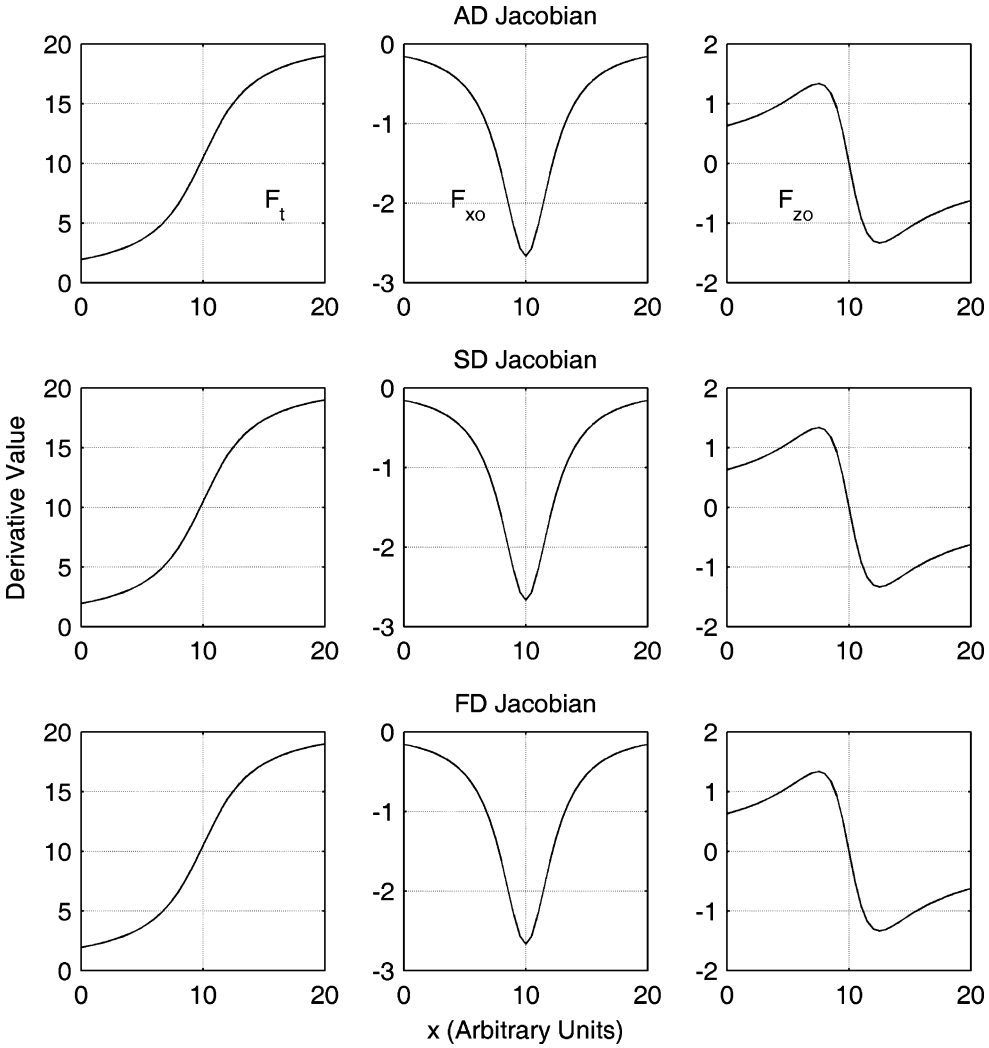

Fig. 3 The Jacobian **J** of the vector-valued function in Eq. 6 with respect to the three-parameter vector $\mathbf{p}=[t, x_o, z_o]$ computed using ADMAT (*top row*) and approximated using SD (*middle row*) and CFD (*lower row*) formulas. Note the apparent similarities of the three approximations

shown in Appendix A, this approach leads to the following centered difference (CSD) approximations formula:

$$\left.\begin{aligned}\mathbf{e}^T\mathbf{f}'(\mathbf{p}) &= \frac{1}{2\delta p}\mathrm{Re}\left[\Delta\mathbf{f}(\mathbf{z}) + h^2\Delta R\right]\\ \mathbf{e}^T\mathbf{f}''(\mathbf{p})\mathbf{e} &= \frac{1}{2h\delta p}\mathrm{Im}\left[\Delta\mathbf{f}(\mathbf{z}) + h^2\Delta Q\right]\end{aligned}\right\} \tag{8}$$

where ΔR and ΔQ are remainder terms and $\mathbf{f}''(\mathbf{p})=\mathbf{H}$ is the Hessian.

Equation 8 provides $O(h^2)$ approximations of the gradient vector $\mathbf{f}'(\mathbf{p})$ in the real part of $\Delta\mathbf{f}(\mathbf{z})$ and of the Hessian $\mathbf{f}''(\mathbf{p})$ in the imaginary part. It must be stressed that, unlike the generalized Squire–Trapp formulas of Eq. 2, the formulas in Eq. 8 are subject to subtractive cancellation because they are based on differencing operations. However, because complex calculations are performed separately for the real and imaginary parts, one may expect that the effects of rounding and cancellation errors are substantially reduced, thus leading to greater accuracy in derivatives approximation than in ordinary FD calculations [3, 12]. This may explain the superior performance of the SD second-order schemes compared to ordinary real-domain FD formalism as we show next.

4.1 Performance evaluation

Performance of the second derivative approximation formulas in Eq. 8 was assessed in the same way as in Section 3.1. The exact derivative f_{Exact} of the analytic test function of Eq. 3 was computed near a minimum point at [x=2.675, α=−0.25] and compared with derivative estimates f_{Est} from real-domain forward and CFD and complex-domain forward and CFD. The relative absolute error $E_r(h)$ for each case was computed from Eq. 4. The results of these tests are summarized in Fig. 5.

The left column of the figure compares the "effective" h range (i.e., range of h values within which accurate derivative estimates are obtained) for the forward (a) and centered (c) difference formulas. Ordinary real-domain FD

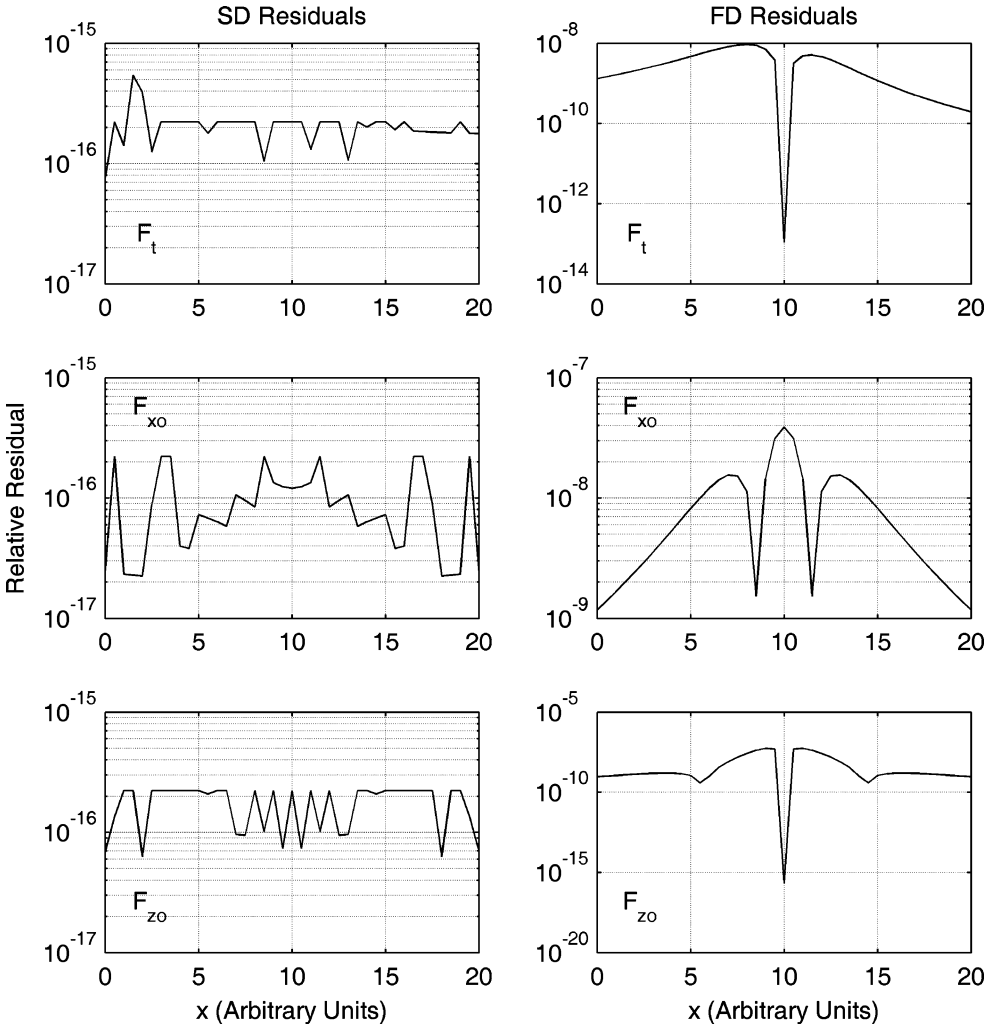

Fig. 4 Relative residuals $\varepsilon_r(h)$ (Eq. 7) of the three components of **J** referred to AD. The SD residuals (*left panels*) of all three components are random-like fluctuations lying within machine precision ($\approx 10^{-16}$), whereas the CFD residuals (*right panels*) are five to six orders of magnitude greater lying in the range [10^{-7}–10^{-13}] and show distinct systematic patterns

schemes (triangles) define a relatively narrow h-range (10^{-1}–10^{-7}), whereas the SD schemes (closed circles) display h-insensitive behavior yielding estimates that coincide with the exact derivative value (open circles) throughout the h range of the test (1–10^{-30}).

The right column compares the relative error curves $E_r(h)$ of the FD (triangles) and SD (filled circles) forward (b) and centered (d) difference formulas. The ordinary FD formulas are seen to attain maximum accuracy only at a single optimum h_{opt} value, beyond which their accuracy deteriorates rapidly before they completely fail near $h \simeq 10^{-17}$. In contrast, the SD formulas converge to their minimum error estimates faster (quadratically) and attain high-precision performance throughout the h range of the test in a manner insensitive to h. Hence, compared to real-domain FD schemes, the complex-domain SD formalism is significantly more stable and several orders of magnitude more accurate with the added advantage of being step-size-independent.

Moreover, Fig. 5 shows that the centered difference SD formula for second-order derivatives (d) performs better than its forward difference counterpart (b) with minimum relative errors nearly six orders of magnitude smaller. Finally, in comparison with the Squire–Trapp original formula for first derivatives, the accuracy [$E_r(h) \simeq 10^{-12}$] of the SD formulas for second-order derivatives (b and d) is distinctly lower by at least five orders of magnitude [$E_r(h) \simeq 10^{-16}$]. This reduction in accuracy is the price paid for the differencing operations involved in the SD schemes for second-order formulas.

4.2 Quality assessment

Quality assessment of the SD approximations in Eq. 8 is made by comparison with AD and FD. The object in this comparison is the Hessian of the following test function:

$$\left.\begin{aligned} \mathbf{F}(\mathbf{x}) &= z^2 \exp\left(-x^2 - y^2\right) \\ \mathbf{x} &= (x, y, z) \end{aligned}\right\} \tag{9}$$

The Matlab script that implements Eq. 8 for both the gradient (**g**) and Hessian (**H**) is *cHg.m* listed in Appendix B. The Hessian of the test function was computed at **p**=(0.5, 0.25, 3.5) using ADMAT toolbox

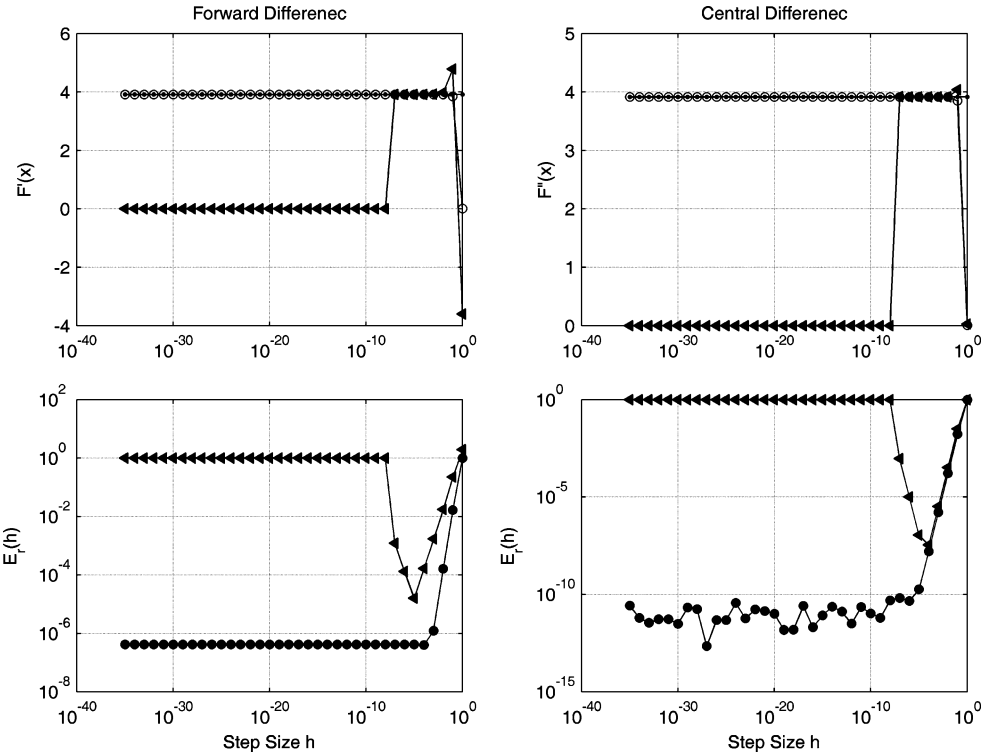

Fig. 5 Performance tests of the second-order approximation using FD and SD forward and centered difference formulas. Note that both FD (*triangles*) formulas define a narrow h range, outside of which they completely fail. In contrast, both SD (*closed circles*) schemes display h-insensitive behavior converging quadratically to a higher-accuracy estimate close to the exact derivatives (*open circles*) and maintain this performance throughout the h range of the test

($\mathbf{H}_{AD}$) and was approximated using the CSD ($\mathbf{H}_{SD}$) formulas of Eq. 8 and the CFD ($\mathbf{H}_{FD}$) central difference formula in Eq. 10. The results are summarized in Table 1, where elements of $\mathbf{H}$ are listed to the first 16 decimal digits.

$$\left.\begin{aligned}\frac{\partial^2 \mathbf{f}}{\partial p_i \partial p_j} &= \frac{1}{4h^2}\left[\mathbf{f}\left(p_i^+, p_j^+\right) - \mathbf{f}\left(p_i^+, p_j^-\right) - \mathbf{f}\left(p_i^-, p_j^+\right) + \mathbf{f}\left(p_i^-, p_j^-\right)\right] \\ p_k^{\pm} &= p_k \pm h\mathbf{e}_k,\ k = i,j\end{aligned}\right\} \tag{10}$$

Table 1 shows that the SD ($\mathbf{H}_{SD}$) and AD ($\mathbf{H}_{AD}$) results agree to within the first 8 to 10 decimal digits, with relative error of the order of $\delta\mathbf{H}_{SD} \simeq 10^{-12}$. The agreement for real-domain CFD Hessian, on the other hand, is limited to the first four to five decimal digits with significantly greater relative errors of the order of $\delta\mathbf{H}_{FD} \simeq 10^{-16}$.

The size of $\delta\mathbf{H}$, in fact, provides a meaningful measure of the quality of an approximation of $\mathbf{H}$. To explain, suppose that the Hessian matrix $\mathbf{H}$ is entered into a computation of the type: $\mathbf{y}=\mathbf{H}\mathbf{x}$, so that $\mathbf{x}=\mathbf{H}^{-1}\mathbf{y}$. Suppose further that $\mathbf{H}$ is in error by an amount $\delta\mathbf{H}$. Then, as shown by Forsythe and Moler [7], the relative errors introduced in $\mathbf{x}$ due to errors $\delta\mathbf{H}$ are bounded by q such that for any norm $\|\cdot\|$:

$$q = \frac{\|\delta H\|}{\|H\|}\kappa \geq \frac{\|\delta x\|}{\|x\|}, \tag{11}$$

where κ is the condition number of $\mathbf{H}$.

For the test case considered here, q was computed for all four commonly used norms: one, Euclidian, infinity and Frobenius (L_1, L_2, L_{inf}, and L_{fro}). The upper bounds for the CSD and CFD approximations were found to be $q \leq 1.1416\times10^{-12}$ and $q \leq 1.540\times10^{-6}$, respectively; and are largest for the L_1 norm and smallest for the L_{fro} norm (1.0146×10^{-10} and 1.2546×10^{-6}). These results clearly indicate that, in addition to its stability, the SD approximation of second-order derivatives in Eq. 8 is of higher accuracy than ordinary FD approximation by several orders of magnitude.

5 Application examples

We present in this section two examples of derivative-based geophysical applications of the SD differentiation technique. The functions to be differentiated will be expressed in terms of two arguments as $\mathbf{f}(\mathbf{p},\mathbf{x})$, where $\mathbf{p}$ is an m vector of physical parameters and $\mathbf{x}$ stands for spatial and time variables (vector or matrix). In some applications such as modeling, simulation, and inversion, the required derivative objects are those of $\mathbf{f}$ with respect to the vector of physical parameters $\mathbf{p}$. In other applications (e.g., geophysical imaging and image enhancement), the derivatives of interest are spatial derivatives of different orders. Examples of both types of applications are presented.

5.1 Data inversion

The first example illustrates use of SD derivative approximations in geophysical data inversion where the required

Table 1 Hessian matrices of the test function in Eq. 9 computed to 16 decimal digits using ADMAT AD ($\mathbf{H}_{AD}$) and approximated using the SD ($\mathbf{H}_{SD}$) formula of Eq. 8 and CFD ($\mathbf{H}_{FD}$) formula of Eq. 10

AD Hessian: $\mathbf{H}_{AD}$

-8.9622914545963610	4.4811457272981805	-5.1213094026264923
4.4811457272981805	-15.6840100455436320	-2.5606547013132461
-5.1213094026264923	-2.5606547013132461	1.4632312578932836

SD Hessian: $\mathbf{H}_{SD}$

-8.962291454*5439034*	4.481145726*9964450*	-5.121309402*4128454*
4.481145727*2719517*	-15.684010045*2222430*	-2.560654701*2064227*
-5.121309402*5046804*	-2.560654701*2982581*	1.463231257*8715996*

CFD Hessian: $\mathbf{H}_{FD}$

-8.9622*886889628699*	4.48114*43444814349*	-5.1213*122631565957*
4.48114*43444814349*	-15.6840*052056850240*	-2.56065*39111322486*
-5.1213*122631565957*	-2.56065*39111322486*	1.46323*39784270696*

SD Relative Error Matrix: $\Delta\mathbf{H}_{SD}$ x 10^{-12}

0.5265615251612231	5.5049717790105133	3.4902153103892508
0.4785276330372875	1.9263302805980638	3.0001066642037637
1.9899646610374981	0.4209341284037375	0.8803066244419301

CFD Relative Error Matrix: $\Delta\mathbf{H}_{FD}$ x 10^{-6}

2.7656334911086	1.3828167455543	2.8605301034546
1.3828167455543	4.8398586081078	0.7901809975230
2.8605301034546	0.7901809975230	2.7205337860270

The underlined digits in the SD and CFD Hessians are where disagreement with AD occurs. The relative absolute error matrices $\Delta\mathbf{H}_{SD}$ and $\Delta\mathbf{H}_{FD}$ were computed term-wise according to $\nabla\mathbf{H} = |\mathbf{H}_{\text{Est}} - \mathbf{H}_{\text{AD}}|/(1 + |\mathbf{H}_{\text{AD}}|)$

derivatives are of $\mathbf{f}(\mathbf{p},\mathbf{x})$ with respect to $\mathbf{p}$. The problem selected is gravity data inversion of a faulted structure. The standard equation of the gravity anomaly $\Delta g(\mathbf{p},\mathbf{x})$ of a 2D fault model (e.g., Telford et al. [18] and Parasnis [15]) is given in the Appendix A. The unknown geophysical parameters $\mathbf{p}$ to be estimated by inversion are:

$\mathbf{p}$ $[\Delta\rho, x_o, z_o, t, d]$
$\Delta\rho$ Density contrast
x_o Position of fault tract (km)
z_o Depth to top of faulted bed (km)
t Thickness of faulted bed (km)
d Dip of fault plane

The inverted data were simulated from the forward model equation by addition of normal random errors with zero mean and unit standard deviation.

I used three different solvers based on different methods from the Matlab toolbox *immoptibox* published online by Nielsen [14]. The first algorithm (*marquardt*) is based on Levenberg–Marquardt damping of the Gauss–Newton method and requires estimate of the Jacobian. The second

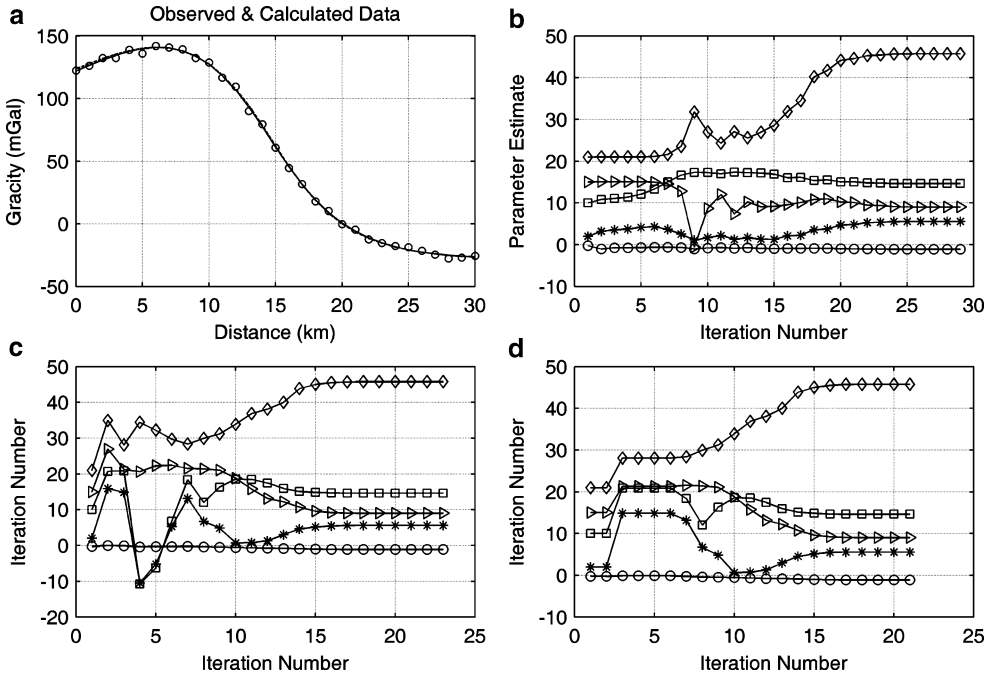

Fig. 6 Inversion of gravity data of a faulted structure (**a**) using three different algorithms: *hybrid* (**b**), *marquardt* (**c**), and *dampnwton* (**d**). The inversion parameters are: density contrast (*circles*), fault dip (*diamonds*), trace position (*squares*), depth to surface (*asterisks*), and thickness (*triangles*)

(*nlshybrid*) is a hybrid between the Gauss–Newton algorithm and a quasi-Newton algorithm; it is based on the Broyden–Fletcher–Goldfarb–Shanno algorithm updating of an approximation to the Hessian computed from the Jacobian. The Jacobian matrix required by these two algorithms is provided by the SD-based function *cJac.m* (Appendix B).

The third algorithm (*dampnewton*) is a variant of Newton method with a Levenberg–Marquardt-type damping, with iteration steps $\Delta\mathbf{p}$ determined by solution of the damped linear system: $\left[\mathbf{f}''(\mathbf{p}) + \lambda I\right]\Delta\mathbf{p} = -\mathbf{f}'(\mathbf{p})$, where λ is the damping factor, $\mathbf{f}''(\mathbf{p})$ is the Hessian matrix (**H**), and $\mathbf{f}'(\mathbf{p})$ is the gradient vector (**g**). Updated versions of **H** and **g** required by this algorithm are computed by function *cHg.m* of Appendix B.

The inversion results are summarized in Fig. 6 and Table 2. All three algorithms converged to nearly the same numerical estimates of **p**, but the Hessian-based algorithm (*dampnewton*) converged faster (21 iterations) than the other two algorithms that use the Jacobian (*marquardt*, 23 iterations) and Jacobian-based estimate of the Hessian (*nlshybrid*, 26 iterations). It is worth noting that the routine *dampnewton.m* internally checks the accuracy of the user's implementation of **g** and **H**; it terminates with error flags if either derivative object fails to pass the preset accuracy level or if **H** exhibits any nonsymmetry.

Table 2 Numerical estimates of the fault parameters by the different inversion algorithms

p	$\mathbf{p}_{True}$	$\mathbf{p}_o$	Parameters estimates		
			dampnewton	*marquardt*	*nlshybrid*
$\Delta\rho$	−1.0	−0.25	−1.0279	−1.0279	−1.0279
x_o	15.0	10.0	14.9826	14.9826	14.9826
z_o	5.0	2.0	5.1041	5.1041	5.1041
t	10.0	15.0	9.8419	9.8419	9.8419
i	45.0	21.0	45.4683	45.4683	45.4683
Number of iterations					
			21	23	29

$\mathbf{p}_{True}$ is the true parameter value and $\mathbf{p}_o$ is the initial iteration value

5.2 Spatial differentiation

Spatial differentiation of functions finds extensive use in geophysical research especially in the area of subsurface

```
function out = GPDiff();
% Returns Spatial Grad, Laplacian and Hessian of the Mexican-Hat
%         function
% Date: 3/8/2006
%
[x,y]=meshgrid(-3:.1:3);
p=[0;0];
xyz{1}=x; xyz{2}=y;
tf=tf2d(p,xyz);

% The Gradient & Laplacian

[cf,sG] = cSGrad('tf2d1',p,xyz);
[cf,sD] = cSDive('tf2d1',p,xyz);
[cf,sH] = cSHess('tf2d1',p,xyz);

% Plot
figure;                 % Spatial Gradient
for ip=1:2;
    subplot(2,2,ip), meshc(x,y,sG{ip});
end

for ip=1:2;             % Spatial Laplacian
    subplot(2,2,ip+2), meshc(x,y,sD{ip});
end

figure,                 % Spatial Hessian
for ip=1:4;
    subplot(2,2,ip), meshc(x,y,sH{ip});
end

out.xyz=xyz; out.tf=tf; out.G=sG;
out.D=sD; out.H=sH;
```

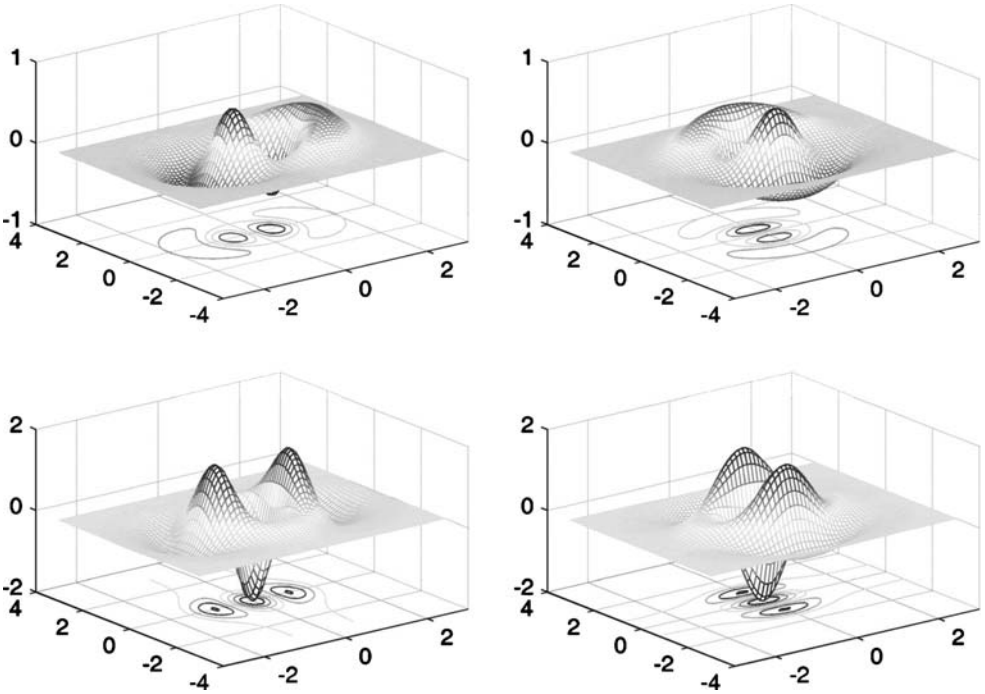

Fig. 7 Spatial derivatives of the function $\mathbf{F}(\mathbf{r}) = -\mathbf{r}^2 e^{-\mathbf{r}^2}$, where $\mathbf{r}$ is the 2D radius ector. The gradient $\mathbf{g}=\nabla\mathbf{F}(\mathbf{r})$ (*top row*) and the divergence $\mathbf{D} = \nabla^2\mathbf{F}(\mathbf{r}) = \mathbf{F}_{xx} + \mathbf{F}_{yy}$ (*bottom row*)

imaging and image enhancement studies. A current focus of research in potential field interpretation theory, for instance, is the use of derivatives of different orders to image subsurface magnetic and gravity source geometry employing techniques, such as analytic signal, horizontal gradients, and Euler deconvolution (e.g., Blakely [1]). The following Matlab script illustrates the ease of SD implementation for computing spatial derivative matrices of different orders of multidimensional functions. The function used is given in Eq. 12, which is 3D in the spatial coordinates *x*,*y*,*z*. The derivatives computed (Fig. 7) are the spatial gradient, the Laplacian, and the Hessian (Fig. 8) with cross partial derivatives.

$$\mathbf{F}(\mathbf{r}) = -\mathbf{r}^2 e^{-\mathbf{r}^2}, \mathbf{r}^2 = \mathbf{x}^2 + \mathbf{y}^2 + \mathbf{z}^2, \tag{12}$$

The three functions (*cSGrad.m*, *cSDive.m*, *cSHess.m*) called by the Matlab script above are slightly modified versions of *cJac.m* and *cHess.m*, which are listed in Appendix B.

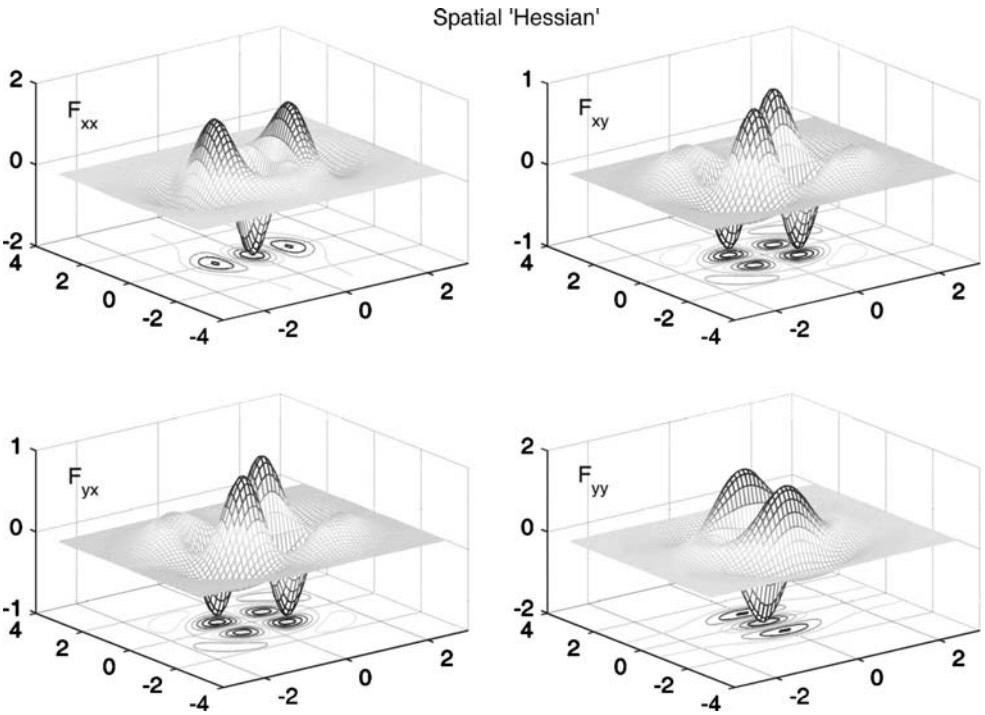

Fig. 8 Spatial "Hessian" of a function. Note that the Hessian includes **D** on its main diagonal and the cross-partial derivatives $\mathbf{F}_{xy}$ and $\mathbf{F}_{yx}$ on the opposite diagonal

6 Summary and conclusions

Derivative approximation is a costly task in any computational exercise. Therefore, development of efficient and accurate differentiation techniques is important. Three different differentiation methods are currently available to researchers: MD (or analytical), AD, and FD. Preference among these techniques is based on the criteria of accuracy, computational expense, and ease of implementation. In actual practice, the MD method is seldom used, being error-prone and highly demanding in terms of labor and computational time. The AD technique, on the other hand, is uncontested in accuracy, yielding exact results in infinite precision arithmetic. However, implementation of this technology is presently in a developmental stage and there is little published information on its real computational cost. Moreover, AD is not competitive with other methods in terms of implementation simplicity; it requires specialized and often bulky software packages not easily accessible to most users. Finally, the classical FD method, though universally popular, is neither efficient nor accurate. It suffers from the problem of step-size dilemma and often leads to ill-conditioned Jacobians and Hessians. Ease of implementation is its only attractive feature, which accounts for its widespread use.

In this paper, an alternative to the FD differentiation method is presented with qualities comparable to AD and with implementation simplicity identical to FD. The suggested method, referred to as SD, is an extension of the Squire–Trapp formula for the complex differentiation of real-valued functions. The original formula for first-order derivatives was generalized and its performance tested and compared with MD, AD, and FD techniques. The results presented confirm previous findings by Martins et al. [12] and by Pauw and Vanrolleghem [16] of the superior performance of the method. Specifically, it was shown that, in terms of accuracy, the SD technique for first derivatives is competitive with AD, and in terms of implementation simplicity, it is similar to the FD method with the added advantage of being step-size insensitive and, hence, unhindered by the step-size dilemma. Matlab code is presented that automates the generalized first-order formula for approximation of gradients, Jacobians, and 2D and 3D spatial derivatives.

Extension of the SD scheme to second-order derivatives was obtained via central differencing in the complex plane, thus allowing approximation of the Hessian. This approach yielded hybrid formulas with many of the desirable features of the original Squire and Trapp first-order formulas but without the ill-conditioning of the FD schemes. Performance of the extension formulas was evaluated and compared with AD and FD methods.

Evaluation results show that the differencing operation involved in the derivation of the second-order formulas decreased their accuracy by four to five orders of magnitude below that of the original Squire–Trapp formula. Nonetheless, compared to FD scheme, the accuracy of the SD scheme is five to six orders of magnitude greater in all tests conducted. In addition, the SD extension formulas exhibit step-size insensitivity over the entire h range of the tests ($1–10^{-30}$), indicating their numerical stability. Finally, the proposed SD method shares with FD its main attractive feature, namely, ease of implementation as illustrated by examples for the Hessian and for partial and cross-partial spatial derivatives.

The key question of the computational cost of the SD method was investigated by Martins et al. [12] in a large-scale sensitivity analysis of a transonic transport wing design. They compared the CPU time and memory usage of MD, AD, and FD relative to SD. Their results indicate that AD is the least efficient, requiring 2.3 times the CPU time and 8.1 times the memory requirement of SD. Whereas MD is by far the fastest, it is more memory-intensive, requiring about 2.5 times that of SD. The FD scheme scored slightly lower time and memory requirements than SD (0.88 and 0.72 relative score), but required considerably more time in pilot runs to establish an optimal step size. Similar results were reported by Burg and Newman III [2] in their study of design space derivative evaluation and by Vatsa [19] in a sensitivity study of Navier–Stokes equations.

Pauw and Vanrolleghem [16] have also assessed the computational cost of SD in a study of local sensitivity analysis of a highly nonlinear bioprocess. They reported SD to be 30 times slower than FD, in large disagreement with other reported results. These authors point out that results of such tests depend on numerous factors, including the complexity of the test problem, the CPU architecture, and the implementation environment (e.g., C++ is less efficient in complex computation than is FORTRAN).

In this regard, it may be relevant to quote Lyness [8], who states: “A machine takes more time to multiply two complex numbers than to multiply two real numbers. Thus a process making direct use of manipulation of complex numbers might have to be four times as efficient as a corresponding process using real numbers to be competitive on a basis of machine running time; it seems unlikely that such processes exits. However,...one might expect the use of complex arithmetic to result in intrinsically simpler processes and possibly more reliable processes.”

In summary, I conclude that, with the extensions introduced in this paper, the complex differentiation method provides a complete system for SD of real-valued functions. It is computationally efficient, highly accurate, and very easy to implement. Compared to FD, it is superior in performance with the added advantage of being step-size-insensitive, ruling out the need to search for optimal step size. It is also competitive with AD in performance and quality of results, with the added advantage of being much easier to implement via simple and maintainable code.

Acknowledgements This research work was conducted at the Department of Earth Sciences, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. I thank Professor G. Korvin and Dr. A. Qahwash for their diligence while reviewing the manuscript and for their very valuable suggestions.

Appendix A

Generalization and extension of SD

Generalization of first-order derivatives

Consider a real vector-valued function $\mathbf{f}$ of a real vector argument $\mathbf{p}$ of M physical parameters, i.e., $\mathbf{f}=\mathbf{f}(\mathbf{p})$, $\mathbf{p}=[p_k]_{k=1}^{M}$. Let $\mathbf{e}$ be an arbitrary unit m vector and define the complex parameters vector: $\mathbf{z}=\mathbf{p}+ih\mathbf{e}$, where $i=\sqrt{-1}$, and h is a small step ($h\ll 1$). If the complex extension $\mathbf{f}(\mathbf{z})$ of $\mathbf{f}(\mathbf{p})$ is analytic, then the function $\mathbf{f}(\mathbf{z})=\mathbf{f}(\mathbf{p}+ih\mathbf{e})$ may be expanded in the complex neighborhood of $\mathbf{p}$ in a Taylor series as:

$$\mathbf{f}(\mathbf{z})=\mathbf{f}(\mathbf{p}+ih\mathbf{e})=\mathbf{f}(\mathbf{p})+ih\mathbf{e}^T\mathbf{f}'(\mathbf{p})-h^2R-ih^3Q, \quad (13)$$

where the superscript T stands for transpose operation and R and Q are remainder terms. Collecting real and imaginary parts:

$$\begin{aligned}\mathbf{f}(\mathbf{z})=\mathbf{f}(\mathbf{p}+i\mathbf{h})&=\left[\mathbf{f}(\mathbf{p})-h^2R\right]+ih\left[\mathbf{e}^{\mathrm{T}}\mathbf{f}'(\mathbf{p})-ih^2Q\right]\\&=\mathrm{Re}[\mathbf{f}(\mathbf{z})]+ih\mathrm{Im}[\mathbf{f}(\mathbf{z})]\end{aligned}$$

Solving in terms of derivatives and rearranging, we get:

$$\left.\begin{aligned}\mathbf{f}(\mathbf{p})&=\mathrm{Re}[\mathbf{f}(\mathbf{z})]+h^2R\\\mathbf{e}^T\mathbf{f}'(\mathbf{p})&=\frac{1}{h}\mathrm{Im}[\mathbf{f}(\mathbf{z})]+h^2Q\end{aligned}\right\} \quad (14)$$

Extension to second-order derivatives

Extension of Eq. 14 to second-order derivatives may be achieved by computing either a forward, backward, or central difference approximation. The forward and central approximation formulas will be derived here so that their performance may be compared against each other and against AD.

Forward difference formula

Let $\mathbf{p}_+=\mathbf{p}+\delta p\mathbf{e}$, where $\mathbf{e}$ is a unit m vector and δp is a small real step, $\delta p\ll 1$. As in the case of single perturbation, define the complex parameters vectors by: $\mathbf{z}_+=\mathbf{p}_++ih\mathbf{e}$ and $\mathbf{z}_0=\mathbf{p}+ih\mathbf{e}$. Under the assumption of analyticity of $\mathbf{f}(\mathbf{z})$, we may expand the functions $\mathbf{f}(\mathbf{z}_+)=\mathbf{f}(\mathbf{p}_++ih\mathbf{e})$ and $\mathbf{f}(\mathbf{z}_o)=\mathbf{f}(\mathbf{p}+ih\mathbf{e})$ in two Taylor series in the complex domain about $\mathbf{p}_+$ and $\mathbf{p}$ as follows:

$$\left.\begin{aligned}\mathbf{f}(\mathbf{z}_+)&=\mathbf{f}(\mathbf{p}_++ih\mathbf{e})=\mathbf{f}(\mathbf{p}_+)+ih\mathbf{e}^T\mathbf{f}'(\mathbf{p}_+)-h^2R_+-ih^3Q_+\\\mathbf{f}(\mathbf{z}_0)&=\mathbf{f}(\mathbf{p}+ih\mathbf{e})=\mathbf{f}(\mathbf{p})+ih\mathbf{e}^T\mathbf{f}'(\mathbf{p})-h^2R_o-ih^3Q_o\end{aligned}\right\} \quad (15)$$

Taking the difference $\Delta\mathbf{f}(z)=\mathbf{f}(z_+)-\mathbf{f}(z_o)$ we obtain:

$$\begin{aligned}\Delta\mathbf{f}(\mathbf{z})&=\mathbf{f}(\mathbf{z}_+)-\mathbf{f}(\mathbf{z}_o)\\&=\Delta\mathbf{f}(\mathbf{p})+ih\mathbf{e}^T\Delta\mathbf{f}'(\mathbf{p})-h^2\Delta R-ih^3\Delta Q\end{aligned} \quad (16)$$

Next, we expand terms in $\mathbf{p}_+$ on the right-hand side (rhs) of the first equation of Eq. 15, retaining only the first two terms to get:

$$\left.\begin{aligned}\mathbf{f}(\mathbf{p}_+)&\simeq\mathbf{f}(\mathbf{p})+\delta p\mathbf{e}^T\mathbf{f}'(\mathbf{p})\\\mathbf{f}'(\mathbf{p}_+)&\simeq\mathbf{f}'(\mathbf{p})+\delta p\mathbf{e}^T\mathbf{f}(\mathbf{p})\end{aligned}\right\}\Rightarrow \left\{\begin{aligned}\Delta\mathbf{f}(\mathbf{p})&\simeq\mathbf{f}(\mathbf{p}_+)-\mathbf{f}(\mathbf{p})=\delta p\mathbf{e}^T\mathbf{f}'(\mathbf{p})\\\Delta\mathbf{f}'(\mathbf{p})&\simeq\mathbf{f}'(\mathbf{p}_+)-\mathbf{f}'(\mathbf{p})=\delta p\mathbf{e}^T\mathbf{f}''(\mathbf{p})\end{aligned}\right. \quad (17)$$

From Eq. 16 into Eq. 17, collecting real and imaginary terms:

$$\Delta\mathbf{f}(\mathbf{z})=\delta p\left[\mathbf{e}^T\mathbf{f}'(\mathbf{p})-h^2\Delta R\right]+ih\delta p\left[\mathbf{e}^T\mathbf{f}''(\mathbf{p})\mathbf{e}-h^2\Delta Q\right] \quad (18)$$

The equation above may be solved for the derivative terms to give:

$$\left.\begin{aligned}\mathbf{e}^T\mathbf{f}'(\mathbf{p})&=\tfrac{1}{\delta p}\mathrm{Re}[\Delta\mathbf{f}(\mathbf{z})]+h^2\Delta R\\\mathbf{e}^T\mathbf{f}''(\mathbf{p})\mathbf{e}&=\tfrac{1}{h\delta p}\mathrm{Im}[\Delta\mathbf{f}(\mathbf{z})]+h^2\Delta Q\end{aligned}\right\} \quad (19)$$

Central difference formula

Let $\mathbf{p}_\pm=\mathbf{p}\pm\delta p\mathbf{e}$, where $\mathbf{e}$ is an arbitrary unit m vector and δp is a small step size $\delta p\ll 1$; define the complex parameters vectors by: $\mathbf{z}_\pm=\mathbf{p}_\pm+ih\mathbf{e}$. Assuming analyticity of $\mathbf{f}(\mathbf{p})$, we can expand the functions $\mathbf{f}(\mathbf{z}_\pm)=\mathbf{f}(\mathbf{p}_\pm+ih\mathbf{e})$ in two Taylor series in the complex domain about $\mathbf{p}_\pm$ to get:

$$\mathbf{f}(\mathbf{z}_\pm)=\mathbf{f}(\mathbf{p}_\pm+ih\mathbf{e})=\mathbf{f}(\mathbf{p}_\pm)+ih\mathbf{e}^T\mathbf{f}'(\mathbf{p}_\pm)-h^2R-ih^3Q \quad (20)$$

The difference function $\Delta\mathbf{f}(\mathbf{p})=\mathbf{f}(\mathbf{p}_+)-\mathbf{f}(\mathbf{p}_-)$ becomes:

$$\begin{aligned}\Delta\mathbf{f}(\mathbf{z}_\pm)&=\mathbf{f}(\mathbf{z}_+)-\mathbf{f}(\mathbf{z}_-)\\&=\Delta\mathbf{f}(\mathbf{p}_\pm)+ih\mathbf{e}^T\Delta\mathbf{f}'(\mathbf{p}_\pm)-h^2\Delta R-ih^3\Delta Q\end{aligned} \quad (21)$$

Carrying out a second Taylor expansion of the terms in $\mathbf{p}_\pm$ on the rhs of Eq. 20, and retaining only the first two terms:

$$\left.\begin{aligned}\mathbf{f}(\mathbf{p}_\pm)&\simeq\mathbf{f}(\mathbf{p})\pm\delta p\mathbf{e}^T\mathbf{f}'(\mathbf{p})\\\mathbf{f}'(\mathbf{p}_\pm)&\simeq\mathbf{f}'(\mathbf{p})\pm\delta p\mathbf{f}''(\mathbf{p})\mathbf{e}\end{aligned}\right\}\Rightarrow\left\{\begin{aligned}\Delta\mathbf{f}(\mathbf{p}_\pm)&\simeq 2\delta p\mathbf{e}^T\mathbf{f}'(\mathbf{p})\\\Delta\mathbf{f}'(\mathbf{p}_\pm)&\simeq 2\delta p\mathbf{f}''(\mathbf{p})\mathbf{e}\end{aligned}\right. . \quad (22)$$

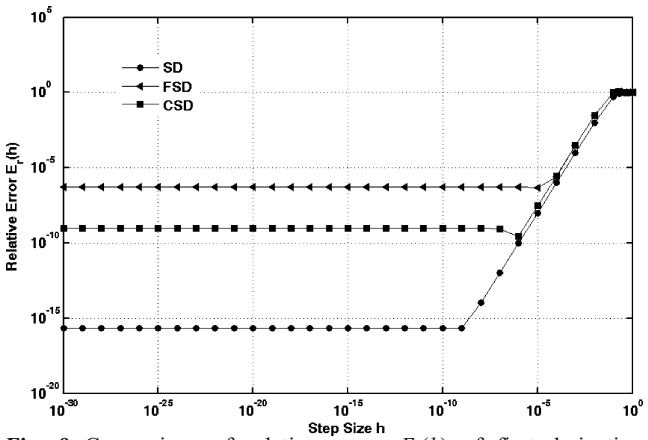

Fig. 9 Comparison of relative errors $E_r(h)$ of first derivative approximations $f'(x)$ using Squire–Trap formula (*circles*) and forward (*triangles*) and centered (*squares*) difference SD formulas. The centered difference SD is lower in accuracy than the Squire–Trap formula by about seven orders of magnitude, the price paid for the differencing operation

From Eqs. 21 into 22 we obtain:

$$\Delta\mathbf{f}(\mathbf{z}) = 2\delta p\left[\mathbf{e}^T\mathbf{f}'(\mathbf{p}) - h^2\Delta R\right] + i2h\delta p\left[\mathbf{e}^T\mathbf{f}''(\mathbf{p})\mathbf{e} - h^2\Delta Q\right]. \tag{23}$$

Solving Eq. 24 for the derivatives yields:

$$\left.\begin{aligned}\mathbf{e}^T\mathbf{f}'(\mathbf{p}) &= \tfrac{1}{2\delta p}\mathrm{Re}[\Delta\mathbf{f}(\mathbf{z})] + h^2\Delta R\\ \mathbf{e}^T\mathbf{f}''(\mathbf{p})\mathbf{e} &= \tfrac{1}{2h\delta p}\mathrm{Im}[\Delta\mathbf{f}(\mathbf{z})] + h^2\Delta Q\end{aligned}\right\} \tag{24}$$

Note that Eqs. 14, 19, and 24 all provide approximations of the first derivative of the target function, and all three formulas have the same $O(h^2)$ degree of approximation. However, while Eq. 14 involves no subtraction, Eqs. 19 and 24 are based on differencing operations. It may be interesting to compare the performance of these three different approximation schemes for first-order derivative. Figure 9 shows the relative errors $E_r(h) = \left|f'_{Exact} - f'_{Est}\right| / \left|f'_{\mathrm{Exact}}\right|$ of the three formulas applied to the test function

$$f(x) = \frac{(x^2+1)}{(x^3-1)}$$

near its point of singularity x=0.9.

It is clear from the figure that all three formulas converge quadratically consistent with their $O(h^2)$ approximation order. The performance of the Squire–Trap formula, which is free of subtractive cancellation, is far superior to the other two, with relative minimum error near machine precision $E_r(h)=10^{-16}$. Moreover, the central difference scheme has higher accuracy than the forward difference scheme, despite the fact that both involve differencing operations and have the same $O(h^2)$ order of approximation.

Gravity anomaly of faulted structure

$$\Delta g(\mathbf{p},\mathbf{x}) = 2G\Delta\rho\left\{F\begin{bmatrix}\sin(d)In(r_2/r_1)\\ +\cos(d)(\phi_2-\phi_1)\end{bmatrix} + z\phi_2 - z_o\phi_1\right\} \tag{25}$$

$$\begin{aligned}F &= x\sin(i) - z_o\cos(i)\\ r_1^2 &= x^2 + z_o^2\\ r_2^2 &= [x + t\cot(i)]^2 + z^2\\ \phi_1 &= \pi/2 + \arctan(x/z_o)\\ \phi_2 &= \pi/2 + \arctan[(x + t\cot(i))/z]\\ \mathbf{p} &= [\Delta\rho, x_o, z_o, t, i]\end{aligned} \tag{26}$$

The parameters to be estimated by inversion are:

$\Delta\rho$	Density contrast
x_o	Position of fault tract (km)
z_o	Depth to top of faulted bed (km)
z	Depth to base of faulted bed (km)
t	$(z-z_o)$=Thickness of faulted bed (km)
d	Dip of fault plane

Appendix B

Matlab implementation of SD

1. *cJac.m*: This Matlab function implements Eq. 14 for the Jacobian **J** of **f**(**p**).

```
function [f,J] = cJac(fun, p, Extra)
% cJac Complex approximation of f = fun(p, Extra) and its Jacobian J
%
% Usage:
%       [f, J] = cJac(fun, p, Extra)
% Input:
%      fun = Name of function to be evaluated
= %         %            fun(p,Extra)
%    Extra = Independant variable
%        p = function parameters
% Output:
%        f = function value f(p,Extra)
%        J = Jacobian at p
%
% See also: cGrad  cDerive  cHg  cDiff

%--------------------------------------------------------------------
% Written by: Dr. A. Abokhodair                    Date: 08/06/2006
% Modified by:                                     Date:
%--------------------------------------------------------------------
h = 1.0e-35;
p = p(:);
Nx = max(size(Extra{1}));
Np = max(size(p));
id = eye(Np);
J = zeros(Nx,Np);

% Evaluate the function
f = feval(fun,p,Extra);

% Evaluate the Jacobian Matrix
for ip = 1: Np
    e = id(:,ip);
    z = complex(p,e*h);       % Create complex parameter p(ip)
    fc = feval(fun,z,Extra);
    J(:,ip) = imag(fc)./h;
end;
```

2. *cHg.m*: This Matlab function implements Eq. 14 for the gradient $\mathbf{g}=\mathbf{f}'(\mathbf{p})$ and Hessian $\mathbf{H}=\mathbf{f}''(\mathbf{p})$.

```
function [f,g,H]=cHg(fun,p,Extra)
% cHg Computes SDC Gradient & Hessian (g & H).
% Usage:
%        [f, g, H] = cHg(fun,p, Extar)
% Input:
%        fun = Handle to function f = f(p,Extra)
%          p = parameter vector
%      Extra = Inactive parameters to be passed to fun
% Output:
%      f = function value at p: f(p)
%      g = Gradieng at p: g(p)
%      H = Hessian at p:  H(p)
%
% See also cJac  cGrad  cDerive  cDiff
%-------------------------------------------------------------------
% Written by: Dr. A. Abokhodair                  Date: 08/06/2006
% Modified by:                                   Date:
%-------------------------------------------------------------------

f=feval(fun,p,Extra);

% The SD routine
p  = p(:);
Np = max(size(p));
if(min(size(p))~=1)
    error('p must be a vector');
end

id = eye(Np);
g  = zeros(Np,1);
H  = zeros(Np,Np);
h  = 1.0e-35;
dp = sqrt(eps);

for r = 1: Np
    er  = id(:,r);
    z   = complex(p,h*er);
    for c = 1:Np
        ec = id(:,c);
        P  = z + ec*dp;
        fp = feval(fun,P,Extra);
        P  = z - ec*dp;
        fn = feval(fun,P,Extra);
        Df = fp-fn;
        if (r==c)
            g(r)=real(Df)/(2*dp);
        end
        H(r,c) = imag(Df)/(2*dp*h);
    end
end
H=(H+H')/2;
```

References

1. Blakely, R.J.: Potential Theory in Gravity and Magnetic Applications. Cambridge University Press, New York (1995)
2. Burg, C.O.E., Newman, J.C. III: Computationally efficient, numerically exact design space derivatives via the complex Taylor's series expansion method. Comput. Fluids. **32**, 373–383 (2003)
3. Cerviño, L.I., Bewley, T.R.: On the extension of the complex-step derivative technique to pseudospectral algorithms. J. Comp. Phys. **187**, 544–549 (2003)
4. Coleman, T.F., Verma, A.: ADMIT-1: Automatic differentiation and MATLAB interface toolbox. ACM Trans. Math. Softw. **26**, 150–175 (2000)
5. Elizondo D., Cappelaere, B., Faurem, C.: Automatic versus manual differentiation to compute sensitivities and solve nonlinear inverse problems. Comput. Geosci. **28**, 309–326 (2002)
6. Fornberg, B.: Numerical differentiation of analytic function. ACM Trans. Math. Softw. **7**, 512–526 (1981)
7. Forsythe, G., Moler, C.B.: Computer Solution of Linear Algebraic Systems. Prentice-Hall, Englewood Cliffs (1967)
8. Lyness, J.N.: Numerical algorithms based on the theory of complex variables. In: Rosenthal, S. (ed.) Proceedings of ACM 22nd National Conference, vol. 1, pp. 124–134. Thompson Book, Washington, DC (1967)
9. Lyness, J.N., Moler, C.B.: Numerical differentiation of analytic functions: SIAM J. Numer. Anal. **4**, 202–210 (1967)
10. Mark, H., Workman, J. Jr.: Derivatives in spectroscopy: Part I—the behavior of the derivative. Spectroscopy **18**, 32–37 (2003)
11. Martins, J.R.R.A., Kroo, H.M., Alonso, J.J.: An automated method for sensitivity analysis using complex variables. AIAA-Paper 2000-0689. In: Proceedings of the 38th Aerospace Sciences Meeting, Reno, January 2000.
12. Martins, J.R.R.A., Sturdza, P., Alonso, J.: The complex derivative approximation. ACM Trans. Math. Softw. **29**, 245–262 (2003)
13. Moŕe, J.J.: Automatic differentiation tools in optimization software. Argon National Laboratory, Mathematics and Computer Science Division, Preprint ANL/MCS-P859-1100, (2000)
14. Nielsen H.B.: Immoptibox: A Matlab toolbox for optimization and data fitting. http://www2.imm.dtu.dk/~hbn/immoptibox/ (2005)
15. Parasnis, D.S.: Principles of Applied Geophysics. Chapman & Hall, New York (1997)
16. Pauw, D.J.W., Vanrolleghem, P.A.: Avoiding the finite difference sensitivity analysis deathtrap by using the complex derivative approximation technique. Paper presented at the 3rd biennial meeting of the International Environmental Modeling and Software Society (iEMSs 2006), Burlington 2006
17. Squire, W., Trapp, G.: Using complex variables to estimate derivatives of real functions. SIAM Rev. **40**, 110–112 (1998)
18. Telford, W.M., Geldart, L.P., Sheriff, R.E.: Applied Geophysics. Cambridge University Press, Cambridge (1990)
19. Vatsa, V.N.: Computation of sensitivity derivatives of Navier–Stokes equations using complex variables. Adv. Eng. Softw. **31**, 655–659 (2000)