JCTC Journal of Chemical Theory and Computation

Arbitrary-Order Density Functional Response Theory from Automatic Differentiation

Ulf Ekström,*,† Lucas Visscher,† Radovan Bast,‡ Andreas J. Thorvaldsen,‡ and Kenneth Ruud‡

Division of Theoretical Chemistry, Amsterdam Center for Multiscale Modeling, VU University - Faculty of Sciences, De Boelelaan 1083, NL-1081 HV Amsterdam, The Netherlands, and Centre for Theoretical and Computational Chemistry, Department of Chemistry, University of Tromsø, N-9037 Tromsø, Norway

Received March 2, 2010

Abstract: We demonstrate how the functional derivatives appearing in perturbative time-dependent density functional theory can be calculated using automatic differentiation. The approach starts from a computer implementation of the exchange-correlation energy functional, from which arbitrary-order derivatives are generated automatically. Automatic differentiation is shown to provide an accurate, general, and efficient implementation of higher-order exchange-correlation functional derivatives that is easy to maintain. When used in combination with an arbitrary-order response solver, the methodology allows us to generate arbitrary-order response functions from time-dependent density functional theory.

1. Introduction

Time-dependent density functional theory[1−4] (TDDFT) has become a popular method for the calculation of excited states, due to its low computational cost and, for certain types of excitations, high accuracy. This is despite the fact that present day TDDFT approximations suffer from well-known deficiencies, e.g., in the description of Rydberg excitations, the underestimation of charge-transfer excitations that are associated with significant density transfer, the failure to access explicit many-electron excitations in the adiabatic approximation, and the overestimation of the dynamic polarizability in conducting polymers due to the locality of standard exchange-correlation (XC) kernels. Significant improvements must be made to the approximate energy functionals before TDDFT becomes as reliable for excited states as DFT is for the ground state.[5] We will here consider response theory, i.e., time-dependent and time-independent perturbation theory based on DFT. We use the term TDDFT to refer to all such approaches, even if the perturbations under consideration may not always be time-dependent. Linear response TDDFT is the simplest and most commonly used method, but the perturbation theory can be carried out to arbitrary order in the perturbation strengths. An application where the present day functionals may be safely used in conjunction with high order perturbation theory is in the calculation of geometric derivatives. In this case, the perturbation expansion converges (within its radius of convergence) to the same DFT ground state potential energy surface that may be obtained from separate DFT calculations at different molecular geometries.

Several implementations of response theory within TDDFT have been reported,[6−14] and the calculation of electronic excitation energies and lower-order molecular electromagnetic properties has become routine.[15−18] In this paper, we discuss the extension of analytic response theory within TDDFT to (in principle) arbitrary order in the Taylor expansion of the energy (or the quasi-energy in the Floquet formalism[19]) with respect to field amplitudes (perturbation strengths). This generalization makes it possible to study a wealth of nonlinear optical properties and spectroscopic parameters with (approximate) inclusion of electron correlation effects.

The implementation of TDDFT is technically challenging since, in contrast to the Hartree−Fock exchange matrix within time-dependent Hartree−Fock theory, the expansion

* To whom correspondence should be addressed: E-mail: ulfek@few.vu.nl.
† VU University.
‡ University of Tromsø.

10.1021/ct100117s
Published on Web 05/24/2010

of the XC potential in orders of the density variables does not vanish for second- and higher-order corrections. Repeated application of the chain rule leads therefore to the need for evaluating a large number of derivatives: XC functional derivatives with respect to density variables and derivatives of density variables with respect to the perturbing field strengths. In addition, we may have geometric and magnetic derivatives of basis functions for perturbations which modify the overlap of basis functions such as geometric displacements[20] or magnetic perturbations with London atomic orbitals.[21−23]

The XC energy density derivatives are needed to the same order as the order of the (quasi-) energy derivative. Since most of the modern XC functionals involve rather complicated expressions, higher-order derivatives are often out of reach for manual differentiation. The required derivatives can alternatively be generated using symbolic differentiation techniques (see, e.g., refs 24 and 25), but this approach still requires manual intervention and verification and typically generates rather lengthy code (since every necessary derivative is implemented separately for each functional). In practice, the computer algebra systems do not take issues related to numerical stability into account and may "optimize" statements into numerically unstable forms, something that we will return to in section 4.

Generating functional derivatives via automated symbolic manipulation is therefore not likely to be a practical approach for calculating higher-order XC contributions, although implementations of response functions based on this scheme have been presented in the literature.[26]

We have for this reason adopted a different approach for calculating higher-order XC energy density derivatives based on automatic differentiation (AD).[27,28] The basic idea of AD is that every computer program, no matter how complicated, performs a (possibly long) series of simple operations: these are the usual arithmetic operations, together with a small number of intrinsic mathematical functions for exponentials, logarithms, trigonometric functions, etc. AD then makes use of the fact that a computer implementation of an analytical function f contains all information needed for the calculation of derivatives of f, to arbitrary order. This calculation can either be done by taking the source code implementing f as input, and automatically generating the code for the derivative f', or can be done using operator overloading features of the programming language itself. In both cases, f' is generated from an *existing* implementation of f without manual intervention.

AD has typically been applied to calculate low-order (first and second) derivatives of models with a large number of variables (see for example the applications described in ref 29). In the field of quantum chemistry, we note the application of AD to the calculation of molecular gradients for semiempirical wave functions,[30] an application where a large number of low-order derivatives are calculated.

For the present application, the situation is different: we need high-order derivatives of a large number of functions of a few variables. For example, to calculate the cubic response function using a GGA functional, we need to evaluate fourth-order derivatives of the XC energy density

$$\varepsilon_{\mathrm{xc}}(\mathbf{r}) = \varepsilon_{\mathrm{xc}}(n_\alpha(\mathbf{r}), n_\beta(\mathbf{r}), |\nabla n_\alpha(\mathbf{r})|^2, |\nabla n_\beta(\mathbf{r})|^2, \nabla n_\alpha(\mathbf{r}) \cdot \nabla n_\beta(\mathbf{r})) \quad (1)$$

a function of five local variables that depend on the spin-up (n_α) and spin-down (n_β) parts of the number density n and their Cartesian gradients, at about a million different grid points for a medium-sized molecule. These derivatives are then multiplied with products of perturbed density variables and integrated to form derivatives of the XC energy with respect to perturbation field strengths, according to the chain rule as described in section 2.

Because of these unusual requirements, we have implemented an AD library based on operator overloading, optimized for high-order derivatives of a small number of variables. The implementation is described in section 3. Numerical stability and performance are tested in sections 4 and 5, respectively, and results of sample applications are presented in section 6. We give some concluding remarks in section 7.

2. Arbitrary-Order Adiabatic Time-Dependent Density Functional Theory

The atomic orbital-based, arbitrary order adiabatic TDDFT formalism employed in this work has recently been presented by Thorvaldsen et al.[31] The general formalism needed to accommodate response theory in the Kohn−Sham approach was discussed in ref 31, but this paper did not give explicit expressions for the contributions arising from the derivatives of the XC functionals. For future reference, we will present these explicit expressions here for a closed-shell reference state up to the quartic response functions. In the following discussion of the working equations, we will restrict ourselves to the extension from TDHF to TDDFT and, for the sake of clarity, neglect the spin density contributions in this presentation. Our ansatz is thus the XC energy

$$\varepsilon_{\mathrm{xc}}(\mathbf{r}) = \varepsilon_{\mathrm{xc}}(n(\mathbf{r}), \nabla n(\mathbf{r}) \cdot \nabla n(\mathbf{r})) = \varepsilon_{\mathrm{xc}}(n(\mathbf{r}), Z(\mathbf{r})) \quad (2)$$

where we have introduced the square gradient norm, $Z = \nabla n \cdot \nabla n$, of the density. Compared to eq 1, we can work with the total density n instead of the spin-up and spin-down parts and set the spin density $s = n_\alpha - n_\beta$ to zero. We will, however, point out where and how additional spin density contributions would show up in the working equations. In the case of electric perturbations for a closed-shell reference, as studied in this paper, spin-density contributions are strictly zero for static perturbations but are in general nonzero if the perturbation is frequency dependent. These expressions can be implemented as a straightforward extension of the method described here. Although neglected in this presentation, the reported library for arbitrary-order XC functional derivatives does implement spin-polarized functionals (and derivatives), applicable both to the spin-unrestricted formalism and the spin-restricted formalism with a spin-polarized response.

The additional terms that appear in the working equations when moving from TDHF to Kohn−Sham TDDFT enter in two specific contributions: the first term appears in the XC contribution to the electronic Hessian during the solution of

the set of linear response equations; the second term arises as an additional XC contribution to the perturbed Fock matrix (or matrices) in the contraction of two-electron integrals with the perturbed density matrix expansion. Both contributions require a numerical integration to form Fock-type matrices in the AO basis, $K_{\mathrm{xc};\kappa\lambda}$, with an integrand that can be expressed in the following computationally advantageous form

$$k_{\mathrm{xc};\kappa\lambda}(\mathbf{r}) = u(\mathbf{r})\,\Omega_{\kappa\lambda}(\mathbf{r}) + 2\mathbf{v}(\mathbf{r})\cdot\nabla\Omega_{\kappa\lambda}(\mathbf{r}) \tag{3}$$

containing the AO distribution, $\Omega_{\kappa\lambda} = \phi_\kappa^\dagger\phi_\lambda$, its Cartesian gradient, $\nabla\Omega_{\kappa\lambda}$, and the scalar and vector prefactors u and $\mathbf{v}$, respectively, which depend on the chosen XC functional and contain functional derivatives of $\varepsilon_{\mathrm{xc}}$ with respect to n and Z, as well as products of functional derivatives and derivatives of (perturbed) density variables. The factor of 2 appearing in eq 3 comes from a product rule differentiation of Z in the AO representation, with respect to the AO density matrix coefficients. In the unperturbed case, the prefactors u and $\mathbf{v}$ are given by

$$u = d_{1,0} \tag{4}$$

$$\mathbf{v} = d_{0,1}\nabla n \tag{5}$$

where $d_{1,0}$ and $d_{0,1}$ represent first-order XC functional derivatives using the short-hand notation:

$$d_{i,j} = \left(\frac{\partial}{\partial n}\right)^i\left(\frac{\partial}{\partial Z}\right)^j\varepsilon_{\mathrm{xc}} \tag{6}$$

The compact notation of eqs 4 and 5 will be convenient when giving the expressions for the higher-order contributions. The implementation of the XC contribution in the solution algorithm of linear response equations requires the evaluation of derivatives of the prefactors u and $\mathbf{v}$ with respect to a field amplitude b

$$u^b = d^b_{1,0} \tag{7}$$

$$\mathbf{v}^b = d^b_{0,1}\nabla n + d_{0,1}\nabla n^b \tag{8}$$

where we have introduced the notation $d^b_{i,j} = (d/db)d_{i,j}$ etc., for field-perturbed quantities. $d^b_{1,0}$ and $d^b_{0,1}$ are to be expanded using the chain rule:

$$d^b_{i,j} = d_{i+1,j}n^b + d_{i,j+1}Z^{0,b} \tag{9}$$

where $Z^{0,b}$ (0 meaning unperturbed) is short-hand notation for the dot product of two density gradients: $Z^{a,b} = 2\nabla n^a\cdot\nabla n^b$. This means that the first-order field-perturbed prefactors u^b and $\mathbf{v}^b$ contain first- and second-order functional derivatives as well as first-order perturbed density variables, n^b and $Z^{0,b}$. These working equations for the XC contribution to the linear response functions have been given in the literature numerous times with varying notations.[6−14] Explicit expressions closest to our implementation can be found in ref 32. Note, however, that the expressions in this reference contain additional contributions due to spin magnetization, which we exclude in this presentation. The higher-order XC contributions to the perturbed Fock matrices can be obtained in a similar manner by straightforward differentiation using the chain rule. The contributions up to the quartic response functions are given in the Appendix.

3. Automatic Differentiation

Our implementation of automatic differentiation is based on replacing all "scalar" floating point operations with operations acting on finite-order Taylor polynomials with floating point coefficients. Each intrinsic mathematical function (exp, log, sin, cos, etc.) of the programming language is first extended to return not only the function value $f(x)$ for a particular argument x, but also the derivatives $f^{(i)}(x)$ for i up to a given order. The derivatives can typically be evaluated using less computational effort than the function value $f(x)$ itself, as is for example the case for the logarithmic function, where the derivative is simply $1/x$. The first derivative of $\sin(x)$ requires the computation of $\cos(x)$, but for the second derivative we obtain again the factor $\sin(x)$, which does not need to be evaluated twice. Similar simplifications can be made for all intrinsic mathematical functions of common programming languages.

Using this code for calculating Taylor expansions of intrinsic mathematical functions f, we are in a position to evaluate expressions $f(g(x))$, where $f(z)$ and $g(x)$ are analytical functions. This is done by first Taylor expanding $g(x)$ up to the desired order as the finite polynomial $P(x)$. This polynomial is then inserted into the Taylor expansion of $f(z)$ around $z = P(0)$. The resulting polynomial is the Taylor polynomial of the composite function. Similar results are obtained for products of functions, i.e., the Taylor expansion of a product of functions can be obtained by first expanding the two functions to a given order and then multiplying their truncated Taylor polynomials. We note that the obtained Taylor coefficients are numerically exact, as they do not arise from any kind of finite difference approximation.

We have implemented efficient arithmetic on multivariate Taylor polynomials in the C++ programming language. Multiplication is performed with a fixed truncation level in every polynomial operation, so that all intermediate values of compound expressions have the same complexity. Using operator overloading techniques, we can convert existing code, working in floating point arithmetic, to a code that works using Taylor polynomials with floating point coefficients. [Operator overloading means that the programmer can give meaning to programming statements such as $z = x*y$, when x, y, and z are of some user defined type. This technique has long been used to implement matrix algebra, and we here use it for Taylor series arithmetic.] As long as the parent code defines an analytical function as a composite expression of intrinsic mathematical functions and arithmetic operations, we can in this way obtain arbitrary-order derivatives of the function. It may seem that this approach is equivalent to rederiving the derivative formula every time the program is run. This is however not the case, since we only compute derivatives at a single expansion point. This is a much simpler task than deriving an analytical expression valid for derivatives at arbitrary points.

In our C++ implementation, we have used the *template* feature of the language to make the number of variables and

polynomial degree compile-time constants. This makes the code very efficient but limits the applicability of the implementation to problems where the number of variables is known at compile time. This is not a limitation for our present application where the number of variables is set by the DFT functional type (two variables for LDA, five for GGA, etc.). The advantage of using templates is that they allow the compiler to produce much more efficient code when the polynomial sizes are known at compile time. This is particularly important for multivariate polynomial multiplication, which has to be formulated recursively. Here, a naive recursive implementation not using templates was found to be 50 times slower than the template-based code.

3.1. Example. In order to illustrate the operating principles of the approach described above, we will give an example of automatic differentiation applied to the simple Slater exchange functional

$$\varepsilon_x(n) = Cn^{4/3} \tag{10}$$

where $C = -0.93$ is a constant. To better illustrate the principles of the approach, we will evaluate the exchange functional as $\varepsilon_x(n) = C(n^4)^{1/3}$.

Suppose we want to Taylor-expand, to third order, $\varepsilon_x(n)$ at $n_0 = 2.0\ a_0^{-3}$. We will then deal with third-order Taylor polynomials throughout the calculation. The procedure starts by introducing a small variation, δn, of the density near the expansion point n_0. The density n can then be written as a polynomial in δn:

$$n = 2.0 + \delta n \tag{11}$$

$$= 2.0 + 1.0\delta n + 0.0\delta n^2 + 0.0\delta n^3 \tag{12}$$

$$\equiv \boxed{2.0\ 1.0\ 0.0\ 0.0} \tag{13}$$

In our implementation, all polynomials have the same order, in this example, order three, so zero coefficients have been explicitly inserted into n for the quadratic and cubic terms. The box notation shows the array of coefficients stored in computer memory and reminds us that the procedure is fully "numerical" in nature—that is, it works with the numerical values of the derivatives and not with their symbolic expressions. Typically the coefficients will be stored as double precision floating point numbers, but in this example, we use two significant digits in the calculation.

Since we choose to evaluate the exchange function as $(n^4)^{1/3}$, the first two steps will be

$$n = \boxed{2.0\ 1.0\ 0.0\ 0.0} \tag{14}$$

$$n^4 = \boxed{16.\ 32.\ 24.\ 8.0}. \tag{15}$$

Here the fourth-order term in n^4 is not needed (because we want only derivatives up to order three), and it is therefore not computed. When the computer now encounters the expression

$$(n^4)^{1/3} = (\boxed{16.\ 32.\ 24.\ 8.0})^{1/3} \tag{16}$$

it will generate a third-order Taylor expansion of the cube root function around the constant term of the argument, which in this case is 16. Introducing a dummy variable z, and using the basic properties of the cube root function, we obtain

$$(16. + z)^{1/3} = 2.5 + 0.052z - 0.0011z^2 + 0.000038z^3 \tag{17}$$

Now, we insert $z = n^4 - 16. = \boxed{0.0\ 32.\ 24.\ 8.0}$ into this expansion, to obtain

$$(\boxed{16.\ 32.\ 24.\ 8.0})^{1/3} = \boxed{2.5\ 1.7\ 0.14\ -0.016} \tag{18}$$

Multiplying, finally, with the constant C, we get

$$C \cdot \boxed{2.5\ 1.7\ 0.14\ -0.016} = \boxed{-2.3\ -1.6\ -0.13\ 0.014}, \tag{19}$$

where the numbers in the last box are now the Taylor coefficients of the Slater exchange functionals at $n = 2.0a_0^{-3}$. This is what we set out to calculate. The result is exact, except for round-off errors, and no finite difference approximation has been used. We note that, except for the Taylor coefficients of the intrinsic mathematical functions such as the cube root used above, we only need to be able to add, subtract, and multiply Taylor polynomials for the scheme to work. For multivariate functions we deal with multivariate Taylor expansions, but the principle remains the same.

The procedure used above may seem like a rather cumbersome way of differentiating eq 10, where we can immediately write down the expression for the derivative to arbitrary order. For more complicated compound expressions, there is however no simpler way of differentiation than to differentiate its parts and combine them using the chain rule. This is exactly what the AD approach does. It is clear that, in some cases, there may be an overhead associated with using AD as described above. In particular, we do not take advantage of sparsity (coefficients that are known *a priori* to be zero) in the derivatives. Since the XC energy and derivatives are evaluated at a large number of grid points, and these evaluations all share the same sparsity pattern, there is an opportunity to optimize the process further. We leave this as a topic for a future study, since the XC energy and derivative evaluation takes only a small amount of time in a typical TDDFT calculation (cf. Figure 2).

4. Numerical Stability

The numerical stability of the scheme described in section 3 depends on the function being differentiated. Loss of precision most often appears when subtracting two almost equal numbers, and if possible such expressions should be reformulated to avoid cancellation error. However, since the AD library provides highly accurate implementations for both the function value *and the derivatives* of intrinsic mathematical functions, there is less possibility for a loss of precision compared to code generated by a symbolic algebra package. With a symbolic derivative approach, statements are typically reordered, and the final program bears little resemblance to the input provided by the programmer.

We investigate the numerical stability issue by performing the same calculation in double precision (about 16 decimal

Table 1. Relative Accuracy ("Number of Correct Digits") of AD Density Functional Derivatives, Defined as $\log_{10}\langle\varepsilon_{xc}^{(N)}\rangle/\langle\Delta\varepsilon_{xc}^{(N)}\rangle$, Where $\langle\varepsilon_{xc}^{(N)}\rangle$ Is the Root Mean Square Average of All Partial Derivatives of Order N, and $\Delta\varepsilon_{xc}$ Indicates the Difference between the Values Computed in Double Precision and the Highly Accurate Quad-Double Precision Values

order N	LDA[a]	LSDA[b]	LDA[c]	BLYP[d]
0	16.6	16.6	13.1	16.2
1	16.1	16.0	13.2	15.0
2	15.8	16.3	12.4	14.5
3	15.9	15.1	12.2	14.6
4	15.4	14.7	12.0	14.5
5	15.4	14.5	12.0	13.8

[a] Evaluated at $n = 1a_0^{-3}$. [b] At $n = 1a_0^{-3}$, $n_\alpha - n_\beta = 0.5a_0^{-3}$. [c] At $n = 10^{-12}a_0^{-3}$. [d] At $n = 2 \times 10^6 a_0^{-3}$, $|\nabla n|^2 = 10^{19}a_0^{-8}$.

digits) and quad-double (64 digits) precision, using the QD library.[33] By taking the quad-double numbers as a reference, we can study the error of the double-precision results. The evaluations are done for densities on the order of unity, as well as very small densities. For BLYP, we use a density that is typical near the nucleus of a heavy atom, where the gradient correction present in BLYP is expected to play a large role. The results are summarized in Table 1 for the LDA (SVWN5)[34,35] and BLYP[36−38] functionals. Using double precision arithmetic, we typically obtain 14−15 correct decimal digits, also for the higher-order derivatives. Since the higher-order derivatives are a result of a large number of arithmetic operations, they suffer from some loss of accuracy. In the present examples, we observe a loss of one to two decimal digits in the fifth-order derivatives compared to the accuracy of the XC energy density itself. In some limiting cases, the method suffers from larger errors, as illustrated by the LDA result of Table 1. The derivatives have in this case been evaluated at a very small density, $n = 10^{-12}a_0^{-3}$, and have in this case a relative accuracy of 12−13 digits. A further loss of accuracy is obtained at even smaller densities, but these errors are less important, because low density regions contribute only very little to the total XC energy in a molecule or solid. For all results presented in section 5, we could safely ignore contributions from grid points with an unperturbed density smaller than $10^{-12}a_0^{-3}$. However, for high-order outer valence properties beyond the properties studied in this work, these contributions may still turn out to become significant.

An issue related to the accuracy of the derivatives for small densities is the problem that the derivatives of a function near a singular point may become very large and cause numerical overflow. The standard density functionals are not differentiable at $n = 0$, which may potentially lead to problems. Taking a close look at the fifth-order partial derivatives of the PBE and BLYP functionals (Figure 1), we see that some partial derivatives are indeed very large. The relative errors are however typically below 10^{-10} even at this high order of derivatives. The few derivatives that have larger relative error all have absolute errors smaller than 10^{-12}, which makes these errors negligible in actual calculations. The BLYP functional suffers less from round-off errors, and here the relative errors are smaller than 10^{-12}.

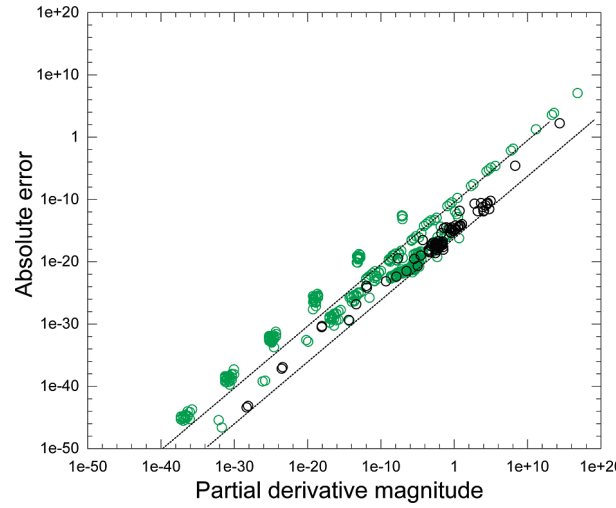

Figure 1. Absolute error as a function of partial derivative magnitude, for the fifth-order derivatives of the BLYP (black) and PBE (green) functionals. Lines are drawn corresponding to relative errors of 10^{-10} (upper line) and 10^{-16} (lower line). The derivatives were evaluated at three different densities: (1.1, 0.9, 17.0, 3.4, 0.1), (1×10^{-6}, 0.9, 1.0, 1×10^3, 0.1), and (1.0, 0.9, 1×10^6, 1×10^6, -1×10^5), using values in atomic units and the variables listed in eq 1.

Locating the exact source of the round-off errors in PBE is left for a future study.

The problem of large derivatives at small densities may be alleviated, if we, instead of expanding $\varepsilon_{xc}(n_0 + x)$ in x, expand $\varepsilon_{xc}(n_0(1 + x))$, which in effect produces weighted derivatives $n_0^m\varepsilon_{xc}^{(m)}(n_0)$. A reciprocal weighting factor n_0^{-m} is introduced in the perturbed density matrices, from which perturbed density variables $n^{b\cdots}$ and $Z^{b\cdots}$ (eqs 28 and 29 in the Appendix) are calculated, which in both cases prevents numerical over- and underflow.

5. Performance

XC derivatives are rarely a bottleneck in TDDFT calculations, compared to the cost of evaluating the density itself at each gridpoint, but we will nevertheless discuss some performance aspects of our approach and implementation. The computational cost for a given density functional depends on the derivative order N and the number of variables K the functional depends on. For spin-polarized LDA functionals, we have $K = 2$, and for GGA functionals, we have $K = 5$. There are a total of $M_N^K = \binom{K+N}{N} = \mathcal{O}(N^K)$ partial derivatives up to order N. Product expressions, $f(n)\,g(n)$, are evaluated using "naive" polynomial multiplication, requiring $\mathcal{O}(N^{2K})$ operations. For the evaluation of intrinsic mathematical functions such as $\exp(f(n))$, a total of $\mathcal{O}(N^{2K+1})$ floating point operations are needed for the evaluation of all partial derivatives (although we have in many parts of the implementation used the "fast" algorithms that exist for the manipulation of Taylor series[39]). We can therefore expect that, for a given XC functional, the asymptotic cost for calculating derivatives up to order N with respect to K variables is $\mathcal{O}(N^{2K+1})$. However, we are rarely interested in the asymptotic behavior since the derivative order N is in practice limited to rather small values. The

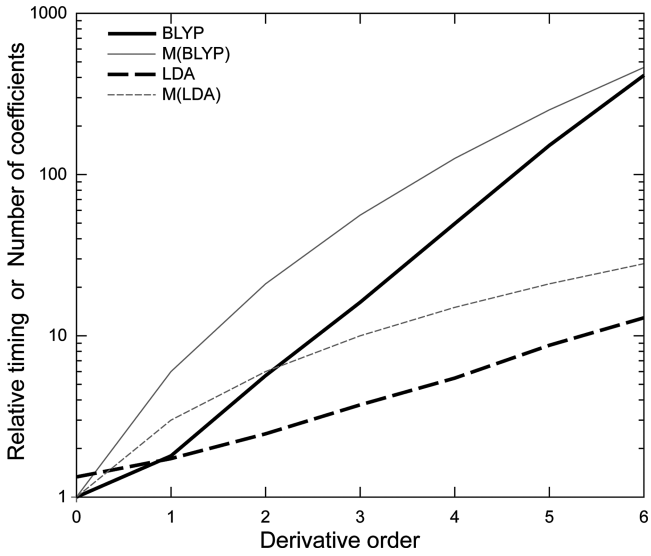

Figure 2. Timings for the evaluation of all partial derivatives of LDA and BLYP functionals up to a given order. Also shown is the number of such partial derivative coefficients, *M*(LDA) and *M*(BLYP). Timings are relative to the time used in evaluating the BLYP XC energy. On a rather modest CPU (1.7 GHz Intel Pentium M), this operation takes 3.0 s for 10^6 grid points.

performance for small N values is strongly affected by the particular implementation, as well as compiler optimizations and CPU architecture. We therefore show in Figure 2 real timings for LDA and BLYP derivatives up to order $N = 6$. These timings are plotted together with the M_N^K factor, showing that up to order six the number of partial derivatives grows faster than the time taken to compute them for the BLYP ($K = 5$) and LDA ($K = 2$) functionals. The reason for this is that, for the calculation of derivatives, almost no additional evaluations of intrinsic mathematical functions are needed. From Figure 2, we conclude that, even for a very high order of the functional derivatives, the XC contribution is unlikely to be a computational bottleneck in TDDFT calculations since the number of evaluations grows linearly with the number of grid points which in turn typically grow linearly with the system size.

6. Results and Discussion

As a demonstration of our approach, we have calculated static and frequency-dependent first, second, and third hyperpolarizabilities, $\beta(-2\omega; \omega, \omega)$, $\gamma(-3\omega; \omega, \omega, \omega)$, and $\delta(-4\omega; \omega, \omega, \omega)$, of the FH molecule. All calculations were performed with a locally modified version of the DIRAC quantum chemistry package,[40] using the radial quadrature proposed by Lindh et al.,[41] Lebedev grids[42] for integration on spheres, and (quadruple augmented) cc-pV{D,T}Z basis sets of Dunning and co-workers.[43]

Selected nonzero components of the calculated (hyper)-polarizability tensors obtained using the HF and LDA methods, respectively, are reported in Table 2. Both the static and the frequency-dependent LDA (hyper)polarizabilities are all consistently larger in magnitude than the HF results. This is in agreement with the smaller HOMO−LUMO energy gap of LDA (0.33 E_h) compared to HF (0.65 E_h). Note that the largest second hyperpolarizability tensor elements (both static and frequency dependent) are the components perpendicular to the molecular axis, $\gamma(x; x, x, x)$ in Table 2, and not the parallel tensor element, $\gamma(z; z, z, z)$. We can also note that, while the static HF and LDA (hyper)polarizabilities have consistent signs, this is not the case for frequency-dependent $\delta(z; x, x, x)$ and $\delta(z; z, z, x)$ elements.

We would like to mention that no GGA results are reported in Table 2, although we have access to numerically stable analytic arbitrary-order GGA functional derivatives and analytic derivatives of perturbed Kohn−Sham matrix elements. During extensive calibration studies, we have observed that the numerical integration of the GGA γ and δ elements of FH is difficult. Using presently available XC numerical integration grids, we are not able to obtain GGA results of FH that are stable with respect to changes in the grid, and we have therefore omitted these results from the discussion. We emphasize, however, that, for a fixed grid, our results for the GGAs correspond in the static cases to the results obtained from a finite difference of lower-order (hyper)polarizabilities.

In order to illustrate the problem in the integration of higher-order GGA contributions we have plotted the distri-

Table 2. Components of the Static and Frequency-Dependent Polarizability, and the First, Second, and Third Hyperpolarizabilities of FH Calculated Using the q-aug-cc-pVTZ Basis Set[a]

	$\omega = 0$		$\omega = 0.06562$ au		$\omega = 0.072$ au	
	HF	LDA	HF	LDA	HF	LDA
$\alpha(x; x)$	4.495	5.930	4.529	6.013	4.537	6.030
$\alpha(z; z)$	5.759	6.854	5.802	6.924	5.811	6.939
$\beta(x; z, x)$	−0.5087	−2.329	−0.6237	−3.074	−0.6519	−3.274
$\beta(z; x, x)$	−0.5087	−2.329	−0.5106	−2.632	−0.5101	−2.701
$\beta(z; z, z)$	−8.397	−10.52	−9.056	−11.72	−9.200	−11.99
$\gamma(x; x, x, x)$	335.9	1148	429.6	1887	453.9	2140
$\gamma(x; z, z, x)$	96.87	309.6	126.3	549.7	134.3	639.6
$\gamma(z; z, x, x)$	96.87	309.6	118.4	446.2	123.4	484.9
$\gamma(z; z, z, z)$	279.6	636.1	342.3	876.7	357.5	942.7
$\delta(x; z, x, x, x)$	111.3	592.6	−199.5	−11218	−393.6	−31464
$\delta(z; x, x, x, x)$	111.3	592.6	257.6	65.90	328.3	−250.5
$\delta(x; z, z, z, x)$	75.95	1618	−302.2	−7668	−554.5	−30618
$\delta(z; z, z, x, x)$	75.95	1619	−39.63	2465	−79.56	2951
$\delta(z; z, z, z, z)$	−1484.2	−2079	−3062	−8340	−3574	−11374

[a] All numbers in atomic units; all perturbing dipole operators carry the same frequency, 0, 0.06562 au, or 0.072 au; $R_e = 1.7328$ bohr; the direction of the positive z axis is from F to H.

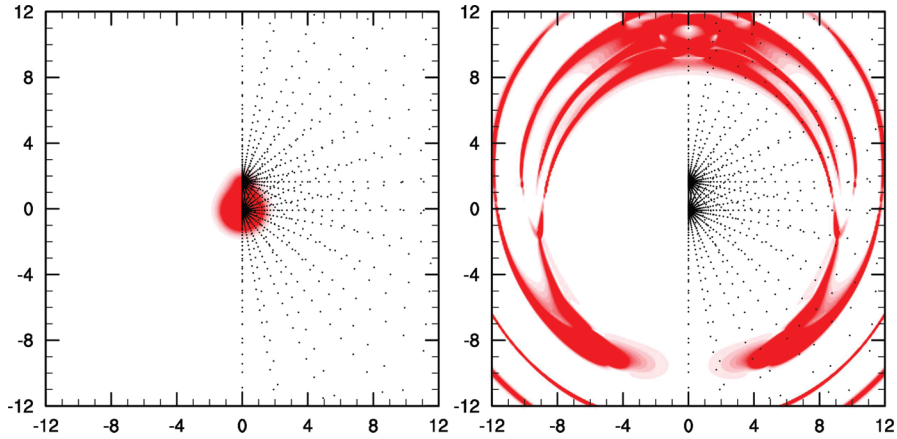

Figure 3. BLYP XC energy density of the FH molecule (left panel) and $|\mathbf{v}^{bcd}|$ of eq 26 (right panel; all perturbations are static and parallel to the molecular axis). The color intensity is proportional to the respective absolute value. The numerical integration grid points are represented as dots which "radiate" from the atom centers at (0, 0, 0) bohr and (0, 0, 1.7328) bohr (dimensions: 24 × 24 bohr).

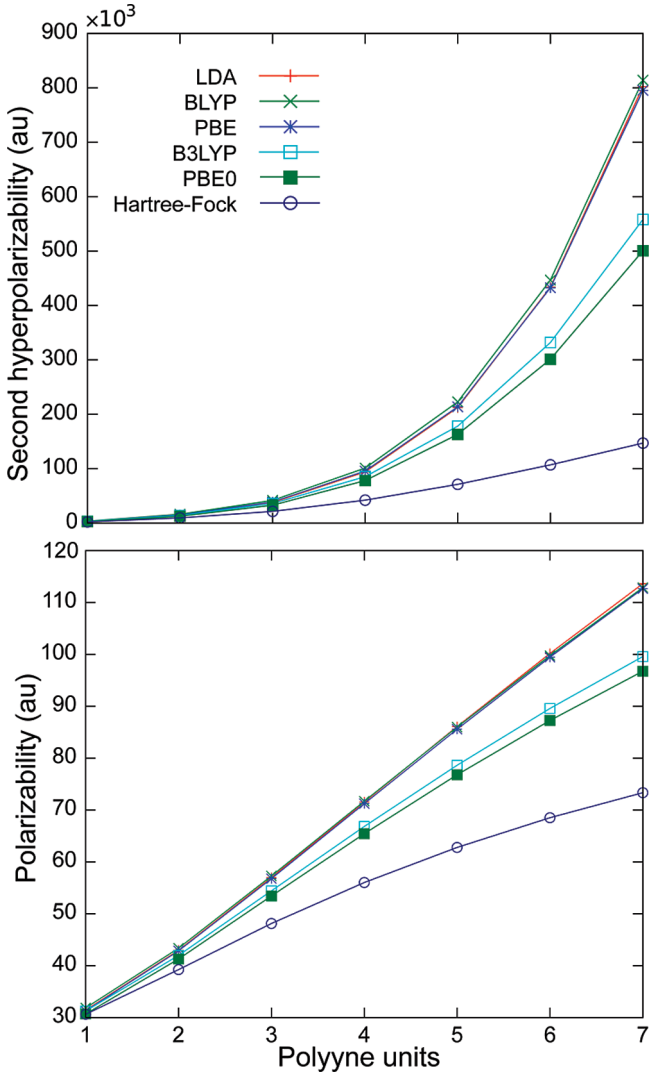

Figure 4. Static polarizability $\alpha(z; z)$ and second hyperpolarizability $\gamma(z; z, z, z)$ per polyyne unit for a linear polyyne ($C_{2N}H_2$) chain (aug-cc-pVDZ; same geometries as in ref 46). LDA, BLYP, and PBE curves are nearly identical at this scale.

bution of grid points and representative functions in Figure 3. In the left panel of Figure 3, we plot the BLYP XC energy density and observe that it is well represented by a grid which is dense in the regions where the XC energy density is significant. The situation is very different in the integration of higher-order GGA terms, where the dominating function values—visible as a series of rings in the right panel in Figure 3—are only poorly sampled by the integration grid. We have found that already β calculations may require a new integration strategy in order to obtain fully converged results.

To remedy the illustrated deficiency of the presently available numerical integration grid, we are currently investigating alternative integration techniques. We focus in particular on adaptive numerical integration schemes which would allow for a more flexible and balanced representation of higher-order valence properties as well as mixed geometric-electromagnetic properties such as Raman optical activity.[44]

We emphasize that these problems are not unique to our AD implementation but are inherent to *all* perturbative TDDFT calculations based on eq 3 and probably strongly dependent on the molecule studied. To give an example where the GGA numerical integration was unproblematic, we plot the parallel static polarizability and second hyperpolarizability for linear polyyne ($C_{2N}H_2$) chains in Figure 4. These quantities are severely overestimated by nonhybrid XC functionals.[45] The numerical integration of the parallel component of α and γ was found to be stable to variations in grid parameters with LDA, BLYP, and PBE curves being nearly identical at the scale of Figure 4.

Finally, we would like to mention that the calculation of higher-order valence properties requires not only well calibrated numerical grids but also increasingly diffuse basis sets. Very diffuse basis sets with an even higher augmentation level than the basis sets employed in this work may cause numerical problems due to linear dependencies already in the ground state calculations in addition to a challenging numerical integration of the higher-order XC contributions.

7. Conclusions

We have shown how high-order, time-dependent density functional theory methods can be reliably and efficiently

implemented, using automatic differentiation when evaluating the XC energy derivatives. Numerical roundoff errors are negligible even at very high (fifth) order derivatives.

We have presented HF and LDA static and frequency-dependent first, second, and third hyperpolarizabilities of the FH molecule and discussed the presently challenging numerical integration to obtain the corresponding GGA results. Applying the arbitrary-order response methods to the second hyperpolarizability of polyyene chains, we show that this quantity is severely overestimated by the LDA XC functional, and that using GGA functionals beyond the adiabatic LDA approximation does not improve the results. We expect the results to improve using time-dependent current-density-functional theory[45,47] or with an exact-exchange DFT approach.[48−50]

To facilitate a more widespread use of the AD method in the DFT community, we have developed a generic software library, XCFun,[51] for calculating arbitrary-order XC derivatives, using the approaches described above. It is similar in scope to the Libxc library[52] but provides derivatives to arbitrary order and works with any set of density variables (for example, n_α and n_β or $n = n_\alpha + n_\beta$ and $s = n_\alpha - n_\beta$). XCFun is therefore suitable both for development of new XC functionals and for calculations of DFT response properties to arbitrary order.

Acknowledgment. This work has received support from the Norwegian Research Council through a Centre of Excellence Grant (Grant No. 179568/V30), a YFF grant to K.R. (Grant No. 162746/V00), and a VIBRON Grant (Grant No. 177558/V30), as well as through a grant of computer time from the Norwegian Supercomputing Program. L.V. thanks The Netherlands Organisation for Scientific Research (NWO) for support through the VICI programme. U.E. acknowledges support from the Wenner-Gren foundations.

Appendix

A. Spin Density Contribution. If the spin-density contributions also were to be included, eq 3 would contain additional terms, one term, $k^z_{\mathrm{xc};\kappa\lambda}$, in the collinear spin density approximation or three terms, $k^x_{\mathrm{xc};\kappa\lambda}$, $k^y_{\mathrm{xc};\kappa\lambda}$, and $k^z_{\mathrm{xc};\kappa\lambda}$, in the noncollinear spin density approach, where

$$k^\mu_{\mathrm{xc};\kappa\lambda} = u_\mu \Omega^\mu_{\kappa\lambda} + 2\mathbf{v}_\mu \cdot \nabla \Omega^\mu_{\kappa\lambda} \tag{20}$$

and $\Omega^\mu_{\kappa\lambda} = \phi^\dagger_\kappa \sigma_\mu \phi_\lambda$ with σ_μ being one of the Pauli spin matrices ($\mu = x, y, z$). In addition, eq 9 and all higher-order contributions given below would include additional perturbed density variables. Within linear response, corresponding noncollinear spin density contributions can be found for instance in ref 32.

B. Second-Order XC Contribution. The second-order XC contribution requires second-order field-perturbed prefactors u^{bc} and $\mathbf{v}^{bc}$

$$u^{bc} = d^{bc}_{1,0} \tag{21}$$

$$\mathbf{v}^{bc} = d^{bc}_{0,1}\nabla n + d^b_{0,1}\nabla n^c + d^c_{0,1}\nabla n^b \tag{22}$$

where $d^{bc}_{1,0}$ and $d^{bc}_{0,1}$ are to be expanded using

$$d^{bc}_{i,j} = d^b_{i+1,j}n^c + d^b_{i,j+1}Z^{0,c} + d_{i,j+1}Z^{b,c} \tag{23}$$

The lower-order terms $d^b_{i+1,j}$ and $d^b_{i,j+1}$ are to be expanded according to eq 9. This means that the second-order field-perturbed prefactors u^{bc} and $\mathbf{v}^{bc}$ contain first-, second-, and third-order functional derivatives as well as first-order derivatives of density variables, and the second-order term $Z^{b,c} = 2\nabla n^b \cdot \nabla n^c$. Observe that terms containing the highest-order density matrix (here, terms containing n^{bc}) are not present due to the $2n + 1$ rule.

C. Third- and Fourth-Order XC Contributions. Third- and fourth-order prefactors for cubic and quartic response functions, respectively, can be obtained accordingly:

$$u^{bcd} = d^{bcd}_{1,0} \tag{24}$$

$$u^{bcde} = d^{bcde}_{1,0} \tag{25}$$

$$\mathbf{v}^{bcd} = d^{bcd}_{0,1}\nabla n + d^{bc}_{0,1}\nabla n^d + d^{bd}_{0,1}\nabla n^c + d^{cd}_{0,1}\nabla n^b + d^b_{0,1}\nabla n^{cd} + d^c_{0,1}\nabla n^{bd} + d^d_{0,1}\nabla n^{bc} \tag{26}$$

$$\begin{aligned}\mathbf{v}^{bcde} = {} & d^{bcde}_{0,1}\nabla n + d^{bcd}_{0,1}\nabla n^e + d^{bce}_{0,1}\nabla n^d + d^{bde}_{0,1}\nabla n^c + d^{cde}_{0,1}\nabla n^b \\ & + d^{bc}_{0,1}\nabla n^{de} + d^{bd}_{0,1}\nabla n^{ce} + d^{be}_{0,1}\nabla n^{cd} + d^{cd}_{0,1}\nabla n^{be} \\ & + d^{ce}_{0,1}\nabla n^{bd} + d^{de}_{0,1}\nabla n^{bc} + d^b_{0,1}\nabla n^{cde} + d^c_{0,1}\nabla n^{bde} \\ & + d^d_{0,1}\nabla n^{bce} + d^e_{0,1}\nabla n^{bcd}\end{aligned} \tag{27}$$

These terms require the evaluation of a growing number of recursive terms and perturbed density variables. The third-order term reads as

$$\begin{aligned}d^{bcd}_{i,j} = {} & d^{bc}_{i+1,j}n^d + d^b_{i+1,j}n^{cd} + d^c_{i+1,j}n^{bd} + d^d_{i+1,j}n^{bc} \\ & + d^{bc}_{i,j+1}Z^{0,d} \\ & + d^b_{i,j+1}(Z^{0,cd} + Z^{c,d}) \\ & + d^c_{i,j+1}(Z^{0,bd} + Z^{b,d}) \\ & + d^d_{i,j+1}(Z^{0,bc} + Z^{b,c}) \\ & + d_{i,j+1}(Z^{b,cd} + Z^{c,bd} + Z^{d,bc})\end{aligned} \tag{28}$$

and the fourth-order term can be written as

$$
\begin{aligned}
d^{bcde}_{i,j} = \; & d^{bcd}_{i+1,j} n^{e} \\
& + d^{bc}_{i+1,j} n^{de} + d^{bd}_{i+1,j} n^{ce} + d^{be}_{i+1,j} n^{cd} + d^{cd}_{i+1,j} n^{be} \\
& + d^{ce}_{i+1,j} n^{bd} + d^{de}_{i+1,j} n^{bc} \\
& + d^{b}_{i+1,j} n^{cde} + d^{c}_{i+1,j} n^{bde} + d^{d}_{i+1,j} n^{bce} + d^{e}_{i+1,j} n^{bcd} \\
& + d^{bcd}_{i,j+1} Z^{0,e} \\
& + d^{bc}_{i,j+1} (Z^{0,de} + Z^{d,e}) \\
& + d^{bd}_{i,j+1} (Z^{0,ce} + Z^{c,e}) \\
& + d^{be}_{i,j+1} (Z^{0,cd} + Z^{c,d}) \\
& + d^{cd}_{i,j+1} (Z^{0,be} + Z^{b,e}) \\
& + d^{ce}_{i,j+1} (Z^{0,bd} + Z^{b,d}) \\
& + d^{de}_{i,j+1} (Z^{0,bc} + Z^{b,c}) \\
& + d^{b}_{i,j+1} (Z^{0,cde} + Z^{c,de} + Z^{d,ce} + Z^{e,cd}) \\
& + d^{c}_{i,j+1} (Z^{0,bde} + Z^{b,de} + Z^{d,be} + Z^{e,bd}) \\
& + d^{d}_{i,j+1} (Z^{0,bce} + Z^{b,ce} + Z^{c,be} + Z^{e,bc}) \\
& + d^{e}_{i,j+1} (Z^{0,bcd} + Z^{b,cd} + Z^{c,bd} + Z^{d,bc}) \\
& + d_{i,j+1} (Z^{b,cde} + Z^{c,bde} + Z^{d,bce} \\
& + Z^{e,bcd} + Z^{bc,de} + Z^{bd,ce} + Z^{be,cd})
\end{aligned}
\tag{29}
$$

Expressions of even higher-order terms as well as the inclusion of spin density variables can be achieved rather straightforwardly using automatic code generation techniques. Note that in this work and in the above discussion we also omit contributions to accommodate perturbations which modify the overlap of basis functions, such as geometric displacements or magnetic perturbations with London atomic orbitals.

References

(1) Gross, E. K. U.; Kohn, W. *Adv. Quantum Chem.* **1990**, *21*, 255.

(2) Casida, M. Time-dependent density-functional response theory for molecules. In *Recent Advances in Density Functional methods, Part I*; Chong, D. P., Ed.; World Scientific: Singapore, 1995; p 155.

(3) van Leeuwen, R. *Int. J. Mod. Phys. B* **2001**, *50*, 1969.

(4) Marques, M. A. L.; Gross, E. K. U. *Annu. Rev. Phys. Chem.* **2004**, *55*, 427.

(5) Casida, M. E. *J. Mol. Struct. (Theochem)* **2009**, *914*, 3.

(6) Bauernschmitt, R.; Ahlrichs, R. *Chem. Phys. Lett.* **1996**, *256*, 454.

(7) Jamorski, C.; Casida, M. E.; Salahub, D. R. *J. Chem. Phys.* **1996**, *104*, 5134.

(8) Petersilka, M.; Gossmann, U. J.; Gross, E. K. U. *Phys. Rev. Lett.* **1996**, *76*, 1212.

(9) Stratmann, R. E.; Scuseria, G. E.; Frisch, M. J. *J. Chem. Phys.* **1998**, *109*, 8218.

(10) Tozer, D. J.; Handy, N. C. *J. Chem. Phys.* **1998**, *109*, 10180.

(11) Hirata, S.; Head-Gordon, M. *Chem. Phys. Lett.* **1999**, *302*, 375.

(12) Görling, A.; Heinze, H. H.; Ruzankin, S. P.; Staufer, M.; Rösch, N. *J. Chem. Phys.* **1999**, *110*, 2785.

(13) van Gisbergen, S. J. A.; Snijders, J. G.; Baerends, E. J. *Comput. Phys. Commun.* **1999**, *118*, 119.

(14) Rinkevicius, Z.; Tunell, I.; Sałek, P.; Vahtras, O.; Ågren, H. *J. Chem. Phys.* **2003**, *119*, 34.

(15) Burke, K.; Werschnik, J.; Gross, E. K. U. *J. Chem. Phys.* **2005**, *123*, 062206.

(16) Dreuw, A.; Head-Gordon, M. *Chem. Rev.* **2005**, *105*, 4009.

(17) Marques, M.; Ullrich, C. A.; Noguiera, F.; Rubio, A.; Burke, K.; Gross, E. K. U. *Time-dependent density functional theory*; Springer: Heidelberg, Germany, 2006.

(18) Elliott, P.; Furche, F.; Burke, K. Excited states from time-dependent density functional theory. In *Rev. Comput. Chem.*; Lipkowitz, K. B., Cundari, T. R., Eds.; Wiley: Hoboken, NJ, 2009; p 91.

(19) Christiansen, O.; Jørgensen, P.; Hättig, C. *Int. J. Quantum Chem.* **1997**, *68*, 1.

(20) Pulay, P. *Mol. Phys.* **1969**, *17*, 197.

(21) London, F. *J. Phys. Radium* **1937**, *8*, 397.

(22) Ruud, K.; Helgaker, T.; Bak, K. L.; Jørgensen, P.; Jensen, H. J. Aa. *J. Chem. Phys.* **1993**, *99*, 3847.

(23) Helgaker, T.; Wilson, P. J.; Amos, R. D.; Handy, N. C. *J. Chem. Phys.* **2000**, *113*, 2983.

(24) Strange, R.; Manby, F. R.; Knowles, P. J. *Comput. Phys. Commun.* **2001**, *136*, 310.

(25) Sałek, P.; Hesselmann, A. *J. Comput. Chem.* **2007**, *28*, 2569.

(26) Jansík, B.; Sałek, P.; Jonsson, D.; Vahtras, O.; Ågren, H. *J. Chem. Phys.* **2005**, *122*, 054107.

(27) Rall, L. B. *Automatic Differentiation: Techniques and Applications*; Springer: Berlin, 1981; Vol. 120.

(28) Griewank, A.; Walther, A. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, 2nd ed.; SIAM: Philadelphia, PA, 2008; Other Titles in Applied Mathematics 105.

(29) *Advances in Automatic Differentiation*; Bischof, C. H., Bücker, H. M., Hovland, P. D., Naumann, U., Utke, J., Eds.; Springer: Berlin, 2008; Vol. 64.

(30) Steiger, R.; Bischof, C.; Lang, B.; Thiel, W. *Future Gen. Comput. Syst.* **2005**, *21*, 1324.

(31) Thorvaldsen, A. J.; Ruud, K.; Kristensen, K.; Jørgensen, P.; Coriani, S. *J. Chem. Phys.* **2008**, *129*, 214108.

(32) Bast, R.; Jensen, H. J. Aa.; Saue, T. *Int. J. Quantum Chem.* **2009**, *109*, 2091.

(33) Hida, Y.; Li, X.; Bailey, D. Quad-Double/Double-Double Computation Package. http://crd.lbl.gov/dhbailey/mpdist/ (accessed May 2010).

(34) Slater, J. C. *Phys. Rev.* **1951**, *81*, 385.

(35) Vosko, S. J.; Wilk, L.; Nusair, M. *Can. J. Phys.* **1980**, *58*, 1200.

(36) Becke, A. D. *Phys. Rev. A* **1988**, *38*, 3098.

(37) Lee, C.; Yang, W.; Parr, R. G. *Phys. Rev. B* **1988**, *37*, 785.

(38) Miehlich, B.; Savin, A.; Stoll, H.; Preuss, H. *Chem. Phys. Lett.* **1989**, *157*, 200.

(39) Brent, R. P.; Kung, H. T. *J. Assoc. Comput. Machinery* **1978**, *25*, 581.

(40) Development version of DIRAC, a relativistic ab initio electronic structure program, release DIRAC08 (2008),written by: Visscher, L., Jensen, H. J. Aa., Saue, T., with new contributions from Bast, R., Dubillard, S., Dyall, K. G., Ekstrm, U., Eliav, E., Fleig, T., Gomes, A. S. P., Helgaker, T. U., Henriksson, J., Iliaš, M., Jacob, Ch. R., Knecht, S.,

Norman, P., Olsen, J., Pernpointner, M., Ruud, K., Sałek, P., Sikkema J. (see http://dirac.chem.sdu.dk, accessed May 2010).

(41) Lindh, R.; Malmqvist, P.-A.; Gagliardi, L. *Theor. Chem. Acc.* **2001**, *106*, 178.

(42) Lebedev, V. I.; Laikov, D. N. *Doklady Mathematics* **1999**, *59*, 477.

(43) Kendall, R. A.; Dunning, T. H., Jr.; Harrison, R. J. *J. Phys. Chem.* **1992**, *96*, 6769.

(44) Ruud, K.; Thorvaldsen, A. J. *Chirality* **2009**, *21*, S54.

(45) van Faassen, M.; de Boeij, P. L.; van Leeuwen, R.; Berger, J. A.; Snijders, J. G. *Phys. Rev. Lett.* **2002**, *88*, 186401.

(46) Dalskov, E. K.; Oddershede, J.; Bishop, D. M. *J. Chem. Phys.* **1999**, *108*, 2152.

(47) Vignale, G.; Kohn, W. *Phys. Rev. Lett.* **1996**, *77*, 2037.

(48) Bulat, F. A.; Toro-Labbé, A.; Champagne, B.; Kirtman, B.; Yang, W. *J. Chem. Phys.* **2005**, *123*, 014319.

(49) Grüning, M.; Gritsenko, O. V.; Baerends, E. J. *J. Chem. Phys.* **2002**, *116*, 6435.

(50) Karolewski, A.; Armiento, R.; Kümmel, S. *Chem. Phys.* **2009**, *5*, 712.

(51) Ekström, U. XCFun library. http://www.admol.org/xcfun (accessed May 2010).

(52) Libxc library. http://www.tddft.org/programs/octopus/wiki/index.php/Libxc (accessed May 2010).

CT100117S