Algorithmic differentiation and the calculation of forces by quantum Monte Carlo

Sandro Sorella[1,2,a)] and Luca Capriotti[3,b)]
[1]*SISSA, International School for Advanced Studies, 34151, Trieste, Italy*
[2]*DEMOCRITOS Simulation Center, CNR-IOM Istituto Officina dei Materiali, 34151, Trieste, Italy*
[3]*Quantitative Strategies, Investment Banking Division, Credit Suisse Group, Eleven Madison Avenue, New York City, New York 10010-3086, USA*

(Received 3 August 2010; accepted 25 October 2010; published online 20 December 2010)

We describe an efficient algorithm to compute forces in quantum Monte Carlo using adjoint algorithmic differentiation. This allows us to apply the space warp coordinate transformation in differential form, and compute all the $3M$ force components of a system with M atoms with a computational effort comparable with the one to obtain the total energy. Few examples illustrating the method for an electronic system containing several water molecules are presented. With the present technique, the calculation of finite-temperature thermodynamic properties of materials with quantum Monte Carlo will be feasible in the near future. © *2010 American Institute of Physics*. [doi:10.1063/1.3516208]

I. INTRODUCTION

In the last few years, we have seen a remarkable progress in the *ab initio* simulation of realistic electronic systems based on first principles quantum mechanics. Despite the power of density functional theory (DFT), with standard local density approximation (LDA) and generalized gradient approximation (GGA) functionals, much effort has been devoted to schemes that are able to describe more accurately the electronic correlations. This is because several materials—such as high temperature superconductors—are indeed strongly correlated. Furthermore, long-range dispersive forces may be extremely important even in simple and fundamental materials, like water, and are notoriously difficult to describe with standard DFT. A promising many-body approach, alternative to DFT, is the so called quantum Monte Carlo (QMC) method, allowing one to include the electronic correlations by means of a highly accurate many-body wave function (WF), sampled by a statistical method. All the basic ingredients of the electronic correlation are described explicitly within this framework. This is particularly appealing because its computational cost scales rather well with the number of electrons N, with a modest power, e.g., N^3 or N^4. Due to this important property, QMC is very promising for large scale calculations especially when compared with standard post-Hartree–Fock methods. In fact, these methods are also capable to describe rather well the electronic correlations. However, they typically require a larger computational cost: from polynomial in the range between N^4 and N^7, to exponential complexity in full configuration interaction schemes.

Despite the clear advantage of QMC for the accurate electronic simulation of materials containing a large number of atoms, its application has been mainly limited to total energy calculations. In particular, no general method to calculate ionic forces, which remains efficient even for a large number of atoms M, is available so far. Indeed, many-body forces usually lead to cumbersome expressions, whose evaluation can be done at present only by means of complicated and computationally expensive algorithms. For this reason, such calculations have been implemented so far only for particularly simple cases. For instance, it was recently possible[1] to simulate ~100 hydrogen atoms in the low temperature, high pressure phase, with the calculation of the ionic forces based on efficient strategies to work with finite variance expressions.[2,3] Unfortunately, this route cannot be followed in general because it works very efficiently only for light atoms.

On the other hand, an accurate algorithm for the calculation of forces that is in principle efficient also with heavy atoms, and with the use of pseudopotentials, was introduced a long time ago.[4] This method is based on the so called *space warp coordinate transformation* (SWCT), allowing a zero-variance expression—i.e., maximum efficiency in QMC—even for isolated atoms. We believe that, without a highly effective variance reduction scheme in the calculation of forces, such as SWCT, there is no hope to extend the applicability of QMC to structural optimization of complex and correlated materials, or to perform *ab initio* molecular dynamics simulation at finite temperature. However, even when using this promising approach, or others based on the zero-variance principle,[3] standard implementations, e.g., based on finite-difference approximations of the derivatives appearing in the expressions for the ionic forces, are still computationally very expensive—to the point of being unfeasible for a large number of atoms.

In this work, we propose a simple strategy for the efficient calculation of the ionic forces—and generally of any arbitrarily complicated derivative of the QMC total energy—by using *adjoint algorithmic differentiation* (AAD). We will show that this method will allow us to achieve two very important targets:

(1) the numerical implementation of all the energy derivatives will be possible in a straightforward way without

a)Electronic mail: sorella@sissa.it.
b)Electronic mail: luca.capriotti@credit-suisse.com.

any reference to complicated expressions, for instance, the terms shown in Ref. 5, when pseudopotentials are used to remove the effect of core electrons;

(2) more importantly, the calculation of *an arbitrarily large* number of energy derivatives will be possible with a computational effort comparable with the one to compute the energy alone. In other words, once the calculation of the energy is well optimized, by means of AAD, also the one for the energy derivatives will be almost optimal.

The latter property is rather remarkable, and suggests that most of the advanced *ab initio* tools belonging to a rather restrictive approximation of the DFT functional, such as structural optimization and molecular dynamics at finite temperature, can be extended to highly accurate many-body approaches based on QMC, with a computational effort that remains affordable even for a large number of atoms.

Though in most of the examples presented here we have used the standard variational QMC method, the technique we propose in this paper directly applies also to more accurate QMC projection schemes, such as (lattice regularized) diffusion Monte Carlo. However, one has to take into account that in this case there are unsolved technical problems—e.g., infinite variance in the most accurate estimators—that do not depend on the technique we propose, and are outside the main scope of this paper. In order to avoid confusion, we anticipate that we apply the AAD technique only to the specific calculation of the wave function and the local energy (see later for their definitions) required for the variational Monte Carlo (VMC) or diffusion Monte Carlo (DMC) [lattice regularized diffusion Monte Carlo (LRDMC)] evaluation of the average energy. In principle AAD can be applied to the whole algorithm, but we have not exploited this possibility that may be interesting for future applications.

II. QMC WAVE FUNCTION, VMC, AND LRDMC

In this section we begin by describing the WF that we have used in our QMC calculations. In the following we will denote with $\mathbf{r}$ a generic three dimensional electronic position, whereas $x = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\}$ stands for a configuration of all electron positions and spins; the $N_\uparrow$ spin-up electrons are at positions $\mathbf{r}_i$ with $1 \le i \le N_\uparrow$, and the $N_\downarrow$ spin-down electrons are at positions $N_\uparrow + 1 \le i \le N$.

The usual trial WF used in the QMC calculation is the product of an antisymmetric part, and a Jastrow factor. The antisymmetric part is a single Slater determinant, while the Jastrow factor is a bosonic many-body function which accounts for the dynamic correlations in the system. Our Slater determinant is obtained with $N/2$ doubly occupied molecular orbitals $\psi_j(\mathbf{r})$, expanded over L atomic Gaussian orbitals $\phi_j(\mathbf{r})$, centered at atomic positions $\mathbf{R}_j$, as:[6]

$$\psi_i(\mathbf{r}) = \sum_{j=1}^{L} \chi_{ij}\phi_j(\mathbf{r}), \tag{1}$$

where the coefficients $\chi_{i,j}$, as well as the nonlinear coefficients, appearing in the exponents of the Gaussians, can be fully optimized by energy minimization as described later.

The molecular orbitals are initialized from a self-consistent DFT-LDA calculation, in the same atomic basis. The Jastrow factor takes into account the electronic correlation between two electrons, and is conventionally split into an homogeneous interaction J_2, depending on the relative distance between two electrons, and two nonhomogeneous contributions J_3 and J_4, depending on the positions of two electrons and one atom, and two electrons and two atoms, respectively. It also contains an inhomogeneous term $\tilde{J}_2$, describing the electron–ion interaction. This is important to compensate for the change in the one particle density induced by J_2, J_3, and J_4, as well as to satisfy the electron–ion cusp conditions. The homogeneous and inhomogeneous two-body terms J_2 and $\tilde{J}_2$ are defined by the following equations:

$$\tilde{J}_2 = \exp\left[\sum_{ia} -(2Z_a)^{3/4} u(Z_a^{1/4} r_{ia}) + \sum_{ial} g_l^a \chi_{al}^J(\mathbf{r}_i)\right], \tag{2}$$

and

$$J_2 = \exp\left[\sum_{i<j} u(r_{ij})\right], \tag{3}$$

where i, j are indices running over the electrons, and l runs over different single particle orbitals χ_{al}^J centered on the atomic center a; r_{ia} and r_{ij} denote the electron–ion and the electron–electron distances, respectively. The corresponding cusp conditions are fixed by the function $u(r) = F[1 - \exp(-r/F)]/2$ (see, e.g., Ref. 8), whereas g_l^a and F are optimizable variational parameters.

The three- and four-body Jastrow terms $J_3 J_4$ are given by

$$J_3(x)J_4(x) = \exp\left(\sum_{i<j} f(\mathbf{r}_i, \mathbf{r}_j)\right), \tag{4}$$

with $f(\mathbf{r}_i, \mathbf{r}_j)$, being a two-electron coordinate function that can be expanded into the same single-particle basis used for $\tilde{J}_2$:

$$f(\mathbf{r}_i, \mathbf{r}_j) = \sum_{ablm} g_{lm}^{ab} \chi_{al}^J(\mathbf{r}_i)\chi_{bm}^J(\mathbf{r}_j), \tag{5}$$

with g_{lm}^{ab} optimizable parameters. Three-body (electron–ion–electron) correlations are described by the diagonal matrix elements g^{aa}, whereas four-body correlations (electron–ion–electron–ion) are described by the matrix elements with $a \neq b$.

The complete expression of the Jastrow factor $J(x) = J_2(x)\tilde{J}_2(x)J_3(x)J_4(x)$ that we adopt in this work allows us to take into account weak and long-range electron–electron interactions, and it is also extremely effective for suppressing higher energy configurations occurring when electrons are too close.

In order to minimize the energy expectation value corresponding to this WF we have applied the well-established energy minimization schemes[8–10] that we have recently adapted for an efficient optimization of the molecular orbitals of the Slater determinant in presence of the Jastrow factor described above.[6]

In VMC, the energy expectation value of a given correlated WF, depending on a set of variational parameters

$c_i, i = 1, \ldots p$, can be computed with a standard statistical method. The energy depends in turn on the atomic positions $\mathbf{R}_a, a = 1, \ldots M$, so that we indicate formally:

$$E(\{c_i\}, \{\mathbf{R}_a\}) = \frac{\langle \Psi_{\{c_i\},\{\mathbf{R}_a\}} | \hat{H}_{\{\mathbf{R}_a\}} | \Psi_{\{c_i\},\{\mathbf{R}_a\}} \rangle}{\langle \Psi_{\{c_i\},\{\mathbf{R}_a\}} | \Psi_{\{c_i\},\{\mathbf{R}_a\}} \rangle}. \tag{6}$$

In VMC the above energy expectation value is computed statistically by sampling the probability $\Pi(x) \propto \langle x|\Psi\rangle^2$. Analogous and more accurate techniques are also possible within QMC. In this work the LRDMC will be also used. The latter method is a projection technique, filtering out the ground state component of a given variational WF by applying the propagator $e^{-H\tau}$ to Ψ for large imaginary time τ. This propagation is subject to the restriction to modify only the amplitudes of the WF without affecting its phases. In this way one can avoid the so called "fermion sign problem" instability in QMC, with an highly accurate technique, providing a rigorous upper bound of the total energy even in presence of pseudopotentials.[11]

III. SPACE WARP COORDINATE TRANSFORMATION AND ITS DIFFERENTIAL FORM

The main purpose of this section is to describe an efficient method to compute the forces $\mathbf{F}$ acting on each of the M nuclear positions $\{\mathbf{R}_1, \ldots, \mathbf{R}_M\}$, namely,

$$\mathbf{F}(\mathbf{R}_a) = -\nabla_{\mathbf{R}_a} E(\{c_i\}, \{\mathbf{R}_a\}), \tag{7}$$

with a reasonable statistical accuracy. Following Ref. 9, we introduce a finite-difference operator $\Delta/\Delta\mathbf{R}_a$ for the evaluation of the force acting on a given nuclear position $\mathbf{R}_a$,

$$\frac{\Delta}{\Delta\mathbf{R}_a} E = \frac{E(\mathbf{R}_a + \Delta\mathbf{R}_a) - E(\mathbf{R}_a)}{\Delta\mathbf{R}_a} \tag{8}$$

so that

$$\mathbf{F}(\mathbf{R}_a) = -\frac{\Delta}{\Delta\mathbf{R}_a} E + O(\Delta R), \tag{9}$$

where $\Delta\mathbf{R}_a$ is a three dimensional vector. Its length ΔR can be chosen as small as 10^{-6} atomic units, yielding negligible finite-difference errors for the evaluation of the exact energy derivative.

In order to evaluate the energy differences in Eq. (8) it is very convenient to apply the SWCT. This transformation was introduced a long time ago in Ref. 4, for an efficient calculation of the ionic forces within VMC. According to this transformation, as soon as the ions are displaced, also the electronic coordinates $\mathbf{r}$ will be translated in order to mimic the displacement of the charge around the nucleus $\mathbf{R}_a$, namely, $x \to \bar{x}$, with

$$\bar{\mathbf{r}}_i = \mathbf{r}_i + \Delta\mathbf{R}_a \omega_a(\mathbf{r}_i), \tag{10}$$

$$\omega_a(\mathbf{r}) = \frac{F(|\mathbf{r} - \mathbf{R}_a|)}{\sum_{b=1}^{M} F(|\mathbf{r} - \mathbf{R}_b|)}, \tag{11}$$

and $F(r)$ is a function which must decay rapidly; here we used $F(r) = 1/r^4$ as suggested in Ref. 12.

The expectation value of the energy depends on $\Delta\mathbf{R}_a$, because both the Hamiltonian and the WF depend on the nuclear positions. Applying the SWCT to the integral involved in the calculation, the expectation value reads

$$E(\mathbf{R_a} + \Delta\mathbf{R}_a) = \frac{\int d\mathbf{r}^{3N} \tilde{J}_{\Delta\mathbf{R}_a}(x) \Psi^2_{\Delta\mathbf{R}_a}(\bar{x}(x)) E_L^{\Delta\mathbf{R}_a}(\bar{x}(x))}{\int d\mathbf{r}^{3N} \tilde{J}_{\Delta\mathbf{R}_a}(x) \Psi^2_{\Delta\mathbf{R}_a}(\bar{x}(x))}, \tag{12}$$

where $\tilde{J}$ is the Jacobian of the transformation and,

$$E_L^{\Delta\mathbf{R}_a} = \frac{\langle \Psi_{\Delta\mathbf{R}_a} | \hat{H} | x \rangle}{\langle \Psi_{\Delta\mathbf{R}_a} | x \rangle} \tag{13}$$

is the so called *local energy* defined by the wave function $\Psi_{\Delta\mathbf{R}_a}$ on a real space electronic configuration x.

In the following, we define $E_L = E_L^{\Delta\mathbf{R}}$ for $\Delta\mathbf{R}_a = 0$.

The importance of the SWCT in reducing the statistical error in the evaluation of the force is easily understood for the case of an isolated atom a. In this case, the force acting on the atom is obviously zero, but only after the SWCT with $\omega_a = 1$ the integrand in Eq. (12) is independent of $\Delta\mathbf{R}_a$, providing an estimator of the force with zero variance.

Starting from Eq. (12), it is straightforward to derive explicitly a differential expression for the force estimator, which is related to the gradient of the previous quantity with respect to $\Delta\mathbf{R}_a$ in the limit of vanishing displacement,

$$\mathbf{F}_a = -\left\langle \frac{d}{d\mathbf{R}_a} E_L \right\rangle + 2\left(\langle E_L \rangle \left\langle \frac{d}{d\mathbf{R}_a} \log(J^{1/2}\Psi) \right\rangle - \left\langle E_L \frac{d}{d\mathbf{R}_a} \log(J^{1/2}\Psi) \right\rangle \right), \tag{14}$$

where the brackets indicate a Monte Carlo like average over the square modulus of the trial WF, namely, over the probability $\Pi(x)$ introduced in Sec. II. In the calculation of the total derivatives $d/d\mathbf{R}_a$, we have to take into account that the electron coordinates are also implicitly differentiated, according to the SWCT. Then, all the terms above can be written in a closed expression once the partial derivatives of the local energy and of the WF logarithm are known, namely,

$$\frac{d}{d\mathbf{R}_a} E_L = \frac{\partial}{\partial\mathbf{R}_a} E_L + \sum_{i=1}^{N} \omega_a(\mathbf{r}_i) \frac{\partial}{\partial\mathbf{r}_i} E_L, \tag{15}$$

$$\frac{d}{d\mathbf{R}_a} \log(J^{1/2}\Psi) = \frac{\partial}{\partial\mathbf{R}_a} \log(\Psi) + \sum_{i=1}^{N} \left[\omega_a(\mathbf{r}_i) \frac{\partial}{\partial\mathbf{r}_i} \log\Psi + \frac{1}{2} \frac{\partial}{\partial\mathbf{r}_i} \omega_a(\mathbf{r}_i) \right], \tag{16}$$

where the term $(1/2)(\partial/\partial\mathbf{r}_i)\omega_a(\mathbf{r}_i)$ in the square brackets gives the contribution of the Jacobian.

Based on the expressions above we can evaluate the forces in Eq. (14) in three different ways, listed below in increasing order of efficiency:

TABLE I. Efficiency of the various types of evaluation of forces in VMC for a water dimer molecule at experimental equilibrium atomic positions. The computational time to calculate the force components on the oxygen atoms within a given statistical error is inversely proportional to the efficiency. The HFM efficiency is taken for reference equal to one, for a statistical accuracy of 0.001 a.u. Pseudopotential is used only for the oxygen.

Method	HFM	No-SWCT	SWCT
Efficiency	1	165	1360

1. Hellmann–Feynman (HFM). By neglecting all the dependence of Ψ on the atomic position, and without the SWCT:

$$\mathbf{F}_a = -\langle \partial_{\mathbf{R}_a} \hat{H} \rangle.$$

This is the least accurate expression, because without the so called Pulay terms derived in Eq. (16) the force is consistent with the energy derivative only when the WF is the exact ground state. This is also the least efficient way to compute the forces as indicated in Table I.

2. No-SWCT. This is obtained by using the terms (15, 16) in Eq. (14) with $\omega_a(\vec{r}) = 0$. This expression is much more accurate and efficient than the previous one, as it fulfills the so called "zero-variance principle": if the WF Ψ coincides with the exact ground state for arbitrary atomic position $\mathbf{R}_a$, it is easy to realize that the estimator of the forces does not have statistical fluctuations, as the local energy and its derivative are just constant, and independent of the real space configuration x.
3. Differential SWCT. The properties above are clearly fulfilled also in this case. Moreover, the SWCT does not change the mean value of the forces, but affects only their statistical fluctuations. In fact, as mentioned before, the SWCT allows us to obtain a zero-variance property for the forces acting on isolated atoms. As a result, this transformation is extremely important for computing forces between atoms at large distance, as in this limit they can be considered isolated.

The advantage to use the SWCT for a water dimer molecule is illustrated in Table I. It is clear that without the SWCT, or even without the differentiation of the local energy with respect to the atomic position (no-SWCT case), the evaluation of forces with a reasonable statistical error is simply not possible.

As discussed in Sec. IV, all partial derivatives involved in the expressions above, namely, the $6N + 6M$ components, $(\partial/\partial \mathbf{r}_i)E_L$, $(\partial/\partial \mathbf{r}_i)\log\Psi$, $(\partial/\partial \mathbf{R}_a)E_L$, $(\partial/\partial \mathbf{R}_a)\log\Psi$, can be evaluated very efficiently with algorithmic differentiation (AD). This is true also when the WF and the expression for the local energy are extremely cumbersome, e.g., when using pseudopotentials. As a result, the quantities in Eqs. (15) and (16) can be evaluated using just a minor computational effort with roughly $\propto NM$ operations. In particular, one of the most involved contribution to the local energy is the one corresponding to the bare kinetic energy $\hat{K} = -\frac{1}{2}\sum_{i=1}^{N} \Delta_i$. The Hamiltonian $\hat{H}$ in Eq. (13) always contains this term, even in presence of pseudopotentials. In the following, we will discuss how to differentiate the contribution $K = \langle \Psi | \hat{K} | x \rangle / \langle \Psi | x \rangle$ to the local energy for the particularly simple but instructive case of a Slater determinant WF with no Jastrow factor.

IV. ADJOINT ALGORITHMIC DIFFERENTIATION

Algorithmic differentiation[13] is a set of programming techniques for the efficient calculation of the derivatives of functions implemented as computer programs. The main idea underlying these techniques is the fact that any such function—no matter how complicated—can be interpreted as the composition of more elementary functions each of which is in turn a composition of basic arithmetic and intrinsic operations that are easy to differentiate. As a result, it is possible to calculate the derivatives of the outputs of a program with respect to its inputs by applying mechanically the rules of differentiation—and in particular the *chain rule*—to the composition of its constituent functions.

What makes AD particularly attractive, when compared to standard (finite-difference) methods for the calculation of derivatives is its computational efficiency. In fact, AD exploits the information on the calculations performed by the computer code, and the dependencies between its various parts, in order to optimize the calculation. In particular, when one requires the derivatives of a small number of outputs with respect to a large number of inputs, the calculation can be highly optimized by applying the chain rule through the instructions of the program in opposite order with respect to the one of evaluation of the original instructions. This gives rise to the so called AAD.

Even if AD has been an active branch of computer science for several decades, its impact in other research fields has been surprisingly limited until very recently.[14] Only over the past 2 years its tremendous effectiveness in speeding up the calculation of sensitivities, e.g., in Monte Carlo simulations, has been first exploited in computational finance applications.[15] In particular, the potential of AD has been largely left untapped in the field of computational physics where, as we demonstrate in the following, it could move significantly the boundary of what can be studied numerically with the computer power presently available.

Griewank[13] contains a detailed discussion of the computational cost of AAD.[16] Here, we will only recall the main ideas underlying this technique to clarify how it can be beneficial in the implementation of the calculation of the forces in QMC. To this end, we consider a particular computer implemented function $X \to Y$

$$Y = \texttt{FUNCTION}(X) \tag{17}$$

mapping a vector X in $\mathbb{R}^n$ in a vector Y in $\mathbb{R}^m$ through a sequence of two sequential steps

$$X \to U \to V \to Y.$$

Here, each step can be a distinct high-level function, or even an individual instruction in a computer code. A general code is usually implemented by several steps of this type, and, more importantly, the output of a particular instance can be used as an input not only for the next one but generally for all instances occurring later in the algorithm. Generally speaking an algorithm can be viewed as a sequential tree or graph with

connectivity larger than one, where each node has more than one children. However, to keep things as simple as possible, in this "warm up" example we do not consider these more complex cases. The generalization to a realistic computational graph is however straightforward.[16]

The adjoint mode of AD results from propagating the derivatives of the final result with respect to all the intermediate variables—the so called *adjoints*—until the derivatives with respect to the independent variables are formed. Using the standard AD notation, the adjoint $\bar{V}$ of any input variable V of an instance $V \to Y$ is defined as the derivative of a given linear combination of the output $\sum_j \bar{Y}_j Y_j$ with respect to the input V, namely,

$$\bar{V}_k = \sum_{j=1}^{m} \bar{Y}_j \frac{\partial Y_j}{\partial V_k}, \tag{18}$$

where $\bar{Y}$ is a given input vector in $\mathbb{R}^m$. In particular, for each of the intermediate variables, using the chain rule, we get

$$\bar{Y}_j \frac{\partial Y_j}{\partial X_i} = \bar{Y}_j \frac{\partial Y_j}{\partial V_k} \frac{\partial V_k}{\partial U_l} \frac{\partial U_l}{\partial X_i},$$

where repeated indices indicate implicit summations. It is simple to realize that in such a simple case we can use the definition in Eq. (19) to evaluate $\bar{X}$, namely,

$$\bar{Y}_j \frac{\partial Y_j}{\partial X_i} = \bar{V}_k \frac{\partial V_k}{\partial U_l} \frac{\partial U_l}{\partial X_i} = \bar{U}_l \frac{\partial U_l}{\partial X_i} = \bar{X}.$$

In other words, once all adjoint instances have been defined, the bar input of each adjoint instance can be obtained from the output of the previous adjoint instance according to a diagram that follows very straightforwardly the original algorithm in reversed sequential order:

$$\bar{Y} \to \bar{V} \to \bar{U} \to \bar{X}. \tag{19}$$

In this way we obtain $\bar{X}$, i.e., the linear combination of the rows of the Jacobian of the function $X \to Y$, with weights given by the input $\bar{Y}$ (e.g., $1, 0, \ldots, 0$), namely,

$$\bar{X}_i = \sum_{j=1}^{m} \bar{Y}_j \frac{\partial Y_j}{\partial X_i}, \tag{20}$$

with $i = 1, \ldots, n$.

In the adjoint mode, the cost does not increase with the number of inputs, but it is linear in the number of (linear combinations of the) rows of the Jacobian that need to be evaluated independently. In particular, if the full Jacobian is required, one needs to repeat the adjoint calculation m times, setting the vector $\bar{Y}$ equal to each of the elements of the canonical basis in $\mathbb{R}^m$. Furthermore, since the partial derivatives depend on the values of the intermediate variables, one generally first has to compute the original calculation storing the values of all of the intermediate variables such as U and V, before performing the adjoint mode sensitivity calculation.

One particularly important theoretical result is that given a computer code performing some high-level function (17), the execution time of its adjoint counterpart

$$\bar{X} = \texttt{FUNCTION\_B}(X, \bar{Y}) \tag{21}$$

(with suffix `_B` for "backward") calculating the linear combination (20) is bounded by approximately four times the cost of execution of the original one. Thus, one can obtain the sensitivity of a single output, or of a linear combination of outputs, to an unlimited number of inputs for little more work than the original computation.

The propagation of the adjoints, being mechanical in nature can be automated. Indeed, several AD tools[14] are available that, given a function of the form (17), generate the adjoint function (21). While the application of such automatic AD tools to large inhomogeneous simulation software is challenging, the principles of AD can be used as a programming paradigm of any algorithm. This is especially useful for the most common situations where simulation codes use a variety of libraries written in different languages, possibly linked dynamically. However, automatic tools are of great utility to generate the adjoint of self-contained functions and subroutines thus effectively reducing the development time of adjoint implementations.

A detailed tutorial on the programming techniques that are useful for adjoint implementations is beyond the scope of this paper. However, when hand-coding the adjoint counterpart of a set of instructions in a general algorithm it is often enough to keep in mind just a few practical recipes, for instance:

(i) As previously mentioned, each intermediate differentiable variable U can be used not only by the subsequent instance but also by several others occurring later in the program. As a result, the adjoint of U has in general several contributions, one for each instruction of the original function in which U was on the right hand side of the equal sign (assignment operator). Hence, by exploiting the linearity of differential operators, it is in general easier to program according to a syntactic paradigm in which adjoints are always updated so that the adjoint of an instruction of the form

$$V = V(U)$$

reads

$$\bar{U}_i = \bar{U}_i + \frac{\partial V_k(U)}{\partial U_i} \bar{V}_k.$$

Clearly, this implies that the adjoints have to be appropriately initialized as discussed in the following paragraphs. In particular, to cope with input variables that are changed by the algorithm (see next point) it is generally best to initialize to zero the adjoint of a given variable in correspondence of the instruction in which it picks its first contribution (i.e., right before the adjoint corresponding to the last instruction of the original code in which the variable was to the right of the assignment operator). For instance, the adjoint of the following sequence of instructions

$$y = F(x)$$
$$z = H(x, y)$$
$$x = G(z)$$

can be written as:

$$\bar{z} = 0$$
$$\bar{z} = \bar{z} + \frac{\partial G(z)}{\partial z}\bar{x}$$
$$\bar{y} = 0$$
$$\bar{x} = 0$$
$$\bar{x} = \bar{x} + \frac{\partial H(x,y)}{\partial x}\bar{z}$$
$$\bar{y} = \bar{y} + \frac{\partial H(x,y)}{\partial y}\bar{z}$$
$$\bar{x} = \bar{x} + \frac{\partial F(x)}{\partial x}\bar{y}.$$

Note in particular that the symbol x represents the input and the output of the algorithm. As a result, also $\bar{x}$ represents both the input and the output of the adjoint algorithm and it is crucial to reinitialize to zero $\bar{x}$ before it picks its first contribution from an adjoint statement, i.e., the one associated with the instruction $z = H(x, y)$. As explained in detail in the following example, the algorithm could be more easily understood by replacing the last statement by another independent output variable $u = G(z)$, and following the straightforward derivation of the adjoint algorithm that has for input $\bar{u}$ and output $\bar{x}$ (see also the example below). One can easily derive that the resulting algorithm coincides with the one above, namely, the same input provides the same output.

(ii) In some situations the input U of a function $V = V(U)$ is modified by the function itself. This situation is easily analyzed by introducing an auxiliary variable U' representing the value of the input after the function evaluation. As a result, the original function can be thought of the form $(V, U') = (V(U), U'(U))$, where $V(U)$ and $U'(U)$ do not mutate their inputs, in combination with the assignment $U = U'$, overwriting the original input U. The adjoint of this pair of instructions clearly reads

$$\bar{U}'_i = 0$$
$$\bar{U}'_i = \bar{U}'_i + \bar{U}_i,$$

where we have used the fact that the auxiliary variable U' is not used elsewhere (so $\bar{U}'_i$ does not have any previous contribution), and

$$\bar{U}_i = 0$$
$$\bar{U}_i = \bar{U}_i + \frac{\partial V_k(U)}{\partial U_i}\bar{V}_k + \frac{\partial U'_l(U)}{\partial U_i}\bar{U}'_l,$$

where, again, we have used the fact that also the original input U is not used after the instruction $V = V(U)$, as it gets overwritten. One can therefore eliminate altogether the adjoint of the auxiliary variable $\bar{U}'$ and simply write

$$\bar{U}_i = \frac{\partial V_k(U)}{\partial U_i}\bar{V}_k + \frac{\partial U'_l(U)}{\partial U_i}\bar{U}_l.$$

Very common examples of this situation are given by increments of the form

$$U_i = a\,U_i + b$$

with a and b constant with respect to U. According to the recipe above, the adjoint counterpart of this instruction simply reads

$$\bar{U}_i = a\,\bar{U}_i.$$

These situations are common in iterative loops where a number of variables are typically updated at each iteration.

In order to better illustrate these ideas, here we consider the calculation of the kinetic energy and of its adjoint counterpart, where for simplicity we consider only one spin component (maximum polarized case), as the calculation of both spin up and spin down contributions can be obtained just by summing them. Given the position of the electrons x and of the ions R, the calculation of the kinetic energy can be performed according to the following steps, as derived in Appendix A:

(1) Calculate $A_{i,j} = \psi_i(\mathbf{r}_j)$ and $B_{i,j} = \Delta_j \psi_i(\mathbf{r}_j)$ according to the definition of the molecular orbitals in Eq. (1).
(2) Calculate $A^{-1}_{i,j}$ by matrix inversion.
(3) Calculate the kinetic energy as

$$K = -\frac{1}{2}\sum_{i,j} A^{-1}_{i,j} B_{j,i}. \tag{22}$$

The corresponding adjoint algorithm can be constructed by associating to each of the steps above its adjoint counterpart according to the correspondence given by Eqs. (17), (20), and (21). As a result, as also illustrated schematically in Fig. 1, the adjoint algorithm for the derivatives of the kinetic energy with respect to the positions of the electrons and ions,

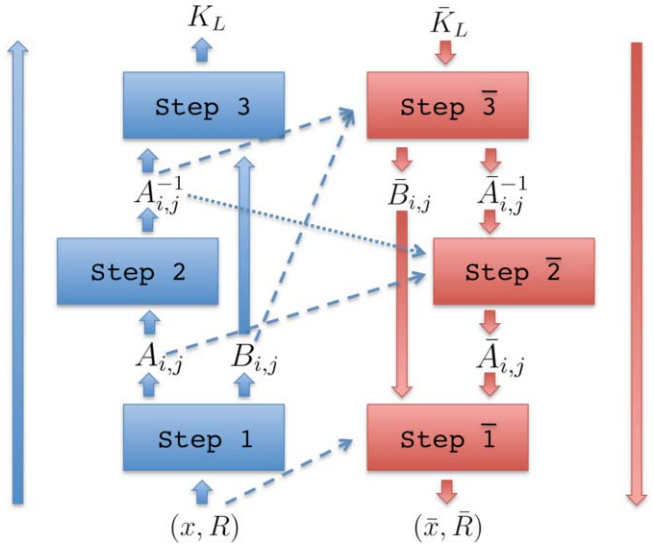

FIG. 1. Schematic representation of the adjoint algorithm for the calculation of the local kinetic energy. Note that the inverse of the matrix A can be passed directly as an input (dotted arrow) of step $\bar{2}$ thus avoiding to repeat the matrix inversion.

consists of steps 1–3 above, and their adjoint counterparts executed in reverse order, namely,

$\bar{3}$. Set $\bar{K}=1$, and evaluate the adjoint of the function $(A^{-1}_{i,j}, B_{j,i}) \to K$ defined in step 3. This is a function of the form $(A^{-1}_{i,j}, B_{i,j}, \bar{K}) \to (\bar{A}^{-1}_{i,j}, \bar{B}_{i,j})$ with $\bar{A}^{-1}_{i,j} = -\frac{1}{2}\bar{K}B_{j,i}$ and $\bar{B}_{i,j} = -\frac{1}{2}\bar{K}A^{-1}_{j,i}$.

$\bar{2}$. Evaluate the adjoint of the function $A_{i,j} \to A^{-1}_{i,j}$ (step 2), namely, $(A_{i,j}, \bar{A}^{-1}_{i,j}) \to \bar{A}_{i,j}$ with $\bar{A} = -(A^{-1})^T \bar{A}^{-1} (A^{-1})^T$ (see App. B).

$\bar{1}$. Evaluate the adjoint of the function $(x, R) \to (A_{i,j}, B_{i,j})$ (step 1), namely, $(x, R, \bar{A}_{i,j}, \bar{B}_{i,j}) \to (\bar{x}, \bar{R})$ with

$$\bar{x}_j = \frac{\partial K}{\partial \mathbf{r}_j} = \sum_i \bar{A}_{i,j}\partial_{\mathbf{r}_j}\psi_i(\mathbf{r}_j) + \bar{B}_{i,j}\partial_{\mathbf{r}_j}\Delta_j\psi_i(\mathbf{r}_j),$$

$$\bar{R}_a = \frac{\partial K}{\partial \mathbf{R}_a} = \sum_{i,j}\left[\bar{A}_{i,j}\partial_{\mathbf{R}_a}\psi_i(\mathbf{r}_j) + \bar{B}_{i,j}\partial_{\mathbf{R}_a}\Delta_j\psi_i(\mathbf{r}_j)\right]$$

$$= \sum_{i,j,k}\chi_{i,k}\delta_{\mathbf{R}_k,\mathbf{R}_a}\left[\bar{A}_{i,j}\partial_{\mathbf{R}_k}\phi_k(\mathbf{r}_j) + \bar{B}_{i,j}\partial_{\mathbf{R}_k}\Delta_j\phi_k(\mathbf{r}_j)\right],$$

where in the latter equality we have expanded the orbitals in terms of atomic orbitals, by means of Eq. (1).

Notice that in the last expression it is the presence of the Kronecker delta, that allows the computation of all the derivatives with respect to the atomic positions $\mathbf{R}_a$ in $\simeq 2N^2L$ operations, namely, the same amount of operations used in the forward step. Indeed by summing only once over the three indices i, j, k in the above expression all the force components acting on all the atoms are obtained.

In AAD this is not accidental, and the structure of the algorithm is automatically optimized for computing several derivatives at the cheapest computational cost.

By applying the chain rule it is immediate to see that $\bar{x}$ and $\bar{R}$, computed according to the steps above, are the derivatives of the kinetic energy with respect to the position of the electrons and the ions, respectively. It is also easy to realize that—as expected according to general results on the computational complexity of adjoint algorithms[13] quoted above—the number of operations involved in each adjoint step is a small constant times the number of operations of the original step, namely, (considering only multiplications) $2N^2L$ vs N^2L, $2N^3$ vs N^3, and $2N^2$ vs N^2 for steps $\bar{1}$ vs 1, $\bar{2}$ vs 2, and $\bar{3}$ vs 3, respectively.

As also anticipated, the propagations of the adjoints (steps $\bar{3}-\bar{1}$) can be performed only after the calculation of the kinetic energy has been completed (steps 1−3) and some of the intermediate results (e.g., the matrices A, B, and A^{-1}) have been computed and stored. This is the reason why, in general, the adjoint of a given function generally contains a *forward sweep*, reproducing the steps of the original function, plus a *backward sweep*, propagating the adjoints. This construction can be clearly applied recursively for each of the steps involved in the calculation.

It is worth noting that each adjoint step, taken in isolation, contains in turn a forward sweep, recovering the information computed in the original step that is necessary for the propagation of the adjoints. However, this can be clearly avoided by storing such information at the time it is first computed in the original step. Strictly speaking, this is necessary to ensure that the computational cost of the overall algorithm remains within the expected bounds. However, there is clearly a tradeoff between the time necessary to store and retrieve this information and the time to recalculate it from scratch, so that in practice it is often enough to store in the main forward sweep only the results of relatively expensive computations. In the example above for instance, significant savings can be obtained by storing the inverse of the matrix A at the output of step 2 and passing it as an input of Step $\bar{2}$ (see Fig. 1).

The main complication in the algorithm above is the implementation of the adjoint of the Laplacian of the WF in step $\bar{1}$. However, the calculation of the Laplacian is a good example of an instance that can be represented by a self-contained, albeit complex, computer function, for which several automatic differentiation tools are available. In particular, in order to complete step $\bar{1}$, it is enough to define the adjoint functions of the calculation of the Laplacian $\vec{r} \to \Delta\phi_j(\vec{r})$, for a set of explicit functions $\{\phi_j\}$ (e.g., Gaussians). The adjoints are then computed by means of the corresponding gradient of the Laplacian, namely, $\bar{\psi} \to \bar{\vec{r}}$ where $\bar{\vec{r}} = \bar{\vec{r}} + \bar{\psi}\nabla_{\vec{r}}\Delta\phi_j(\vec{r})$. For this application we have used TAPENADE, developed at INRIA by Hascoët and collaborators.[17]

V. RESULTS

After implementing the adjoint counterpart of the two main instances corresponding to the evaluation of the log WF and the local energy, we have computed the exact energy derivatives and compared with the straightforward finite-difference evaluation, finding perfect agreement within numerical accuracy. However, the finite-difference method presents a well known bottleneck: in order to evaluate the $3M$ energy derivatives, one has to evaluate the local energy and the log WF at least $3M$ times more. Since the computation of such quantities is the most relevant part in QMC, with a computational effort scaling as N^3, we end up with a very inefficient algorithm for large number of atoms. As shown in Fig. 2, this slowing down can be completely removed by using AAD, as the cost to compute all the force

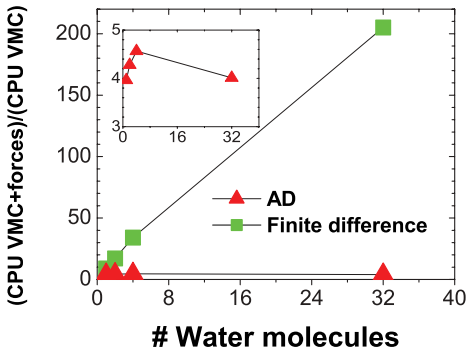

FIG. 2. Ratio of CPU time required to compute energies and all force components referenced to the one required for the simple energy calculation within VMC. The calculations refer to 1, 2, 4, and 32 water molecules. The inset is an expansion of the lower part of the plot.

components in a system containing several water molecules, remains approximately four times larger than the cost to compute only the total energy. This factor 4 is a very small cost, if we consider that the main adjoint instance has to be evaluated twice, one for the local energy and the other for the WF logarithm, and that, on the other hand, VMC is the fastest method in QMC. For instance, we can evaluate forces within LRDMC with only a small overhead, as the cost to generate a new independent configuration within LRDMC is about ten times larger than VMC, and therefore, for this more accurate method, the cost to compute all force components will be essentially negligible. Analogous consideration holds during an energy optimization. We have to consider that in this case AAD can be used to compute not only the force components, but also all the energy derivatives with respect to all variational parameters $\{c_i\}$ of the WF, essentially at the same computational cost, even when the number p of variational parameters is extremely large.

Though we have not implemented AAD for this general task, we expect a further speed up (and simplification) of the code, once AAD will be fully implemented for all possible energy derivatives. We believe this will become common practice for future quantum Monte Carlo packages. At present, in order to have consistent forces within VMC, all variational parameters have to be optimized,[18] and to this purpose we have used the standard way to compute energy derivatives.

We have applied the efficient evaluation of the forces for the structural optimization of the water monomer. We have used energy-consistent pseudopotentials[19] only for the oxygen atom. In the calculation we have adopted a huge basis set to avoid basis superposition errors. The molecular orbitals are expanded in a primitive basis containing 24s22p10d6f1g on the oxygen and 6s5p1d on the hydrogen atom. The exponents of the Gaussians are optimized by minimizing the energy of a self-consistent DFT calculation within the LDA approximation.[7] The accuracy in the total DFT energy is well below 1 mHa for the water dimer, implying that we are essentially working with an almost complete basis set. For the Jastrow factor we have also used a quite large basis, to achieve similar accuracy in the total energy, within a VMC calculation on a WF obtained by optimizing the Jastrow over the LDA Slater determinant. The final optimized basis for the Jastrow contains a contracted basis 6s5p2d/3s3p1d on the oxygen and an uncontracted 1s1p basis on the hydrogen atom.

In the following we describe the first application of this method for optimizing the structure of simple water compounds. The variational parameters of the WF—molecular orbitals and Jastrow factor—are optimized, by energy minimization, with the method described in Ref. 6. At each step of optimization, we compute the ionic forces by AAD, and we employ a standard steepest descent move of the ions $\mathbf{R}_a \rightarrow \mathbf{R}'_a$:

$$\mathbf{R}'_a = \mathbf{R}_a + \Delta\tau \mathbf{F}_a, \tag{23}$$

where $\Delta\tau = 1/2$ a.u. After several hundred iterations both the variational parameters and the atomic positions fluctuate around average values, and we use the last few hundred iterations to evaluate the error bars and the mean value of the atomic positions, as illustrated in Fig. 3.

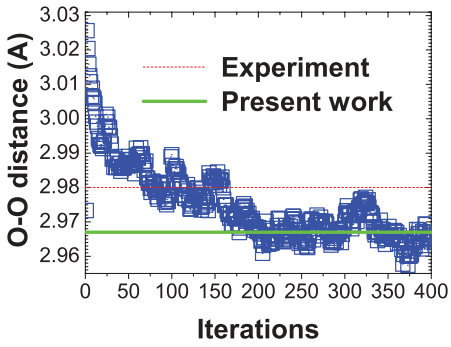

FIG. 3. Oxygen–oxygen distance as a function of the number of iterations for determining the equilibrium zero-temperature structure of the water dimer. All the 18 atomic coordinates, as well as about 1000 variational parameters of the electronic many-body WF are fully optimized with an iterative scheme (Refs. 6 and 8).

In Table II we show the optimized structure of the water monomer. As it is clearly evident our final atomic positions are almost indistinguishable from the experimental ones. Generally speaking our calculation appears more accurate than simple mean field DFT methods, and comparable with state of the art quantum chemistry techniques, such as CCSD(T). The accuracy of the VMC method has been also confirmed recently in another context.[20]

In the dimer structure the situation is slightly different. As shown in Table III, the oxygen–oxygen distance is in quite good agreement with experiments, whereas the *OHO* angle is overestimated by few degrees. Probably in this case the quantum corrections should affect the hydrogen position between the two oxygens, because the dimer bond is very weak. Indeed we have also checked that, with the more accurate LRDMC calculation, the equilibrium structure obtained by the VMC method remains stable as all the force components are well below 10^{-3} a.u. On the other hand LRDMC increases the binding of the dimer by about 1 kCal/mol, showing that, from the energetic point of view, the LRDMC calculation may be important, as also confirmed in previous studies.[6,21] All the above calculations can be done with a relatively small computational effort (few hours in a 32 processor parallel computer), and therefore the same type of calculation, with the same level of accuracy, can be extended to much larger systems containing several atoms with modern supercomputers.

Stimulated by the above success we have tested the finite-temperature molecular dynamics simulation introduced some time ago,[1] using 4 water molecules in a cubic box with 4.93 Å side length, mimicking the density of liquid water at ambient conditions. Since we are interested in static equilibrium properties we have used for the oxygen the same mass of hydrogen. Though the system is very small we have been

TABLE II. VMC optimized structure of the water monomer.

	Exp	VMC	LDA[a]	BLYP[a]	BP[a]	CCSD(T)[b]
$d_{OH}(A)$	0.957[c]	0.954(1)	0.973	0.973	0.974	0.95829
$\angle HOH$ (deg)	104.5[d]	104.61(10)	104.4	104.6	104.1	104.454

[a]From Ref. 23
[b]From Ref. 24
[c]From Ref. 25
[d]From Ref. 26

TABLE III. VMC optimized structure of the water dimer. The LRDMC calculation was done only for the binding energy, by projecting the VMC WF.

	Exp	VMC	LRDMC	LDA[a]	BLYP[a]	PBE[b]	CCSD(T)[c]
$d_{OO}(A)$	2.98[d]	2.969(2)	–	2.70	2.95	2.89	2.9089
$\angle OHO$ (deg)	174[e]	177.6(3)	–	169	173	172	-
Binding (kCal/mol)	5.0(7)[f]	3.84(14)	4.76(6)	8.8	4.3	5.55	=

[a]From Ref. 23
[b]From Ref. 27
[c]From Ref. 28
[d]From Ref. 29
[e]From Ref. 30
[f]From Ref. 31

able to perform several thousands steps. For each step all variational parameters are optimized using a given number n of stochastic reconfiguration (SR) optimizations.[8] For the first 18 000 steps we used $n = 1$ and a time integration step Δt for the MD ranging from 20 to 40 a.u. Several iterations were possible because in QMC we can decide to work with a relatively small number of samples to accumulate statistics for the energy derivatives and the forces. In these conditions the forces are rather noisy but the molecular dynamics with noise correction[1] allows us to have sensible results, at the price to have an overdamped dynamics. However, as it is shown in Fig. 4, it is difficult to remain within the Born–Oppenheimer energy surface, because at selected times, we have fully optimized the wave function using further 200 SR iterations, and found 6 mHa difference between the energy on fly and the optimized energy. In order to overcome this bias in the dynamics, in the final part of the MD simulation, we have used $n = 10$ (and $\Delta t = 40$ a.u.) and found that we remain sufficiently close to the Born–Oppenheimer energy surface. This very preliminary application is clearly limited by the too small number of water molecules considered in the simulation, and therefore does not allow us to determine the equilibrium properties of liquid water. Nevertheless, we believe that this result is rather encouraging because it shows that all the possible sources of errors in the MD driven by QMC forces, can be controlled in a rather straightforward way.

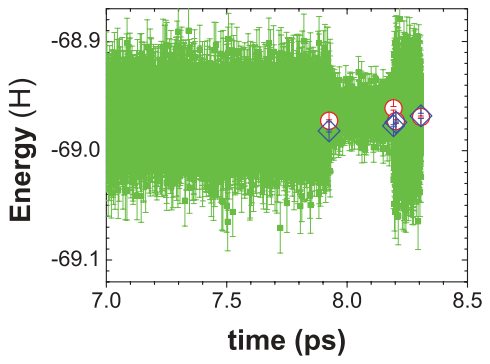

FIG. 4. Average internal energy as a function of the simulation time, for the molecular dynamics with QMC forces[1] and four water molecules in a cubic box. Empty dots indicate VMC energy values corresponding to the time evolved WF with much smaller error bars (<0.4 mHa). Empty squares are obtained after optimizing the WF at fixed atomic positions, starting from the previous initial state and with quite accurate statistical accuracy (<0.4 mHa).

VI. CONCLUSIONS

In this work we have shown that the calculation of all the force components in an electronic system containing several atoms, can be done very efficiently using adjoint algorithmic differentiation (AAD). In particular it is possible to employ the very efficient space warp coordinate transformation (SWCT) in differential form in a straightforward and simple way, even when pseudopotentials and/or complicated many-body wave functions are used. More importantly, we have shown that, using AAD, one can compute all these force components, and in principle all the energy derivatives with respect to any variational parameter contained in the many-body wave function, in about four times the cost to compute the expectation value of the energy.

So far, for large number of atoms, the use of quantum Monte Carlo methods have been generally limited to total energy calculations. We believe that our work opens the way for new and more accurate tools for *ab initio* electronic simulation based on quantum Monte Carlo. In particular we have shown that it is possible to perform an *ab initio* molecular dynamics simulation for several picoseconds, in a system containing four water molecules. Since the cost of a variational Monte Carlo calculation with fixed statistical accuracy in the energy per atom (total energy) increases with the number of atoms as M^2 (M^4) the simulation of about 32 water molecules should be possible with less than 10^5 (10^7) CPU hours, a figure that is nowadays possible (at the limits of present possibilities) with modern massively parallel supercomputers. It is not known at present if it is sufficient to target a fixed statistical error in the energy per atom in order to obtain well-converged thermodynamic extensive quantities. Otherwise a computationally more expensive calculation with a statistical error on the total energy of the order of kT is necessary, as in the penalty method.[22]

In the example we have presented, we have also seen that the accuracy of variational Monte Carlo in determining the equilibrium structure of the water monomer and the water dimer is rather remarkable and comparable to post-Hartree–Fock methods, requiring much more computer resources for large number of atoms. Therefore we believe that, in view of the efficiency in the evaluation of forces obtained by AAD, realistic and very accurate *ab initio* simulation based on quantum Monte Carlo will be within reach in the near future.

ACKNOWLEDGMENTS

This work was partially supported by COFIN2007 and CNR.

APPENDIX A: CALCULATION OF THE LAPLACIAN

In order to compute the Laplacian K of a Slater determinant WF:

$$\langle x|S\rangle = \det A, \tag{A1}$$

where A is the $N \times N$ matrix defined by

$$A_{i,j} = \psi_i(\mathbf{r_j}), \tag{A2}$$

and the molecular orbitals ψ_i are defined in Eq. (1) and the spin index is omitted for simplicity.

We notice that if we change the electron position of the kth electron the matrix A change only by a single column:

$$A'_{i,j} = A_{i,j} + \delta_{j,k}[\psi_i(\mathbf{r}'_k) - A_{i,k}]. \tag{A3}$$

In this way we can single out the dependence on $\mathbf{r}'_k$ by evaluating explicitly the ratio of the two WF's:

$$\frac{\det A'}{\det A} = \det A^{-1}A' = \sum_j A^{-1}_{k,j}\psi_j(\mathbf{r}'_k). \tag{A4}$$

The evaluation of the Laplacian with respect to the kth electrons easily follows by linearity of differential operations such as the Laplacian, so that we finally arrive to Eq. (22), by summing over all the electrons.

APPENDIX B: CALCULATION OF THE ADJOINT OF AN INVERSE MATRIX OPERATION

In this appendix we derive the adjoint instance of the calculation of the inverse A^{-1}_{ij} of an input $N \times N$ square matrix $A_{i,j}$. To this purpose we notice that an arbitrary linear combination of the output (the inverse matrix) can be written as:

$$\sum_{i,j} \bar{A}^{-1}_{i,j} A^{-1}_{i,j} = \mathrm{Tr}[(\bar{A}^{-1})^T A^{-1}], \tag{B1}$$

where the subscript T indicate the transpose of the corresponding matrix, and Tr the conventional matrix trace (sum of the diagonal elements). As explained in Sec. IV by differentiating the above equation with respect to an arbitrary variation of the input we obtain the adjoint instance. To this purpose we denote the differential change of the input A with the matrix DA, so that the corresponding variation can be conveniently written as:

$$(A + DA) = A(I + A^{-1}DA)$$

and therefore:

$$(A + DA)^{-1} = (I - A^{-1}DA)A^{-1} + O(|DA|^2).$$

Thus, by simply substituting the above relation in Eq. (B1), and by using the cyclic invariance of the trace, we obtain

$$d\mathrm{Tr}[(\bar{A}^{-1})^T A^{-1}] = -\mathrm{Tr}[A^{-1}(\bar{A}^{-1})^T)A^{-1}DA], \tag{B2}$$

which implies that the adjoint matrix $\bar{A}$, after this instance, is updated as follows:

$$\begin{aligned}\bar{A} &= \bar{A} + \frac{d\mathrm{Tr}[(\bar{A}^{-1})^T A^{-1}]}{dA} \\ &= \bar{A} - [A^{-1}(\bar{A}^{-1})^T)A^{-1}]^T \\ &= \bar{A} - (A^{-1})^T \bar{A}^{-1}(A^{-1})^T,\end{aligned} \tag{B3}$$

which concludes this appendix.

[1] C. Attaccalite and S. Sorella, Phys. Rev. Lett. **100**, 114501 (2008).
[2] R. Assaraf and M. Caffarel, J. Chem. Phys. **113**, 4028 (2000).
[3] R. Assaraf and M. Caffarel, J. Chem. Phys. **119**, 10536 (2003).
[4] C. J. Umrigar, Int. J. Quant. Chem. Symp. **23**, 217 (1989).
[5] A. Badinski and R. J. Needs, Phys. Rev. E **76**, 036707 (2007).
[6] M. Marchi, S. Azadi, M. Casula, and S. Sorella, J. Chem. Phys. **131**, 154116 (2009).
[7] S. Azadi, C. Cavazzoni, and S. Sorella, Phys. Rev. B. **82**, 125112 (2010).
[8] S. Sorella, M. Casula, and D. Rocca, J. Chem. Phys. **127**, 014105 (2007).
[9] M. Casula, C. Attaccalite, and S. Sorella, J. Chem. Phys. **121**, 7110 (2004).
[10] C. J. Umrigar, J. Toulouse, C. Filippi, S. Sorella, and R. G. Hennig, Phys. Rev. Lett. **98**, 110201 (2006).
[11] M. Casula, C. Filippi, and S. Sorella, Phys. Rev. Lett. **95**, 100201 (2005); see also M. Casula, S. Moroni, S. Sorella, and C. Filippi, J. Chem. Phys. **132**, 154113 (2010).
[12] C. Filippi and C. J. Umrigar, Phys. Rev. B **61**, R16291 (2000).
[13] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation* (Frontiers in Applied Mathematics, Philadelphia, 2000).
[14] A good source of information can be found at http://www.autodiff.org.
[15] See for instance: L. Capriotti and M. Giles, Risk Mag. **23**, 79 (2010), and references therein.
[16] A tutorial can be also found in L. Capriotti, J. Comp. Finance (in press); available at http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1619626.
[17] See: http://www-sop.inria.fr/tropics/tapenade.html.
[18] See, e.g., M. Casalegno, M. Mella, and M. Rappe, J. Chem. Phys. **118**, 7193 (2003).
[19] M. Burkatzki, C. Filippi, and M. Dolg, J. Chem. Phys. **126**, 234105 (2007).
[20] O. Valsson and C. Filippi, J. Chem. Theory Comput. **6**, 1275 (2010).
[21] G. Gurtubay and R. J. Needs, J. Chem. Phys. **127**, 124306 (2007).
[22] D. M. Ceperley and M. Dewing, J. Chem. Phys. **110**, 9812 (1999).
[23] M. Sprik, J. Hutter, and M. Parrinello, J. Chem. Phys. **105**, 1142 (1996).
[24] D. Feller and K. A. Peterson, J. Chem. Phys. **131**, 154306 (2009).
[25] W. S. Benedict, N. Gailer, and E. K. Plyler, J. Chem. Phys. **24**, 1139 (1956).
[26] A. A. Clough, Y. Beers, G. P. Klein, and L. S. Rothman, J. Chem. Phys. **59**, 2254 (1973).
[27] P. Sit and N. Marzari, J. Chem. Phys. **122**, 204510 (2005).
[28] G. S. Tschumper, M. L. Leininger, B. C. Hoffman, E. F. Valeev, H. F. Schaefer, and M. Quack, J. Chem. Phys. **116**, 690 (2002).
[29] R. Bentwood, A. Barnes, and W. Orville Thomas, J. Mol. Spectrosc. **84**, 391 (1980).
[30] K. Kuchitsu and Y. Morino, Bull. Chem. Soc. Jpn. **38**, 805 (1965); L. A. Curtiss, C. L. Frurip, and M. J. Blander, J. Chem. Phys. **71**, 2703 (1979).
[31] E. Mas, R. Bukowski, K. Szalewicz, G. Groenenboom, P. Wormer, and A. van der Avoird, J. Chem. Phys. **113**, 6687 (2000).