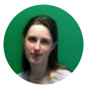
A high performance grid-based algorithm for computing QTAIM properties

ARTICLE *in* CHEMICAL PHYSICS LETTERS · APRIL 2009

Impact Factor: 1.9 · DOI: 10.1016/j.cplett.2009.02.081


Carine Michel
Ecole normale supérieure de Lyon
40 PUBLICATIONS **531** CITATIONS

Andreas W Götz
University of California, San Diego
38 PUBLICATIONS **1,013** CITATIONS

Carles Bo
ICIQ Institute of Chemical Research of Catal…
153 PUBLICATIONS **4,099** CITATIONS

Chemical Physics Letters 472 (2009) 149–152

Contents lists available at ScienceDirect

Chemical Physics Letters

journal homepage: www.elsevier.com/locate/cplett

A high performance grid-based algorithm for computing QTAIM properties

Juan I. Rodríguez [a,*], Richard F.W. Bader [a], Paul W. Ayers [a], Carine Michel [b], Andreas W. Götz [b], Carles Bo [c]

[a] Department of Chemistry, McMaster University, Hamilton, Ontario, Canada L8S4M1
[b] Theoretische Chemie, Vrije Universiteit Amsterdam, De Boelelaan 1083, 1081 HV Amsterdam, The Netherlands
[c] Institut Catalá d'Investigació Química, Av. Paisos Catalans 16, 43007 Tarragona, Spain

ARTICLE INFO

Article history:
Received 28 October 2008
In final form 27 February 2009
Available online 4 March 2009

ABSTRACT

An improved version of our method for computing QTAIM [J.I. Rodríguez, A.M. Köster, P.W. Ayers, A. Santos-Valle, A. Vela, G. Merino, J. Comput. Chem. (2009), in press, doi:10.1002/jcc.21134] is presented. Vectorization and parallelization of the previous algorithm, together with molecular symmetry, make the present algorithm as much as two orders of magnitude faster than our original method. The present method scales linearly with both system size and the number of processors. The performance of the method is demonstrated by computing the QTAIM atomic properties of a series of carbon nanotubes. Our results show that the CPU time for a QTAIM property calculation is comparable to that of a SCF-single point calculation. The accuracy of the original method is also improved.

1. Introduction

Atomic properties are useful for interpreting the results of electronic structure calculations on molecules and materials. (For example, atomic charges and multipole moments provide valuable information about intermolecular forces.) An appropriate definition of the properties of atoms inside molecules, based on quantum mechanical principles, is provided by the quantum theory of atoms in molecules (QTAIM), the physics of an open system [1].

The importance of QTAIM properties has been widely recognized in the quantum chemistry and solid state physics communities [2–4]. QTAIM has been extensively exploited in fields ranging from solid state physics and X-ray crystallography to drug design and biochemistry [4]. However, it is not always feasible to apply QTAIM to large systems due to its computational cost. (QTAIM property calculations are often several orders of magnitude slower than a SCF-single point calculation.) Researchers developing more efficient QTAIM computer algorithms have explored a variety of intricate numerical schemes [5–19]. The basic concepts of grid-based algorithms date back to the early days of QTAIM [13,14], but these methods have received renewed attention due to their easy numerical implementation and relatively high efficiency [15–19]. Although recent grid-based methods seem less suitable for high accuracy calculations than the standard approaches, they can be about 30 times faster than the standard approaches [18,19]. Our goal is to reduce the computational cost of a QTAIM calculation so that it is comparable to an SCF-single point calculation.

In this Letter, we present an improved version of the grid-based method we recently introduced [19]. This improved algorithm was implemented in the Amsterdam Density Functional program (ADF) [20–22]. There are four main improvements in the new implementation: (I) Accuracy of the integration method. (II) Vectorization of the procedure for constructing the steepest ascent paths of the density. (III) The use of the molecular symmetry. (IV) Parallelization. The algorithm is presented in Section 2 (overview of the previous algorithm) and Section 3 (new developments). Sections 4 and 5 contain the details about our computational tests.

2. The original grid-based method

Within QTAIM, a property P_A of the atom A is defined as the expectation value of an effective single-particle property density $p(\vec{r})$ over its so-called atomic basin Ω_A [2–4],

$$P_A = \int_{\Omega_A} p(\vec{r})d\vec{r}. \tag{1}$$

The atomic basins must be determined and may have complicated shapes, which makes the integration in Eq. (1) difficult. The explicit numerical construction of the boundaries of the atomic basins, the zero-flux surfaces, is the bottleneck in most QTAIM software [5–12]. In grid-based approaches, the atomic integration in Eq. (1) is performed without the explicit construction of the zero-flux surfaces. Instead, an integration grid in real space is partitioned into subsets, ω_A. Each subset ω_A is composed of all grid points contained in the

* Corresponding author. Present address: Departamento de Química, CINVESTAV, Avenida Instituto Politécnico Nacional 2508, A.P. 14-740, México D.F. 07000, Mexico.
E-mail addresses: jirodriguez@cinvestav.mx, jirh73@gmail.com (J.I. Rodríguez).

Table 1
Atomic properties. E_G is the energy obtained using our original method [19] as implemented in ADF. $E = E_G \times F_{\text{virial}}$ is the energy scaled by the virial factor, F_{virial} [2,26]. The SCF energy is shown is the last column.

Molecule	Charge	E_G[a]	E	L	SCF energy
C_6H_6					
C	−0.0346	−37.48890	−37.77683	-6.0×10^{-4}	
H	0.0346	−0.58243	−0.58690	6.0×10^{-4}	
Total	0.0000	−228.42798	−230.18242	-3.0×10^{-5}	−230.18239
CH_2O					
C	0.8889	−36.91922	−37.15282	6.4×10^{-4}	
O	−0.9852	−74.86336	−75.33705	-9.3×10^{-3}	
H	0.0482	−0.57204	−0.57566	4.3×10^{-3}	
Total	0.0000	−112.92666	−113.64119	3.4×10^{-6}	−113.64119
$Fe(C_5H_5)_2$					
Fe	0.6617	−1259.66270	−1263.07930	1.0×10^{-3}	
C	−0.1274	−37.52635	−37.62813	-1.4×10^{-3}	
H	0.0612	−0.56769	−0.56923	1.3×10^{-3}	
Total	0.0000	−1640.60310	−1645.05290	-2.2×10^{-5}	−1645.05290

[a] $E_G = G(\Omega) = \frac{N}{2}\int_\Omega \int \ldots \int \nabla_{\vec r}\Psi^*(\vec r, \vec r_2, \ldots, \vec r_N) \cdot \nabla_{\vec r}\Psi(\vec r, \vec r_2, \ldots, \vec r_N) d\vec r_2 \ldots d\vec r_N d\vec r$; Ψ and N are the total wave function and the number of electrons, respectively (see Refs. [2,19]).

atomic basin Ω_A so that Eq. (1) is reduced to a simple quadrature over the grid points in ω_A [14–19],

$$P_A = \int_{\Omega_A} p(\vec r) d\vec r = \sum_{\vec r_i \in \omega_A} w_i p(\vec r_i). \tag{2}$$

We determine which atomic basin a given grid point $\vec r_i$ belongs to by tracing the steepest-density-ascent path from $\vec r_i$ to the corresponding atom nucleus (it is worth mentioning that the earliest version of this algorithm was implemented in the program Promega [14]). This partitioning procedure is based on the fact that any steepest ascent path of the density that starts within an atomic basin will end at the attractor of that basin [13]. However the steepest ascent path does not have to be constructed for every point in the grid because some points can be unambiguously assigned to a nearby atom and other points are so far from the atoms that they contribute minimally to the atomic properties (see Refs. [18,19] for details). The CPU time is proportional to the grid size, but, because the zero-flux surfaces are not explicitly constructed, the CPU time of our previous method is smaller than the CPU time of the standard methods by at least one order of magnitude [18,19]. The attractor assignation step, i.e., the construction of the steepest ascent path of the density for every grid point is the time consuming step in our original method. In the following sections we show how the attractor assignation can be done more efficiently.

3. The new method

3.1. Accuracy

We introduced the virial factor [2,26] for accurately computing the QTAIM energies (see Table 1). The accuracy in computing the other atomic properties (charges, dipole and quadrupole moments) was also improved by using the ADF symmetry adapted grids [20–22,27] (see Table 1).

3.2. Vectorization

In ADF the grid points are divided in *NB* blocks (sets), each with *NBP* grid points. This is done to facilitate both *automatic* vectorization in inner loops over the points in a block and the parallelization of the code (see Section 3.4). Many of the subroutines in ADF are designed to perform function evaluations on a block of grid points at once, which reduces the CPU time [20–22]. This fact motivated our vectorization procedure: Instead of constructing steepest ascent paths one after the other as in the original method, these paths are constructed at once for a block (vector) of points. Thus we *explicitly* transform the scalar code for constructing the path of the density gradient[1],

$$\vec r^{\mu}_{i+1} = \vec r^{\mu}_i + \Delta_{step} \frac{\nabla\rho(\vec r^{\mu}_i)}{|\nabla\rho(\vec r^{\mu}_i)|}, \tag{3}$$

into a vector (matrix) code,

$$\begin{bmatrix} \vec r^{\mu}_{i+1} \\ \vec r^{\mu+1}_{i+1} \\ \vdots \\ \vec r^{\mu+NBP}_{i+1} \end{bmatrix} = \begin{bmatrix} \vec r^{\mu}_{i} \\ \vec r^{\mu+1}_{i} \\ \vdots \\ \vec r^{\mu+NBP}_{i} \end{bmatrix} + \Delta_{step} \begin{bmatrix} \frac{\nabla\rho(\vec r^{\mu}_i)}{|\nabla\rho(\vec r^{\mu}_i)|} \\ \frac{\nabla\rho(\vec r^{\mu+1}_i)}{|\nabla\rho(\vec r^{\mu+1}_i)|} \\ \vdots \\ \frac{\nabla\rho(\vec r^{\mu+NBP}_i)}{|\nabla\rho(\vec r^{\mu+NBP}_i)|} \end{bmatrix}. \tag{4}$$

Here μ runs over the points in the block and i over the points in the density steepest ascent path. One of the biggest advantages of the explicit vectorization procedure, Eq. (4), is that the density and its gradient are computed over the block of points at once, which significantly speeds up this algorithm, and other algorithms in ADF. ADF subroutines for computing the density, its gradient, and the Hessian matrix, are designed to achieve this goal [21–23]. These subroutines also scale linearly with system size [21]. The number of points on each block (*NBP*) is determined dynamically for each calculation and is usually set to a value around 128. (The value can be adjusted by the user.) Notice that this number defines the 'width' of our explicit vectorization procedure. Vectors are declared in our Fortran 90/95 code as they appear in Eq. (4) so that the Fortran intrinsic functions for vector operations are also exploited. Notice also that this vectorization procedure is not the same as a parallelization procedure, so it can be implemented even as a serial program. The vectorization procedure reduced the CPU time drastically (see Table 2).

3.3. The use of molecular symmetry

We can use the molecular symmetry so that the attractor assignation is performed only for symmetry unique points in the integration grid. The use of the symmetry was implemented as follows. Let $\vec r_i$ a symmetry unique point, which is assigned to the attractor *A*. Let $\hat\tau$ be a point group symmetry operation so that

$$\vec r' = \hat\tau \vec r. \tag{5}$$

[1] The method that was vectorized and used for constructing the steepest ascent path was actually the 2nd order Runge–Kutta. Here we show the vectorization of the Euler method for didactic purposes. See next section.

Table 2
Comparison of the CPU time (in seconds) for different versions of the algorithm. The third column shows the maximum of the absolute value of L_Ω for all the atomic basins Ω. $T_{original}$ is the CPU time for the original method [19] as implemented in ADF; T_{vec} is the CPU time of the vectorized version of the method; $T_{vec+sym}$ is the vectorized version plus the use of molecular symmetry; T is the CPU time for the method with all features together: vectorization, symmetry and parallelization (4 processors).

Molecule	Sym	Max. $\|L_\Omega\|$	$T_{original}$	T_{vec}	$T_{vec+sym}$	T
H_2O	C_{2v}	2.5×10^{-4}	15.55	1.32	0.37	0.11
$Fe(C_5H_5)_2$	D_{5h}	1.4×10^{-3}	4060	711.1	40.9	10.5
C_{70}	D_5	4.3×10^{-3}	35 675	6948	737.9	257.2

Then $\vec{r}_i'$ will be assigned to the attractor,

$$A' = \hat{\tau}A. \qquad (6)$$

We obtained exactly the same numerical values of the atomic properties with and without using molecular symmetry. Thus the CPU time of the new method is roughly equal to the CPU time of the original method divided by the number of symmetry operations in the molecular point group (see Table 2).

3.4. Parallelization

The attraction assignation for a block of grid points is independent of the attractor assignation of any other block. So the procedure is amenable to parallelization. Sets of points to be assigned to an attractor are distributed to different processors. Therefore the parallelization of the attractor assignation over a blocks of points of our grid-based method is a natural extension of in the ADF parallelization for the integration procedure, which distributes blocks of points over the nodes of a parallel machine using cyclic distribution [22,23]. The paradigm of parallelization that was implemented in ADF is based on the 'single program multiple data' model and is not communication-bound. Ref. [23] describes the ADF parallelization procedure in detail.

4. Computational methods and hardware

All calculations were performed with a development version of ADF using a TZP Slater basis. We used the Dirac exchange functional [24] with the Vosko–Wilk–Nusair correlation functional [25]. Unless otherwise stated, the ADF default settings for the SCF procedure were used [20–22]. The 2nd order Runge–Kutta algorithm with a step size equal to 0.2 a.u. was used for constructing the steepest ascent paths of the density [19,26]. The initial radius of each atomic trust sphere as defined in the original algorithm was set to 0.23 a.u. (Bohr). Unless otherwise stated, the values for all the other parameters in the original algorithm were used [19]. All calculations were performed on a 8-node cluster with single-cpu Quad-core Intel Xeon X3353 (2.66 GHz, 6 MB cache) on each node. The compiler used was the Intel Fortran 10.1.008 (Intel MKL 10.0.011; HP-MPI 2.2.5.1). The -x and -ax options are used to enable SSE instructions.[2]

5. Results and discussion

A single point calculation at the optimized geometry[3] was used to generate the electron density for our atomic property calculations. Table 1 shows the QTAIM atomic properties for some

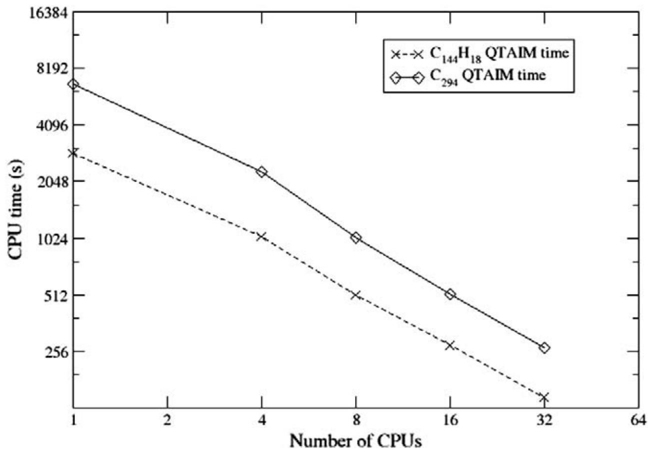


Fig. 1. Parallel performance for QTAIM atomic property calculation.

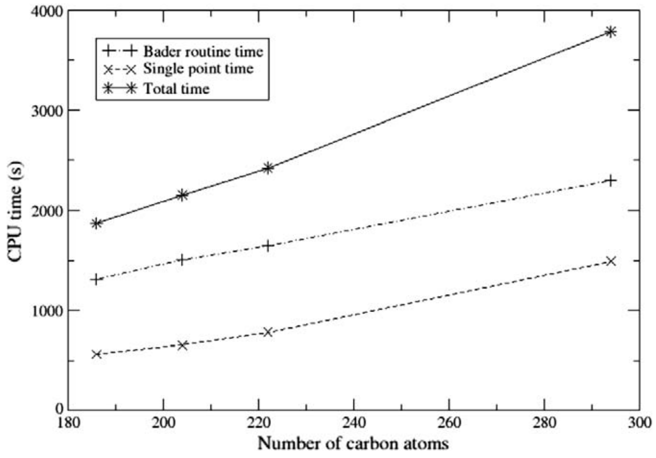


Fig. 2. The CPU time for determining the atomic charges, dipole moments, and quadrupole moments for a series of carbon nanotubes (C_{186}-D_{3h}; C_{204}-D_{3d}; C_{222}-D_{3h}; C_{294}-D_{3h}). All calculations were performed on 4 processors.

representative molecules. From Table 1, we can see that the accuracy of our calculations is good enough to reproduce closely the molecular property as a sum of the atomic properties even for the cases of the lowest accuracy ($L \sim 10^{-3}$).[4] The problem in the original method related to the disparity between charges of some symmetry equivalent atoms was also overcome even when symmetry was not used (see Table 5 of Ref. [19] and the related comment). The use of the virial correction factor [2,27] improved significantly the accuracy in computing the atomic energies with respect to the original method [19] (see Refs. [2,27] for details about the conditions under which the virial factor must be used). A minimum accuracy of $L_\Omega \sim 10^{-3}$ a.u. was required for all atomic property calculations (energies, charges, dipole and quadrupole moments) reported in this work. The ADF default integration grid [20–22,28] (~4500 grid points per atom) suffices for reaching this minimum accuracy except for the iron atom's basin in the ferrocene molecule, where a larger grid was required (~11 000 grid points). Table 2 shows the reduction of the computational cost due to the different improvements introduced here. Notice that the vectorization procedure alone made the new method about one order of magnitude faster, on average, than the original algorithm. Using the molecular symmetry reduces the CPU time

[2] Relevant Fortran compiler flags, most functions: -nothreads -xW -mcmodel = medium -O2; performance-critical functions: -nothreads -xW -mcmodel = medium -O3 -axP.

[3] We did not perform geometry optimizations for some of the larger nanotubes. In those cases, we analytically verified that the initial geometry had the correct symmetry and visually confirmed that the molecular geometry was reasonable.

[4] $L_\Omega = \int_\Omega -\frac{1}{4}\nabla^2\rho(\vec{r})d\vec{r} = 0$ for any atomic basin. Verifying this equation is the standard way to measure the accuracy of any QTAIM method.

in proportion to the number of the symmetry operations in the point group of the molecule. With modest parallelization (4 processors), the overall algorithm can be more than 300 times faster than the original algorithm. Moreover, because the CPU time of the new method decreases linearly with the number of CPUs (see Fig. 1), the CPU time of the atomic property calculations can be decreased further by increasing the number of processors.

Fig. 1 shows the parallel performance of the QTAIM atomic property calculation. The average parallel efficiency of the QTAIM atomic property calculation is equal to 70% (79.3% for C_{294} and 61.9% for $C_{144}H_{18}$). Fig. 2 shows the CPU time for both the single point calculation and the QTAIM calculation as a function of the system's size. From Fig. 2 we can see that the cost of the QTAIM atomic property calculation is comparable to the single point calculation and that the difference between these two timings becomes smaller as the size of the system increases. Fig. 2 shows that the new method scales linearly with system size.

6. Conclusions

We report, for the first time, an algorithm for computing the properties of atoms (as defined through the quantum theory of atoms in molecules) that has computational cost comparable to a standard single point LDA/GGA Kohn–Sham DFT calculation. This breakthrough is built upon our previous grid-based method for computing atomic properties (thus avoiding the explicit construction of the zero-flux surfaces), but the implementation presented here is much faster. The increase in speed is due to three main improvements: vectorization, symmetry handling, and parallelization. Even for single processor calculations without symmetry, the vectorized method is about 10 times faster than its predecessor. However, small-scale parallelization (4 processors) and the use of symmetry make the method even faster, so that the resulting approach is over 200 times faster on average than our previous program. The parallel efficiency of the method is high, and the computational cost can be further decreased by further increasing the number of processors. For example, using 32 CPU cores it takes only 4.4 min to compute atomic properties of a 294-atom carbon nanotube (in the D_{3h} point group). In addition, the accuracy for computing QTAIM atomic energies and other properties has improved.

ADF is supported on many platforms and ADF's testing procedure ensures new functionality works on all of them. For example, the vectorization procedure is fully portable and the speed-up due to it will be similar on all supported platforms. This is because the biggest speed-up is achieved by reducing computational overhead, rather than by exploiting compiler- or platform-specific vectorization features. Of course, platform-specific vectorization capabilities are exploited, when available.

This method, like its predecessor, is targeted for applications where computational efficiency is critical and moderate accuracy suffices. For such calculations, this method is the fastest that we know of.

Acknowledgements

We would like to thank Dr. Erik Van Lenthe, Dr. Pier Philipsen, Dr. Alexei Yakovlev, Dr. Stan Van Gisbergen, and Dr. Todd Keith for useful discussions about this work and for proofreading this manuscript. J.I.R. gratefully acknowledge the *Vrije Universiteit Amsterdam* theoretical chemistry group and all SCM people for hosting him during the development of this work and for sharing their computer facilities. J.I.R. also thanks Miss. Silvia Dimitrova for her insightful discussions and for LMBABB. J.I.R. and P.W.A. acknowledge research support from NSERC.

References

[1] R.F.W. Bader, Phys. Rev. B 49 (1994) 13348.
[2] R.F.W. Bader, Atoms in Molecules: A Quantum Theory, Oxford University Press, New York, 1990.
[3] P.L.A. Popelier, Atoms in Molecules: An Introduction, Prentice Hall, Edinburg, 2000.
[4] C.F. Matta, R.J. Boyd (Eds.), The Quantum Theory of Atoms in Molecules. From Solid State to DNA and Drug Design, Wiley-VCH, Weinheim, 2007.
[5] F.W. Biegler-Köning, R.F.W. Bader, T.H. Tang, J. Comput. Chem. 3 (1982) 317.
[6] J. Cioslowski, B.B. Stefanov, Mol. Phys. 84 (1995) 707.
[7] B.B. Stefanov, J. Cioslowski, J. Comput. Chem. 16 (1995) 1394.
[8] P.L.A. Popelier, Mol. Phys. 87 (1996) 1169.
[9] P.L.A. Popelier, Comput. Phys. Commun. 108 (1998) 180.
[10] M. Rafat, P.A.L. Popelier, J. Comput. Chem. 28 (2007) 2602.
[11] J. Cioslowski, Chem. Phys. Lett. 194 (1992) 73.
[12] J. Cioslowski, A. Nanayakkara, M. Challacombe, Chem. Phys. Lett. 203 (1993) 137.
[13] F.W. Biegler-König, T.T. Nguyen-Dang, Y. Tal, R.F.W. Bader, A.J. Duke, J. Phys. B: At. Mol. Phys. 14 (1981) 2739.
[14] T.A. Keith, Ph.D. Dissertation, McMaster University, Hamilton, ON, Canada, 1993.
[15] G. Merino, Ph.D. Dissertation, Cinvestav, Mexico City, Mexico, 2003.
[16] G. Henkelman, A. Arnalsson, H. Jónsson, Comput. Mater. Sci. 36 (2006) 354.
[17] E. Sanville, S.D. Kenny, R. Smith, G. Henkelman, J. Comput. Chem. 28 (2007) 899.
[18] J.I. Rodríguez, Ph.D. Dissertation, McMaster University, Hamilton, ON, Canada, 2008.
[19] J.I. Rodríguez, A.M. Köster, P.W. Ayers, A. Santos-Valle, A. Vela, G. Merino, J. Comput. Chem. (2009), in press, doi:10.1002/jcc.21134.
[20] G. te Velde, F.M. Bickelhaupt, S.J.A. van Gisbergen, C. Fonseca Guerra, E.J. Baerends, J.G. Snijders, T. Ziegler, J. Comput. Chem. 22 (2001) 931.
[21] C. Fonseca Guerra, J.G. Snijders, G. te Velde, E.J. Baerends, Theor. Chem. Acc. 99 (1998) 391.
[22] ADF2008, SCM, Theoretical Chemistry, Vrije Universiteit, Amsterdam, The Netherlands. <http://www.scm.com>.
[23] C. Fonseca Guerra, O. Visser, J.G. Snijders, G. te Velde, E.J. Baerends, in: E. Clementi, C. Corongiu (Eds.), Methods and Techniques for Computational Chemsitry, STEF, Cagliary, 1995, p. 303.
[24] P.A.M. Dirac, Proc. Cambridge Philos. Soc. 26 (1930) 376.
[25] S.H. Vosko, L. Wilk, M. Nusair, Can. J. Phys. 58 (1980) 1200.
[26] W.H. Press, B.P. Flannery, S.A. Teukolsky, T. Vetterling, Numerical Recipes in Fortran, Cambridge University Press, Cambridge, 1992.
[27] C.F. Matta, R.J. Boyd, in: C.F. Matta, R.J. Boyd (Eds.), The Quantum Theory of Atoms in Molecules. From Solid State to DNA and Drug Design, Wiley-VCH, Weinheim, 2007.
[28] G. te Velde, E.J. Baerends, J. Comput. Phys. 99 (1992) 84.