Local MP2 with Density Fitting for Periodic Systems: A Parallel Implementation

Lorenzo Maschio*

Dipartimento di Chimica IFM and Centre of Excellence NIS (Nanostructured Interfaces and Surfaces), Università di Torino, via P. Giuria 5, I-10125 Torino, Italy

ABSTRACT: A parallel implementation is presented for the evaluation of local second-order Møller—Plesset perturbation theory (LMP2) energies in periodic, nonconducting crystalline systems with a density-fitting approximation of two-electron repulsion integrals. Peculiarities of the periodic case with respect to parallel LMP2 implementations in molecular codes, such as the use of translational and point symmetry, impose different strategies in order to achieve good parallel performance. The implementation is benchmarked on a few systems, representing a choice of the most interesting solid state quantum-chemistry problems where the MP2 approach can be decisive. Good parallel efficiency of the algorithms is demonstrated for up to 54 processors. Test systems include a metal organic framework (MOF-5) 3D crystalline structure with a triple-ζ-quality basis set: this is the largest calculation performed so far with 106 atoms, 532 correlated electrons, and 2884 atomic orbitals per unit cell.

1. INTRODUCTION

Molecular quantum chemistry has undergone a significant revolution in recent years. Thanks to the implementation of fast electron correlation techniques in modern quantum chemistry codes,[1,2] accurate post-Hartree—Fock methods are now applicable to very large molecular complexes[3] at a reasonable computational cost, thus becoming a serious competitor to DFT also for routine calculations. This has become possible after the assessment of fast integral evaluation techniques, like multipole-based screening,[4−6] resolution of identity (RI),[7] or density fitting (DF),[8−10] and due to the introduction of approximate yet efficient variants of the traditional quantum chemistry methods, like local correlation[11−13] and Laplace transform[14] techniques. The efficient parallelization of the algorithms had, and still has, a central role in this process,[15−24] following the growing availability of cheap and powerful multicore machines.

The progress of solid state quantum chemistry along this path is a few years behind. Several groups, in the past decade, have proposed different and often complementary approaches, successfully taking up the challenge of evaluating *ab initio* electron correlation in periodic systems.[25−31] Still, the application of such methods has mainly played a benchmark role, since it has been limited, up to date, to the study of simple model systems, with a small number of atoms in the reference cell.[32−34]

The present work presents a step forward in the direction of the assessment of a powerful and usable tool for real-life quantum chemical calculations at the correlated level for crystalline solids. The Cryscor program,[35] jointly developed by the theoretical chemistry groups of Torino and Regensburg, is part of a project aimed at implementing local correlation methods for the study of crystalline periodic systems. The first public (serial) version of the code[36] implements density-fitted local Møller-Plesset perturbative correction at second order (DF-LMP2) to the Hartree—Fock (HF) solution provided by the Crystal code.[37,38] Both programs adopt a local basis set consisting of Gaussian-type atomic orbitals.

The implemented method is an adaptation to the periodic case of the LMP2 method first proposed by Pulay[11,13] and then efficiently implemented for molecules by Schütz and Werner in the Molpro code.[1,39,40] Periodic LMP2 relies on a localized solution of the HF equations, expressed in terms of symmetry-adapted Wannier functions (WF),[41,42] and on a representation of the virtual space by projected atomic orbitals (PAO; *vide infra*). The adoption of approximate integrals evaluation techniques like the DF approximation allows one to speed up the calculation by several orders of magnitude.[43] The serial periodic LMP2 code has already been successfully applied to the study of cohesive energy of molecular crystals[44−46] and rare gas crystals,[34] surface adsorption,[47,48] the relative stability of crystalline polymorphs,[49] pressure-induced phase transitions,[50] and simulation of Compton scattering experiments.[51] The largest calculations performed so far with Cryscor on a single processor have been a CO_2 bulk crystal with the aug(d,f)-cc-pVQZ basis set (12 atoms, 696 basis functions in the reference cell)[45] and a sodalite crystal with a triple-ζ-quality basis set[52] (36 atoms, 1128 basis functions in the reference cell).

A great amount of literature exists concerning implementation of electronic structure methods for molecules, a good review of which can be found in ref 21. In particular, a number of parallel MP2 implementations have been reported in the past 10 years.[53−56] Among these, a massively parallel implementation of the local MP2 method for molecules, which is of particular relevance to the scopes of the present work, has been presented by Nielsen and Janssen.[20] In that work, it is shown that due to peculiarities of the local approach, it is very difficult to obtain a good parallel efficiency. Earlier work on local MP2 parallel implementation, using nonorthogonal occupied orbitals, has been reported by Nakao and Hirao.[55] Finally, parallel implementation of canonical MP2 with RI approximation of two-electron repulsion integrals

Received: May 26, 2011
Published: August 01, 2011

```
STEP1:
Define N segments
Set up N empty buckets
Loop over all blocks
 read integral block
 put block into appropriate bucket n
 If n full then
     write n to disk
     empty bucket n
 EndIf
End Loop
Flush to disk partly filled buckets
STEP2:
Loop N segments
    gather all buckets of this segment
    sort the segment and write to disk
End Loop segments
```

Figure 1. Standard scheme of bucket sort algorithm.

has been proposed by Hättig et al.[56] and by Katouda and Nagase.[24]

The presentation of the periodic LMP2 algorithms is structured as follows: In section, 2 the main strategies for parallel efficiency and I/O are presented. In section 3, the structure of the parallel implementation is described and discussed in detail, with reference to the main formalism of the periodic LMP2 method. In section 4, benchmark calculations are reported; the performance of the different parts of the parallel code is analyzed.

2. PARALLELIZATION STRATEGIES AND I/O

The main priority, when implementing a parallel version of a quantum chemistry algorithm, is to extend its applicability to larger and more complex systems. In this sense, saving memory resources is the most important goal to achieve. General principles, which have been followed in the development, are as follows. (i) Code linearity: parallel instructions should not be too invasive—only the outermost loops of very complex routines are parallelized. (ii) Code maintainability: when possible the parallel code source is the same as the serial, and parallel libraries can be linked or not at the compilation stage. (iii) Code portability: pure MPI instructions have been used throughout, with no use of OpenMP.

Since the periodic DF-LMP2 code is very heterogeneous, with distinct features and peculiarities in its different parts, different strategies have been followed for distributing CPU load and memory:

- *Distributed memory linear algebra*. The use of Scalapack routines has been necessary in the PAO generation step, where the size of the matrices involved scales quadratically with the number of atoms in the unit cell. In all other parts, the peculiarities of the local method allow for the use of matrices whose size is not closely related to the number of atoms in the system (though it depends on basis set choice and local domain definition).
- *Parallelization according to atomic index in the unit cell*. This is the strategy followed for the parallelization of the different steps of the Periodic Density Fitting code and in the evaluation of integrals for the multipolar expansion. The advantages of such a coarse grain parallelization are a clean and straightforward implementation and a good efficiency (see section 4). Drawbacks such as load unbalancing arise if the number of processors is larger than the number of atoms in the unit cell, or in the case of atoms with very different numbers of electrons.
- *Parallelization according to pair index*. The LMP2 equations can be conveniently factorized in terms of WF pair contributions. Given the large number of WF pairs involved in a fairly large calculation (several thousands), load balancing issues are not likely to appear at this stage.

```
 STEP1:
 Define N segments
 Set up N empty buckets
*classified=false
 Loop over irreducible blocks b
p  If ( my b) then
     read integral block
p  EndIf
*  Loop over all symm.  operators V
*    Obtain rotated block b'=V b
*    If (classified(b')) skip
*    If (not needed(b')) skip
*    classified(b')= true
p    If (my b) then
*       obtain rotated (b')
p       If( my b') then
         put block into bucket
         if bucket full write to disk
p       Else
p         send (b') to processor which wants it
p       EndIf
p    Else if (I want b' ) then
p      recv (b')
       put block into bucket
       if bucket full write to disk
p    EndIf
   End Loop V
 End Loop blocks
 Flush to disk partly filled buckets
 STEP2:  same as the serial one
```

Figure 2. Modified bucket sort algorithm with symmetry exploitation and data redistribution. The lines colored in red and marked by a p letter on the left are specific of the parallel implementation. The instruction lines colored in blue and marked by an asterisk on the left perform exploitation of symmetry.

Local post-Hartree–Fock methods are characterized by the huge amount of intermediate data produced, especially if density fitting techniques are adopted. Data (i.e., integrals, MP2 amplitudes) is by far too large to be kept in memory even for fairly simple calculations. Efficient strategies have to be devised in order to keep the I/O overhead under control and, at the same time, reduce communication among processors. Discussion of such strategies will be covered in the remainder of this section.

2.1. Bucket Sort Algorithm. As will become clear in the remainder of this paper, the parallelized bucket sort plays a central role in the implementation of periodic density fitting.

Bucket sort, also known in quantum chemistry as the bin sort, is a standard, well-known algorithm that allows one to efficiently sort arrays, which must be kept on disk because it is considerably larger than the available system memory. For a recent discussion on this algorithm, see for instance ref 57. Bucket sort was first introduced in electron correlation theory by Yoshimine,[58] and a parallel implementation was described in 2002 by Baker and Pulay in the context of canonical MP2 energy calculation.[54] In the present LMP2 implementation, a new parallel version of this algorithm is introduced, which features redistribution of data among processors and exploitation of symmetry at the same time.

The serial algorithm, which has been adapted from the corresponding one in the Molpro code, is sketched in Figure 1. A set of N "segments" is defined such that the length of a single segment is equal to the available memory. A buffer is allocated and formally divided in a set of N empty "buckets". In a first step, atomic integral blocks are read from the disk, one at a time, and assigned to the corresponding destination bucket. When a bucket is full, it is written to disk and emptied. After all blocks have been processed, the second step is performed: all buckets corresponding to the first segment are loaded in memory, sorted, and written to disk. By repeating the process for all segments, the file with the sorted elements is generated. If memory is large enough to hold full segments, no intermediate buckets are written to disk. In this case, the sort is $\mathcal{O}(n)$ scaling (since the final order is known *a priori*); otherwise the scaling depends on the amount of available memory.

The parallel version of this algorithm is reported in Figure 2; instructions which are specific for symmetry and parallelization are highlighted as explained in the caption. After the processor, which has computed *symmetry irreducible* block b, has read it, all symmetry operators are applied to it (this includes identity), generating several b′ new objects. It is checked whether b′ is actually needed or if it has already been obtained by some other rotation. b′ is then sent to the processor to which it is assigned according to memory distribution strategies (this happens only if the target processor is not the same as b). The target processor receives b′ and puts it in its bucket. The routine proceeds as for the serial one: step 2 is performed by each processor on the quantities it holds, and no further communication takes place.

Note that parallel step 1 is not *globally blocking*, since only one-to-one (or one-to-many) communications take place, and processors which are neither owners of a given b nor targets for the rotated b′ are allowed to proceed independently. Still, it is not completely *asynchronous* like the algorithm outlined by Baker and Pulay, who achieved this result through spawning of additional "listening" processes per each node, dedicated to I/O.

In the present context, the bucket sort is used in two parts of the DF procedure. The symmetry is applied only in the second sort.

2.2. Paging Algorithm. A paging strategy has been implemented to deal with matrices that are too large to fit into the available memory. A buffer, as large as the available memory, is allocated. The big matrix is divided in *blocks*, having the size of all shells belonging to a pair of atoms. Initially, the buffer is empty. During the program execution, when a block is needed, it is loaded in memory and fills an empty part of this buffer. If the buffer is full and a new block must be loaded, a suitable portion of it is freed by discarding one block. The choice is made according to a scoring system: every time a block is used, it gains one point. The block, currently in memory, which has the lowest score is discarded. If this block has changed in the meantime (this is the case of the updated LMP2 amplitudes), it is written back to disk; otherwise (most frequently), its memory slot is just overwritten. The blocks used more frequently are thus always kept in memory, if possible.

2.3. Disk Sharing. In addition to the paging algorithm just described, a further scheme has been set up to efficiently handle large, replicated matrices. If two or more nodes share the same disk space, the disk itself can be used as a communication medium among them instead of message passing. The CPUs are classified according to groups, where each group is formed by those processors sharing the same disk, and one processor of each group is nominated the "leader" of the group. The processors automatically recognize if they are sharing the disk with others (no input is required by the user in this sense).

Direct writes can be done, within a group, to the same physical file, and information is accessible by all processors of that group. Replication of data on the disk is avoided, and all-to-all communication steps are reduced to a communication among group leaders. In the case of the SP6 AIX machine, used for the benchmark calculations in this work, all of the processors share the same disk. As a consequence, message passing instructions are avoided when this scheme is applied. Although not as advanced as the Array Files system,[59] this very simple technique can be quite effective in achieving good overall parallel performance. The disk sharing strategy is adopted for the two-index density fitting integrals (section 3.3.1) and for excitation amplitudes (section 3.4). In combination with the paging algorithm described in section 2.2, it represents a powerful tool to improve the parallel efficiency of the LMP2 equations.

3. THE PARALLEL CODE AND PERIODIC LMP2 ESSENTIAL FORMALISM

The general scheme of the Cryscor code has been outlined in our previous works[28,60] but will be reported here for ease of reference and to support the discussion of the parallel algorithms. After a preliminary stage, in which information of the system is recovered from Crystal (geometry, symmetry data, basis set, HF solution expressed in terms of Wannier functions), the main computational steps of a local MP2 calculation are (i) generation of the localized functions used to represent the virtual space, that is projected atomic orbitals (PAO),[13] (ii) integrals evaluation via multipolar expansion and density fitting techniques, and (iii) periodic LMP2 iterative equations.

The notation adopted in the following is meant to be consistent with previous papers:[28,35]

- Greek letters μ, ν, etc. represent an atomic orbital, i.e., a basis function.
- Letters i, j, etc. represent an occupied orbital, i.e., a Wannier function.
- Letters a, b, etc. represent a virtual orbital, i.e., a PAO.
- Capital letters P, Q, etc. indicate an auxiliary basis function, used for density fitting.
- Calligraphic capital letters $\mathcal{J}$,$\mathcal{A}$,$\mathcal{P}$, etc. indicate a lattice vector label, while symbols such as $\mathbf{R}_{\mathcal{J}}$, $\mathbf{R}_{\mathcal{A}}$, and $\mathbf{R}_{\mathcal{P}}$ will indicate the corresponding vectors. Where no lattice vector label is indicated, the reference cell is implicitly understood. Primed lattice vectors $\mathcal{J}'$,$\mathcal{A}'$,$\mathcal{P}'$, etc. indicate that the corresponding index has been shifted to take advantage of translational symmetry. The entity of the shifting is always clear from the context (e.g., $(P\mathcal{P}|Q\mathcal{Q}) = (P|Q\mathcal{Q}')$ where $\mathcal{Q}' = \mathcal{Q}\ominus\mathcal{P}$).

The algorithms will be presented in the order they are executed by the code. For this reason, evaluation of preliminary quantities and integrals will be discussed before the LMP2 energy calculation.

3.1. Generation of Local Virtual Space and Related Matrices. Wannier functions are used to represent the occupied

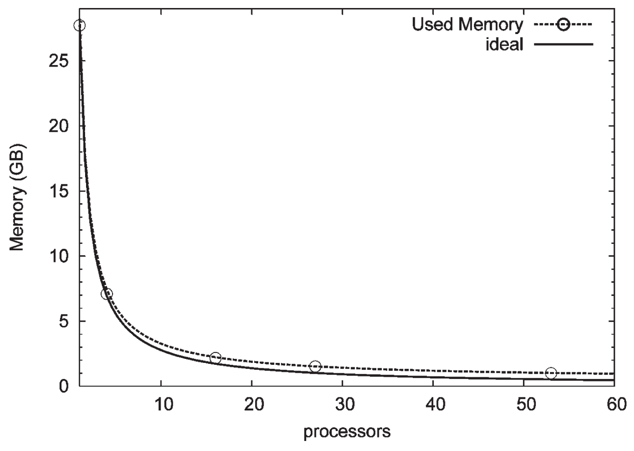


Figure 3. Use of memory in the PAO generation step, for the largest test case of this work (MOF-5).

space. These symmetry-adapted and well-localized objects are calculated by the Crystal program.[37,38] To describe the virtual manifold, following the proposal of Saebø and Pulay,[13] atomic orbitals projected out from the occupied space (PAO) have to be generated. Each PAO $\chi_a(\mathbf{r})$ is described by a linear combination of the AO basis set: $\chi_a(\mathbf{r}) = \sum_{\mu_{\mathcal{M}}} c_a^{\mu_{\mathcal{M}}} \phi_{\mu_{\mathcal{M}}}$. They form a redundant nonorthogonal set but at the same time are appreciably well localized and have the symmetry of the parent AOs. The projection that produces the coefficients of a reciprocal space image of a PAO, $\chi_a{}^{\mu}(\mathbf{k})$, can be cast as

$$c_a^{\mu}(\mathbf{k}) = \sum_{\nu}[\delta_{\mu,\nu} - \sum_{i} |u_{\mathbf{k}}^{i,\mu}\rangle\langle u_{\mathbf{k}}^{i,\mu}|]S^{*}_{\nu,\mu}(\mathbf{k}) \tag{1}$$

where $|u_{\mathbf{k}}^{i,\mu}>$ are the occupied Fock eigenvectors and $S_{\nu,\mu}(\mathbf{k})$ is the overlap matrix in the AO basis, both expressed in reciprocal space. The PAOs in direct space are obtained through a back Fourier transform:

$$c_a^{\mu,\mathcal{M}} = \sum_{\mathbf{k}} c_a^{\mu}(\mathbf{k})\, e^{i\mathbf{k}\cdot\mathbf{R}_{\mathcal{M}}} \tag{2}$$

The most demanding steps within this part of the code are the orbital transformation of overlap (**S**) and Fock (**F**) matrices from the AO to the PAO basis, which is also performed in reciprocal space,

$$X_{a,b,\mathcal{B}} = \sum_{\mathbf{k}} X_{a,b}(\mathbf{k})\, e^{i\mathbf{k}\cdot\mathbf{R}_{\mathcal{B}}}$$
$$= \sum_{\mathbf{k}}[\sum_{\mu} c_a^{\mu}(\mathbf{k})[\sum_{\nu} X_{\mu,\nu}(\mathbf{k})\, c_b^{*\nu}(\mathbf{k})]]e^{i\mathbf{k}\cdot\mathbf{R}_{\mathcal{B}}} \tag{3}$$

where $X = S, F$.

This step represents mainly a memory bottleneck, since square matrices having the full size of the basis functions in the unit cell are involved. For small and medium-sized systems, eqs 2 and 3 might be conveniently factorized according to the **k**-mesh sampling points. This approach would imply just a little communication (a global sum for the inverse Fourier transform at the end) but would limit the number of processors that can be used to be equal or less than the number of **k** points and cause load balancing problems, without solving the problem of square matrices to be kept in memory.

When the system grows large, the number of needed **k** points reduces considerably, while the user almost certainly strives for a larger number of CPUs to handle the computational expense of the calculation. For this reason, this step is parallelized, according to a *distributed memory* strategy by means of standard parallel distributed linear algebra routines.[61] All matrices in eqs 2 and 3 are distributed among all processors. In Figure 3, the use of memory is reported for a test case: deviation of the memory distribution from ideal behavior is small, showing that 96% of the allocated arrays are handled through distributed linear algebra routines.

Thanks to the features of the local approach, the other steps of the LMP2 code exhibit a need for memory that can be made independent of the size of the system.

3.2. Fast Integral Evaluation: The Multipolar Expansion. The main bottleneck of the LMP2 method is constituted by the calculation of the four index two-electron repulsion integrals in the basis of WFs and PAOs:

$$K^{ij\mathcal{J}}_{a\mathcal{A},b\mathcal{B}} = (ia\mathcal{A}|j\mathcal{J}b\mathcal{B})$$
$$= \int d\mathbf{r}_1 \int d\mathbf{r}_2\, \chi^i(\mathbf{r}_1)\, \chi^a(\mathbf{r}_1 - \mathbf{R}_{\mathcal{A}}) \frac{1}{|\mathbf{r}_2 - \mathbf{r}_1|} \chi^j(\mathbf{r}_2 - \mathbf{R}_{\mathcal{J}})\, \chi^b(\mathbf{r}_2 - \mathbf{R}_{\mathcal{B}}) \tag{4}$$

Approximate techniques for the evaluation of such integrals must be devised, which are described in this subsection and the following.

An efficient multipolar expansion can be applied if the two $\rho_{ia\mathcal{A}}$ and $\rho_{jb\mathcal{B}}$ product distributions are distant enough that they can be considered as enclosed in separate spheres. This is a well-known and widely used technique in quantum chemistry (see for instance ref 62). The integrals are obtained through the interaction of two sets of point multipoles:

$$K^{ij\mathcal{J}}_{a\mathcal{A},b\mathcal{B}} \approx \sum_{\ell,\ell'} Q^{\ell}_{ia\mathcal{A}} V^{\ell,\ell'}_{i,j\mathcal{J}} Q^{\ell'}_{jb\mathcal{B}'} \tag{5}$$

where the interaction operator $V^{\ell\ell'}_{i,j\mathcal{J}}$ is an interaction matrix which depends only on the relative position $\mathbf{r}_j + \mathbf{R}_{\mathcal{J}} - \mathbf{r}_i$ of the two centers and is easily calculated. The evaluation of the multipole moments of each product distribution with respect to its WF centroid $\mathbf{r}_i$,

$$Q^{\ell}_{ia\mathcal{A}} = \sum_{\nu,\mathcal{N}} c_a^{\nu\mathcal{N}'}[\sum_{\mu,\mathcal{M}} c_i^{\mu\mathcal{M}} \phi^{\ell}_{\mu\mathcal{M},\nu\mathcal{N}}(\mathbf{r}_i)] \tag{6}$$

becomes the most demanding step. The index ℓ indicates the multipole order, and by default it runs up to 4 (hexadecapoles): a total of 25 multipole moments have to be evaluated for each $ia\mathcal{A}$ product distribution. Note that the multipole moments $\phi^{\ell}_{\mu\mathcal{M},\nu\mathcal{N}}(\mathbf{r}_i)$ are not, in general, translationally invariant, since they are computed with respect to a center $\mathbf{r}_i$. A suitable translation operator must be applied.[63]

Parallelization of eq 6 is performed according to the ν index: each processor computes only a subset of the $\phi^{\ell}_{\mu,\nu\mathcal{N}}$ shell multipoles and uses them to compute a partial result for *all* of the $Q^{\ell}_{ia\mathcal{A}}$ objects. No intermediate communications are needed, and through a global sum after the end of the algorithm, all processors hold all computed multipole moments. The time required by eq 5 is negligible. The load is distributed according to the same strategy as in the LMP2 equation (see below), so that no further redistribution of the K integrals is required.

3.3. Fast Integral Evaluation: The Density Fitting Approximation. Density fitting (DF) approximation[9,10] is a powerful technique that allows one to speed up the calculation of two electron repulsion integrals of eq 4 by orders of magnitude, with

Table 1. Different Steps of the Periodic Density Fitting Parallel Code[a]

section	step	order of indices	parallelization index
3.3.1	two-index integrals	$\mathcal{Q}$, P, Q	$\mathcal{Q}$
3.3.1	three-index integrals	P, i, a, $\mathcal{A}$, $\mathcal{P}$	P
2.1	bucket sort		
3.3.2	solve for $d^{P\mathcal{P}}_{ia\mathcal{A}}$ coefficients	i, a, $\mathcal{A}$, P, $\mathcal{P}$	i, a, $\mathcal{A}$
2.1	bucket sort with symmetry		
3.3.3	assembly	P, i, a, $\mathcal{A}$, $\mathcal{P}$	P

[a] In the first column, reference is made to the section in the text where the corresponding algorithm is discussed. Column 3 reports the order according to which the quantities are evaluated/handled, from slowest to fastest index. The last column explicitly reports the index according to which the parallelization is made. In all cases, this coincides with the slowest index. Bucket sort routines redistribute the quantities among processors.

negligible errors.[43,60] The simple idea is to expand each product distribution between a PAO and a WF in a set of *auxiliary basis functions* $\{\Xi_P\}$:

$$\rho_{ia\mathcal{A}} = \chi_i(\mathbf{r})\,\chi_a(\mathbf{r}-\mathbf{R}_{\mathcal{A}}) \approx \sum_{P\mathcal{P}} d^{P\mathcal{P}}_{i,a\mathcal{A}}\,\Xi_P(\mathbf{r}-\mathbf{R}_{\mathcal{P}}) \tag{7}$$

A mixed auxiliary basis of Gaussian and Poisson functions[64] is used in this work.

Different flavors of the DF approximation have been developed and implemented for LMP2 in periodic systems (see ref 52 for a complete discussion). Among these, the local direct-space approach is the most similar to the approach adopted in molecular codes and in some of its aspects is similar to the one implemented by Scuseria and Izmaylov.[65] This is the simplest periodic DF approach, and at the same time, it is the most suitable for large unit cell systems. Thus, it was the method of choice in the parallel implementation of the LMP2 code.

The different steps of the implementation are outlined in Table 1. For each step, the corresponding strategy of parallelization is sketched and will be discussed in detail in the following. As can be seen, parallelization is always performed on the slowest running index, and MPI communications are almost entirely enclosed within the bucket sort steps.

3.3.1. Three- and Two-Index Integrals Calculation. Evaluation of three-index integrals of the type $(ia\mathcal{A}|P\mathcal{P})$ is distributed according to blocks containing all integrals sharing the same P index. Each processor computes independently all the needed $(\mu\nu\mathcal{N}|P\mathcal{P})$ three-index integrals in the AO basis, which are selected by each CPU through prescreening of the set of $(ia\mathcal{A}|P\mathcal{P})$ owned.[35] This coarse grain parallelization totally avoids communication among processors at this stage.

The number of computed integrals is huge, so they are kept on disk. Once computed and stored, integrals must be resorted: in the solve step, the index running the P atom has to be the fastest index, since to obtain a coefficient d^P_{ia} (eq 8) all fitting functions in a fitting domain are needed at once. This is achieved through the bucket sort algorithm described in section 2.1. The atomic object handled by the routine (a *block*) includes all $(ia|P)$ integrals where all PAOs a belong to the same atom, and all fitting functions P belong to the same atom. Redistribution of the integrals among processors takes place contextually. No symmetry operators are applied at this stage.

```
Loop over all $P$ fitting functions
   If (my $P$) Then
     read integrals $(ia\mathscr{A}|P\mathscr{P})$ for all $i$, all $a\mathscr{A}$, all $\mathscr{P}$
     read coefficients $d^{P\mathscr{P}'}_{j,bB'}$ for all $j$, all $b\mathscr{B}'$, all $\mathscr{P}'$
     Loop over all $i,j,\mathscr{J}$ pairs
       add contribution from atom $P$ to $K$ local buffer
       local sum $K$ on processor who owns $K(i,j,\mathscr{J})$
     End Loop $i,j,\mathscr{J}$ pairs
   EndIf
End Loop $P$
```

Figure 4. Parallel algorithm for the density fitting assembly step.

Two-index integrals of the type $(P|Q\mathcal{Q})$ are treated differently. These simple objects are evaluated quickly but are large, so the costly part is to store them on disk. Calculation is distributed according to the cell index of the second function, following a disk-sharing strategy (section 2.3), so that all unit cells are computed by processors of one *group*. This avoids replication of this matrix on disk, which can be very large, and totally avoids communication.

3.3.2. The Solve Step. The coefficients $d^{P\mathcal{P}}_{ia\mathcal{A}}$ of eq 7 are defined by the linear equation system

$$\sum_{P\mathcal{P}} d^{P\mathcal{P}}_{ia\mathcal{A}}(P|Q\mathcal{Q}') = (ia\mathcal{A}|Q\mathcal{Q}) \tag{8}$$

Both the indices $P\mathcal{P}$ and $Q\mathcal{Q}$ run over all fitting functions in the fit-domain, which is *specific for each separate set of i, a,* and $\mathcal{A}$ indices, that is, for all PAOs on a same atom. This allows for fairly small matrices in eq 8, leading at the same time to a greater number of different coefficients to be calculated and handled. A linear equation solver allows one to solve eq 8 efficiently and stably, given that no linear dependency issues arise in the $(P|Q\mathcal{Q})$ matrix. The eigenvalues of this matrix are thus always computed beforehand: if they are all above a threshold (10^{-5} by default), the linear equation solver is invoked; otherwise, inversion is carried out through a much more costly singular value decomposition procedure. Such linear dependency issues are likely to appear in periodic systems, which are considerably more closely packed than molecules, since molecular fitting basis sets are adopted.

After the solve step, a second bucket sort procedure is needed, in which point symmetry is exploited in order to obtain all objects needed for the assembly step in eq 10. This is applied twice, to the integrals $(ia\mathcal{A}|P\mathcal{P})$ file and to the coefficients $d^{P\mathcal{P}}_{ia\mathcal{A}}$ file. Once more, redistribution of data among processors takes place.

At this stage, $d^{P\mathcal{P}}_{ia\mathcal{A}}$ coefficients are in memory and distributed in the most convenient order, so that the following intermediate quantities are conveniently evaluated:

$$D^{P\mathcal{P}}_{ia\mathcal{A}} = \sum_{Q\mathcal{Q}} d^{Q\mathcal{Q}}_{ia\mathcal{A}}(Q|P\mathcal{P}') \tag{9}$$

These quantities are needed in the assembly step described in the next subsection. They are written on disk, and the bucket sort routine with symmetry and redistribution is applied in the same way as for three-index integrals and coefficients.

3.3.3. The Assembly Step. According to Dunlap's robust expression,[9] which guarantees that the error in the integral is second order with respect to the error in the fitting, the four-index two-electron integrals are approximated as

$$K^{ij\mathcal{J}}_{a\mathcal{A},b\mathcal{B}} \approx \sum_{P\mathcal{P}}(ia\mathcal{A}|P\mathcal{P})d^{P\mathcal{P}'}_{jb\mathcal{B}'} + \sum_{P\mathcal{P}} d^{P\mathcal{P}}_{ia\mathcal{A}}[(P\mathcal{P}'|jb\mathcal{B}') - D^{P\mathcal{P}'}_{jb\mathcal{B}'}] \tag{10}$$

with $D_{jb\mathcal{B}'}^{P\mathcal{P}'}$ factors as defined in eq 9. The three-index integrals (and coefficients), after the preceding bucket sort step, are stored on disk with the fitting function center as the slowest index and distributed among processors according to the same index. Symmetry has been exploited as well, so no rotations are needed at this stage.

The parallel implementation of the algorithm for the first term of eq 10 is sketched in Figure 4. Each processor performs *one* single complete read of integrals and coefficients, thus loading in memory all the quantities having the same set of P indices in common. The K integrals, at the end of the procedure, are distributed among CPUs according to the same strategy as in the LMP2 equations (next section). A partial sum is performed by each processor, and an all-to-one global sum is needed for each pair. This allows each processor to allocate only one K buffer, which is reused for each pair, and the number of such global sums depends on the size of the fitting basis set.

The parallel algorithm for the second term is identical, with "corrected" $[(P\mathcal{P}'|jb\mathcal{B}') - D_{jb\mathcal{B}'}^{P\mathcal{P}'}]$ integrals instead of normal ones. This algorithm guarantees a reasonable memory occupation: three buffers are allocated by each CPU, one for the three index integrals, one for the coefficients (these have the size of one fitting function "block"), and one for the K integrals.

3.4. Local MP2 Equations. Once all of the ingredients are available (integrals, Fock, and overlap matrices in the PAO basis) the orbital invariant local MP2 energy per unit cell can computed. It is expressed as a sum over pair contributions:

$$E_{\mathrm{cell}}^{\mathrm{LMP2}} = \sum_{i,j,\mathcal{J}} E_{i,j,\mathcal{J}}^{\mathrm{LMP2}} \tag{11}$$

Each pair contribution is obtained as

$$E_{i,j,\mathcal{J}}^{\mathrm{LMP2}} = \sum_{a,\mathcal{A},b,\mathcal{B}} K_{a\mathcal{A},b\mathcal{B}}^{ij\mathcal{J}} [2T_{a\mathcal{A},b\mathcal{B}}^{ij\mathcal{J}} - T_{b\mathcal{B},a\mathcal{A}}^{ij\mathcal{J}}] \tag{12}$$

where $T^{ij\mathcal{J}}$ is the amplitude corresponding to a two-electron excitation from a pair of WFs to a pair of PAOs.

Summations over $a\mathcal{A}$ and $b\mathcal{B}$ indices are truncated through the selection of non-negligible excitations according to the *locality Ansatz*. To the general WF i, a *domain* $\mathcal{D}_i$ is associated, consisting of a number of atoms close to it. $\mathcal{D}_i$ is usually defined through a Boughton−Pulay criterion.[66] A *pair-domain* $\mathcal{D}_{(ij)}$, assigned to a WF pair *ij*, is defined as the union of the corresponding WF domains. Only those excitations are retained for which, first, both PAOs a and b belong to atoms in $\mathcal{D}_{(ij)}$ and, second, the distance d_{ij} between the centers of the two WFs is within a certain value D. Following the so-called *frozen core* approximation, only *valence* WFs are usually considered.

The unknown amplitudes $T_{b\mathcal{B},a\mathcal{A}}^{ij\mathcal{J}}$ are obtained solving a set of linear equations which imposes on the $R_{a\mathcal{A},b\mathcal{B}}^{ij\mathcal{J}}$ residuals to be zero:

$$R_{a\mathcal{A},b\mathcal{B}}^{ij\mathcal{J}} = K_{a\mathcal{A},b\mathcal{B}}^{ij\mathcal{J}} + A_{a\mathcal{A},b\mathcal{B}}^{ij\mathcal{J}} + B_{a\mathcal{A},b\mathcal{B}}^{ij\mathcal{J}} = 0 \tag{13}$$

where the matrices **A** and **B** incorporate *internal* and *external* contributions to the residual, respectively, and are defined as follows:

$$A_{a\mathcal{A},b\mathcal{B}}^{i,j\mathcal{J}} = \sum_{c\mathcal{C},d\mathcal{D}} [F_{ac}^{\mathcal{C}'} T_{c\mathcal{C},d\mathcal{D}}^{i,j\mathcal{J}} S_{db}^{\mathcal{B}'} + S_{ac}^{\mathcal{C}'} T_{c\mathcal{C},d\mathcal{D}}^{i,j\mathcal{J}} F_{db}^{\mathcal{B}'}] \tag{14}$$

$$\begin{aligned} B_{a\mathcal{A},b\mathcal{B}}^{i,j\mathcal{J}} = & \sum_{c\mathcal{C},d\mathcal{D}} S_{ac}^{\mathcal{C}'} \sum_{k\mathcal{K}} [T_{c\mathcal{C},d\mathcal{D}}^{i,k\mathcal{K}} F_{kj}^{\mathcal{J}'} \\ & + F_{ik}^{\mathcal{K}} T_{c\mathcal{C}',d\mathcal{D}'}^{k,j\mathcal{J}'}] S_{db}^{\mathcal{B}'} \end{aligned} \tag{15}$$

S_{ab} (F_{ab}) denotes the element of the overlap (Fock) matrix between functions a and b, and the simplifications due to translational symmetry are implicitly introduced. The range of the $k\mathcal{K}$ summation is suitably truncated according to *locality* criteria in order to include only significant contributions.

The condition expressed by eq 13 is reached iteratively by minimizing a suitable Hylleraas functional. At each step of the iterative procedure, the residuals are used to update the amplitudes. Due to the redundant character of PAOs, the update must be performed in the orthogonal basis proper of the pair $i,j\mathcal{J}$, the so-called local orthonormal (LON) basis. The matrices to perform this unitary transformation are obtained by diagonalizing the Fock matrix in the local pair domain. The set of transformations, one for each WF pair, is generated once before entering the iterative procedure and stored on disk.

The update is then expressed as

$$\Delta\tilde{T}_{a\mathcal{A},b\mathcal{B}}^{i,j\mathcal{J}} = \frac{\tilde{R}_{a\mathcal{A},b\mathcal{B}}^{i,j\mathcal{J}}}{\varepsilon_a + \varepsilon_b - f_{ii} - f_{jj}} \tag{16}$$

with the tilde symbol indicating that $\tilde{T}$ and $\tilde{R}$ are in the LON basis representation. Only symmetry irreducible amplitudes are computed, updated, and stored during the procedure. The amplitudes needed in eq 14 are obtained on the fly from the irreducible ones by applying suitable symmetry operators.

Parallelization of the iterative LPM2 equations is conveniently performed by distributing the workload according to $i,j\mathcal{J}$ pairs (eq 11), but an efficient parallel algorithm is difficult to achieve. In fact, in the evaluation of *external* residuals, term B in eq 14, which accounts for the majority of the time spent in the iterative procedure, amplitudes from different WF pairs contribute to each $i,j\mathcal{J}$ pair. Nielsen and Janssen[20] have proposed two different parallel algorithms for the molecular LMP2 equations: (i) based on replication of the amplitudes and (ii) based on distribution of the amplitudes according to a smart assignment of the workload for each processor. They conclude that scheme i is more load-balanced but less efficient, since it involves a global summation step, while scheme ii is more efficient, since only a redistribution of amplitudes is needed but suffers from load imbalance when a large number of processors is used. A 60% performance has been demonstrated for scheme i when 50 processors are used.

A fundamental feature of periodic LMP2 equations is the extensive use of translational and point symmetry. This makes algorithm ii of Nielsen and Janssen not practicable, since a huge number of additional, globally blocking, communications would be needed, along with a significant overhead for the bookkeeping of all these operations. A replicated memory (scheme i) strategy has therefore been adopted, with a difference: communication among processors takes place only at the end of one iteration, and not after each update. This has some drawbacks: the updates performed by a single processor cannot take advantage of already updated amplitudes (Gauss−Siedel), and therefore the convergence can be slower. Thanks to the use of disk sharing (and paging) strategies for the amplitudes, this problem is softened, since all processors within a group can read from disk amplitudes that other members of the group have updated and stored.

Regarding the distribution of data, LON transformation matrices, **R** residuals, and **K** integrals are distributed according to the pair index, while **T** amplitudes are replicated—but buffered. Parallelization is straightforward, since no communication is needed during one iteration.

The implemented algorithm (Figure 5) contains also instructions to avoid one processor sending the data to its own group leader. As a consequence of that, in the case of all CPUs sharing the same disk, no MPI communications are needed but rather a final I/O step in which all processors write to disk (all in the same file) the updated amplitudes.

```
Loop over all $i, j, \mathcal{J}$ pairs
  If (my $i, j, \mathcal{J}$ pair) then
    If (not in memory) then
      read $T_{i,j\mathcal{J}}$
    Else
      write $T_{i,j\mathcal{J}}$
    End If
    send $T_{i,j\mathcal{J}}$ to leaders of other CPU groups
  Else if (I am a group leader) then
    receive $T_{i,j\mathcal{J}}$
    write $T_{i,j\mathcal{J}}$
  EndIf
End $i, j, \mathcal{J}$ loop
```

Figure 5. Scheme of the communication algorithm for the updated amplitudes in LMP2 equations.

Thanks to paging routines, only the amplitudes that can fit in memory are loaded. As a limiting case, in case of very small memory or a very large basis set, only a block of each matrix is kept in memory at once, and each of these blocks has the size of the number of functions *in a local domain* squared—which is constant with the increasing number of atoms in the unit cell.

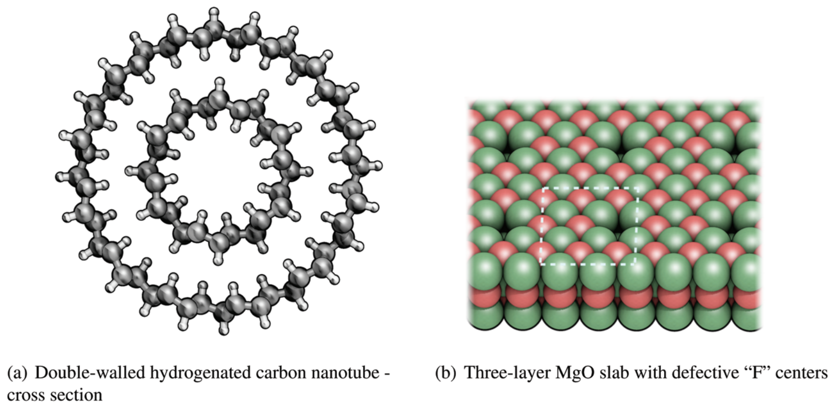
4. RESULTS

As discussed in the previous section, the periodic LMP2 program is complex and formed by several parts. The parallel efficiency of these different parts has been benchmarked on a small but significative set of periodic crystalline systems, representing a choice of interesting problems, in the field of solid state quantum chemistry, in which the LMP2 method can give

Table 2. Information on the Crystalline Systems Used As a Benchmark[a]

system	$n_{\text{atoms}}^{\text{cell}}$	n_{elec}^{cell}	basis set	$n_{\text{ao}}^{\text{cell}}$	n_{pairs}
MgO slab with "F" center	54	210	Mg: [4s3p2d1f]	1916	15663
			O: [4s3p3d2f]		
double-walled nanotube	144	360	aug(p,d)-cc-pVDZ	1944	14439
aspirin on SiO_2	141	652	slab: 88-31G*	2465	72426
			molecule: TZP		
MOF-5	106	532	cc-pVTZ	2884	28562

[a] $n_{\text{elec}}^{\text{cell}}$ is the number of correlated electrons per unit cell. $n_{\text{ao}}^{\text{cell}}$ is the number of basis functions in a unit cell; n_{pairs} is the number of Wannier function pairs included in the calculation with the chosen thresholds for the LMP2 truncations.


(a) Double-walled hydrogenated carbon nanotube - cross section
(b) Three-layer MGO slab with defective "F" centers

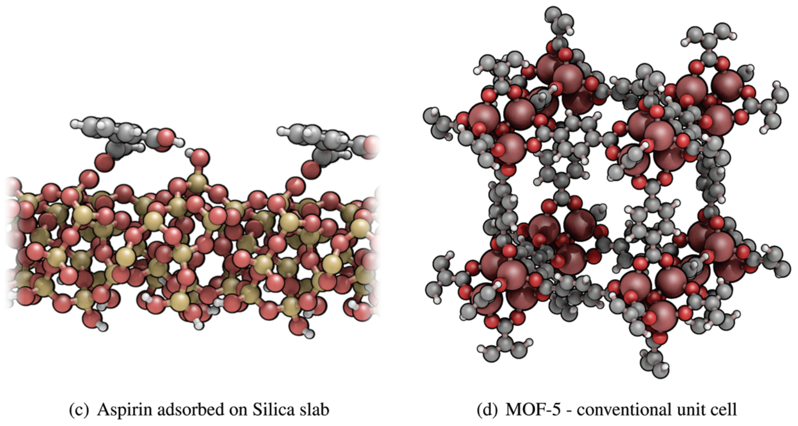

(c) Aspirin adsorbed on Silica slab
(d) MOF-5 - conventional unit cell

Figure 6. The systems used for benchmark calculations. In panel b, the repetitive unit is marked by a dashed white box.

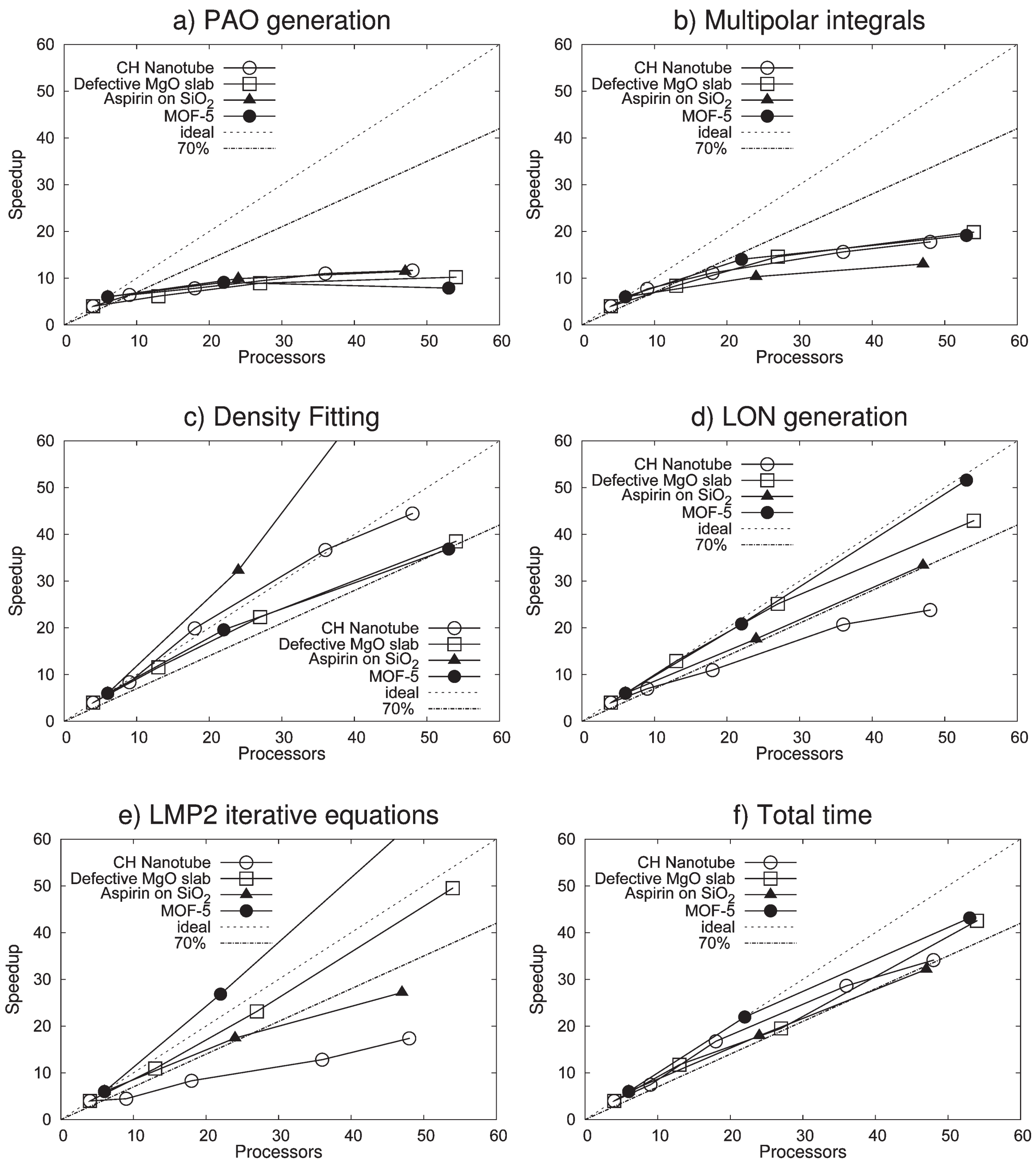


Figure 7. Parallel performance of the different parts of the LMP2 algorithm. Dotted lines indicate the ideal performance (100%) and the 70% performance, considered as an optimal target for the present work.

important answers. The efficiency and features of the scalar algorithm have been discussed in previous works[35] and will not be addressed here. In particular, quasi-linear scaling behavior has been demonstrated with respect to the number of atoms in the unit cell.[52]

4.1. The Test Systems. The set of test systems has been defined in order to cover many of the several geometrically and chemically different situations that can be encountered when dealing with periodic crystals. Systems periodic in one, two, and three dimensions have been considered. Ionic, covalent, and weakly bound, closely packed, as well as microporous materials are included. Let us discuss more in detail the features of each benchmark system. Pictures of the crystalline structure can be found in Figure 6, and some reference data about each test case are reported in Table 2.

Double-Walled Hydrogenated Carbon Nanotube. Graphene and carbon nanotubes (CNTs) are among the "hot" nanomaterials in today's science. One way to tune the chemical and physical properties of CNTs is by means of chemical functionalization, for example, by hydrogenation.[67] A double-walled CNT, composed of two fully hydrogenated tubes, has been chosen as an interesting test case, because of the dispersive interactions that take place among the two tubes. The unit cell of this 1D periodic system is made of 144 atoms and represents the section perpendicular to the tube length (see Figure 6a). The basis set adopted is a cc-pVDZ[68] augmented with polarization functions for high angular momenta.

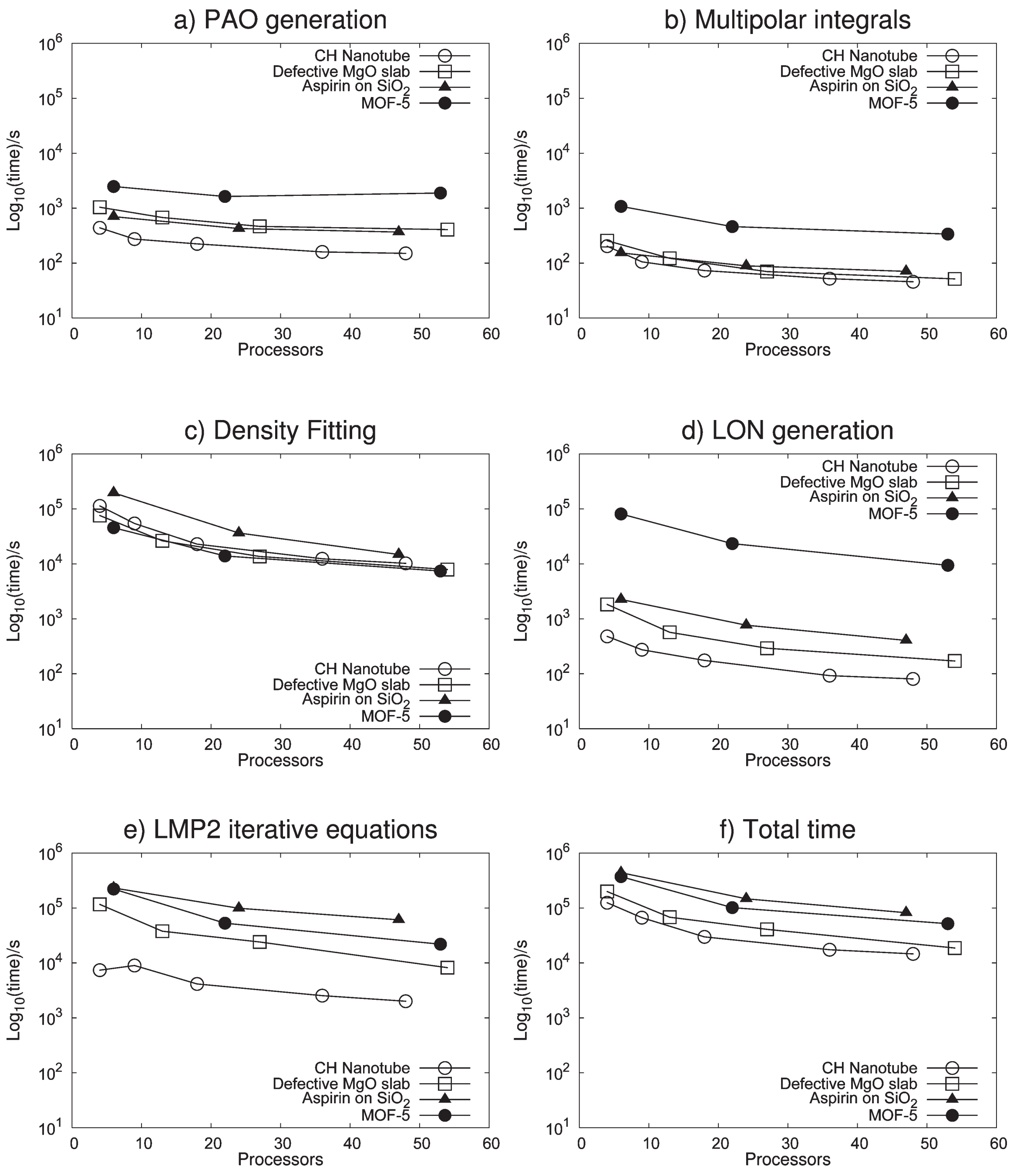


Figure 8. Total wall clock timings (reported in logarithmic scale, in seconds) of the different parts of the LMP2 parallel algorithm.

MgO Three-Layer Slab with a Defective "F" Center. Magnesium oxide, as a bulk crystalline system or as a modeled surface, is and has been a favorite system for periodic *ab initio* calculation, both for its simple crystalline structure and for its interesting chemical properties. In particular, defective sites on the surface are the key to many catalysis processes, of great relevance in modern industry. When modeling a defect in a periodically repeated structure, one has to define a supercell in order to mimic the defect concentration of a real sample. In this way, the computational cost of the calculation is greatly increased. The thickness of the slab is an important factor as well, since one wants generally to reproduce the features of a semi-infinite system. The test case used in this work is a three-layer slab, commonly recognized as a reasonable thickness for the modeling of many surface properties, with a 3×3 supercell, resulting in 54 atoms per cell. One oxygen atom of the upper surface is then removed and replaced by two paired electrons, thus creating a diamagnetic F center, one of the most well-known widely diffused defects in rocksalt structures. The basis set is also a good one, with [4s3p2d1f] functions on Mg and [4s3p3d2f] on oxygen,[52] and cc-pVTZ for hydrogen on the F center.

Aspirin Molecule on Silica Substrate. Adsorption of molecules on surfaces is both a technologically interesting process and a challenge for quantum chemistry, since it is ruled mainly by dispersive interactions. A particular case is that of the adsorption of organic or pharmaceutical molecules on biocompatible

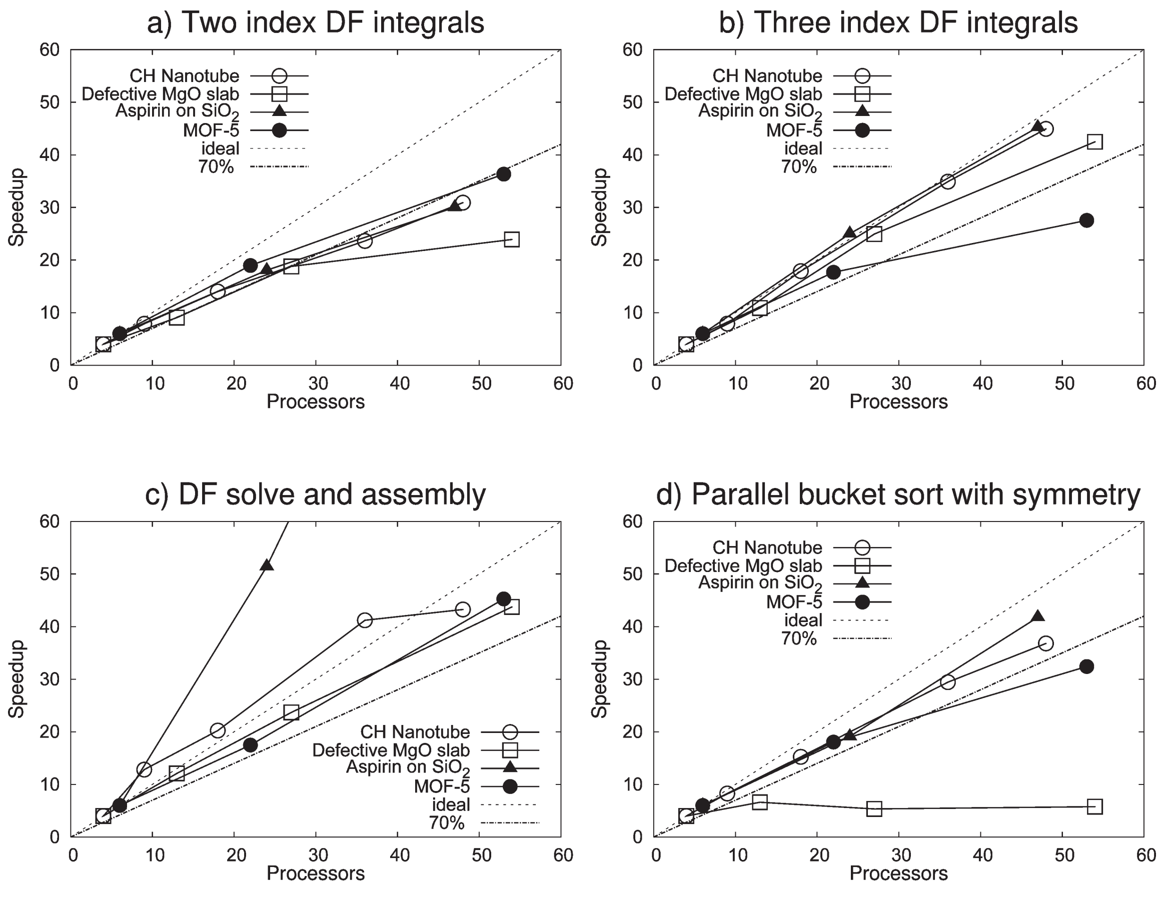


Figure 9. Detail of parallel performance in different steps of the periodic density fitting procedure. See caption of Figure 7 for details.

substrates.[69] In the present test, kindly provided by Piero Ugliengo, the substrate is represented by a ~10-Å-thick slab of SiO_2, with 141 atoms in the unit cell, and the adsorbate is a molecule of aspirin, a well-known pharmaceutical compound. A good basis set has been used: 88−31G* on Si and O atoms in the slab, TZP on the molecule's atoms.

Metal−Organic Framework (MOF-5). Storage of small gases like hydrogen or CO_2 inside microporous materials is presently one of the main technological challenges. In the past few years, MOFs have become universally popular and recognized as one of the most promising classes of porous materials, with surface areas higher than that of commonly used zeolites.[70] Sillar et al., in 2009, wrote: "...use of even the simplest of [post Hartree-Fock methods], second-order Møller-Plesset perturbation theory (MP2), is far beyond present computational resources for periodic structures with large unit cells, such calculations are performed only on finite-sized models that represent individual structural elements of the whole MOF framework".[71] It is here demonstrated that this is now feasible, with a basis set such as cc-pVTZ with Stuttgart−Dresden ECPs on the Zn atom, at a reasonable computational cost. Problems have been encountered though, trying to reach Hartree−Fock SCF convergence with the unmodified molecular cc-pVTZ basis; since it is beyond the scope of the present work to optimize a suitable basis set for periodic calculations, Wannier functions obtained through a B3LYP Hamiltonian have been used instead of HF ones. The energy results are not to be discussed in this paper, and since this procedure is totally equivalent from a computational point of view, this detail is not relevant to the conclusions on code performance.

4.1.1. Computational Parameters. The parameters relevant to the following discussion—that is, with a significant impact on the cost of the LMP2 calculation—are those related to the local treatment of electron correlation. In all test cases, parameters compatible with real-life calculations have been chosen. Only valence electrons are correlated, according to the commonly accepted *frozen core* approximation. WF pairs are explicitly included up to 10 Å in the smallest cases, 9 Å in the largest. Integrals are evaluated through density fitting for pairs up to 6 Å and through multipolar expansion beyond that. Excitation domains are always defined according to a Boughton−Pulay parameter[66] of 0.990, larger than the commonly used value of 0.985.

4.2. Benchmark Calculations. Computations reported in this section were performed on the IBM SP Power 6 cluster at CINECA supercomputing facilities in Bologna, Italy. The system features 4.7 GHz cores, Infiniband network connection, and shared disk file system. In all calculations, 8 GB of RAM memory have been made available to each processor, independently from the number of processors used.

In Figure 7, the parallel performance of the different parts of the code is reported. The speedup is defined, for a given number of processors N, as $(t_{ref} \times N_{ref})/t_N$. The reference number of processors is for all systems set to 4 or 6, since it is not possible to run these calculations on a smaller number of processors (this comes out quite clearly from the use of memory in Figure 3). In Figure 8, the wall clock timings are reported, on a logarithmic scale, for the same pieces of code, in order to highlight the different calculation weights and to give a feeling of the relative cost of the same algorithm on different systems, which can be quite different.

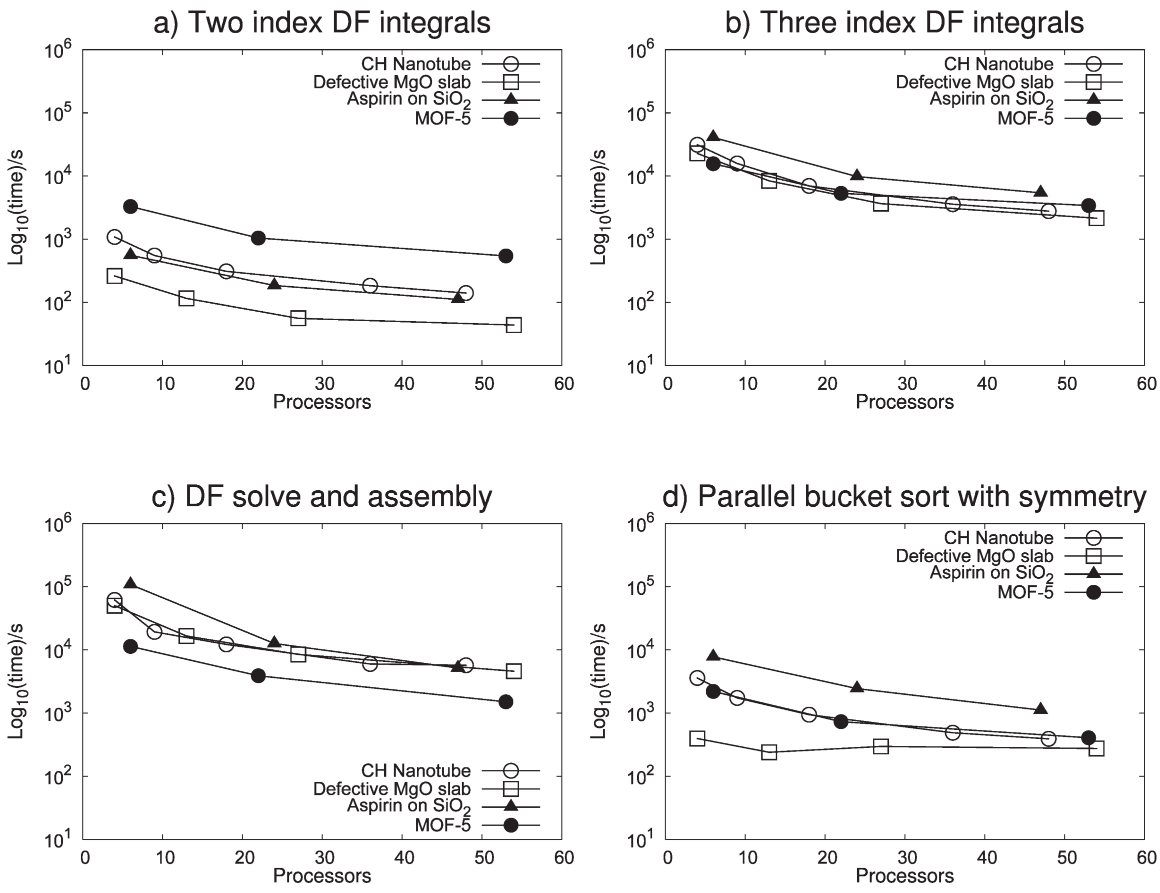


Figure 10. Total wall clock timings (reported in logarithmic scale, in seconds) of the different steps of periodic density fitting procedure.

At first glance, the reader can notice that the four different test systems show a very different behavior in the different parts, especially in panels c, d, and e, which cover most of the execution time. Despite these differences, the global performance of the code, evaluated on wall clock timings, is very similar for all the systems and is comprised between 70% and 80% at the highest number of processors. Correspondingly, the total execution times are reduced by almost an order of magnitude.

The first two panels concern the generation of PAOs and related matrices and the multipolar integrals calculation. The poor parallel scaling behavior is explained by the small elapsed time of these pieces of code, since serial code and I/O take a leading role when the cost of the parallelized part is minor. Besides, it is known that Scalapack routines do not have an ideal scaling for this matrix size. As a matter of fact, as already discussed the purpose of this piece of code is to mostly distribute the memory load, which is achieved quite efficiently (Figure 3).

Evaluation of integrals through the density fitting approximation is definitely more relevant to the present discussion. The cost on a low number of processors ranges from half a day to two days but is generally very similar, if compared with the large differences (several order of magnitude) which are observed in the elapsed time of other parts of the code. The largest calculation, MOF-5, has the lowest time since in this microporous lattice the prescreenings are very efficient, and thus the number of integrals to be computed is reduced. The behavior of the test set when increasing the number of processors is very heterogeneous, even showing superlinear scaling phenomena, and it needs more careful examination.

In Figures 9 and 10, speedups and total timings are reported for the most significant steps of the DF algorithm, in analogy with Figures 6 and 7. Two-index integrals are quite cheap, and no worries come from some nonoptimal speedup, which is actually observed when the CPUs outnumber the unit cells to be computed. The performance of the three-index integrals code is generally very good and deteriorates at a high number of processors only for MOF-5. This system contains different elements such as zinc and hydrogen and suffers from load-balancing problems, due to the coarse grain parallelization according to the atomic center, as discussed in section 3.3.1. The most expensive step, solve and assembly, shows superlinear scaling behavior for two of out the four systems. The two-index auxiliary integrals are here handled through a paging strategy (see section 2.2), and the algorithm greatly benefits from the larger amount of total memory available.

The analysis of the parallel bucket sort with symmetry exploitation is quite surprising. First, its cost is non-negligible, though generally an order of magnitude lower than the most important DF steps. Second, it scales very well in three out of the four test cases, with speedups up to 90% at 50 CPUs. This is in fact a combination of different factors: (i) the better performance of the algorithm due to the increasing of the total available memory, which brings superlinear scaling, (ii) sublinear scaling due to concurrent access to the disk and locally blocking steps, and (iii) the fact that no all-to-all communications take place, but always one-to-one, making this step non-globally-blocking. In the case of a globally blocking algorithm, the speedup would be constant. In the case of MgO, the overall cost is small, thus making irrelevant point i above and point ii dominant.

The last two steps of the LMP2 algorithm to be discussed are the LON generation and the LMP2 iterative equations, reported in panels d and e of Figures 6 and 7. The first is important only in the case of MOF-5; this fact is a consequence of a very good basis set (cc-pVTZ) and large excitation domains, since the diagonalization of matrices of the size of the number of basis functions in the domain is involved—which scales as $\mathcal{O}(n^3)$. Fortunately, the parallel performance is close to ideal for this system, while it is worse for others only due to the fact that, since the elapsed time is small, serial parts of the code or noise become significant.

The LMP2 equations, described in section 3.4, are the most delicate and complex part of the parallel implementation, and the same experience is reported for nonperiodic implementations.[20] Again, a very heterogeneous behavior is observed in this case, ranging from superlinear scaling to sublinear. For this reason, let us analyze separately the four test cases: (i) The CH double-walled nanotube has a small basis set and is not a dense system. The pair excitation domains contain few centers and few functions per center; the number of external contributions to the update of a single amplitude is also small. As a consequence, the cost of the iterations is not very significant and is mainly dominated by I/O. (ii) The aspirin on SiO_2 two-dimensional system is rather dense, and the calculation involves a high number of WF−WF pairs (Table 2), but excitation domains are constituted of few centers, since WFs are very localized, and the cost of evaluating a single residual is small. A large part of the time resides also in this case in the I/O. (iii) MgO is the most dense among the test cases, and the number of external contributions in eq 14 is very large. The cost of computing residuals then dominates with respect to I/O, so the performance is close to ideal. (iv) MOF-5 is a microporous system, so even if it is periodic in three dimensions, not many pairs are included in the calculation. The WFs are not well localized though, due to aromatic rings, so large excitation domains are defined by the chosen Boughton−Pulay threshold. The local approach is then not at its peak efficiency, and large matrices have to be handled. For this reason, the paging algorithm is intensively used, and as already observed for the solve step of the DF procedure, superlinear scaling is observed due to the increasing of the global amount of memory when the number of processors is increased. We recall here that, thanks to the disk-sharing strategy described in section 2.3, no communication among processors takes place in all of the LMP2 iterations, since this is substituted by writing on the disk unit shared by all processors.

As mentioned above, all of the different parts of the code, behaving differently on the different systems, do average in the end, resulting in quite similar speedups for all of the test systems. The performance is satisfactory in all cases, and the total time required to complete a job ranges from 4 h in the smallest case (CH nanotube) to 23 h (MOF-5).

5. CONCLUSION AND PROSPECTS

A parallel implementation of the periodic local MP2 code of the Cryscor program has been presented and thoroughly tested. It is seen that scaling with the number of processors is favorable up to more than 50 processors, with parallel performance up to 70−80% with respect to the ideal behavior.

The challenges posed by the full exploitation of symmetry at different levels present throughout the code have been overcome by adopting simple yet powerful strategies: (i) to enclose all communication in well-defined and separate pieces of code, thus relying on coarse-grain parallelization, and (ii) to fruitfully use the disk, as far as it is possible, as a communication medium among CPUs who share the same disk space.

The behavior of the four periodic systems in the test set, chosen to be different as concerns the number of periodic dimensions, density, and chemical composition, is very heterogeneous in specific parts of the code, while the speedups evaluated on the total wall clock time are very similar. This can be a good indication that this general purpose code will perform reasonably well on any kind of crystalline system.

As a final result, the calculation of MP2 energy of a MOF-5 three-dimensional periodic crystal with a cc-pVTZ basis (106 atoms, 2884 basis functions in the unit cell) is achieved in less than 24 h on 53 processors. The parallelized LMP2 code will be soon released as an update to the Cryscor program, making this work available to the scientific community.

A massive parallel implementation, i.e., capable of efficiently running on hundreds of processors, would require a complete recoding of the LMP2 procedure, extending the use of linear algebra routines (Scalapack) to the whole code. As a perspective, the power of implementations taking advantage of GPU cores is emerging nowadays,[72] which opens the way to an innovative concept of the parallelization codes.

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: lorenzo.maschio@unito.it.

■ ACKNOWLEDGMENT

The author is grateful to Prof. Cesare Pisani, Dr. Ian J. Bush, Dr. Denis Usvyat, and Prof. Martin Schütz for fruitful discussions. Mikko Linnolahti, Piero Ugliengo, and Bartolomeo Civalleri have kindly provided the input structures for the double walled nanotube, the aspirin adsorbed on silica, and MOF-5 test systems, respectively. 3D images of crystalline structures have been realized with Qutemol.[73] Calculations have been performed at the CINECA facilities thanks to grants HP10C5J7E1-QCECSSC and HP10CNWQPB-TPC.

■ REFERENCES

(1) Werner, H.-J. et al. *MOLPRO*, version 2010.1; University College Cardiff Consultants Limited: Cardiff, Wales, 2010. See http://www.molpro.net (accessed Aug 2011).

(2) *TURBOMOLE*, V6.2; University of Karlsruhe and Forschungszentrum Karlsruhe GmbH: Karlsruhe, Germany, 2010. Available from http://www.turbomole.com (accessed Aug 2011).

(3) Doser, B.; Zienau, J.; Clin, L.; Lambrecht, D. S.; Ochsenfeld, C. *Z. Phys. Chem.* **2010**, *224*, 397–412.

(4) Lambrecht, D. S.; Ochsenfeld, C. *J. Chem. Phys.* **2005**, *123*, 184101.

(5) Lambrecht, D. S.; Doser, B.; Ochsenfeld, C. *J. Chem. Phys.* **2005**, *123*, 184102.

(6) Izmaylov, A.; Scuseria, G. E.; Frisch, M. J. *J. Chem. Phys.* **2006**, *125*, 104103.

(7) Feyereisen, M.; Fitzgerald, G.; Komornicki, A. *Chem. Phys. Lett.* **1993**, *208*, 359.

(8) Pedersen, T. B.; Aquilante, F.; Lindh, R. *Theor. Chem. Acc.* **2009**, *124*, 1–10.

(9) Dunlap, B. *Phys. Chem. Chem. Phys.* **2000**, *2*, 2113–2116.

(10) Werner, H.; Manby, F.; Knowles, P. *J. Chem. Phys.* **2003**, *118*, 8149.

(11) Pulay, P. *Chem. Phys. Lett.* **1983**, *100*, 151.

(12) Saebø, S.; Pulay, P. *Chem. Phys. Lett.* **1985**, *113*, 13.
(13) Saebø, S.; Pulay, P. *Annu. Rev. Phys. Chem.* **1993**, *44*, 213.
(14) Häser, M.; Almlöf, J. *J. Chem. Phys.* **1992**, *96*, 489.
(15) Schütz, M.; Lindh, R. *Theor. Chem. Acc.* **1997**, *95*, 13–34.
(16) Mitin, A.; Baker, J.; Wolinski, K.; Pulay, P. *J. Comput. Chem.* **2003**, *24*, 154–160.
(17) Aikens, C.; Gordon, M. *J. Phys. Chem. A* **2004**, *108*, 3103–3110.
(18) Fossgard, E.; Ruud, K. *J. Comput. Chem.* **2006**, *27*, 326–333.
(19) Ishimura, K.; Pulay, P.; Nagase, S. *J. Comput. Chem.* **2006**, *27*, 407–413.
(20) Nielsen, I. M. B.; Janssen, C. L. *J. Chem. Theory Comput.* **2007**, *3*, 71–79.
(21) Lotrich, V.; Flocke, N.; Ponton, M.; Yau, A. D.; Perera, A.; Deumens, E.; Bartlett, R. J. *J. Chem. Phys.* **2008**, *128*, 194104.
(22) Janowski, T.; Pulay, P. *J. Chem. Theory Comput.* **2008**, *4*, 1585–1592.
(23) Harding, M. E.; Metzroth, T.; Gauss, J.; Auer, A. A. *J. Chem. Theory Comput.* **2008**, *4*, 64–74.
(24) Katouda, M.; Nagase, S. *Int. J. Quantum Chem.* **2009**, *109*, 2121–2130.
(25) Ayala, P.; Kudin, K.; Scuseria, G. *J. Chem. Phys.* **2001**, *115*, 9698.
(26) Stoll, H. *Phys. Rev. B* **1992**, *46*, 6700–6704.
(27) Paulus, B. *Phys. Rep.* **2006**, *428*, 1–52.
(28) Pisani, C.; Busso, M.; Capecchi, G.; Casassa, S.; et al. *J. Chem. Phys.* **2005**, *122*, 094113.
(29) Manby, F.; Alfè, D.; Gillan, M. *Phys. Chem. Chem. Phys.* **2006**, *8*, 5178–5180.
(30) Hirata, S. *J. Chem. Phys.* **2008**, *129*, 204104.
(31) Marsman, M.; Grüneis, A.; Paier, J.; Kresse, G. *J. Chem. Phys.* **2009**, *130*, 184103.
(32) Gruneis, A.; Marsman, M.; Kresse, G. *J. Chem. Phys.* **2010**, *133*, 074107.
(33) Hermann, A.; Schwerdtfeger, P. *Phys. Rev. Lett.* **2008**, *101*, 183005.
(34) Halo, M.; Casassa, S.; Maschio, L.; Pisani, C. *Chem. Phys. Lett.* **2009**, *467*, 294–298.
(35) Pisani, C.; Maschio, L.; Casassa, S.; Halo, M.; Schütz, M.; Usvyat, D. *J. Comput. Chem.* **2008**, *29*, 2113–2124.
(36) Pisani, C.; Casassa, S.; Maschio, L.; Schütz, M.; Usvyat, D. CRYSCOR09. See www.cryscor.unito.it (accessed Aug 2011).
(37) Dovesi, R.; Saunders, V. R.; Roetti, C.; Orlando, R.; Zicovich-Wilson, C. M.; Pascale, F.; Doll, K.; Harrison, N. M.; Civalleri, B.; Bush, I. J.; D'Arco, P.; Llunell, M. CRYSTAL09 User's Manual. http://www.crystal.unito.it/Manuals/crystal09.pdf (accessed Aug 2011).
(38) Dovesi, R.; Orlando, R.; Civalleri, B.; Roetti, C.; Saunders, V.; Zicovich-Wilson, C. *Z. Kristallogr.* **2005**, *220*, 571–573.
(39) Schütz, M.; Werner, H. *J. Chem. Phys.* **2001**, *114*, 661.
(40) Schütz, M.; Hetzer, G.; Werner, H. *J. Chem. Phys.* **1999**, *111*, 5691.
(41) Zicovich-Wilson, C.; Dovesi, R.; Saunders, V. *J. Chem. Phys.* **2001**, *115*, 9708.
(42) Casassa, S.; Zicovich-Wilson, C.; Pisani, C. *Theor. Chem. Acc.* **2006**, *116*, 726–733.
(43) Maschio, L.; Usvyat, D. *Phys. Rev. B* **2008**, *78*, 73102.
(44) Maschio, L.; Usvyat, D.; Civalleri, B. *CrystEngComm* **2010**, *12*, 2429–2435.
(45) Maschio, L.; Usvyat, D.; Schütz, M.; Civalleri, B. *J. Chem. Phys.* **2010**, *132*, 134706.
(46) Erba, A.; Pisani, C.; Casassa, S.; Maschio, L.; Schütz, M.; Usvyat, D. *Phys. Rev. B* **2010**, *81*, 165108.
(47) Martinez-Casado, R.; Mallia, G.; Usvyat, D.; Maschio, L.; Casassa, S.; Schütz, M.; Harrison, N. M. *J. Chem. Phys.* **2011**, *134*, 014706.
(48) Halo, M.; Casassa, S.; Maschio, L.; Pisani, C.; Dovesi, R.; Ehinon, D.; Baraille, I.; Rèrat, M.; Usvyat, D. *Phys. Chem. Chem. Phys.* **2011**, *13*, 4434–4443.
(49) Halo, M.; Pisani, C.; Maschio, L.; Casassa, S.; Schütz, M.; Usvyat, D. *Phys. Rev. B* **2011**, *83*, 035117.
(50) Erba, A.; Maschio, L.; Salustro, S.; Casassa, S. *J. Chem. Phys.* **2011**, *134*, 074502.
(51) Erba, A.; Itou, M.; Sakurai, Y.; Ito, M.; Casassa, S.; Maschio, L.; Terentjevs, A.; Pisani, C. *Phys. Rev. B* **2011**, *83*, 125208.
(52) Schütz, M.; Usvyat, D.; Lorenz, M.; Pisani, C.; Maschio, L.; Casassa, S.; Halo, M. Density fitting for correlated calculations in periodic systems. In *Accurate Condensed-Phase Quantum Chemistry*; Manby, F. R., Ed.; CRC Press: Boca Raton, FL, 2010; pp 29−56.
(53) Ishimura, K.; Pulay, P.; Nagase, S. *J. Comput. Chem.* **2006**, *27*, 407.
(54) Baker, J.; Pulay, P. *J. Comput. Chem.* **2002**, *23*, 1150.
(55) Nakao, Y.; Hirao, K. *J. Chem. Phys.* **2004**, *120*, 6375.
(56) Hättig, C.; Hellweg, A.; Köhn, A. *Phys. Chem. Chem. Phys.* **2006**, *8*, 1159.
(57) Shutler, P. M. E.; Sim, S. W.; Lim, W. Y. S. *Comput. J.* **2008**, *51*, 451–469.
(58) Yoshimine, M. *J. Comput. Phys.* **1973**, *11*, 449.
(59) Ford, A. R.; Janowski, T.; Pulay, P. *J. Comput. Chem.* **2006**, *28*, 1215.
(60) Maschio, L.; Usvyat, D.; Manby, F.; Casassa, S.; Pisani, C.; Schütz, M. *Phys. Rev. B* **2007**, *76*, 75101.
(61) Blackford, L. S.; Choi, J.; Cleary, A.; D'Azevedo, E.; Demmel, J.; Dhillon, I.; Dongarra, J.; Hammarling, S.; Henry, G.; Petitet, A.; Stanley, K.; Walker, D.; Whaley, R. C. *ScaLAPACK User's Guide*; Society for Industrial and Applied Mathematics: Philadelphia, PA, 1997.
(62) Hetzer, G.; Pulay, P.; Werner, H. *Chem. Phys. Lett.* **1998**, *290*, 143.
(63) Helgaker, T.; Jørgensen, P.; Olsen, J. *Molecular Electronic-Structure Theory*; Wiley: New York, 2000; pp 410−416.
(64) Manby, F.; Knowles, P. *Phys. Rev. Lett.* **2001**, *87*, 163001.
(65) Izmaylov, A.; Scuseria, G. *Phys. Chem. Chem. Phys.* **2008**, *10*, 3421–3429.
(66) Boughton, J. W.; Pulay, P. *J. Comput. Chem.* **1993**, *14*, 736–740.
(67) Tanskanen, J.; Linnolahti, M.; Karttunen, A. J.; Pakkanen, T. A. *ChemPhysChem* **2008**, *9*, 2390–2396.
(68) Kendall, R.; T.H. Dunning, J.; Harrison, R. *J. Chem. Phys.* **1992**, *96*, 6796.
(69) Rimola, A.; Civalleri, B.; Ugliengo, P. *Phys. Chem. Chem. Phys.* **2010**, *12*, 6357–6366.
(70) Civalleri, B.; Napoli, F.; Noel, Y.; Roetti, C. *CrystEngComm* **2006**, *8*, 364–371.
(71) Sillar, K.; Hofmann, A.; Sauer, J. *J. Am. Chem. Soc.* **2009**, *131*, 4143–4150.
(72) Vogt, L.; Olivares-Amaya, R.; Kermes, S.; Shao, Y.; Amador-Bedolla, C.; Aspuru-Guzik, A. *J. Phys. Chem. A* **2008**, *112*, 2049–2057.
(73) Tarini, M.; Cignoni, P.; Montani, C. *IEEE Trans. Visualization Comput. Graphics* **2006**, *12*, 1237–1244.