Parallel Computing: Technology Trends
I. Foster et al. (Eds.)

doi:10.3233/APC200060

BERTHA and PyBERTHA: State of the Art for Full Four-Component Dirac-Kohn-Sham Calculations

Loriano Storchi [a,b] Matteo De Santis [b,c] Leonardo Belpassi [b]

[a] *Dipartimento di Farmacia, Universitá degli Studi 'G. D'Annunzio', Via dei Vestini 31, 66100 Chieti, Italy*

[b] *Istituto di Scienze e Tecnologie Molecolari, Consiglio Nazionale delle Ricerche c/o Dipartimento di Chimica, Biologia e Biotecnologie, Universitá degli Studi di Perugia, Via Elce di Sotto 8, 06123 Perugia, Italy*

[c] *Dipartimento di Chimica, Biologia e Biotecnologie, Universitá degli Studi di Perugia, Via Elce di Sotto 8, 06123 Perugia, Italy*

Abstract.
While since mid-seventies it has been clearly shown that relativistic effects play a crucial role for the complete understanding of the chemistry of molecules especially when heavy elements are involved, still nowadays in most of the case the relativistic effects are introduced via approximate methods. The main motivation behind the introduction of such approximation, respect to the natural and most rigorous component (4c) formalism derived from the Dirac equation, is the computational burden. In the present paper we are proposing a review of the BERTHA code that, together with the recently introduced Python bindings, represents the state of the art for full 4c calculations both in term of performances as well as in terms of code usability.

Keywords. four-component Dirac-Kohn-Sham

1. Introduction

Relativistic effects, arising by the fast moving of the core electrons and propagating into the valence region, it has been largely shown to become very important for the proper understanding of the chemical properties of molecules [1,2]. Indeed, expecially when heavy or Super-Heavy atom are involved the inclusion of relativistic effects is fudamental also in the deep understanding of the chemical bond [3,4]

More recently the developments of new Free Electron Lasers (FEL) provide a range of opportunities to achieve significant advances that extend the boundaries of our knowledge in the field of atomic and molecular science and promise to obtain direct information on the basis processes of energy relaxation/ transfer in molecules (how charge, spin, orbital and eventually nuclear degrees of freedom interact to redistribute the energy). The development of accurate theoretical and computational methods, based on first principles, for the accurate characterization of electronic dynamics including spin-orbit cou-

pling in molecular systems containing heavy atoms is now one of the most important challenges of theoretical chemistry and computational science [5].

Because of the computational cost of the rigorous way to include relativity (including spin-orbit effect) in the modelling of molecular systems, a disparate set of approximate methods have been derived from the strictly relativistic 4-component (4c) formalism derived from the Dirac equation by Bertha Swirles [6]. Among these the so called 2-components approximation, deriving from the decoupling the "large" and "small" components of the Dirac spinors [7], is the most used. Maybe the Douglas-Kroll-Hess [8] and Zero Order Regular Approximation hamiltonians [9] are, amoung others, the most popular two-component schemes. Both of them have found a wide range of applications with implementations in several modern commercial codes.

Clearly the introduction of the cited approximation scheme is motivated by the intrisic computational difficulty realted to a proper full 4c approach. The BERTHA code, we will describe here, is basically built around a smart and efficient algorithm for the analytical evaluation of relativistic electronic repulsion integrals, developed by Quiney and Grant in Oxford more than a decade ago [10], representing the relativistic generalization of the well-known McMurchie-Davidson algorithm [11]. As we will show in the following we have extended the applicability range of all-electron DKS calculations, exploiting density fitting techniques and parallelization strategies, to large clusters of heavy metals [12].

Aside what previously stated more recently we introduced a maior improvement in the BERTHA code usability introducing a set of Python bindings [13]. In the present paper, after a review of the parallelization strategies adopted, we will describe in details the PyBERTHA charecteristics and implementation. Finally we will conclude with a test case of the code involving the interection of a flerovium atom [14] with some gold atoms cluster.

The achievements, we will describe in the following, represent the state of the art for full 4c calculations and give us the ideal starting point for the necessary further development of methods of relativistic theory.

2. Computational details

In the following subsections we will give all the details about the fundamental features of the BERTHA code. Specifically we will summarizing the basic strategy behind the parallelization of the code, and ,maybe more importantly, we will give all the details about the newly developed Python interface of BERTHA.

2.1. Parallelization strategy

Historically the main motivation for the use of approximate methods to include relativity (e.g. including spin-orbit effect) in the modelling of molecular systems is the assumption that full 4c approach is computationally too demanding [15,16]. While this is in principle an obvious assumption, we have shown that one can drastically reduce the computational cost of a Dirac-Kohn-Sham (DKS) calculation, by implementing various parallelization and memory distribution [17,18,12] schemes and by introducing new algorithms, such as those based on the method "density fitting" [19]. We already shown that is now possible

to carry out DFT calculations at full relativistic 4c level in an efficient way, and thus exploit the full power behind the DKS equations, in a wide range of molecular systems [4, 20,3].

The main goal we prosecuted has been to almost nullify any serial portion of the code, being inspired by the basic idea behind Amdahl's law [21]. On the other hand, considering the amount of memory required to perform a DFT calculations at full relativistic 4c level, we designed an ad-hoc method to simulate shared memory for distributed memory computers following a path already designed by other quantum chemistry softwares (e.g., ADF [22] uses GlobalArray [23], GAMESS-US [24] uses the Distributed Data Interface [25] library). The ad-hoc method we designed as been specifically shaped on the data distribution induced by the problem, more specifically dictated by the grouping of G-spinor basis functions in sets characterized by common origin and angular momentum.

In order to recall the main aspects of the implemented parallelization strategy, it is important to point out the fundamental steps of each BERTHA run. Once the molecular system geometry has been specified, together with the basis and fitting set to be used, provided an initial guess density (i.e. cast as a superposition of atomic densities), the **density fitting** is carried out. Soon after the software builds the **Coulomb plus exchange-correlation matrix**, and at this stage, only during the first iteration, also the **one-electron** and **overlap** matrices are computed and stored in memory as they do not vary from cycle to cycle. Finally the **DKS matrix** is assembled and the **eigenvalue problem** is solved.

While the full description of the implemented parallelization strategy has been reported in our previous works[17,18,12], for the sake of completeness here we are reporting only a quick overview starting from the results presented in Figure 1. As the reported results indicate we are able to achieve good results both in term of speed-up as well in terms of memory distribution.

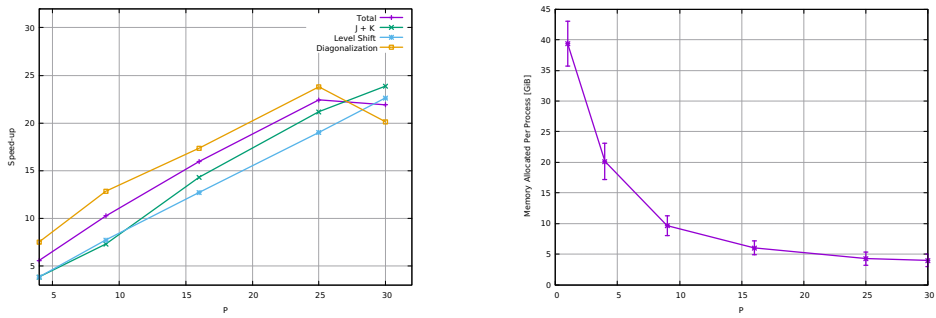

Figure 1. On the left panel we are reporting the speed-up obtained for the $Au_{20} - Fl$ complex with the current implementation of BERTHA. On the right panel we are instead reporting the memory allocated per process, for the same molecular system, when running the code using an increasing number of processes P. The results are obtained compiling the code with Intel Parallel Studio XE 2019 and Intel Math Kernel Library , and running on a two nodes cluster quipped with Intel(R) Xeon(R) CPU E5-2650 v2 at 2.60GHz and InfiniBand interconnection.

Whenever a linear algebra operation is needed we took advantage of the widely available ScaLAPACK [26] library. We recall here that the P processes of a generic parallel execution are, in ScaLAPACK, mapped onto a $P_r x P_c$ two-dimensional process grid of P_r process rows and P_c process columns. Almost consequently any dense matrix is then decomposed into blocks of suitable size, which are uniformly distributed along

each dimension of the process grid according to a specific the so-called Block Cyclic Distribution (BCD) pattern.

We implemented some utility functions (see [17,18,12] for details) that are used to efficiently map the main matrices as they are computed into the BCD distribution schema. It is indeed important to underline as the memory consumption per process, thanks to the cited approach, is always well under control as well reported in the right panel of Figure 1. We even observe cases of superlinear performance when the small size of the local arrays permits improved cache reuse to prevail over other factors. Also considering that the **PZHEGVX** function we are using to carry out the complex DKS matrix diagonalization is able to converge on a given subset of eigenvectors. That is, in our case, we are converging only in the occupied spinors subset and this gives us an evident advantage respect to the serial code, where instead we are always converging on the full set of spinors.

The grid shape affects appreciably the performance of the linear algebra routines and different routines may be differently influenced depending on the block size, number of processes, and size of molecular systems. We already reported a spread in performance of up to 50% when rectangular processes grids are used, this appear to be unfavorable mainly for the diagonalization step compared to square grids [18]. Clearly this explain the decrease on performances observed when 30 process are used. Indeed we are using a rectangular processes grid, that is a 5x6 or equivalently 6x5 one, thus a rectangular one.

2.2. *PyBERTHA: a Python API for the BERTHA code*

Undoubtedly the Python programming language is emerging as one of the most important and used HLL [27,13] also in the field of scientific computing. Python HLL, besides providing an extensive range of modules to be used to solve comprehensive set of computational problems, enables for a quick prototyping, being so a natural choice in the BERTHA project.

The very first step toward a Python binding started reworking the original "monolithic" FORTRAN [28] BERTHA code so that it becomes a set of SO libraries: **libbertha** containing all the basic kernel functions, **libberthaserial** to perform the serial run, and **libberthaparalleshm** to execute the parallel computation. Once this very first step has been completed the computational kernel of the DKS calculation it is driven by a FORTRAN module named **bertha_wrapper**. The FORTRAN module contains a set of methods to access to all the basic quantities, such as energy, density and DKS matrices and more. That same FORTRAN module is used to access all the basic functionality such as: **bertha_init** to perform all the memory allocations, **bertha_main** to run the main SCF iterations, clearly **bertha_finalize** to basically free all the memory, and more.

Finally the main PyBERTHA module, named **berthamod**, has been developed using the **ctypes** Python module. The cited module provides C compatible data types, and allows calling functions in shared libraries. In order to simplify the direct interlanguage communication between Python and FORTRAN, we developed a C layer called **c_wrapper**, as well summarized in Figure 2.

In the actual version of the code the input geometry, basis and fitting set are specified via a file and the related **set_fnameinput** and **set_fittfname** methods. Additionally the **pybertha** class is populated with all the basic functionality need to easily implements basic procedure as a single-point energy calculation (i.e. using the **run** and **get_etotal**

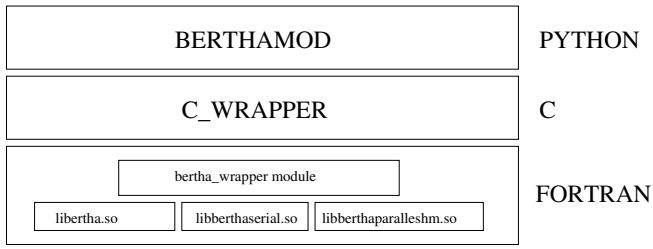

Figure 2. An overview of the software and HLL layers.

methods), or geometry optimization and also much more complex procedures as for example a real-time TDDFT [5] (i.e. using **realtime_init**, **get_realtime_dipolematrix** and **get_realtime_fock** to obtain the DKS matrix given as an input a density matrix).

All data produced at the FORTRAN layer and retrieved at the Python level can be easily handled in case of scalars. Instead when we need to transfer matrices, and in our case matrices of complex numbers, we used the FORTRANC interoperability **iso_c_binding** module and two Python functions: **doublevct_to_complexmat** to convert array of double into Python complex **numpy.array**[29], and **complexmat_to_doublevct** to perform the opposite operation. For the sake of completeness we are reporting in the following the code of the two cited functions:

```
def complexmat_to_doublevct (inm):

    if len(inm.shape) != 2:
        return None

    if inm.shape[0] != inm.shape[1]:
        return None

    dim = inm.shape[0]

    cbuffer = numpy.zeros((2*dim*dim), dtype=numpy.double)
    cbuffer = numpy.ascontiguousarray(cbuffer, dtype=numpy.double)

    cbuffer[0::2] = inm.flatten().real
    cbuffer[1::2] = inm.flatten().imag

    return cbuffer

def doublevct_to_complexmat (invector, dim):

    if (invector.size != (2*dim*dim)):
        return None

    outm = numpy.zeros((dim,dim), dtype=numpy.complex128)

    inmtxreal = numpy.reshape(invector[0::2], (dim,dim))
    inmtximag = numpy.reshape(invector[1::2], (dim,dim))
    outm[:,:] = inmtxreal[:,:] + 1j * inmtximag[:,:]

    return outm
```

The approach to move data from the FORTRAN layer up to the Python one we just described it is not necessarily the most efficient, indeed one can maybe use a direct memory mapping between the FORTRAN array and the numpy ones. Nevertheless, given the results of the Python overhead we will shortly illustrate, we believe that the way we used it is the less error-prone and the best compromise in term of code portability and efficiency.

Indeed adopting the described technique we performed some test to exactly estimate the overhead related to the Python binding. All the results, reported in Table 1, have

been obtained compiling the code with the Intel(R) FORTRAN and Python compilers (version: 2018.3.222), but similar results, in term of percentage of the Python binding overhead, have been obtained using the GNU compilers.

Table 1. We are reporting the impact of the Python binding in the total execution time using 10 SCF iterations. The code has been executed on a Intel(R) Xeon(R) CPU E3-1220 compiling the code with the Intel(R) compiler version: 2018.3.222

System	Matrix Dimension	Wall-time 10 SCF iterations with Python (s)	Wall-time 10 SCF iterations without Python (s)	Python overhead 10 SCF iterations
H_2O	140	3.910	3.906	0.09 %
Au_2	1560	104.458	104.354	0.99 %
Au_4	3152	613.912	613.483	0.07 %
Au_8	6304	3965.911	3964.078	0.05 %

Looking at Table 1 it is evident that the impact of the Python binding is almost always lower then 0.1 % and thus it is clearly negligible. The only system where the overhead is higher then 0.1 % is the Au_2 gold cluster. We may only speculate that in such a case there is some effect related to the cache memory size, indeed the most demanding part of the Python binding is essentially related to a memory-to-memory copy.

As we already pointed-out the Python binding overhead is almost solely related to the arrays copying process that, in the case of a single point ENERGY calculation, is executed just once at the end. Thus clearly, in the case of a standard single point energy calculation, the Python binding has no impact on the serial execution time of BERTHA at all.

3. Bond Analysis of $Au_{20} - Fl$ complex with NOCV/CD method

In the last decade, experiments were carried out to compare the chemical behaviour of SHEs (Super Heavy Elements) with that of their lighter homologues of the 6th period. For example by gas-phase thermochromatography studies of volatility through adsorption on gold surfaces [30].

Here we will try shed some light on the $Au_{20} - Fl$ interaction using the NOCV/CD analysis scheme [4]. In a previous work we presented the formalism used to decompose the CD function in terms of NOCVs in the context of the relativistic four-component framework where spin-orbit coupling is included variationally [4].

The core idea of the approach is the decomposition, via natural orbitals for chemical valence (NOCV), of the so-called charge-displacement (CD) function into additive chemically meaningful components.

Firstly we start recalling the CD function that is defined as a partial integration along a suitable z axis of the difference $\Delta\rho(x,y,z')$ between the electron density of the adduct and that of its non-interacting fragments (that is the Au_{20} gold cluster and Fl atom in our specific case) placed at the same equilibrium position they occupy in the adduct:

$$\Delta q(z) = \int_{-\infty}^{z} dz' \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Delta\rho(x,y,z')dxdy \tag{1}$$

In the previous equation the integration axis is obviously chosen following some physical criteria, for instance the bond axis between the fragments appears to be the most convenient choice in our case. Any CD function features positive values along z when, upon formation of the bond, charge is transferred from right to left across a plane perpendicular to the bond axis through z. On the contrary negative values of the CD function identify charge flow in the opposite direction.

Finally, it is worth noting that the electron density difference can be further partitioned when both the adduct and its constituting fragments belong to the same symmetry group, $\Delta\rho$ can be expressed in terms of additive symmetry components. More generally, a different scheme can been applied to provide the decomposition of the $\Delta\rho$ in terms of contributions arising from the molecular spinors most involved in the bonding. Natural orbitals for chemical valence (NOCV) were introduced by Mitoraj and Michalak [31] as descriptors of chemical bond. The cited formalism allows a very compact description of the bonding phenomenon, indeed the electron density difference $\Delta\rho$ can be brought into diagonal contributions in terms of NOCVs (i.e. additive chemically meaningful components).

The cited NOVC/CD analysis has been applied to the $Au_{20} - Fl$ complex, as reported in Figure 3, using a molecular geometry as reported in a previous work [20]. It is somehow important to underline the fact that, while the geometry optimization of the gold cluster has been performed using the zero-order regular approximation (ZORA)[32], both the Flerovium gold cluster distance and clearly all the calculation needed to produce the final NOCV/CD results, have been performed using BERTHA.

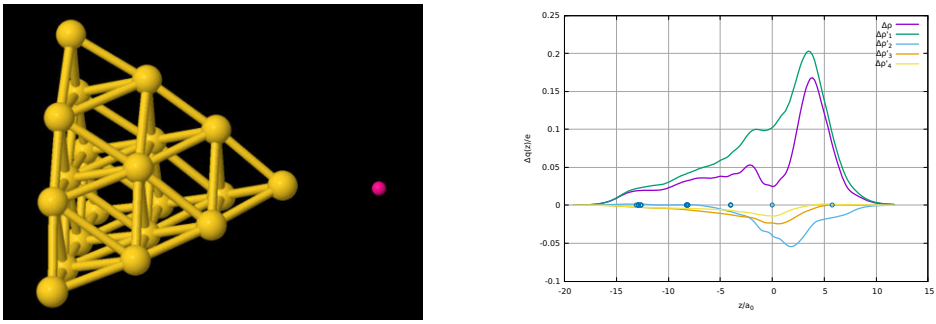

Figure 3. On the left panel we are reporting $Au_{20} - Fl$ complex. On the right panel instead the CD analysis for the $Au_{20} - Fl$ bond is reported, where the dots on the axis mark the z coordinate of the atoms. We are reporting the contribution to deformation density, $\Delta\rho$, of the four most significant NOCV-pairs ($\Delta\rho'_1$, $\Delta\rho'_2$, $\Delta\rho'_3$ and $\Delta\rho'_4$,).

In Figure 3 we are reporting the complex geometry, on the left panel, and the CD curves on the right. The fundamental feature of the $\Delta\rho$ CD curve is that $\Delta q(z)$ is appreciably positive everywhere in the cluster region. This means that there is a shift of charge from the Fl atom towards the gold cluster. More interestingly, the shift of charge does not stop at the nearest Au layer but extends appreciably down to the fourth layer. Another interesting feature of the $\Delta\rho$ CD curve is showing are the two peaks, one corresponding to the zone between the first and second Au layers, and the other corresponding to the gold-Fl binding region.

Finally looking at Figure 3, it is undoubtedly interesting to see how we are able to split the total CD curve into several additive chemically meaningful components. In

the figure we are reporting only the first four NOCVs, that are quantitatively the most relevant. It is clear how the total curve, that it is highlighting an shift of charge from the Fl atom towards the gold cluster, is instead made of two different contributions. The first, $\Delta\rho'_1$, is indeed representing a shift of charge from the Flerovium toward the gold cluster, but the other three curves are instead displaying and an opposite charge flow, from the Au_{20} cluster toward the Fl atom.

The reported results give a clear view of the power the NOVC/CD analysis, that it is able to give clear insights on the nature of a chemical bond in a simple and visual way. It is somehow important here to remark the fact that these results have been made possible only by the previously reported effort in terms of code optimization and code parallelization.

4. Conclusions and perspectives

As already stated mainly because of the computational cost of the rigorous way to include relativity in the modelling of molecular systems, a disparate set of approximate methods have been derived from the strictly relativistic 4-component (4c) formalism derived from the Dirac equation by Bertha Swirles [6].

In the present work we described the BERTHA code, that as a result of our effort, can be considered the state of the art for full 4c calculations. Indeed, thanks to the the parallelization strategies and density fitting techniques adopted, we have been able to extend the applicability range of all-electron DKS calculations to extremely big molecular systems.

In addition, we introduced also a set of Python bindings (so called PyBERTHA), that we demonstrated to have almost a negligible impact in terms of time consumption. Instead the introduction of such software layer improved enormously the code usability, especially respect to the original FORTRAN version.

Specifically, due to the actual enormous diffusion of the Python programming language, we hope to spread the use of full 4c calculations to a larger scientific population. Indeed, thanks to the introduced Python API, it is now simple and quick to implement and test new approaches based on the basic building blocks provided by the current version of PyBETHA.

Clearly, given the described abstraction layer it is now easy to extend the API as needed. In a future coming version we are planning to add all the fundamental functions to specify the input geometry and basis set in a more user-friendly (i.e. pythonic) way.

References

[1] L. Belpassi, L. Storchi, H. M. Quiney, and F. Tarantelli, "Recent advances and perspectives in four-component Dirac-Kohn-Sham calculations," vol. 13, pp. 12368–12394, 2011.

[2] P. Pyykkö and J. P. Desclaux, "Relativity and the periodic system of elements," *Accounts of Chemical Research*, vol. 12, no. 8, pp. 276–281, 1979.

[3] M. De Santis, L. Belapassi, F. Tarantelli, and L. Storchi, "Relativistic quantum chemistry involving heavy atoms," *Rendiconti Lincei. Scienze Fisiche e Naturali*, vol. 29, no. 2, pp. 209–217, 2018.

[4] M. De Santis, S. Rampino, H. M. Quiney, L. Belpassi, and L. Storchi, "Charge-displacement analysis via natural orbitals for chemical valence in the four-component relativistic framework," *Journal of chemical theory and computation*, vol. 14, no. 3, pp. 1286–1296, 2018.

[5] M. Repisky, L. Konecny, M. Kadek, S. Komorovsky, O. L. Malkin, V. G. Malkin, and K. Ruud, "Excitation energies from real-time propagation of the four-component dirac–kohn–sham equation," *Journal of chemical theory and computation*, vol. 11, no. 3, pp. 980–991, 2015.

[6] B. Swirles, "The relativistic self-consistent field," *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 152, no. 877, pp. 625–649, 1935.

[7] M. Reiher and A. Wolf, *Relativistic quantum chemistry: the fundamental theory of molecular science*. John Wiley & Sons, 2014.

[8] M. Reiher and A. Wolf, "Exact decoupling of the dirac hamiltonian. ii. the generalized douglas–kroll–hess transformation up to arbitrary order," *The Journal of chemical physics*, vol. 121, no. 22, pp. 10945–10956, 2004.

[9] E. v. van Lenthe, J. Snijders, and E. Baerends, "The zero-order regular approximation for relativistic effects: The effect of spin–orbit coupling in closed shell molecules," *The Journal of chemical physics*, vol. 105, no. 15, pp. 6505–6516, 1996.

[10] I. P. Grant, *Relativistic quantum theory of atoms and molecules: theory and computation*, vol. 40. Springer Science & Business Media, 2007.

[11] L. E. McMurchie and E. R. Davidson, "One-and two-electron integrals over cartesian gaussian functions," *Journal of Computational Physics*, vol. 26, no. 2, pp. 218–231, 1978.

[12] S. Rampino, L. Belpassi, F. Tarantelli, and L. Storchi, "Full parallel implementation of an all-electron four-component Dirac-kohn-Sham program," vol. 10, no. 9, pp. 3766–3776, 2014.

[13] M. F. Sanner *et al.*, "Python: a programming language for software integration and development," *J Mol Graph Model*, vol. 17, no. 1, pp. 57–61, 1999.

[14] P. Schwerdtfeger, "One flerovium atom at a time," *Nature chemistry*, vol. 5, no. 7, p. 636, 2013.

[15] H. M. Quiney, H. Skaane, and I. P. Grant, "Relativistic calculation of electromagnetic interactions in molecules," vol. 30, no. 23, p. L829, 1997.

[16] H. M. Quiney, H. Skaane, and I. P. Grant, "*Ab initio* relativistic quantum chemistry: four-components good, two-components bad!," vol. 32 of *Advances in Quantum Chemistry*, pp. 1–49, Academic Press, 1998.

[17] L. Storchi, L. Belpassi, F. Tarantelli, A. Sgamellotti, and H. M. Quiney, "An efficient parallel all-electron four-component Dirac-Kohn-Sham program using a distributed matrix approach," vol. 6, no. 2, pp. 384–394, 2010.

[18] L. Storchi, S. Rampino, L. Belpassi, F. Tarantelli, and H. M. Quiney, "Efficient parallel all-electron four-component Dirac-Kohn-Sham program using a distributed matrix approach II," vol. 9, no. 12, pp. 5356–5364, 2013.

[19] L. Belpassi, F. Tarantelli, A. Sgamellotti, and H. M. Quiney, "Poisson-transformed density fitting in relativistic four-component Dirac-Kohn-Sham theory," vol. 128, no. 12, p. 124108 (11), 2008.

[20] S. Rampino, L. Storchi, and L. Belpassi, "Gold–superheavy-element interaction in diatomics and cluster adducts: A combined four-component dirac-kohn-sham/charge-displacement study," *The Journal of chemical physics*, vol. 143, no. 2, p. 024307, 2015.

[21] J. L. Gustafson, "Reevaluating amdahl's law," *Communications of the ACM*, vol. 31, no. 5, pp. 532–533, 1988.

[22] G. t. Te Velde, F. M. Bickelhaupt, E. J. Baerends, C. Fonseca Guerra, S. J. van Gisbergen, J. G. Snijders, and T. Ziegler, "Chemistry with adf," *Journal of Computational Chemistry*, vol. 22, no. 9, pp. 931–967, 2001.

[23] J. Nieplocha and R. Harrison, "Shared memory programming in metacomputing environments: The global array approach," *The Journal of Supercomputing*, vol. 11, no. 2, pp. 119–136, 1997.

[24] M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. Su, *et al.*, "General atomic and molecular electronic structure system," *Journal of computational chemistry*, vol. 14, no. 11, pp. 1347–1363, 1993.

[25] G. D. Fletcher, M. W. Schmidt, B. M. Bode, and M. S. Gordon, "The distributed data interface in gamess," *Computer Physics Communications*, vol. 128, no. 1-2, pp. 190–200, 2000.

[26] J. Choi, J. J. Dongarra, R. Pozo, and D. W. Walker, "Scalapack: A scalable linear algebra library for distributed memory concurrent computers," in *[Proceedings 1992] The Fourth Symposium on the Frontiers of Massively Parallel Computation*, pp. 120–127, IEEE, 1992.

[27] J. M. Perkel, "Programming: pick up python," *Nature News*, vol. 518, no. 7537, p. 125, 2015.

[28] S. J. Chapman and S. J. Chapman, *Fortran 95/2003 for scientists and engineers*. McGraw-Hill, 2008.

[29] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical

computation," *Computing in Science & Engineering*, vol. 13, no. 2, p. 22, 2011.

[30] I. Zvára, *The Inorganic Radiochemistry of Heavy Elements: Methods for Studying Gaseous Compounds*. Springer Science & Business Media, 2008.

[31] M. A. Mitoraj M., "Natural orbitals for chemical valence as descriptors of chemical bonding in transition metal complexes," *J. Mol. Model.*, vol. 13, pp. 347–355, 2007.

[32] E. van Lenthe, A. Ehlers, and E.-J. Baerends, "Geometry optimizations in the zero order regular approximation for relativistic effects," *The Journal of chemical physics*, vol. 110, no. 18, pp. 8943–8953, 1999.