Introduction

Within the density functional theory (DFT), the system of Kohn-Sham equations

$$H\psi_i = \left(\frac{1}{2}\nabla^2 + V_{\text{ext}} + V_{\text{H}} + V_{\text{xc}}\right)\psi_i(\vec{x}) = e_i\psi_i(\vec{x}) \tag{1}$$

for electronic states ψ_i has to be solved to obtain the electronic structure of the investigated material and to determine its material properties via subsequent calculations, as needed. This approach is called *ab-inito*, because it makes possible to calculate the material properties from the first principles of the quantum theory, not using any experimental data. The *Hartree* (V_{H}) and *exchange-correlation* (V_{xc}) potentials included in the equations are functions of the *charge density* ρ, which arises itself from the solution of the Kohn-Sham equations. Therefore, these equations need to be solved repeatedly until the iterative cycle, called *self-consistent cycle* in the DFT terminology, converges.

For the performance of such a DFT iterative scheme, the crucial component is the *mixing scheme* — the way of generating a new *input charge density* ρ^{in}_{i+1} for the next iteration of the self-consistent cycle from the set of previous input and *output charge densities* $\rho^{\text{in}}_{j\in\{1\ldots i\}}$ and $\rho^{\text{out}}_{j\in\{1\ldots i\}}$ coming from the previous iterations.

The importance of the topic is emphasized by the fact that the developers of DFT software packages in the course of decades since publishing the well-known Anderson-Pulay mixing scheme [1, 2] (which can be considered as the standard approach for solving the mixing problem) still have been making their effort to improve the mixing schemes and their reliability, e.g. [3, 4, 5, 6, 7, 8].

This paper analyses several well-known and newly-proposed mixing schemes and proposes a new modification of the Anderson mixing scheme, which improves the convergence of the scheme at least in some cases.

The first section defines the problem and key terms. An overview of the various existing mixing methods and techniques is given in the second section.

The third section contains a numerical analysis of various mixing schemes, implemented into the FENNEC code (our package based on the discretization of DFT via finite element method or via isogeometric analysis [9, 10, 11]). To confirm that the results of the analysis and the observed phenomena are not method-dependent, the properties of the newly proposed schemes are illustrated also by means of the SPR-KKR code, which is based on the Green-function Korringa-Kohn-Rostoker method [12]. The performed numerical analysis shows that the standard Anderson mixing scheme [13] performs the best (at least in the presented cases within the set of the considered mixing schemes) and that further improvement could be achieved rather by a proper choice of the *(linear) mixing parameter* than by altering the mixing scheme.

The essential finding of the paper is that the optimum choice of the mixing parameter in each iterative step can be derived from the *mixing coefficients* of the densities in previous steps that result from the Anderson mixing scheme. This fact makes it possible to suggest a modification of the mixing scheme that can be called *Adaptive Anderson Mixing*, which is presented in the fourth section.

The fifth chapter presents the results of tests of the new mixing scheme which prove the advantages of the adaptivity of the mixing parameter. The last chapter contains a brief documentation of the implementation of the new mixing scheme in the Fortran language. It is available as a freeware library for possible incorporation into any existing DFT (and possibly other – e.g. Hartree-Fock) software packages.

1 Definition of the problem

From the mathematical point of view, the iterative scheme solves a *fixed point* problem: to find ρ, such that

$$F(\rho) = \rho\,. \tag{2}$$

In our particular case, the operator F calculates a new *output charge density* from the *input* one using the Kohn-Sham equations (1) (which include $V_{\mathrm{H}}(\rho^{\mathrm{in}})$ and $V_{\mathrm{xc}}(\rho^{\mathrm{in}})$). The operator F is an operator on the Hilbert space $\mathcal{H}$.[1].

A fixed-point problem like that cannot be solved analytically. Thus, an iterative numerical method — a *mixing scheme* — has to be employed. The simplest approach is to take a reasonable guess of the initial charge density and then evaluate the function $F(\rho)$ repeatedly, taking the output of the previous iteration as the input for the next iteration until the process converges, i.e. until the following relation is satisfied:

$$\left\|\rho^{\mathrm{in}} - \rho^{\mathrm{out}}\right\| < t\,, \tag{3}$$

where $t \in \mathbb{R}, t > 0$ is a given threshold.

Sometimes the convergence is expressed in terms of a *relative convergence threshold*

$$\frac{\|\rho^{\mathrm{in}} - \rho^{\mathrm{out}}\|}{\|\rho^{\mathrm{in}}\|} < t_{\mathrm{REL}}\,. \tag{4}$$

A “reasonable guess” for the initial charge density can be given by a superposition of charge densities of isolated atoms, by a charge density obtained from an alike configuration, from a faster and less accurate computation (e.g. with a lower number of basis elements), or even by a uniform charge density.

Unfortunately, the direct approach to solving the self-consistent problem described above results generally in poor convergence or (often) no convergence at all. Therefore, more sophisticated approaches have been developed. The simplest one is the *linear mixing*: instead of the output density, a linear combination of the input and output densities is used as the input in the next iteration of the self-consistent cycle.

$$\rho^{\mathrm{in}}_{i+1} = a_i\rho^{\mathrm{out}}_i + (1 - a_i)\rho^{\mathrm{in}}_i\,. \tag{5}$$

This approach could damp the oscillations around the desired result. However, the performance of linear mixing in real-world problems is often unsatisfactory.

To discuss more powerful mixing schemes, we introduce a term *residual.* The i_{th} *residual* τ_i is defined as

$$\tau_i = \rho^{\mathrm{out}}_i - \rho^{\mathrm{in}}_i\,. \tag{6}$$

[1] In practice, the operator is evaluated after a discretization (e.g. by planewaves or FEM); so the problem could be expressed rather in terms of vectors, not functions. However, this distinction is insignificant for further considerations.

The linear mixing (Eq. 5) can be rewritten using residuals as:

$$\rho_{i+1}^{\text{in}} = \rho_i^{\text{in}} + a_i \tau_i \,. \tag{7}$$

By means of the newly introduced term, a new insight into the problem can be obtained: we try to find an input for which the residual is zero. Therefore we need to solve a multidimensional nonlinear *root finding problem*:

$$\text{Find } \rho \text{ such that } G(\rho) = 0, \text{ where } G(\rho) = F(\rho) - \rho \,. \tag{8}$$

2 Overview of commonly used mixing algorithms

The same or analogous problem (as mentioned above), either in its fixed point or the root-finding form, arises in various fields and it has been studied by many authors. Below, just a brief overview of the algorithms to solve the problem will be given. A rigorous mathematical survey of the presented methods can be found e.g. in [13].

Here, we restrict ourselves to the methods that are suitable for DFT calculations, which means that the following must be taken into account : (i) the evaluation of the function F (or G) is computationally very expensive in this case — therefore the number of its evaluations should be kept as low as possible. This makes it hardly possible to employ e.g. line-search techniques and other methods like that. (ii) there is no efficient way to evaluate the gradient of the function, which disqualifies a wide range of gradient-based methods — from the classical Newton method (including its modifications Newton-Krylov methods [14]) to the well-established BFGS methods [15]. For these reasons we focus on the class of so-called (multi)secant methods such as Broyden's first and second method and particularly on the Anderson/Pulay mixing schemes [2, 16, 1, 17] (which can be considered as a standard method for solving DFT selfconsistent problem), and on their modifications.

2.1 Generalized Broyden methods

Anderson and Pulay mixing Both Pulay and Anderson view the problem as a fixed-point problem. Their approach results in an iterative mixing scheme that takes into account a longer history of previous iterations than simple linear mixing. Anderson's class schemes calculate a new input density as a linear combination of previous input and output charge densities:

$$\rho_{i+1}^{\text{in}} = \sum_{j=1}^{i} \left(b_{i,j} \rho_j^{\text{in}} + a_i b_{i,j} \tau_j \right) \quad , \qquad \sum_{j=1}^{i} b_{i,j} = 1 \quad . \tag{9}$$

where the scalars $b_{i,j}$ (called *mixing coefficients*) minimize the L^2 norm of the residual over a subspace of n last input densities

$$\min \left\| \sum_{j=1}^{i} b_{i,j} \tau_j \right\|_2 \quad . \tag{10}$$

The constrained minimization is usually done by means of Lagrange multipliers, which leads to a low-dimensional system of linear equations [18], or it is treated as a least-square problem. [19]

The convergence rate of the Anderson method depends on the number of charge densities taken into account, i.e. on the *history length*: a method allowed to have nonzero coefficients $b_{i,j}$ only for $j > i - n$ has the *history length* n. If a mixing scheme has a history length n, the new density is sought in the space of n previous densities (and residuals). The small study performed by Eyert in [17] indicates that an efficient history length could be pretty small: his calculations show no differences for $n > 4$. Marks in [7] recommends as the default history length 8. In most calculations performed by us using FENNEC (see e.g. [20]), there was no benefit from rising the history above six, which agrees with Marks' experience.

Nowadays, the *Pulay mixing* and the *Anderson iterative scheme* are essentially the same algorithms, viewed from various sides. Pulay [2] has introduced the scheme, in which he sought the new density only in the space spanned by the input densities, which is effectively the same as setting all a_i in (9) to zero.

This mixing was named after Pulay but he was not the first who invented the method: the same method in a bit different context had been published earlier[2] by a less known (at least in the physics community) scientist with Czechoslovak roots Márian Mešina. [22] Few years before both Pulay and Mešina, Anderson suggested mixing with nonzero a_i. However, originally, his approach used the history length of only two. [1].

Both approaches — Pulay's and Anderson's — evolved: their advantages (nonzero a_i and a longer history) have been combined to form a single algorithm called either the *Pulay mixing*, *DIIS* (Direct inversion of the iterative subspace), *Anderson mixing* or *Anderson iterative scheme*. The recently published convergence results of the Anderson method can be found in [23, 24] and the relation of DIIS with GMRES and with other generalized Broyden methods has been investigated by Rohwedder and Schneid in [25].

Broyden methods Broyden's work, contemporary with Anderson's one, was aimed at a root-finding problem. Broyden introduced a modification of the well-known Newton method for solving nonlinear equations.

One iteration of the Newton method is given by

$$\rho_{i+1} = \rho_i - \left(J_G(\rho_i)\right)^{-1} \tau_i \tag{11}$$

where $J_G(\rho_i)$ is the Jacobian[3] of the function G. Broyden's methods replace the Jacobian with its approximation: *Broyden's first method* constructs the Jacobian approximation and then inverts it, whereas *Broyden's second method* constructs directly the inversion.[26] Since the direct inversion of the Jacobian, which is needed

[2] The even earlier work of Cabay and Jackson [21] — again in a different context — shares the basic idea of finding the fixed point via eliminating the residual. However, they do not formulate this mixing as an iterative process, but as a posteriori analysis of the already generated sequences of the vectors.

[3] The Jacobian of a function is the matrix of its partial derivatives: $J^f_{a,b}(\mathbf{x}) = \frac{\partial f_a}{\partial \mathbf{x}_b}(\mathbf{x})$ (we assume the finite-dimensional case for the sake of simplicity).

in the first method (see (11)), is computationally expensive, the Sherman-Morisson-(Woodbury) formula is applied to the Jacobian approximation.[4] The resulting formulation then differs from the second method just in the choice of vector to be used for generating the update of the Jacobian. [13] In both methods, the updates of the Jacobian satisfy the so-called *secant condition*

$$(J_G(\rho_i))\,(\rho_i - \rho_{i-1}) = \tau_i - \tau_{i-1} \tag{12}$$

and the changes of the Jacobian between the iterations are kept as small as possible (in the terms of squared Frobenius norm), as it is explained e.g. in [13].

Generalized Broyden's methods Broyden's work is based on the theoretical background of a class of methods called *generalized Broyden's* or *multisecant methods*, which follow Broyden's idea of approximating the Jacobian, but — unlike the original Broyden's methods — they use a longer, generally an arbitrary history length for updating the Jacobian [16]: while the original Broyden's methods enforce "the optimality" of the Jacobian approximation only with regard to the result of the last iteration (see (12)), these methods replaces the secant condition with so-called *multisecant condition*:

$$(J_G(\rho_i)) \begin{pmatrix} \rho_i - \rho_{i-1} \\ \rho_{i-1} - \rho_{i-2} \\ \dots \\ \rho_{i-h+1} - \rho_{i-h} \end{pmatrix} = \begin{pmatrix} \tau_i - \tau_{i-1} \\ \tau_{i-1} - \tau_{i-2} \\ \dots \\ \tau_{i-h+1} - \tau_{i-h} \end{pmatrix}, \tag{13}$$

which keeps the Jacobian approximation optimal with regard to the last h residuals. [13]

The Pulay/Anderson mixing belongs to this class of generalized Broyden's methods, too. [13, 17] Among other methods of this class we can mention (although in the DFT community rarely used) e.g. methods proposed by Martínez [27, 28, 29], or Broyden's method with projected updates [30], or the recently published Anderson type-I mixing [31], which is the Anderson-like generalization of Broyden's first method, in a similar way as the original Anderson mixing is the generalization of Broyden's second method. Marks in [7, 3] has published a hybrid method that adaptively combines the first and the second generalized Broyden's method.

In the presented results of Anderson type-I mixing [31], the algorithm (with the regularization and including linear mixing steps, which will be mentioned below) outperforms the original Anderson algorithm, especially in the latter stages of the computation where the presented implementation of the original Anderson algorithm suffers from convergence stagnation. However, from the information provided in the article, it is not clear whether the improvement is caused by better numerical properties of the Anderson type-I algorithm or due to the additional techniques incorporated into the newly proposed algorithm.

The Anderson mixing is nowadays the de facto standard in DFT calculations, and it (at least usually) performs substantially better than Broyden's original

[4] Broyden's first method constructs Jacobian approximation using directional derivatives obtained in the iterations of the algorithm, which are applied as rank-updates to the initial Jacobian approximation.

methods — as we can see in Fig. 1. Therefore, we focus particularly on the Anderson method and its modifications.

2.2 Non-Broyden approaches and recent developments

Relatively recently, several papers were published that suggest modifications of the Pulay mixing scheme using the "non-Broyden" approaches. These are represented by the periodical Pulay method and the guaranteed residual Pulay method.

Periodical Pulay The more recent one is the *Periodical Pulay method.* Its idea is simple: to alternate linear mixing (Eq. 5) and Anderson mixing (Eq. 9) steps to achieve a better convergence [32]. However, it performs slightly worse than the Anderson method in our numerical examples (see Fig. 1).

Guaranteed residual Pulay method The second mentioned modification, the *Guaranteed residual Pulay method* [33] (abbreviated as *GR-Pulay*), shares the idea of alternating linear and Anderson mixing steps with the Periodical Pulay method. However, all linear steps are performed with $a_i = 1$, and all Anderson steps are performed with $a_i = 0$ in this scheme. Moreover, the results obtained by each linear step are discarded from the history just after performing the next "Anderson step" which effectively replaces it. Thus, in each Anderson step, only the results of previous Anderson steps and of the last linear step are taken into account.

In further analysis, we will consider a generalization of this concept in which linear steps are done with varying values of a_i (Anderson steps are still done with $a_i = 0$), which sometimes improves the convergence rate slightly.

Authors of the GR-Pulay method suggest that the whole "linear-Anderson multistep" should be considered as a single iteration. However, since it is the number of function evaluations that determines the performance of the scheme, each evaluation of a new charge density should be considered as one iteration. Taking this into account, the improvement of the convergence rate doesn't look so convincing in our numerical tests (Fig. 1).

Other methods The methods mentioned above do not represent a complete summary of "non-gradient" methods (or at least methods that can be formulated without an explicit calculation of the gradient) applicable for solving nonlinear problems. It is not possible to list all of them, but some of the other methods are worth mentioning since they are — in some sense — based on a reformulation or on a modification of the multisecant methods which could bring new insight into the issue.

There are also e.g. nonlinear variants of conjugate-gradient [34] or GMRES [35]. In the linear case, the GMRES is very tightly related [36, 37] to the Eirola-Nevanlinna methods [38], which can be extended to a nonlinear case as well [39]. The connection between the Anderson method and GMRES is shown in [25].

Also, the class of methods based on Aitken's $\triangle^2$ method [40] exists which evolved to many *residual based methods* [41] (e.g. Lemaréchal [42] or Irons and Tuck method [43] and many similar approaches, see [41]). Those methods in their original forms require an extra evaluation of the objective function in each iteration, which would make them less suitable for our purpose. However, using Shanks transformations [44], these extra evaluations can be included in the iterative process which leads to "Anderson-like" methods. [23, 41]

Thus – many of the on-the-first-sight-very-different methods in fact differ from the Anderson mixing only slightly and their analysis could bring new insights into the issue. We encourage the interested reader to see e.g. Fang's and Saad's work [13] for the classification of Broyden-like methods including the Eirola-Nevanlinna one, more general Brune et al.'s survey [45] of nonlinear methods and Ramiére's and Helfer's classification of Residual methods [41].

2.3 Other mixing-related approaches

The efficiency of the mixing can be further enhanced using the following techniques.

Position dependent mixing So far we considered a_i to be the same for the whole charge density vector. In fact, the a_i can be different for each component of the vector: from the "Broyden's point of view," it follows (see Eq. 11) that $-1/a_0$ should approximate the Jacobian of the function [13]. Therefore, for a heterogeneous material, setting the parameters independently in different parts of the domain can be a viable approach, supposing that the correct normalization of the resulting charge density after mixing (i.e. the correct total charge) is ensured. A similar approach was published by Lin and Chao, who suggested using a spatially dependent preconditioner [46]. Heide and Ono demonstrated that a substantial improvement of convergence can be achieved by using different mixing schemes for different parts of the state vector [47].

Preconditioning The preconditioning techniques allow a better approximation of the Jacobian than the constant $-1/a_0$. Several preconditioners have been developed, e.g. Kerker preconditioner [48, 4, 5, 6, 49, 50, 8], elliptic preconditioner [46], Resta preconditioner [51, 49] or HIJ [52, 53]. However, all these preconditioners are material-dependent and thus cannot be used as a general recipe for all problems.

Potential mixing (and density matrix) The mixing of the charge density is not the only option for solving the self-consistent problem. The First Hohenberg-Kohn theorem states [54] that the ground charge density is a function of the potential and vice versa. Therefore, taking the potential ($V_{\mathrm{Hxc}} = V_{\mathrm{H}} + V_{\mathrm{xc}}$) as a "state of the system" and iterating it instead of the charge density represents a viable choice.

The exploratory calculations performed by means of the electronic structure code SPRKKR [12] show that in some cases this approach could lead to better performance than mixing the charge density (see Fig. 1). In addition, no regard for normalization is necessary in that case.

Expressing the state of the system in terms of density matrices [55] is the third possible form (besides the charge density and the potential).

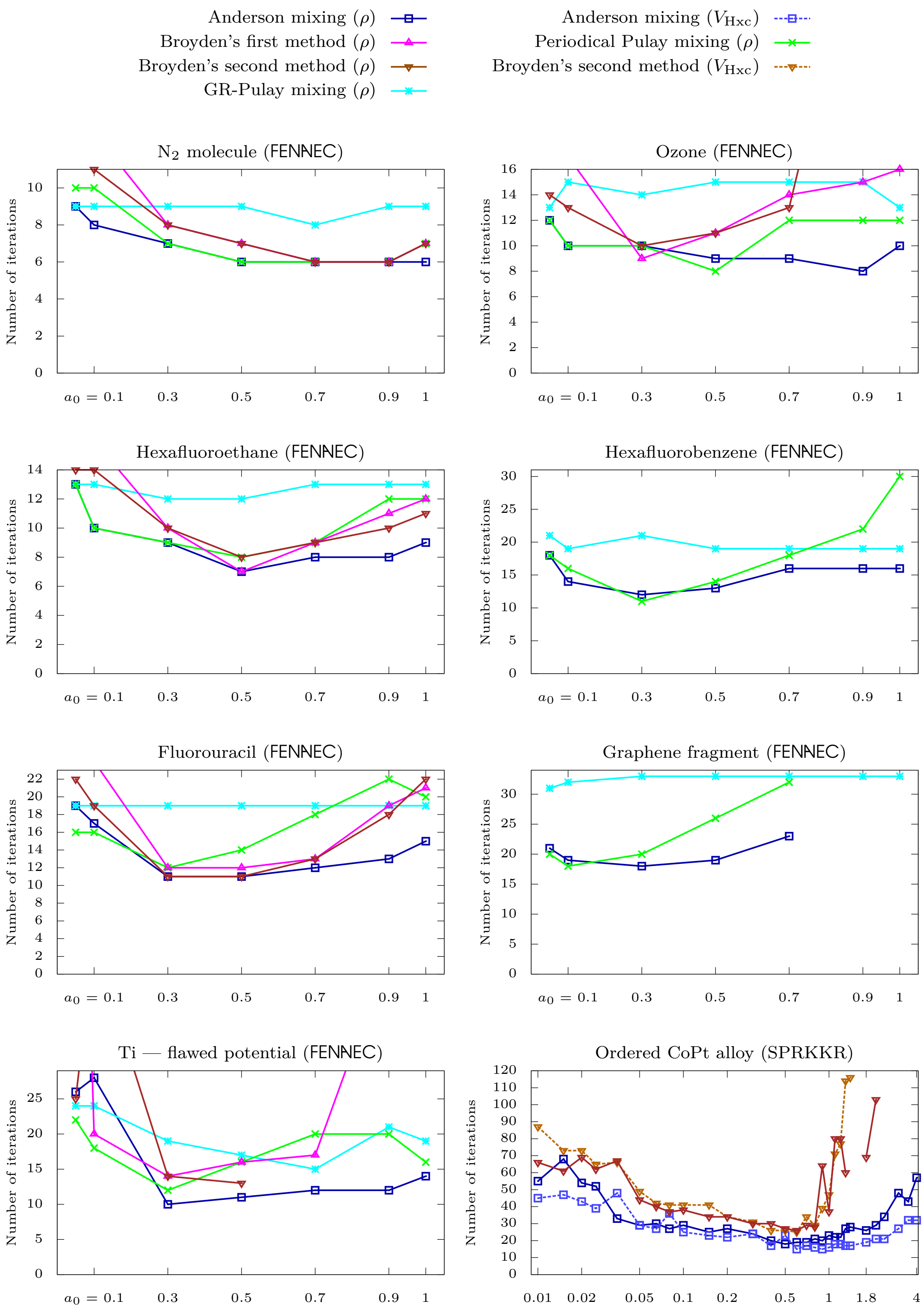


Fig. 1: Dependence of the number of iterations required for the convergence of the DFT self-consistent cycle, on the a_0 coefficient, for various mixing algorithms. Two different codes have been employed (FENNEC and SPRKKR [12]). The convergence criteria $t_{\mathrm{REL}} = 1 \times 10^{-9}$ for FENNEC and $t_{\mathrm{REL}} = 1 \times 10^{-8}$ for SPRKKR have been used (see Eqs. 3 and 4). Algorithms denoted as V_{Hxc} mix the V_{Hxc} potential. The other ones, denoted by ρ, mix the charge density. Missing values indicate that the self-consistent calculation did not converge. The system for which the calculations have been performed is quoted in the title of each graph. The FENNEC calculations employed an adaptive hexahedral mesh and cubic Lagrange elements.

Energy minimalization Wang et al. have suggested linear-expansion shooting techniques (LIST) [56] as an improvement of the classical Anderson mixing where the energy instead of charge density residual is minimized [57, 58]. These techniques can be expressed as a classical Anderson mixing with a special norm applied when the error is minimized [59]. However, this approach can be applied only if there is a way to express the energy efficiently from the iterated quantity (e.g. from the charge density). Therefore these techniques are employed primarily within the density matrix formalism.

Geometry optimization The basic way to perform geometry optimization (sometimes called relaxation)[5] is to utilize two nested minimization algorithms. However, as Bend and Zunger showed [60], both problems can be treated at once as different degrees of freedom of a single problem (see e.g. [3] for the application of the principle).

Regularization, stabilization A *regularization* is an approach that makes it possible to deal with nearly collinear residual vectors, which ordinarily cause numerical difficulties in mixing. Several papers on that topic have been published quite recently. [61, 62, 63, 64] Another well-known approach to handle nearly-collinear vectors is *stabilization* (published e.g. in [23]).

Trust region approaches In some cases, a *charge sloshing* can appear (see e.g. [8]) during self-consistent cycle iterations. This term denotes the phenomenon of charge oscillations appearing either between two spatially distinct parts of the system or between states with similar eigenvalues, which can lead to poor convergence or even non-convergence. To prevent this behavior, some authors recommend using suitable preconditioning (e.g. [49]). The second published approach (in [3]) is to use the *trusted region approach* — which limits the size of steps to prevent the non-convergence.

Some algorithms in this family consist just in adding a "distance-penalty" to the minimized function, which makes them close to the regularization algorithms (see e.g. [65]). Other adaptively restrict the step size, see e.g. [3].

3 Analysis of the mixing algorithms and the parameter a_0

The *mixing parameters* a_i have not been discussed so far. All the algorithms introduced above keep it constant during iterations ($a_i = a_0$)[6], and the value of a_0 has to be supplied by the user.

Therefore, a suitable way to determine the value of a_0 should be found. For illustration, the numbers of iterations needed to achieve convergence of the ab-initio electronic structure self-consistent calculations by the FENNEC code and, for comparison, also by the SPRKKR code [12] for various systems and for various mixing algorithms are presented in Fig. 1.

[5] I.e. to solve the problem of minimal energy of the system with regard to positions of atoms.
[6] The parameter a_0 is often called α.

The results of Broyden mixing for FENNEC were obtained using the implementation in the SciPy package [66]. SPRKKR results use the implementation included in the package. The other mixing schemes have been implemented by us. The performance of our implementation of Anderson mixing (with a stabilization incorporated in a similar way — although not inspired by — as Brezinsky et al. published recently in [23]) has been compared with its SciPy implementation. They provide the same results for common problems, which confirms the correctness of the implementation. Mixing problems arising from the finite element method in FENNEC generally evince better and more consistent convergence properties. For this reason, SPRKKR problem(s) were examined in a wider range of a_0 and with finer samplings.

We can observe that the convergence rate depends on the value of a_0. Since a_0 is related to the approximation of the Jacobian, it is not surprising that the optimal values for a_0 are equal or at least close to each other for various mixing algorithms. The algorithms differ more in how they are able to handle a non-optimal a_0 than in the minimal achieved number of iterations or in the value of an optimal a_0. If the parameter a_0 differs from its optimal value, the self-consistent algorithm converges slowly or even does not converge at all.

The only exception is the GR-Pulay scheme, which seems to be much less sensitive to the value of a_0 than other schemes. However, its convergence is overall quite slow. This behavior of the GR-Pulay mixing is quite expectable since it "wastes" half of its iterations on finding the best next guess. This approach is capable of eliminating bad guesses but the extra required iterations indicate that it can be hardly competitive with other algorithms if they are supplied with a good value of a_0.

We can conclude that the crucial question is not which mixing scheme to pick but how to choose the right a_0 for a given problem. If we choose a proper value of a_0, the classical Anderson scheme performs best (at least, for the classes of problems we have investigated).

Moreover, for more complicated examples, a good estimate for the value of a_0 seems to be necessary for achieving a reasonable convergence rate or even to have the calculation converged at all (see Graphene fragment in Fig. 1).

Although the preconditioning techniques like those described above exist, unfortunately, there is no general prescription on how to choose the mixing parameter a_0 to obtain the best possible convergence. The derivation of the optimal choice of a_0 exists only for original Broyden's history-length-one methods in the case of a linear problem [67], however, there is no generalization available for the longer-history methods or non-linear problems. The common practice is to let the user supply a_0 and keep the $a_i = a_0$ fixed during the self-consistent cycle. In his recent work, Anderson suggests an adaptation of the mixing parameter [62], however, no numerical examples of the advantages have been provided. Probably the most interesting work in this field has been published recently by Marks [3], where the problem was formulated using the Broyden approach, incorporating the mixing parameter into the Jacobian approximation and suggesting a kind of adaptive scheme (see also his earlier paper [7], based on the previous work of Shano and Pula [68]).

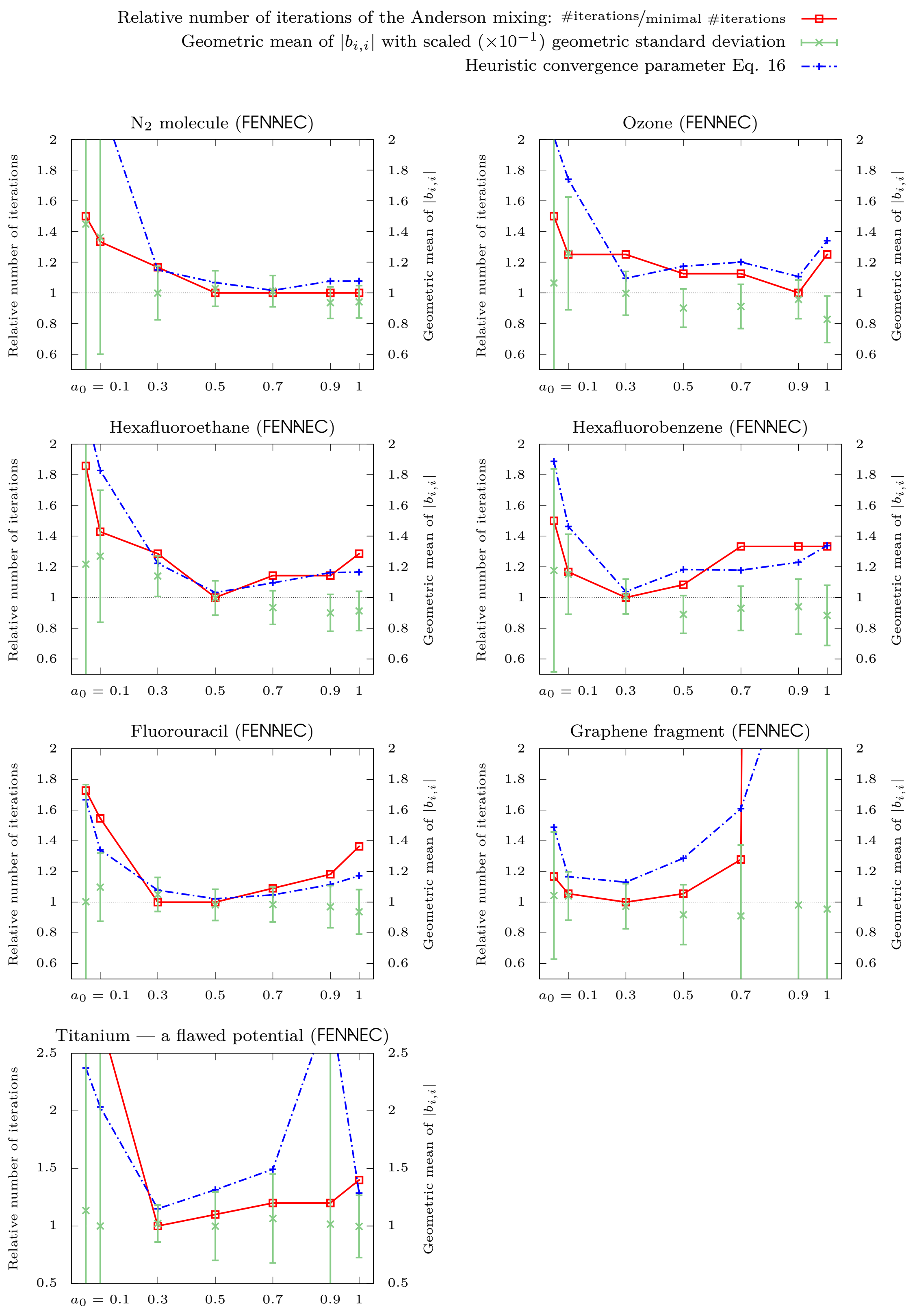


Fig. 2: Influence of the coefficients $b_{i,i}$ on the number of iterations needed to achieve convergence for the Anderson mixing. $b_{i,i}$ are given in terms of their geometric mean with geometric standard deviation and in terms of the heuristic convergence parameter (Eq. 16).

However, in most of the current DFT codes, the mixing parameter a_0 is viewed as a kind of a “magic parameter”. The main merit of the present paper is a new original algorithm for determining suitable mixing parameters by means of an adaptive scheme during Anderson’s iterations.

4 The Adaptive Anderson mixing

The coefficient $b_{i,i}$ in Eq. 9 expresses the quotient of the last input density/potential and residual in the next input density/potential. The observation crucial for our further considerations is that the convergence of the Anderson mixing scheme correlates with how these coefficients are close to one during the iterations.

In further reasoning we will use the geometric mean value b of these coefficients (over all the iterations of the cycle):

$$b = \sqrt[n]{\prod_i \mathrm{abs}(b_{i,i})} \tag{14}$$

We use the geometric mean since $b_{i,i}$ is used for multiplication in Eq. 9 and thus we are interested in the mean with respect to the logarithmic scale. Geometric standard deviation is then defined as

$$\mathrm{GSD}\left(\{b_{i,i}\}\right) = \exp\left(\sqrt{\frac{\sum_i \log\left({}^{b_{i,i}}/_{b}\right)}{n}}\right) . \tag{15}$$

A large value of $\mathrm{GSD}\left(\{b_{i,i}\}\right)$ indicates large oscillations of the values of $b_{i,i}$ around the mean value b.

If the coefficients $b_{i,i}$ differ significantly from one, which can be indicated either by the mean value b or by the presence of large oscillations indicated by large $\mathrm{GSD}\left(\{b_{i,i}\}\right)$, the convergence of the self-consistent cycle is poor: compare the green geometric means of $|b_{i,i}|$ with the red curves in Fig. 2.

On the basis of this observation, the following convergence criterion has been suggested: the convergence is the better, the lower is the heuristic convergence parameter c.

Definition 1 (Heuristic convergence parameter)**.** *For a given Anderson self-consistent cycle, the* heuristic convergence parameter *is a real number c defined as*

$$c = \max\left(b, \frac{1}{b}\right) \sqrt[4]{\mathrm{GSD}\left(\{b_{i,i}\}\right)} . \tag{16}$$

This heuristic convergence parameter (see its visualization in Fig. 3) has been constructed empirically. However, it seems that it can describe the speed of convergence quite well: see the agreement of the red and blue curves in Fig. 2.

4.1 Relation of a_i and $b_{i,i}$

The observation mentioned above suggests that the optimal value of a_i could be guessed using the coefficients $b_{i,i}$. In this paper, we suggest a new adaptive mixing scheme based on these observations.

However, until this correlation is explored in a mathematically more rigorous way, an explanation of the functionality of the suggested adaptivity scheme can be based merely on correlations that commonly occur, within an empirical/heuristic approach, which we will present as follows:

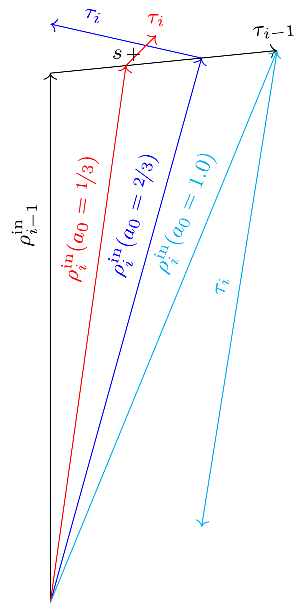
First let us assume that $b_{i,i} \approx 1$. Recalling Eq. 9, it follows that $\sum_{j=1}^{i-1} b_{i,j} \approx 0$ and the new density ρ^{in}_{i+1} can be written as

$$\rho^{\text{in}}_{i+1} \approx \rho^{\text{in}}_{i} + a_i\tau_i + \sum_{j=1}^{i-1}\left(b_{i,j}\rho^{\text{in}}_{j} + a_i b_{i,j}\tau_j\right) \ . \tag{17}$$

Therefore, the latest iteration forms the major part of the new input density. Its other components (the sum in Eq. 17) can be viewed as small corrections to the mix of the latest input and output density. Since the coefficients $b_{i,j}$ minimize the residual, the last step has lowered the residual more than the previous steps: the self-consistent cycle converges well. This is illustrated by the red lines in Fig. 4.

If $b_{i,i}$ is substantially lower than one, then the substantial part of the new input density is formed by the previous steps, not the last one. It can have two reasons: Either the latest residual has a larger magnitude than the previous ones — in this case, the latest input density ρ^{in}_{i} is too far away from the desired solution (see the cyan curve in Fig. 4) — or the lastest residual can be (at least partially) canceled out by a previous one (the blue curve). In both cases, the latest guess

$$\rho^{\text{in}}_{i} = \rho^{\text{in}}_{i-1} + a_{i-1}b_{i-1,i-1}\tau^{\text{in}}_{i-1} + \dots \tag{18}$$



τ_i
τ_i
τ_{i-1}
s
ρ^{in}_{i-1}
$\rho^{\text{in}}_i(a_0 = 1/3)$
$\rho^{\text{in}}_i(a_0 = 2/3)$
$\rho^{\text{in}}_i(a_0 = 1.0)$
τ_i

Fig. 4: An overestimation of a_0 leads to overstepping of the desired solution s and to oscillations.

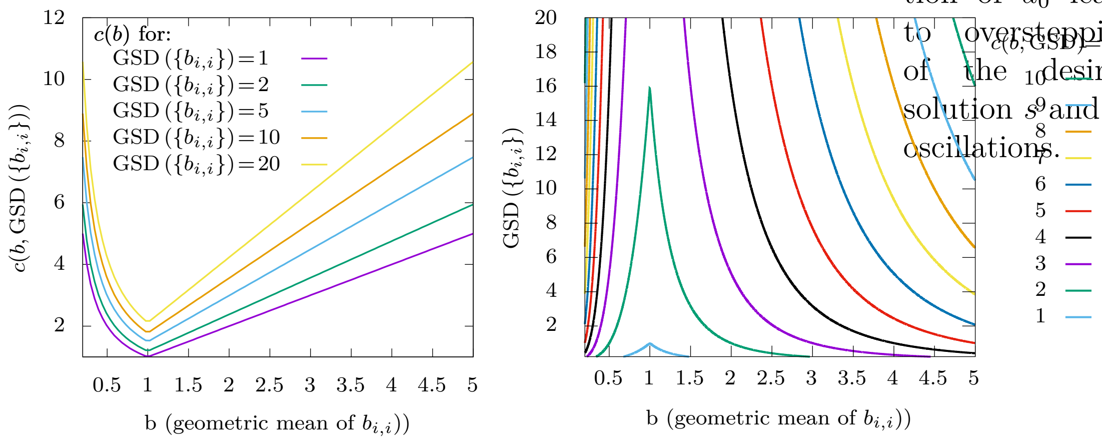


Fig. 3: Visualisation of the heuristic convergence parameter $c(b, \text{GSD}(\{b_{i,i}\}))$: the function values for given GSDs (left) and the contour lines of the function (right).

overstepped the solution[7]: the amount of residua added to ρ_i^{in} is too large. Decreasing a_i, we would obtain a better guess.

The opposite case, $b_{i,i} \gg 1$, can be explained using similar heuristic argumentation. In this case, $b_{i,i}$ must be counterbalanced by a negative $b_{i,k}$ (since $\sum_j b_{i,j} = 1$). Because the coefficients $b_{i,j}$ minimize Eq. 10, the τ_i and τ_k have to aim at similar directions (they have to cancel each other at least partially, otherwise the residual had not been minimized). Moreover, there must be (if we do not consider the cases of overdetermined linear dependent sets of residuals) no τ_j of an opposite direction and of a similar magnitude as τ_i: if such τ_j existed, the minimal linear combination of residuals could be obtained by summing these two residuals and so it could hardly be $b_{i,i} \gg 1$. Therefore, $b_{i,i} \gg 1$ indicates that we are approaching the solution still from the "same side" and thus we should enlarge a_i to speed up the convergence.

From these observations, a heuristic scheme for adapting the a_i coefficient during the Anderson iterations can be derived: if $b_{i,i}$ is greater than one, increase a_i, whereas if $b_{i,i}$ is smaller than one, decrease a_i. We call this algorithm *Adaptive Anderson scheme* onward.

The adaptation of a_i could (and for "reasonable" functions would) bring the values of $b_{i,i}$ close to one since the adaptation is in fact the approximation of the Jacobian of the function. We can demonstrate this link between the Jacobian and the coefficient $b_{i,i}$, if we simplify the case to the linearized problem. Let G be a multidimenzional linear function $G : \mathbb{R}^n \to \mathbb{R}^n$, defined by an invertible matrix D as follows:

$$G(\rho) = D\tau_\rho = D(\rho_{\text{root}} - \rho) \qquad D \in \mathbb{R}^{n\times n}\,, \tag{19}$$

for which its root ρ_{root} is sought. The application of the Anderson mixing (Eq. 9) with a given fixed $a_i = a_0$ and with $\rho_0^{\text{in}} = 0$ yields

$$\rho_0^{\text{in}} = 0 \qquad\qquad \tau_0 = G(\rho_0^{\text{in}}) - \rho_0^{\text{in}} = D\rho_{\text{root}} \tag{20}$$

$$\rho_1^{\text{in}} = a_0 d\rho_{\text{root}} \qquad\qquad \tau_1 = G(\rho_1^{\text{in}}) - \rho_1^{\text{in}} = D(I - a_0 D)\rho_{\text{root}}\,. \tag{21}$$

To obtain ρ_2^{in}, we have to minimize Eq. 10, under constraint $b_{1,0} + b_{1,1} = 1$, which yields:

$$\min \left\| D\left(b_{1,1} a_0 D\rho_{\text{root}} - \rho_{\text{root}}\right)\right\| \tag{22}$$

Let's recall that $-{a_0}^{-1}$ should be an approximation of the Jacobian J_G of function G, which is (from 19) equal to $-D$. Let us assume the best (and well) scalar approximation of the Jacobian a_0^{ideal}:

$$-\frac{1}{J_G} = D^{-1} \approx a_0^{\text{ideal}} \tag{23}$$

Then, if we substitute Eq. 23 into Eq. 22, we obtain

$$\min \left\| D\left(b_{1,1} D^{-1} D\rho_{\text{root}} - \rho_{\text{root}}\right)\right\|\,, \tag{24}$$

[7] We assume that the residuals are pointing (at least roughly) towards the solution; fortunately, the opposite case is not common in DFT calculations.

from which simply follows that

$$b_{1,1} = 1\,. \tag{25}$$

Thus, if we try to keep the $b_{i,i} = 1$ by adapting a_i, we, in fact, try to approximate the Jacobian of the function in so far unknown directions by the (implicitly) obtained (directional) derivatives of the function.

In a real case, we commonly do not guess the best Jacobian approximation. We again substitute to Eq. 22 the ideal scalar approximation of the Jacobian a_0^{ideal}

$$\min \left\| D \left(b_{1,1} \frac{a_0}{a_0^{\text{ideal}}} \rho_{\text{root}} - \rho_{\text{root}} \right) \right\| . \tag{26}$$

The above-derived expression is minimized if

$$b_{1,1} \frac{a_0}{a_0^{\text{ideal}}} = 1 \tag{27}$$

which insinuates the adaptation scheme for a_i:

$$a_i = a_{i-1} b_{i,i}\,. \tag{28}$$

However, in a real case, the Jacobian may not be approximable by one scalar number, and the use of the last directional directive (as in Eq. 28) could lead to oscillations. Thus, some kind of damping should be utilized. The details of the adaptation function used for the following numerical examples will be presented later.

5 Analysis and discussion of the Adaptive Anderson mixing algorithm

In Fig. 5, we compare the results obtained using the Adaptive Anderson mixing with the results obtained using the original Anderson mixing with a fixed $a_i \equiv a_0$ in the FENNEC code. We can note that the Adaptive Anderson mixing performs, in most cases, at least as well as the original Anderson algorithm for an optimal a_0 (or, in a few cases, only slightly worse). Additionally, the Adaptive Anderson mixing suffers much less from a bad initial guess of a_0, since the final adapted coefficients a_i approach the optimal a_0, see Fig **??**.

The above holds with two exceptions: The first one is the case of ozone O_3, where the convergence is the same for a wide range of a_0 and the optimal $a_0 = 0.9$ obtained in our calculation is evidently a “random result”. The second exception is titanium, which will be discussed below.

An interesting situation has been encountered when dealing with the graphene fragment. In this case, the values of adapted coefficients a_i vary during the self-consistent cycles. They tend to be slightly lower in the first iterations than in the latter ones (see Fig. 6). The change of the optimal value of a_i also indicates that the Jacobian of the function is not fixed. As a result, the final a_i differs from the optimal fixed $a_i \equiv a_0$ coefficient, and the adapted version of the algorithm performs better than the original version with any fixed a_0 since it allows to change the Jacobian approximation a_i during the iterations.

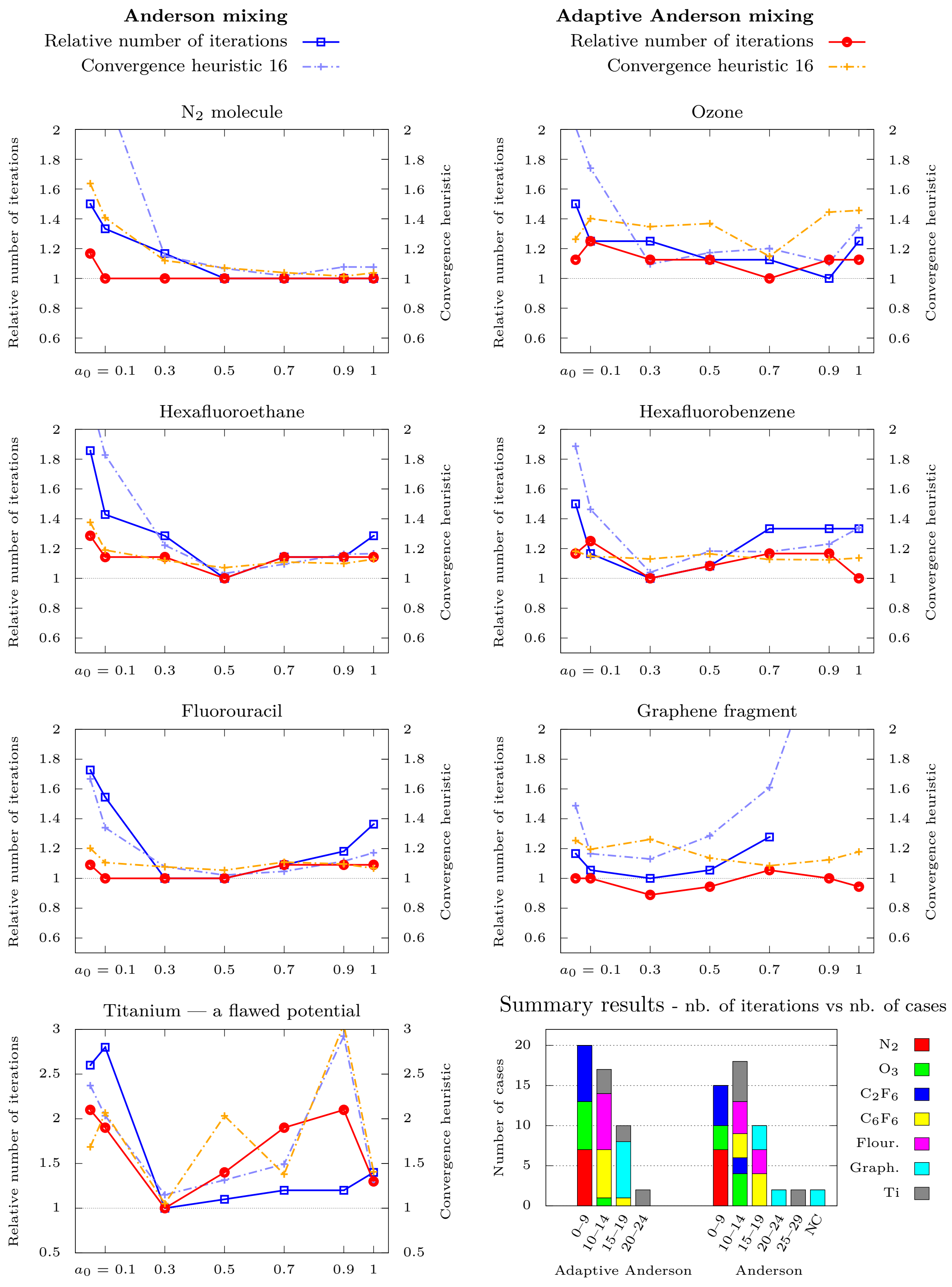


Fig. 5: Comparing the efficiency of the original Anderson mixing and of the Adaptive Anderson mixing suggested in this paper in FENNEC: each plot depicts the relative (with regard to the achieved minimal number of iterations for Anderson mixing) number of iterations, required for the convergence of the DFT self-consistent cycle for a specific system, in dependence on the a_0 coefficient. The relative convergence criteria were 1×10^{-9}. Missing values indicate that the self-consistent calculation has not converged.
The last plot depicts the statistics of the results in the form of a histogram: it shows the number of cases that require the given number of iterations to converge. 'NC' means not converged.

Thus, this case demonstrates that the adaptation is not only a tool for determining the right a_0 but it can improve the performance of the self-consistency cycle generally.

This observation can be further confirmed by analyzing the $b_{i,i}$ coefficients, employing the heuristic convergence parameter c — see Fig. 5: we can see that the largest oscillations of $b_{i,i}$ were substantially reduced.

Limitations of the approach The only exception to the previous observations is the case of the titanium atom for which the pseudopotential (see e.g. [69]) has been intentionally tuned to a wrong energy window. This is the reason why this simple example (a single atom) required more iterations to converge than e.g. hexafluorethane.

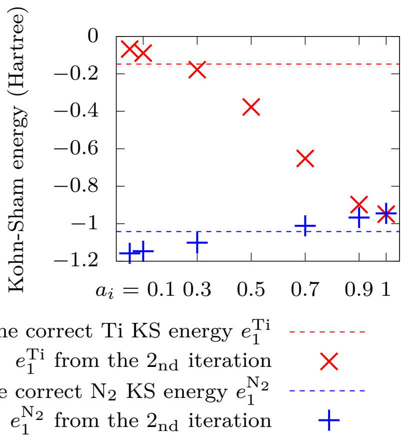


Fig. 7: Comparing the effects of having a "good" and a "bad" pseudopotential: dependence of the Kohn-Sham energy of the lowest-energy state during the second self-consistent iteration on the parameter a_0 for a nitrogen dimer and for a titanium atom with the system pseudopotential intentionally constructed with the shifted energy window. The correct final energies are marked by dashed lines.

Having a pseudopotential tuned to a given energy means that it acts properly just on the wave functions with eigenvalues not far from that specific energy.[70, 71] It is not necessary to have this energy tuned absolutely precisely, because the energy range, where the pseudopotential acts properly (denoted often as energy window), is usually — for norm-conserving pseudopotentials and their equivalents and derivatives — relatively wide. However, if the reference energy is too far from the correct energy of the true wave function, problems may arise. The wave functions naturally change their energy during the self-consistent cycle: therefore, the effect of such an improper choice of energy range changes too, which spoils the convergence of the cycle — due to the fact that the Anderson iterative process cannot guess the change of the pseudopotential-related error and thus the currently obtained derivative in the Jacobian of the function does not provide a good estimation for the value of Jacobian in the next iterations.

This effect is especially strong for a large initial a_0, which results in poor estimates of the charge densities in the first iterations where the energies of wave functions differ substantially from the final values (see Fig. 7).

This example has been added to illustrate the limitations of the proposed adaptive mixing. All fluctuations of the DFT self-consistent cycle are reflected in $b_{i,i}$, not only the ones caused by a wrong estimate of a_i. If such a secondary source of error is present (be it a wrongly tuned pseudopotential, a solver instability, or just a too "wild" optimized function), the adaptive mechanism for a_i not only does not help, but it can even completely spoil the convergence. Obviously, the adaptive scheme can just cure a bad guess of a_0 or it can help if there is a need for

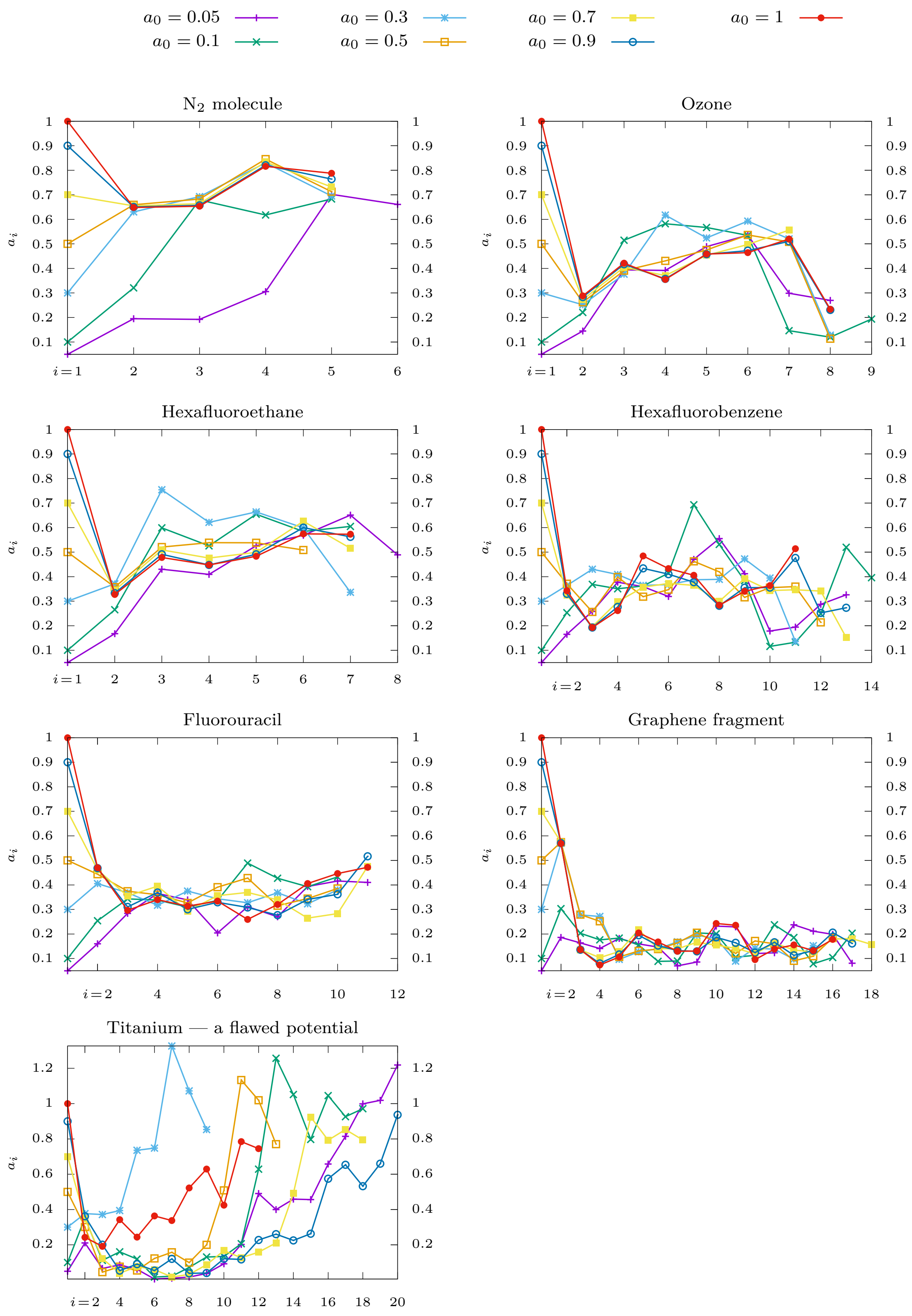


Fig. 6: Adaptive Anderson Mixing - adaptation of the coefficient a_i during iterations for various systems calculated by FENNEC. The iteration numbers are on the x-axis. Each plotted line denotes the evolution of coefficient a_i during iterations for a given starting a_0.

Algorithm 2: Adaptive Anderson mixing

```
1: $\rho_1^{\text{in}}$ = a guess of the charge density
2: **for** $i = 1, 2, \ldots$ **do**
3:     $\rho_i^{\text{out}} = F(\rho_i^{\text{in}})$
4:     $\tau_i = \rho_i^{\text{out}} - \rho_i^{\text{in}}$                                  ▷ New residual
5:     **if** $\|\tau_i\| <=$ threshold **then return**
6:     If some residuals are collinear to the newer ones, discard them from history
7:     Solve coefficients $b_{i,j}$ from Eq. 10                                  ▷ [18]
8:     $a_i = a_{i-1} * f_i(b_{i/i})$                                  ▷ Adaptation of $a_i$
9:     $\rho_{i+1}^{\text{in}} = \sum_{j=1}^{i} b_{i,j}\rho_i^{\text{in}} + a_i b_{i,j}\tau_i$                                  ▷ Set the new input density
```

a smooth change of Jacobian approximation, but it cannot resolve other kinds of problems that affect the mixing algorithm performance. Therefore, the adaptive mixing should be used with proper care.

On the other hand, even in such a case, we can use the analysis of $b_{i,i}$, and the convergence heuristic (Eq. 16), as indicators for the best choice of a_0. In addition, the oscillations of $b_{i,i}$ can be also used to detect instabilities in the DFT self-consistent cycle. Moreover, we can still observe that even in this case, the convergence heuristic (Eq. 16) correlates with the speed of convergence. Thus, maybe a better adaptation function could handle even such a case of a DFT cycle with a secondary source of “stochastic” errors. This topic is to be a subject of further research.

5.1 Details of the adaptation function

The adaptation algorithm has been determined just empirically and we do not claim that it is the best possible choice under any conditions. Above, we show that the algorithm converges most efficiently if $b_{i,i} \doteq 1$. Further numerical tests showed that the optimal $b_{i,i}$ should be even slightly greater than one, the more the longer the available history is in the given step (to slightly underestimate the Jacobian for the sake of stability).

Thus, the optimal $b_{i,i}$ is set to

$$g_i = d_{\text{base}} + dh_i \,, \tag{29}$$

where d_{base} and d are the algorithm parameters with the default values of 10.0 and 0.02 and h_i is the available history length in the i_{th} step.

Special attention should be payed to the case, where the coefficient $b_{i,i}$ is negative. This case can (and usually is) in fact very similar to the "blue" case in Fig 4 — two residuals are subtracted from each other. However, (generally) due to the linear dependency of the residuals, the current one switched its "position of the best one" with a linear combination of the other residuals. Thus, the coefficient $b_{i,i}$ has a negative sign, but its magnitude is approximately the same as if the switching would not occur. Therefore, $b_{i,i}$ is used in its absolute value for the purpose of the adaptation.

The adaptation of a_i is determined via the following expression:

$$a_i = c_i a_{i-1} \qquad c_i = f_i(|b_{i,i}|/g_i)\,. \tag{30}$$

The function f should be designed so that it allows both to reach the correct a_i quickly and to prevent large fluctuations of a_i. For FENNEC that generally shows good convergence properties, f works well if designed as follows:

$$\bar{f} = \tag{31}$$
$$= 1/(t + \log(1/x/t)) \qquad \text{for} \qquad x < 1/t\,, \tag{32}$$
$$= x \qquad \text{for} \qquad 1/t \le x \le t\,, \tag{33}$$
$$= t + \log(x/t) \qquad \text{for} \qquad t < x\,, \tag{34}$$
$$f_i(x) = \bar{f}(x)^{1/p}\;. \tag{35}$$

Threshold t is set to two. The coefficient p can be one, two, or three, depending on (respectively) whether two, one, or none of the following criteria are met: (i) the adaptive change of a_i is in the same direction as in the previous step, and (ii) the adaptive change is in the same direction during all previous steps.

5.1.1 Further improvements of the adaptation function for the SPR-KKR code

The mixing in SPR-KKR code (based on KKR method [12]) seems to be a much more difficult problem than in the finite elements of FENNEC, probably due to the nature of the KKR Green function basis (which is non-local and highly non-orthogonal).

To overcome the difficulty, several empirical heuristic criteria that help to recognize the situations when not to apply the adaptivity have been introduced, see table 1. Their basic idea is, not to perform the adaptation, if either the last input has not been formed dominantly using the last residual, or if the coefficient associated with the last residual is only the “byproduct” of the linear dependency of the other vectors. In both cases, the reasoning in chapter **??** is not valid.

Moreover, in SPR-KKR code, the mixing algorithm sometimes flounders in the first steps, till it (sometimes “by pure chance”) catches the solution sufficiently close to the right solution. Especially the second iteration is often an “off-target shooting” – this fact is reflected by the first condition in 1. Thus, to improve the robustness of the scheme, it is undesirable to do large modification of a_i during the first iterations, ie. the rules to determine the coefficient p from (35) have to be modified: p is set to 2 either if the direction of the adaptation did not change or if the calculation converges quickly ($|\tau_i|/|\tau_{i-1}| < 0.1$), and to 3 otherwise.

The results obtained using this adaptation function are depicted in Fig. 8. To demonstrate the advantages of the newly proposed algorithm, we choose the structures with convergence problems [8]. Four of them were calculated for a wide

[8] Since we were focused on the hard-mixing problems, we did not attempt to tune the input parameters for SPR-KKR to achieve the best convergence.

#	Criteria	Description
1	$i = 2, \lvert\tau_i\rvert > \lvert\tau_{i-1}\rvert$	misconvergence in the second iteration
2	$\exists j : \lvert b_{i-1,i-1}\rvert < 5\lvert b_{i-1,j}\rvert$	the last ρ^{in} doesn't contain a significant part of the last τ
3	$c_i > 1 \,\&\, \exists j : \lvert b_{i-1,i-1}\rvert < {}^3\!/\!_4 \lvert b_{i-1,j}\rvert$	do not enlarge a_i if the last τ is not dominant in the last ρ^{in}
4	$c_i > 1 \,\&\, \lvert\tau_i\rvert / \lvert\tau_{i-1}\rvert > 5$	do not enlarge a_i if the mixing did not converge in the last iteration
5	$\lvert b_{i,i}\rvert > 1.2 \,\&\, \exists j,k : \lvert b_{i,j}\rvert \geq \lvert b_{i,k}\rvert > b_{i,i}$	do not enlarge a_i if $b_{i,i}$ is not one of the two 'dominant' coefficients
6	$\lvert b_{i,i}\rvert < 1.2 \,\&\, \exists j : b_{i,j} < -0.5\lvert b_{i,i}\rvert$	do not reduce a_i if there is a ('significant') negative coefficient $b_{i,j}$

Tab. 1: The do-not-adapt-a_i criteria. The number in the first column is contained in the verbose output of the code as the reason for not performing the adaptation. The second column contains the exact condition, that has to be satisfied to skip the adaptation. The descriptions in the third column are approximate — their purpose is to capture the idea of the criterion.

range of a_0 using both potential (labeled V_{Hxc}) and charge density (labeled ρ) mixing. The most problematic structure seems to be silicon carbide, which converges only for a few values of starting a_0 and only if the potential mixing is used.

Our newly proposed algorithm shows its advantage just for such hard problems, where the range of starting a_0 that converges successfully was substantially increased: we can observe this phenomenon in the last "summary" graph in Fig. 8. This histogram summary graph also shows that the average number of iterations achieved by adaptive Anderson mixing has been substantially lower than by original Anderson mixing. The NaCl case, where some starting a_0 converges using the Anderson algorithm, but not using the adaptive Anderson algorithm (however, and vice versa) suggests, that further improvements for the adaptation function could be made.

6 Adaptive Anderson mixing Fortran package

Implementation of the algorithm described above is provided in the Adaptive Anderson mixing Fortran package. The use of the package is very easy — the quick-start instructions are:

To run a self-consistent cycle in Fortran, first, call the function `adaptive_anderson_init(n,` ρ_0^{in}`)` where ρ_0^{in} is the initial guess of the charge density and `n` is the dimension of ρ_0^{in}. The result of this function is a pointer[9] to the newly allocated `state` object that keeps the whole inner state of the mixer.

[9] It has the type of `adaptive_anderson_state`, however, it needs not be typed and can be declared as a `class* pointer`. Nevertheless, it has to be allocatable.

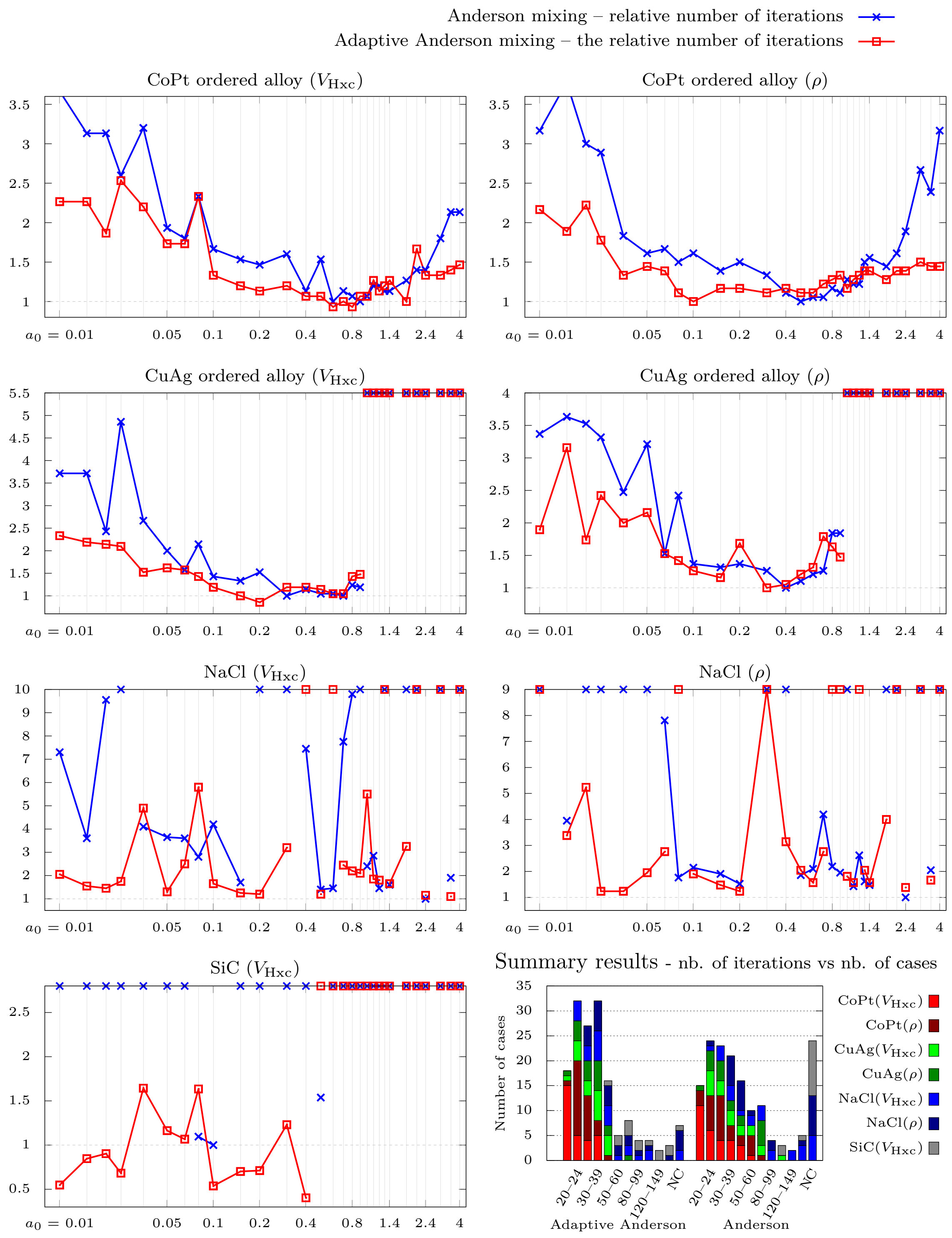


Fig. 8: Comparing the efficiency of the original Anderson mixing and the Adaptive Anderson mixing in SPRKKR: each plot depicts the relative (with respect to the achieved minimal number of iterations for Anderson mixing) number of iterations, required for the DFT self-consistent cycle to converge for a specific system, in dependence on the a_0 coefficient. The relative convergence criteria were set to 1×10^{-8}. Not-connected points on the upper edge denote a misconvergence for a given a_0.
The last plot depicts the statistics of the results in the form of a histogram: it shows the number of cases that require the given number of iterations to converge. ‘NC’ means not converged. Cases, where the both algorithms fail, are omitted here.

In addition to the parameters mentioned above, the function accepts several optional arguments, see table 2. Some of the arguments are marked as "experimental". Their effects are not thoroughly tested — they are intended as a playground for further possible enhancing of the algorithm.

After initialization of the algorithm, repeatedly evaluate the function the root of which is sought for the current input charge density (i.e., in the first iteration, for the initial guess ρ_0^{in}).

$$\tau_i = G(\rho_i) \tag{36}$$

and pass the resulting residual into the function `adaptive_anderson_step(state,` τ_i`,` ρ_{i+1}^{in}`)`. If a positive `threshold` parameter had been passed to `adaptive_anderson_init` and the threshold has been reached, the function returns `.TRUE.`. Otherwise, it forms the new input charge density (for which the function G should be evaluated in the next iteration) in the array ρ_{i+1}^{in} (which is given as the third argument) and returns `.FALSE.`.

And finally, when the desired accuracy is reached (either when the function returns `.TRUE.`, or when a custom check is satisfied), call `adaptive_anderson_end(state)` to clean up the memory. See algorithm 3 for the basic scheme of the usage of the package.

Algorithm 3: Usage of the Adaptive Anderson mixing package

```
1: $\rho$ = a guess of the charge density of the length n
2: state = adaptive_anderson_init(n, $\rho$, threshold, ...)
3: for $i = 0, 1, 2, \ldots$ do
4:     $\tau = G(\rho)$
5:     if adaptive_anderson_step(state, $\tau$, $\rho$) then
6:         break
7: adaptive_anderson_end(state)
8: return $\rho$
```

For Python programmers, the Cython[72] wrapper of the package is provided in the `adaptive_anderson_solver` package. The package provides the `solve` function with nearly the same interface as root-finding functions from the `scipy.optimize` package [66]. The function accepts (in addition to the arguments in table 2) as its first argument a callable that is able to evaluate the function G — it should accept ρ^{in} and return τ^{in}. All array arguments should be given as NumPy arrays [73]. If you want to implement your own stopping criterion, you can interrupt the iteration by raising `StopIteration` exception.

Alternatively, you can use a reverse communication strategy in the same way as in Fortran: you can create an object of the `AdaptiveAndersonSolver` class (also provided from the mentioned package `adaptive_anderson_solver`) and repeatedly call its `step` method, giving the current residuum and obtaining a new ρ^{in}. Freeing the memory is not needed here since it is done automatically during destructing the object.

Conclusion

In this paper, several existing mixing algorithms have been analyzed after a brief survey with regard to the suitability for electronic structure calculations within the density functional theory. Although the results vary, most of the analyzed algorithms evince similar performance for an ideal (optimally chosen) mixing parameter. Thus, the results suggest that determining the correct mixing parameter is more important than the choice of the particular mixing algorithm.

The main result of the paper is a finding that — within the Anderson mixing — there is a correlation between the degree of "optimality" of the mixing parameter and the mixing coefficient associated with the last residual. This fact has been used to propose a new mixing scheme called Adaptive Anderson mixing. This new mixing scheme — as it is summarized in Algorithm 2 — differs from the standard Anderson mixing scheme by adding a mechanism to determine the optimal mixing parameter at each iterative step, adaptively, according to charge densities in previous steps. It can be easily implemented within the existing software packages.

This mixing scheme has been implemented into two DFT codes, FENNEC and SPR-KKR. In both of them, it has proved its advantages. However, the SPR-KKR requires a bit more attention to design the function that adapts the mixing parameter. The performed numerical tests show that, especially for complex systems and/or hard-to-converge cases, using the adaptive a_i can lead to a better performance than relying on constant $a_i \equiv a_0$, even in the case of optimal choice of a_0 (see graphene fragment results in Fig. 5 or silicon carbide in Fig. 8). Besides that, the adaptive scheme substantially reduced the dependence on the initial guess of a_0 and can substantially increase the range of starting a_0, for which the calculation converges.

There are still open questions regarding the proposed mixing. The adaptation function has been designed empirically to examine the idea of adaptive mixing, without any ambition to claim that it is the best feasible mixing scheme. A function proposed in a more sophisticated way could probably provide even better results, particularly as regards the robustness of the scheme (see the titanium case in Fig. 5).

Also, the principles of the mixing themselves — especially the criteria assessing the a_i and $b_{i,i}$ coefficients — are worth a more rigorous investigation. Such an effort could shed more light on the scheme's functionality and maybe allow designing a more efficient scheme. We hope to deal with those topics in a more rigorous way in the future.

The Fortran implementation of the Adaptive Anderson mixing and Python (Cython) wrapper comes with the article and the paper contains brief documentation for the users of the provided package.

Arguments of `adaptive_anderson_init`

parameter	type	default	description
`n`	`I`		The size of the ρ^{in}.
`x0`	`R(n)`		The array containing initial ρ_0^{in}.
optional arguments			
`history`	`I`	10	History length — the number of remembered input density/residual pairs.
`tolerance`	`R`	(None)	The convergence threshold. See the discussion of `adaptive_anderson_step`.
`alpha`	`R`	0.5	Mixing parameter a_0.
`adaptive_alpha`	`L`	`.TRUE.`	If false, use the standard (non-adaptive) Anderson mixing.
`delta`	`R`	1.0	Adaptation coeficient d_{base}, see (29).
`delta_per_vector`	`R`	0.02	Adaptation coeficient d, see (29).
`weights`	`R(n)`	(None)	If the vector of weights (w_i) is given, the L^2 norms used thorough the algorithm are evaluated as follows: $\sqrt{\sum x_i^2 w_i}$. To be used e.g. if components of the ρ vector correspond to spatial elements with varying volumes.
`norm_tolerance`	`L`	`.TRUE.`	If true, the `threshold` is adjusted to be "`weights` independent", i.e. it is multiplied by $\sqrt{n/\sum_i w_i}$
`collinearity_ threshold`	`R`	`1e-10`	Residuals, whose linearly-independent component has norm lower than $\mathtt{V} * \lvert\tau_i\rvert$ are omitted from minimizing.
`adapt_from`	`I`	`0`	Do not adapt a_0 in the first $\mathtt{V} - 1$ iterations.
experimental optional arguments			
`regularization_ lambda`	`R`	`0.`	A simple regularization: this coefficient is added to the diagonal of the matrix of the scalar product of the residuals.
`restart_ threshold`	`R`	`0.`	In each step, discard the residuals whose norm is larger than $\lvert\tau_i\rvert$/`restart_threshold`.
`b_ii_switch_to_ linear`	`R`	`0.`	If $b_{i,i}/b_{i-1,i-1} > \mathtt{V}$ or $< 1/\mathtt{V}$, switch to linear mixing.
`linear_if_ cycling`	`R`	`0.`	If $\exists j \le i : \lvert\rho_{i+1}^{\text{in}} - \rho_j^{\text{in}}\rvert / \lvert\rho_i^{\text{in}}\rvert < a_i * \mathtt{V}$, switch to linear mixing.
optional arguments for debugging			
`debug_store_ to_file`	`I`	`0`	If nonzero, file handlers `V` and $\mathtt{V} + 1$ are used for storing ρ_i^{in}, `weights` and τ_i to files `and_(inputs|residuals|weights).data`.
`verbosity`	`I`	`0`	The amount of the printed information about the mixing. All lines are prepended by `AAMIX`. Zero means no output.

Tab. 2: Arguments of the function `adaptive_anderson_init`. Type `I` stands for integer, `R` for real*8, `L` for logical (boolean). `V` in the description stands for the value of the described option.

References

[1] Donald G Anderson. “Iterative procedures for nonlinear integral equations”. In: *Journal of the ACM (JACM)* 12.4 (1965), pp. 547–560.

[2] Péter Pulay. “Convergence acceleration of iterative sequences. The case of SCF iteration”. In: *Chemical Physics Letters* 73.2 (1980), pp. 393–398.

[3] L. D. Marks. “Predictive Mixing for Density Functional Theory (and Other Fixed-Point Problems)”. In: *Journal of Chemical Theory and Computation* 17.9 (2021). PMID: 34398610, pp. 5715–5732. DOI: `10.1021/acs.jctc.1c00630`. eprint: `https://doi.org/10.1021/acs.jctc.1c00630`. URL: `https://doi.org/10.1021/acs.jctc.1c00630`.

[4] Miriam Winkelmann et al. “Kerker mixing scheme for self-consistent muffin-tin based all-electron electronic structure calculations”. In: *Physical Review B* 102.19 (2020), p. 195138.

[5] Jongmin Kim, Andris Gulans, and Claudia Draxl. “Robust mixing in self-consistent linearized augmented planewave calculations”. In: *Electronic Structure* 2.3 (2020), p. 037001.

[6] Shashikant Kumar, Qimen Xu, and Phanish Suryanarayana. “On preconditioning the self-consistent field iteration in real-space Density Functional Theory”. In: *Chemical Physics Letters* 739 (2020), p. 136983.

[7] LD Marks. “Fixed-point optimization of atoms and density in DFT”. In: *Journal of chemical theory and computation* 9.6 (2013), pp. 2786–2800.

[8] Georg Kresse and Jürgen Furthmüller. “Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set”. In: *Physical review B* 54.16 (1996), p. 11169.

[9] Robert Cimrman et al. “Isogeometric analysis in electronic structure calculations”. In: *Mathematics and Computers in Simulation* 145 (2018), pp. 125–135.

[10] Robert Cimrman et al. “Convergence study of isogeometric analysis based on Bézier extraction in electronic structure calculations”. In: *Applied Mathematics and Computation* 319 (2018), pp. 138–152.

[11] Jiří Vackář et al. “Finite Element Method in Density Functional Theory Electronic Structure Calculations”. In: *Advances in the Theory of Quantum Systems in Chemistry and Physics*. Ed. by P. E. Hoggan et al. Vol. 22. Progress in Theoretical Chemistry and Physics. Springer, 2012. Chap. 12, pp. 199–217. ISBN: 978-94-007-2075-6, 978-94-007-2076-3. DOI: `10.1007/978-94-007-2076-3\_12`.

[12] H Ebert, D Ködderitzsch, and J Minár. “Calculating condensed matter properties using the KKR-Green’s function method—recent developments and applications”. In: *Reports on Progress in Physics* 74.9 (Aug. 2011), p. 096501. DOI: `10.1088/0034-4885/74/9/096501`. URL: `https://doi.org/10.1088%2F0034-4885%2F74%2F9%2F096501`.

[13] Haw-ren Fang and Yousef Saad. "Two classes of multisecant methods for nonlinear acceleration". In: *Numerical Linear Algebra with Applications* 16.3 (2009), pp. 197–221.

[14] Peter N Brown and Youcef Saad. "Hybrid Krylov methods for nonlinear systems of equations". In: *SIAM Journal on Scientific and Statistical Computing* 11.3 (1990), pp. 450–481.

[15] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[16] Duane D. Johnson. "Modified Broyden's method for accelerating convergence in self-consistent calculations". In: *Physical review. B, Condensed matter* 38 (18 Jan. 1989), pp. 12807–12813. DOI: `10.1103/PhysRevB.38.12807`.

[17] V Eyert. "A comparative study on methods for convergence acceleration of iterative vector sequences". In: *Journal of Computational Physics* 124.2 (1996), pp. 271–285.

[18] DePrince research group. *Programming Projects: DIIS convergence acceleration in SCF*. URL: `https://www.chem.fsu.edu/~deprince/programming_projects/diis/` (visited on 12/09/2019).

[19] Homer F Walker and Peng Ni. "Anderson acceleration for fixed-point iterations". In: *SIAM Journal on Numerical Analysis* 49.4 (2011), pp. 1715–1735.

[20] Robert Cimrman et al. "Convergence study of isogeometric analysis based on Bézier extraction in electronic structure calculations". In: *Applied Mathematics and Computation* (2017).

[21] Stan Cabay and LW Jackson. "A polynomial extrapolation method for finding limits and antilimits of vector sequences". In: *SIAM Journal on Numerical Analysis* 13.5 (1976), pp. 734–752.

[22] M Mešina. "Convergence acceleration for the iterative solution of the equations X= AX+ f". In: *Computer Methods in Applied Mechanics and Engineering* 10.2 (1977), pp. 165–173.

[23] Claude Brezinski et al. "Shanks and Anderson-type acceleration techniques for systems of nonlinear equations". In: *arXiv preprint arXiv:2007.05716* (2020).

[24] Alex Toth et al. "Local improvement results for Anderson acceleration with inaccurate function evaluations". In: *SIAM Journal on Scientific Computing* 39.5 (2017), S47–S65.

[25] Thorsten Rohwedder and Reinhold Schneider. "An analysis for the DIIS acceleration method used in quantum chemistry calculations". In: *Journal of mathematical chemistry* 49.9 (2011), p. 1889.

[26] Charles G Broyden. "A class of methods for solving nonlinear simultaneous equations". In: *Math. Comput.* 19.92 (1965), pp. 577–593. DOI: `10.1090/S0025-5718-1965-0198670-6`.

[27] José Mario Martínez. "A quasi-Newton method with modification of one column per iteration". In: *Computing* 33.3 (1984), pp. 353–362.

[28] José Mario Martínez and Mário C. Zambaldi. “An inverse column-updating method for solving large–scale nonlinear systems of equations”. In: *Dynamical Systems* 1.2 (1992), pp. 129–140.

[29] José Mario Martínez. “Algorithms for solving nonlinear systems of equations”. In: *Algorithms for Continuous Optimization*. Springer, 1994, pp. 81–108.

[30] David M Gay and Robert B Schnabel. “Solving systems of nonlinear equations by Broyden’s method with projected updates”. In: *Nonlinear Programming 3*. Elsevier, 1978, pp. 245–281.

[31] Junzi Zhang, Brendan O’Donoghue, and Stephen Boyd. “Globally convergent type-I Anderson acceleration for nonsmooth fixed-point iterations”. In: *SIAM Journal on Optimization* 30.4 (2020), pp. 3170–3197.

[32] Amartya S Banerjee, Phanish Suryanarayana, and John E Pask. “Periodic Pulay method for robust and efficient convergence acceleration of self-consistent field iterations”. In: *Chemical Physics Letters* 647 (2016), pp. 31–35.

[33] DR Bowler and MJ Gillan. “An efficient and robust technique for achieving self consistency in electronic structure calculations”. In: *Chemical Physics Letters* 325.4 (2000), pp. 473–476.

[34] William W Hager and Hongchao Zhang. “A survey of nonlinear conjugate gradient methods”. In: *Pacific journal of Optimization* 2.1 (2006), pp. 35–58.

[35] H De Sterck. “A nonlinear GMRES optimization algorithm for canonical tensor decomposition”. In: *SIAM Journal on Scientific Computing* 34.3 (2012), A1351–A1379.

[36] Cornelis Vuik and Henk A van der Vorst. “A comparison of some GMRES-like methods”. In: *Linear Algebra and its Applications* 160 (1992), pp. 131–162.

[37] Henk A Van der Vorst. *Iterative Krylov methods for large linear systems*. 13. Cambridge University Press, 2003.

[38] Timo Eirola and Olavi Nevanlinna. “Accelerating with rank-one updates”. In: *Linear Algebra and its Applications* 121 (1989), pp. 511–520.

[39] Ulrike Meier Yang. *A family of preconditioned iterative solvers for sparse linear systems*. 1904. Citeseer, 1995.

[40] Alexander Craig Aitken. “XXV.—On Bernoulli’s Numerical Solution of Algebraic Equations”. In: *Proceedings of the Royal Society of Edinburgh* 46 (1927), pp. 289–305.

[41] Isabelle Ramière and Thomas Helfer. “Iterative residual-based vector methods to accelerate fixed point iterations”. In: *Computers & Mathematics with Applications* 70.9 (2015), pp. 2210–2226.

[42] C Lemaréchal. “Une méthode de résolution de certains systèmes non linéaires bien posés”. In: *CR Acad. Sci. Paris, sér. A* 272 (1971), pp. 605–607.

[43] Bruce M Irons and Robert C Tuck. “A version of the Aitken accelerator for computer iteration”. In: *International Journal for Numerical Methods in Engineering* 1.3 (1969), pp. 275–277.

[44] Daniel Shanks. “Non-linear transformations of divergent and slowly convergent sequences”. In: *Journal of Mathematics and Physics* 34.1-4 (1955), pp. 1–42.

[45] Peter R Brune et al. “Composing scalable nonlinear algebraic solvers”. In: *siam REVIEW* 57.4 (2015), pp. 535–565.

[46] Lin Lin and Chao Yang. “Elliptic preconditioner for accelerating the self-consistent field iteration in Kohn–Sham density functional theory”. In: *SIAM Journal on Scientific Computing* 35.5 (2013), S277–S298.

[47] Marcus Heide and Tomoya Ono. “Convergence of the Broyden Density Mixing Method in Noncollinear Magnetic Systems”. In: *Journal of the Physical Society of Japan* 82.11 (2013), p. 114706.

[48] G. P. Kerker. “Efficient iteration scheme for self-consistent pseudopotential calculations”. In: *Physical Review B* 23 (6 Mar. 1981). DOI: `10.1103/PhysRevB.23.3082`. URL: `http://gen.lib.rus.ec/scimag/index.php?s=10.1103/PhysRevB.23.3082`.

[49] Yuzhi Zhou et al. “Applicability of Kerker preconditioning scheme to the self-consistent density functional theory calculations of inhomogeneous systems”. In: *Physical Review E* 97.3 (2018), p. 033305.

[50] Yoshinori Shiihara, Osamu Kuwazuru, and Nobuhiro Yoshikawa. “Real-space Kerker method for self-consistent calculation using non-orthogonal basis functions”. In: *Modelling and Simulation in Materials Science and Engineering* 16.3 (2008), p. 035004.

[51] Raffaele Resta. “Thomas-Fermi dielectric screening in semiconductors”. In: *Phys. Rev. B* 16 (6 Sept. 1977), pp. 2717–2722. DOI: `10.1103/PhysRevB.16.2717`. URL: `https://link.aps.org/doi/10.1103/PhysRevB.16.2717`.

[52] P-M Anglade and Xavier Gonze. “Preconditioning of self-consistent-field cycles in density-functional theory: The extrapolar method”. In: *Physical Review B* 78.4 (2008), p. 045126.

[53] Kai-Ming Ho, J Ihm, and JD Joannopoulos. “Dielectric matrix scheme for fast convergence in self-consistent electronic-structure calculations”. In: *Physical Review B* 25.6 (1982), p. 4260.

[54] Pierre Hohenberg and Walter Kohn. “Inhomogeneous electron gas”. In: *Physical review* 136.3B (1964), B864.

[55] P Haynes. “Linear-scaling methods in ab initio quantum-mechanical calculations”. PhD thesis. University of Cambridge, 1998.

[56] Yan Alexander Wang et al. “Communication: Linear-expansion shooting techniques for accelerating self-consistent field convergence”. In: *The Journal of chemical physics* 134.24 (2011), p. 241103.

[57] Konstantin N Kudin, Gustavo E Scuseria, and Eric Cances. “A black-box self-consistent field convergence algorithm: One step closer”. In: *The Journal of chemical physics* 116.19 (2002), pp. 8255–8261.

[58] Xiangqian Hu and Weitao Yang. “Accelerating self-consistent field convergence with the augmented Roothaan–Hall energy function”. In: *The Journal of chemical physics* 132.5 (2010), p. 054109.

[59] Alejandro J Garza and Gustavo E Scuseria. “On the equivalence of LIST and DIIS methods for convergence acceleration”. In: *The Journal of chemical physics* 142.16 (2015), p. 164104.

[60] Paul Bendt and Alex Zunger. “Simultaneous relaxation of nuclear geometries and electric charge densities in electronic structure theories”. In: *Physical Review Letters* 50.21 (1983), p. 1684.

[61] Wenqing Ouyang et al. “Nonmonotone Globalization for Anderson Acceleration Using Adaptive Regularization”. In: *arXiv preprint arXiv:2006.02559* (2020).

[62] Donald GM Anderson. “Comments on “Anderson acceleration, mixing and extrapolation””. In: *Numerical Algorithms* 80.1 (2019), pp. 135–234.

[63] Damien Scieur, Alexandre d’Aspremont, and Francis Bach. “Regularized nonlinear acceleration”. In: *Mathematical Programming* 179.1 (2020), pp. 47–83.

[64] LD Marks and DR Luke. “Robust mixing for ab initio quantum mechanical calculations”. In: *Physical Review B* 78.7 (2008), p. 075114.

[65] Ya-xiang Yuan. “Recent advances in trust region algorithms”. In: *Mathematical Programming* 151.1 (2015), pp. 249–281.

[66] Pauli Virtanen et al. “SciPy 1.0: fundamental algorithms for scientific computing in Python”. In: *Nature methods* 17.3 (2020), pp. 261–272.

[67] Peter Deuflhard, Roland Freund, and Artur Walter. “Fast secant methods for the iterative solution of large nonsymmetric linear systems”. In: *IMPACT of Computing in Science and Engineering* 2.3 (1990), pp. 244–276.

[68] David F Shanno and Kang-Hoh Phua. “Matrix conditioning and nonlinear optimization”. In: *Mathematical Programming* 14.1 (1978), pp. 149–160.

[69] Jiří Vackář, Antonín Šimůnek, and Raimund Podloucky. “Ab initio pseudopotentials for interacting atoms”. In: *Phys. Rev. B* 53 (12 Mar. 1996), pp. 7727–7730. DOI: `10.1103/PhysRevB.53.7727`. URL: `https://link.aps.org/doi/10.1103/PhysRevB.53.7727`.

[70] Jiří Vackář and Antonín Šimůnek. “Adaptability and accuracy of all-electron pseudopotentials”. In: *Physical Review B* 67.12 (2003), p. 125113.

[71] J Vackar and A Simunek. “Modelling the valence electronic structure of the core region of an atom in a solid within a local-density approximation pseudopotential framework: reintroduction of the full nodal form”. In: *Journal of Physics: Condensed Matter* 6.16 (1994), p. 3025.

[72] Stefan Behnel et al. “Cython: The best of both worlds”. In: *Computing in Science & Engineering* 13.2 (2011), pp. 31–39.

[73] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. “The NumPy array: a structure for efficient numerical computation”. In: *Computing in Science & Engineering* 13.2 (2011), pp. 22–30.