IOP Publishing Electron. Struct. 6 (2024) 037003 https://doi.org/10.1088/2516-1075/ad610f

Electronic Structure

OPEN ACCESS

RECEIVED
1 March 2024

REVISED
30 June 2024

ACCEPTED FOR PUBLICATION
9 July 2024

PUBLISHED
26 July 2024

TECHNICAL NOTE

Facilities and practices for linear response Hubbard parameters U and J in ABINIT

Lórien MacEnulty[1,*], Matteo Giantomassi[2], Bernard Amadon[3,4], Gian-Marco Rignanese[2] and David D O'Regan[1]

[1] School of Physics, Trinity College Dublin, The University of Dublin, Dublin 2, Ireland
[2] Université Catholique de Louvain, Institute of Condensed Matter and Nanosciences, Chemin des Étoiles 8, Louvain-la-Neuve B-1348, Belgium
[3] CEA, DAM, DIF, 91297 Arpajon Cedex, France
[4] Université Paris-Saclay, CEA, Laboratoire Matière en Conditions Extrêmes, 91680 Bruyères-le-Châtel, France
[*] Author to whom any correspondence should be addressed.

E-mail: lmacenul@tcd.ie

Keywords: DFT+U, ABINIT, Hubbard U, Hunds J, linear response, PAW, electronic correlation

Abstract

Members of the density functional theory (DFT)+U family of functionals are increasingly prevalent methods of addressing errors intrinsic to (semi-) local exchange-correlation functionals at minimum computational cost, but require their parameters U and J to be calculated *in situ* for a given system of interest, simulation scheme, and runtime parameters. The self-consistent field (SCF) linear response approach offers *ab initio* acquisition of the U and has recently been extended to compute the J analogously, which measures localized errors related to exchange-like effects. We introduce a renovated post-processor, the `lrUJ` utility, together with this detailed best-practices guide, to enable users of the popular, open-source ABINIT first-principles simulation suite to engage easily with *in situ* Hubbard parameters and streamline their incorporation into material simulations of interest. Features of this utility, which may also interest users and developers of other DFT codes, include *n*-degree polynomial regression, error analysis, Python plotting facilities, didactic documentation, and avenues for further developments. In this technical introduction and guide, we place particular emphasis on the intricacies and potential pitfalls introduced by the projector augmented wave method, SCF mixing schemes, and non-linear response, several of which are translatable to DFT+U(+J) implementations in other packages.

1. Introduction

ABINIT [1–5] is an open-source electronic structure suite developed in the mid-1990s by Xavier Gonze and colleagues. The suite is equipped with a variety of *ab initio* techniques and the softwares that support them, including, but not limited to, time-dependent density functional theory (DFT), dynamical mean field theory, density functional perturbation theory, many-body perturbation theory, electron-phonon calculations, and projector-augmented wave (PAW) dataset generation.

Of particular interest here is ABINIT's primary implementation of approximative DFT and the family of Hubbard-like corrective extensions thereof (referred to here as DFT+U+J). The ground-state DFT scheme relies on a plane-wave self-consistent field (SCF) algorithm, into which developers comprehensively integrated the PAW method [6] in 2008 [7, 8]. Shortly thereafter, DFT + U was built on top of this PAW implementation, and programs were developed to calculate its namesake parameter, the Hubbard U, and other corrective objects *in situ* via linear response [9, 10].

How all these formalisms—DFT+U, PAW, linear response, SCF algorithms—overlap in ABINIT is not trivial, and there exists a need for comprehensive, more didactic documentation making explicit the link between the programs and the theory that inspired them [11]. Furthermore, state-of-the-art Hubbard corrective techniques and practices have advanced in the 15 years since the initial implementation of the

DFT+U and linear response utilities. For example, the Hund's exchange coupling J and the DFT + U + J functionals have seen more scientific attention than before [12–17], and ground has been made in calculating the Hund J via linear response [9, 10, 12, 18, 19]. ABINIT's DFT+U+J linear response implementation and support utilities were due for reevaluation and expansion.

In addition to the aforementioned aims, this manuscript seeks to illuminate the under-the-hood workings of Hubbard corrective protocol within the PAW formalism, which works in distinct ways among popular electronic structure codes like QuantumESPRESSO [20, 21] and VASP [22].

We address these matters in the current work. In section 2, we provide an overview of the many formalisms involved in ABINIT's DFT+U+J software, including the PAW method (section 2.1), Hubbard corrective protocol (section 2.2), and the linear response method for determining the Hubbard U and Hund's J parameters (collectively referred to herein as the 'Hubbard parameters,' the *in situ* determination of which is described in section 2.3). The review of this formalism is designed to reinforce our description of the technical implementation of DFT+U+J in ABINIT, which constitutes section 3.1 and includes details on how to invoke various Hubbard parameter-related features via the ABINIT input file (section 3). A closer look at ABINIT's relevant SCF mixing schemes can be found in appendix B.

Through the evaluation of ABINIT's linear response faculties in section 4, we found ABINIT-specific quirks requiring additional clarification as well as evidence of a defect that prompted its renovation and expansion to calculate the Hund's exchange J via first-principles. We describe these renovations in section 4.1 by comparing and contrasting the improved `UJdet` functionalities with their successor, the `lrUJ` post-processor, a user manual of which we document in 4.2. We then dissect the linear response procedure, following a SCF calculation and documenting the algorithm that transforms potential perturbations to occupancy responses to Hubbard parameters, a chronicle recounted in section 4.3.

A table of contents is provided for ease of navigation.

Contents

2. Background

The following sections comprise a description of the formalism on which the ABINIT DFT+U(+J) functionalities were built. We begin, in section 2.1 with a relatively detailed description of the PAW formalism, introducing its canonical basis functions and their origins. This description constructs the necessary foundation on which PAW basis set generators like *atompaw* are built, providing useful information on how to select a PAW dataset and pseudopotential for use in one's system. Moreover, the PAW and DFT+U(+J) formalisms are particularly entangled when it comes to the use of projectors to isolate the Hubbard subspaces for treatment. The following reiteration of the PAW formalism and its manifestations in DFT is intended to be convenient and accessible reference information for section 4.3, and within that, specifically section 4.3.2, as we describe how the DFT+U(+J) method is embedded within ABINIT's PAW implementation.

2.1. PAW Formalism

In 2008, ABINIT developers comprehensively integrated the PAW method, introduced by Peter Blöchl in 1994, into ABINIT DFT in order to simplify the numerical treatment of core electron wavefunctions, which are typically encumbered by tight oscillations near the nucleus of atoms. While we summarize the relevant details of the PAW formalism here, we refer the reader to Blöchl's original paper ([6]) for more information. Details on the PAW implementation in ABINIT can be found in [7, 8].

The PAW formalism starts by positing that there exists a true, full all-electron (AE) wavefunction Ψ—a Slater determinant of one-electron Kohn–Sham orbitals tightly oscillating about each atomic nucleus—that can be linked to a well-behaved pseudized wavefunction $\tilde{\Psi}$ with few or no oscillations by a linear transformation τ,

$$\Psi = \tau\tilde{\Psi}. \tag{1}$$

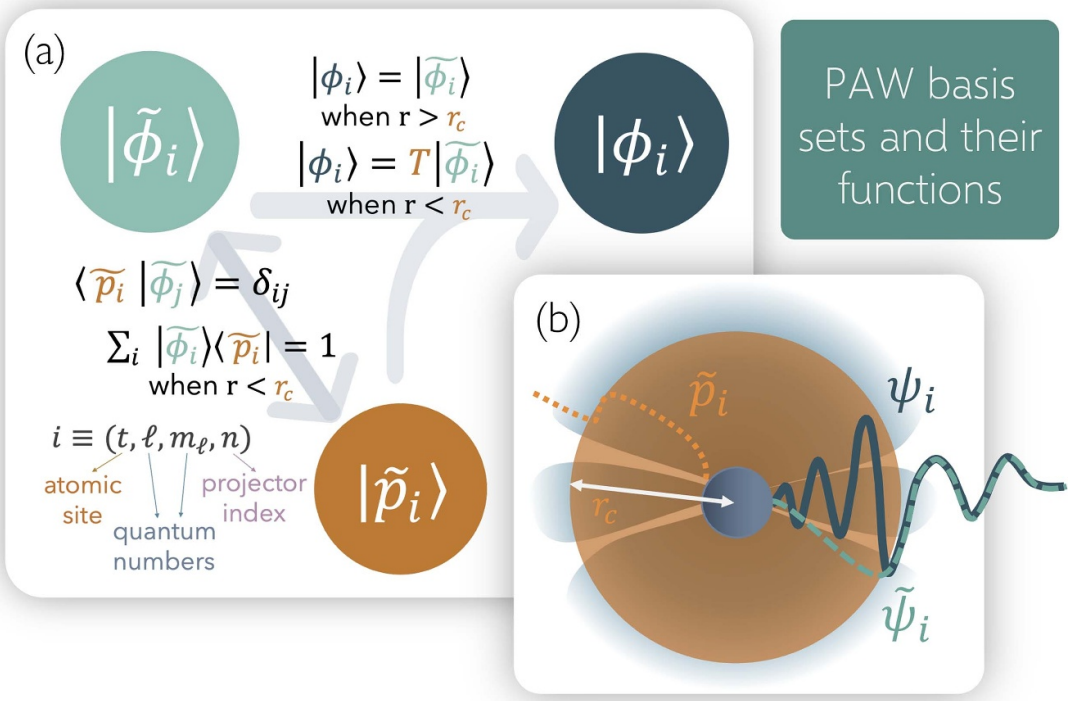

Figure 1. (a) Schematic outlining the mathematical relationship between the three canonical bodies used to construct PAW datasets. (b) The atom-centered augmentation sphere, defined by projector p_i, where i refers to the set of four indices $\{t, \ell, m_\ell, n\}$ inside cut off radius r_c. When $r > r_c$, the AE wavefunction ψ_i and the pseudo wavefunction $\tilde{\psi}_i$ are equivalent. T is a typo referring to the transformation operator τ.

With this postulate in hand, we can derive physical quantities using the expectation value of an operator A by sandwiching it between the transformation operator τ and its Hermitian conjugate $\tau^\dagger$ to yield the operator's pseudized counterpart, $\tilde{A}$. Accordingly, the variational principle for the total energy is

$$\frac{\partial E\left[\tau|\tilde{\Psi}\rangle\right]}{\partial\langle\tilde{\Psi}|} = \varepsilon\tau^\dagger\tau|\tilde{\Psi}\rangle \tag{2}$$

such that one may derive the pseudized equivalent of the Kohn–Sham equations. Thus, instead of looking for the ground state of our system in real space, one may seek the ground state energy in this particular pseudospace.

A better-defined transformation operator τ is necessary to pursue this method any further. As a prerequisite, τ must modify the smooth, pseudo-valence wavefunction within an atomic region in order to yield the correct nodal structure for the AE wavefunction. This modification is only necessary in the regions closest to the atomic nuclei. We then tailor our treatment specifically to this atom-centered region, spherically symmetric about the nucleus, by defining a cut-off radius, inside which we consider only the smooth, pseudized representation of the AE wavefunction, and outside of which the AE and pseudo wavefunctions are equivalent. Figure 1 illustrates this concept.

In this way, we effectively define an augmentation sphere: a spherically symmetric region of radius r_c centered around each atomic site in our system. Consider constructing a pseudized wavefunction for a particular orbital with quantum numbers ℓ and m_ℓ on atom t. When $r < r_c$, we must ensure that $\tilde{\psi}_i$ is a well-behaved projection of the AE wavefunction ψ_i. Moreover, when $r > r_c$, $\tilde{\psi}_i = \psi_i$. To satisfy these requirements, the transformation operator τ may be defined as an identity operator plus the sum of atomic orbital-based modifications,

$$\tau = \mathbb{1} + \sum_i \left(|\phi_i\rangle - |\tilde{\phi}_i\rangle\right)\langle\tilde{p}_i|, \tag{3}$$

where ϕ_i ($\tilde{\phi}_i$) are the AE (pseudo) partial wave basis functions with which we define the AE (pseudo) wavefunctions ψ_i ($\tilde{\psi}_i$). Here, i refers to the set of four indices $\{t, \ell, m_\ell, n\}$, representing the atomic site,

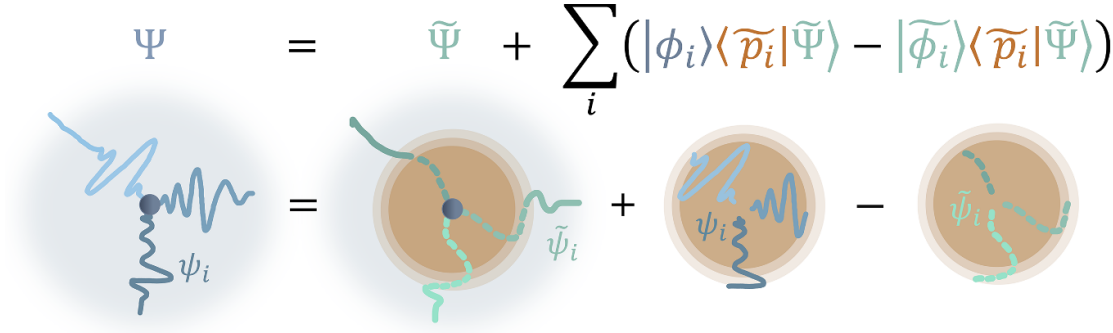

Figure 2. Schematic of the structure of the AE wavefunction in the PAW formalism. In this way, we can restrict our treatment of the cumbersome AE wavefunction to the interior of a particular region.

angular quantum number, magnetic quantum number, and projector index, respectively. The AE partial waves ϕ_i may be defined in any way, although an organic choice would be the bound and scattering state solutions to the Schrödinger equation for an isolated atom. In a manner analogous to the wavefunctions they construct, the pseudo partial waves $\tilde{\phi}_i$ corresponding to each $\tilde{\psi}_i$ are well-behaved projections of ϕ_i when $r < r_c$ and identical to ϕ_i when $r > r_c$. The objects that allow us to restrict these projections to the pertinent regions of space are the projector functions $\tilde{p}_i$. These are the three mathematical objects intrinsic to the PAW formalism (and those of any ultrasoft pseudo potential method, for that matter). Figure 1 makes explicit the nature of the mathematical duality between these functions. Crucially, note the relation between the pseudo partial waves and the projectors; they are dual to one another, and their projections are normalized inside the augmentation region.

We can represent the Kohn–Sham wavefunctions—which can be approximated as plane waves (i.e. waves with wave fronts parallel to flat planes that are azimuthally symmetric about the direction of propagation)—by an infinite sum of spherically symmetric constituents called partial waves (i.e. waves with spherical wave fronts that propagate along a radius emanating from a central point). Such partial waves, also known as spherical waves, are the products of spherical Bessel functions and the spherical harmonics, which are functions of angular momentum ℓ and azimuthal quantum number m_ℓ. In the PAW context, therefore, a partial wave refers to a wave that is spherically symmetric about an atom and a function of a given angular momentum ℓ.

Per equation (3), the atomic orbital based modifications that transform properties into their pseudo-space counterparts comprise the cumulative differences between (i) the projection of the AE partial wave on the augmentation region, and (ii) the pseudo partial waves on the augmentation region. The transformation operator, when applied to the wavefunction, is described visually in figure 2. Due to the structure of the transformation operator τ, objects such as the density and the energy are formulated analogously.

2.1.1. Construction of PAW datasets

The three PAW basis sets ψ_i, $\tilde{\psi}_i$, and $\tilde{p}_i$, are typically read into one's chosen DFT program from a file, called a PAW dataset, not so unlike a pseudopotential. A PAW dataset to be read into ABINIT should adhere to either XML format (extension `.xml`) or the Abinit proprietary format (extension `.abinit`). These datasets will typically include the following information.

(i) AE and PS partial waves, ψ_i and $\tilde{\psi}_i$, and projector functions, $\tilde{p}_i$, for all valence orbitals
(ii) AE core charge density, n_c
(iii) PS core charge density, $\tilde{n}_c$
(iv) PS valence charge density, $\tilde{n}$[1]
(v) A local ionic pseudopotential, $\tilde{\nu}_{\text{loc}}$ (see appendix A)
(vi) Information for the construction of the compensation charge, $\hat{n}$, (i.e. a shape function)

There exist a variety of publicly available programs to generate these datasets, the one with ties to ABINIT being *atompaw* [23–25].

These algorithms start by defining the AE partial waves as descriptions of all orbitals of the element under scrutiny. As mentioned earlier, the eigenfunctions of the solution to the Schrödinger equation for the isolated atom are a good starting point as they are partial waves, thus comprising a sum of products—radial

functions (typically of polynomial form, although other options are open for use) multiplied by spherical harmonics. DFT calculations are performed using the exchange-correlation potential, ν_{xc}, of choice to obtain the AE basis functions defined on a radial grid of variable mesh density. For this reason, PAW datasets are categorized in terms of the associated XC functional (e.g. PAW-LDA, PAW-PBE). The *atompaw* algorithm, specifically, mandates that the partial waves be defined as the atomic eigenfunctions resulting from these calculations.

We assume here that the electronic states resulting from this calculation can be separated into core orbitals—to be 'frozen' only in the sense that they are grouped with the nucleus and not represented individually by PAW basis sets—and valence orbitals, which will be represented individually in the PAW basis [25]. In principle, the frozen core approximation that inspired this treatment would necessitate the core electron density remain unchanged from that of the relaxed isolated atom despite immersion in different environmental potentials and configurations. By drawing more or less orbitals out of the core and into the valence, the accuracy of this frozen core approximation may be tailored to the system at hand. The number of PAW basis functions, used to describe these valence orbitals, follows from this decision.

In practice, and in ABINIT, a soft-core scheme is adopted to restore the core electrons' involvement in and response to the physics of the system. A pseudo density representing the nucleus and core electrons, $\tilde{n}_{Zc}$, is used to generate its namesake share of the Hartree potential, and a soft-core density, $\tilde{n}_c$, contributes to calculations of any non-linear core corrections [26]. Additionally, a compensation charge density, $\hat{n}$, is introduced to restore the correct multipole moments of the AE charge density $n^1 + n_{Zc}$, evaluated on a radial grid and exclusively inside the augmentation region [22].

Often, only one or two partial waves are necessary to accurately describe a valence orbital's angular momentum, as the partial wave expansion of orbitals rapidly converges. At least one partial wave constructing a PS plane wave will represent a bound electronic state of the atom. Often, when there are two partial waves representing a subspace, the PAW dataset will feature one bound and one unbound electronic state.

The pseudo partial waves and the associated projectors are then constructed from one of many pseudization scheme designed to ensure that these bodies fulfill the requisites established in earlier in this section. The PS partial waves are solutions to the PAW Hamiltonian, which features a screened, pseudized local potential that is equal to the AE atomic potential outside of a radius r_{pp}. One option for the pseudization scheme is that of Blöchl, in which the pseudized basis functions are solutions to the non-relativistic Schrödinger equation of an isolated atom immersed in a pseudopotential defined for each AE partial wave. In other words, the projector functions are chosen first, and the PS basis functions are derived [24]. This procedure is described more in section VI B of [6]. Yet another option involves the RRKJ optimization scheme [27], which represents the pseudo wavefunction as a sum of two Bessel functions.

The pseudization scheme with which we are most concerned, for reasons that will be made known in section 4.3.2 is the so-called Vanderbilt [28] scheme implemented in *atompaw*. Via this scheme, the pseudo wavefunctions are eighth-degree polynomials inside the augmentation region,

$$\tilde{\psi}_i(r) = r^{\ell+1} \sum_{k=0}^{4} c_k r^{2k}, \tag{4}$$

from which the basis and projector functions are deduced by fitting the five coefficients c_k such that $\tilde{\phi}_i = \phi_i$ in the vicinity of r_c. The Vanderbilt scheme thus ensures that (a) $\tilde{\psi}_i(r)$ is an eigenfunction of the atomic PAW Hamiltonian; and (b) the $\tilde{\psi}_i(r)$ satisfies the generalized norm-conserving condition $Q_{ij} = 0$, where

$$Q_{ij} = \int_{r<r_{c,i}} \left[\phi_i^*(\mathbf{r})\,\phi_j(\mathbf{r}) - \tilde{\phi}_i^*(\mathbf{r})\,\tilde{\phi}_j(\mathbf{r})\right] \mathrm{d}\mathbf{r}. \tag{5}$$

And j represents a set of quantum numbers $\{t, \ell, m_\ell, n\}$ distinct from that of i. The integrand of Q_{ij} is known as the pseudized augmentation function between two partial waves i and j. For more details on this scheme, we refer the reader to [24, 28].

Following the pseudization process, the pseudo basis functions and their accompanying projectors are orthogonalized according to a chosen orthogonalization scheme, such as the Gram–Schmidt scheme or Vanderbilt's own scheme, also published in [28].

With the AE basis functions, pseudo basis functions and projector functions defined, PAW dataset generators output their values on a radial grid in the form of a pseudopotential in an increasing number of formats compatible with popular electronic structure theory codes. Predefined, open-source, and tested PAW datasets for many elements can be acquired from, for example, the *atompaw* or PseudoDojo [25, 29] websites, among others.

2.2. DFT+U+J

Inspired by the Hubbard model [30], DFT+U+J [31–35] offers treatment of self-interaction and static-correlation errors in highly localized electronic subspaces while minimizing additional computational expense [36]. There are a variety of functionals occupying the Hubbard-like corrective class, ranging from the recognizable and widely implemented Dudarev (DFT+U_{eff} = U-J) [33], Himmetoglu (DFT+U+J) [35], and Liechtenstein [37] functionals, to the ultra-modern BLOR functional [17], derived to explicitly address the flat-plane condition. The treatment provided by the additional Hubbard U and Hund's J terms is prescribed exclusively for subspaces that require numerical attention in excess of those for which the base XC functional is descriptively sufficient.

We assume that in consulting this technical report at all, the reader is already operationally familiar with DFT+U, if not DFT+U+J, as a method. Furthermore, the modifications we present here did not extend to the implementation of the Hubbard functionals themselves, only the calculation of their parameters. For this reason, rather than provide an overall review, we train our focus on the linear response determination of the Hubbard U and Hund's J in addition to the specifics of how DFT+U+J is implemented in ABINIT. (See section 3.1). For more information on the formalism and relative advantages of DFT+U+J method, we redirect the reader to references. [8, 17, 18, 31–35, 37, 38].

2.3. Linear response determination of the Hubbard parameters

In the relevant literature, the Hubbard U parameter for a valence subspace is most often determined semi-empirically. That is, the parameter space is swept and a U value chosen for its ability to get a particular DFT-derived property closer to some more concrete benchmark. Otherwise, the U and J parameters are reappropriated from similar studies, in perpetual and serial reuse. However, since these parameters are ground state properties of subspaces described by a particular XC functional (and moreover specific to code, pseudopotential, and other runtime convergence parameters), they are inherently non-transferable on the one hand, but also derivable from first-principles on the other.

Following Pickett *et al*'s [9] lead, Cococcioni and de Gironcoli picked up and developed a linear response-based protocol for calculating the Hubbard U *in situ* [10]. ABINIT researcher Donat Adams, alongside Bernard Amadon and Silke Biermann, developed an ABINIT utility, manifested in the PAW formalism, that determines the strength of the Coulombic repulsion, the Hubbard U, and other metrics via linear response. Details of this utility can be found in appendix C. The homologous linear response protocol for the Hund's J was published by Linscott *et al* in 2018 [18], taking inspiration from the earlier exploration of the same by Himmetoglu *et al* [12]. We refer the reader to these articles for the mathematical formalism and theory.

Formulated practicably, the SCF linear response procedure for calculating a scalar Hubbard U parameter is as follows.

(i) Serially apply several (preferably more than three) small perturbations $\pm\alpha$ in equal magnitude to both the up and down spin channels of the external potential of the chosen error-afflicted subspace.
(ii) Once the potential perturbation α is applied to the treated subspace, the charge occupations on the spin-up $n^{\uparrow}$ and spin-down $n^{\downarrow}$ channels of that atom and the surrounding atoms change in response. The change in occupation as a direct result of this potential perturbation is nothing more than the non-interacting response function, χ_0, which is harvested here, after the first SCF iteration but before the density and Hamiltonian are updated to begin a new iteration. Therefore, extract the up and down subspace occupations of the subspace, $n_0^{\uparrow}$ and $n_0^{\downarrow}$, after the first self-consistency iteration.
(iii) The perturbation is screened, its charge reorganized to compensate for the disturbance once again until, after the last self-consistent iteration, it reaches an equilibrium state. At the end of the SCF cycle, then, extract the up and down subspace occupations, $n^{\uparrow}$ and $n^{\downarrow}$. The derivative of this equilibrium occupation with respect to the perturbation magnitude is the interacting response function, χ.
(iv) Perform a linear (or higher-order polynomial) regression to the collected data sets and differentiate at $\alpha = \beta = 0.0$ eV to find the slopes of the response functions, $\chi_0 = \frac{\mathrm{d}(n_0^{\uparrow}+n_0^{\downarrow})}{\mathrm{d}\alpha}$ and $\chi = \frac{\mathrm{d}(n^{\uparrow}+n^{\downarrow})}{\mathrm{d}\alpha}$.
(v) Insert the response functions into the following equation to acquire U

$$\begin{aligned} \mathrm{U} &= \frac{\mathrm{d}\alpha}{\mathrm{d}\left(n_0^{\uparrow}+n_0^{\downarrow}\right)} - \frac{\mathrm{d}\alpha}{\mathrm{d}\left(n^{\uparrow}+n^{\downarrow}\right)} \\ &= \chi_0^{-1} - \chi^{-1}. \end{aligned} \tag{6}$$

The extension to polynomial regressions in step (iv) accounts for the fact that the response behavior is not always linear. Figure C1 partially demonstrates this for the Ni 3*d* orbitals in a ferromagnetically ordered

NiO system. Based on the visuals alone, one can see the data demonstrate a noticeable degree of curvature. The use of a linear regression on this response is not entirely justified. A system demonstrating exceptionally ill-behaved linear response, wherein a third-order polynomial or higher would be needed to accurately fit the data, can be seen in figure 2 of [39]. Linearity is expected in the limit of small perturbations, but this region is not always accessible if one wants to amplify the signal-to-noise ratio. If the perturbations are too large, one can expect some non-linear behavior, or even asymmetry across the zero-perturbation axis, particularly if the system has a shallow energy landscape.

Ideally, U is calculated for a cell of infinite size such that the perturbed subspace is isolated from its periodic images. Since this is unfeasible computationally, U must be converged with respect to an increasing number of atoms, ideally organized into a roughly cubic supercell to isotropically distribute the effect of the perturbation.

The Hund's coupling J parameter is calculated analogously [18, 31]. Instead of monitoring the change in total subspace occupancy as a function of the applied perturbation α, however, Hund's J monitors changes in magnetization M, or difference between the up and down spin occupancies (i.e. $\mathrm{M} = n^{\uparrow} - n^{\downarrow}$), in response to perturbations $\pm\beta$ applied in positive magnitude to the spin-up potential and in negative magnitude to the spin-down potential. Furthermore, the sign convention in the calculation of J is the opposite of that for the Hubbard U (i.e. a positive J corresponds to the curvature of the total energy with respect to fractional magnetization, minus its non-interacting analogue, demonstrating concavity). Formally,

$$\begin{aligned} \mathrm{J} &= \frac{\mathrm{d}\beta}{\mathrm{d}\left(n^{\uparrow} - n^{\downarrow}\right)} - \frac{\mathrm{d}\beta}{\mathrm{d}\left(n_0^{\uparrow} - n_0^{\downarrow}\right)} \\ &= \chi_{\mathrm{M}}^{-1} - \chi_{0_{\mathrm{M}}}^{-1}. \end{aligned} \tag{7}$$

> NOTE: Equivalence of Unscreened Response Functions for α and β Perturbations
>
> It is important to note, for verification purposes, that the unscreened response matrices χ_0 and $\chi_{0_{\mathrm{M}}}$ are equivalent. That is, for the same material subspace, we can show that $\chi_0 = \chi_{0_{\mathrm{M}}}$. The proof of this can be found in appendix A of [39].

3. Implementation of DFT+U+J in ABINIT

This and following sections are color- and font- coded for clarity. Mutable variables found in the ABINIT input file are displayed in `blue code` text. Immutable, internal ABINIT variables, functions and subroutines are displayed in `orange code` text.

3.1. Running DFT+U in ABINIT

The DFT+U formalism is built into ABINIT's PAW functionality. As of version 9, DFT + U and PAW are inseparable in ABINIT and its users have no choice but to use PAW datasets as pseudopotentials when administering a correction via the Hubbard functionals. Moreover, the Hubbard functional implementation and related utilities in ABINIT are, for the moment, restricted to the case of collinear magnetism (i.e. when the variable `nspinor = 1`).

When DFT+U and PAW are simultaneously activated, the total energy becomes

$$E_{\mathrm{DFT+U}} = E_{\mathrm{DFT}} + E_{\mathrm{ee}} - E_{\mathrm{dc}} \tag{8}$$

where E_{DFT} is the standard DFT energy functional, E_{ee} is the electron–electron interaction energy expanded in equation (1) of [8], and E_{dc} is the double-counting term, which corrects for the interaction already encompassed in E_{DFT}.

DFT+U is activated via the `usepawu` input variable, a single integer which may adopt several non-zero values, each corresponding to the treatment of the double-counting term. If `usepawu = 0`, DFT+U is unactivated. If `usepawu = 1`, the double-counting term is assessed via the Full Localized Limit formulation proposed by Anisimov *et al* [31], which takes on the following form,

$$E_{\mathrm{dc}} = \frac{U}{2} N(N-1) - \frac{J}{2} \sum_{\sigma} N^{\sigma} (N^{\sigma} - 1). \tag{9}$$

When only the Hubbard U is defined via input variable upawu, the Hund's J is assumed to be 0.0, and equation (9) inserted in equation (8) becomes the Dudarev functional (DFT+U_{eff}) [33]. When the Hund's J is set to a non-zero value via jpawu, equation (9) contributes to a Hubbard corrective protocol sometimes called the Liechtenstein [37] DFT+U+J functional. Its double-counting expression is derived from a reference system that assumes the diagonal elements of the diagonalized occupation matrix are integers. Similarly, this expression is evaluated if usepawu = 4, except it is done so without spin polarization in the exchange-correlation functional [40]. If usepawu = 2, the Around Mean Field double counting expression, found in equation (7) of [41], is evaluated. Other options for the usepawu variable exist, which are related to DMFT and GW methods. For the standard DFT protocol with Hubbard corrections, use usepawu = 1, the Full Localized Limit.

Declaration of the usepawu compels ABINIT to read three more input values: lpawu, upawu and jpawu. The variable lpawu accepts an array of integers of length ntypat (the number of types of atoms) to determine on which atomic subspaces we will apply U and J values. If lpawu is negative, no Hubbard parameters are applied. If lpawu is positive, ABINIT will apply a Hubbard U and Hund's J to the atomic subspace indexed by the angular quantum number of the subspace (e.g. lpawu = 2 applies it to *d* orbitals, lpawu = 3 to *f* orbitals). The U and J can be applied to any orbitals, including *s* orbitals. The variables upawu and jpawu, subsequently, define respectively the Hubbard U and Hund's J parameters to be applied to those subspaces. By default, upawu and jpawu are read in atomic units but can be specified in other units of energy, notably eV.

Optional variables for ABINIT's DFT+U implementation include usedmatpu and dmatpawu, which work together to allow the user to propose an initial density matrix to facilitate ABINIT in finding the DFT+U ground state.

4. Determination of the Hubbard parameters *in situ* in Abinit

There are two ways to determine the Hubbard parameters *in situ* with ABINIT: linear response (lrUJ or UJdet), or cRPA. The cRPA protocol is beyond the scope of the present article, but the interested reader may take a look at the cRPA Abinit tutorial in addition to [42, 43] to get started.

Prior to ABINIT version 9.9, only the UJdet internal and post-processing utilities existed as a means of calculating the Hubbard parameters via linear response in ABINIT. In 2022, users alerted ABINIT to some inconsistencies in its implementation. These inconsistencies are explained in appendix C. For technical reasons, however, these issues could not be easily remedied, and the decision was taken to decommission the UJdet post-processing utility and to renovate its internal functionality.

As of version 9.10, the ABINIT DFT suite is equipped with both the renovated UJdet utility in addition to a new post-processing tool, the Linear Response U(J) (lrUJ) utility, which is built upon the same core UJdet programming. Most of UJdet's data processing functionalities have been preserved throughout this renovation. However, we emphasize that the functionalities of UJdet and lrUJ serve distinct purposes and implement different levels of theory, which we discuss further in the following sections.

Although older versions of ABINIT preserve the UJdet deprecated internal functions and post-processing utility, their use is strongly disadvised for the reasons outlined in appendix C.

4.1. Clarification of available linear response utilities

The primary differences between the lrUJ and UJdet as implemented in current versions of ABINIT are outlined in table 1, where item (2) highlights the most obvious difference between the two: the number of data points used to compute a linear regression of the response functions χ and χ_0. The UJdet utility uses only two points: the unperturbed case—in which the perturbation applied is zero and the subspace occupations are those of the ground state—and one perturbed case, in which the potential perturbation is equal in magnitude to the value of input variable pawujv.

By contrast, the lrUJ utility requires, at minimum, three data points (one unperturbed case and at least two perturbations) to conduct a distinct regression analysis. With n data points, the lrUJ utility computes not only a linear regression of the response functions χ and χ_0, but all polynomial regressions up to degree $n-2$. Furthermore, the lrUJ utility conducts RMS error analysis on the fits and factors that into an approximative RMS error on the resulting Hubbard parameters.

Table 1. Comparison of preserved and renovated ABINIT linear response functionalities.

	UJdet	lrUJ
1	Embedded in ABINIT core routine	Post-processor
2	Two-point linear regression	3+ point polynomial (variable degree) regression
3	$\boldsymbol{\chi}$ and $\boldsymbol{\chi_0}$ responses treated as matrices; interatomic response monitored; matrices augmented by total system charge	χ and χ_0 responses treated as scalars
4	Supercell extrapolation scheme	RMS Error analysis
5	Atomic Sphere Approximation projector extensions/normalizations	Outputs *`LRUJ.nc` NetCDF files with details of perturbative run

NOTE: Insufficiency of two-point regression

Adequate sampling of the $N(\alpha)$ (in the case of the Hubbard U) or $M(\beta)$ (for the Hund's J) terrains is crucial. It is arguable that the two-point interpolation scheme implemented in the ABINIT `UJdet` utility, as demonstrated by the difference taken in equations (D.10) and (D.11), is insufficient in terms of sampling. Furthermore, no insight into the regression error may be achieved using a two-point regression. For these reasons, the authors recommend performing multiple perturbations.

Another crucial difference between the two utilities is Item (3) in table 1: the `UJdet` utility treats the response functions as matrices, whereas the `lrUJ` utility treats them as scalars. This means that the `UJdet` Hubbard parameters are, to some degree, informed by the Hubbard interactions on and between the other atomic subspaces of the system as well as the total charge bath.

The protocol is expanded upon in [44], wherein an extrapolation scheme aiming to accelerate the determination of the Hubbard parameters is proposed. This scheme involves (a) augmenting the response matrices (collecting the response functions while moving the site of perturbation) with the negative of their total response to enforce charge neutrality, and (b) capitalizing on the assumption that the occupancy response to the potential perturbation attenuates for atoms further away from the site of the perturbation [10].

NOTE: Using a supercell

All atoms in the unit cell of a bulk material are bound to each other by symmetry. Altering the subspace potential and/or occupations of one atom will alter the subspaces potentials of the other symmetry-related atoms of the same species, both inside the explicitly defined cell and among their mirror images.

This ripple effect is counterproductive to the perturbation cardinal to linear response; the occupancy of the perturbed subspace must be allowed to evolve separately to both its mirror images and other atoms of the same species.

To ensure that ABINIT isolates the perturbed atom from its mirror images, there's really only one option: perform linear response on the biggest supercell possible. It's not ideal, and protocols have been implemented to approximatively circumvent this requisite (such as the supercell extrapolation scheme). However, the authors here suggest performing linear response on an actual supercell of the material.

By contrast, the `lrUJ` utility provides the scalar Hubbard parameters, informed only by the change in occupancy on the perturbed subspace. Certainly, a more scientific investigation, one beyond the scope of this technical note, is needed to determine which protocol (the scalar or matrix treatment) yields faster convergence with respect to supercell size, and, more abstractly, which is better suited for addressing the self-interaction error and static correlation error within isolated subspaces. In the present work, we do not

offer any particular recommendation or endorsement either way, that being beyond the scope of the work. For now, the lrUJ post-processor does not have the infrastructure needed to perform the matrix response protocol, which would require performing polynomial regression analysis on all elements of the response matrices, a much more computationally complex, taxing, and of yet uncharted endeavor. For those wishing to undergo such an endeavor, we provide details on how to access all of its necessary components in appendix D.

For all other purposes, it can be said that lrUJ offers a simplified data processing procedure to that of UJdet, provided that the user commits to more than three LR data points (i.e. at least two separate DFT runs). By design, these data points can be run in parallel, and so the use of the lrUJ utility over the UJdet utility is strongly encouraged.

4.2. Running linear response with lrUJ

The explanation that follows is a more detailed version of the corresponding ABINIT tutorial, conducted now for the Hund's J parameter as opposed to the Hubbard U. The reasoning for including such slightly redundant information is three-fold. First, in the same way that the official Abinit tutorial describes the calculation and provides example output of the Hubbard U parameter, it is fitting to provide example output files for the same system, but for the calculation of the lesser known yet increasingly emphasized Hund's J parameter. Furthermore, we take advantage of the visualization opportunities presented by this journal publication to comprehensively label all output variables pertaining to the linear response functionalities implemented in Abinit.

Second, the tutorial is extended to include, in section 4.2.4, a proof on the data-processing modifications necessary to correctly plot the linear response data coming from Abinit, and (in appendix E) a step-by-step guide of the AbiPy functionalities available to plot the results from the lrUJ utility. Neither of these is included in the Abinit documentation.

And third, section 4.3 on the implementation of the Abinit linear response utilities relies heavily on the chronological outline and input variables best introduced in tutorial format. The current format, starting with a reiteration of the tutorial and ending with a description of the hard-coding mechanisms in Abinit, was designed to make explicit, in an organized fashion, how Abinit's input variables are connected to its output variables.

The linear response procedure can be carried out in three steps:

1. Run a ground state ABINIT calculation of your supercell to generate WFK files.
2. Run a series of perturbative ABINIT calculations to generate LRUJ.nc files.
3. Execute the lrUJ post-processing utility.

4.2.1. Ground state calculation and generation of WFK files

We need to establish a ground state system whose subspace potential we can perturb. For all intents and purposes, this should be your ordinary DFT calculation, aside from a few minor modifications to the input file.

> **NOTE: Which atom should be perturbed?**
>
> The short answer is, any atom, at least for lrUJ. If you wish to avail of the UJdet functionalities such as the supercell extrapolation scheme, the atom on which you intend to apply the linear response perturbation MUST be the first atom listed in xred. See appendix D for more details.

First, we specify as a separate species the atom whose subspace we wish to apply a potential perturbation. This will alert ABINIT that we want to allow the perturbed subspace to vary its external potential independently to its kin atoms in the cell. To this end, we increase ntypat by 1 and adjust the parameters typat, znucl, lpawu, upawu, jpawu, pseudos, and all other variables dependent on ntypat, to reflect that change. This will remain true for Step (2), as well.

Table 2. Variables `macro_uj` and `nspden` combinations available in ABINIT and their corresponding Hubbard parameter.

	Possible Combinations			
`macro_uj`	1	2	3	4
`nsppol`	1 2	2	2	2
Parameter	Hubbard U			Hund's J
Perturbation applied to:	α on both spin $\uparrow$ and spin $\downarrow$	α on spin $\uparrow$	α on spin $\uparrow$	$+\beta$ on spin $\uparrow$; $-\beta$ on spin $\downarrow$
Response monitored on:	spin $\uparrow$ + spin $\downarrow$	spin $\uparrow$	spin $\downarrow$	spin $\uparrow$ − spin $\downarrow$

NOTE: Another way to break symmetries

By specifying the perturbed atom as a separate species, ABINIT will only harvest the changes in occupation of the perturbed atom. This information is sufficient for the `lrUJ` procedure, but not for `UJdet`. To avail of the supercell extrapolation technique, you will need to set the symmetry relations, `symrel`, explicitly. The symmetry relations are specified in terms of 3×3 matrices, and they represent in some form the Wyckoff positions of the atoms. Making these explicit in the input will tell ABINIT that (a) the perturbed atom should vary independently to its kin, and (b) it should still collect occupation information for all atoms containing the subspace to be treated, not just that of the perturbed atom. The `UJdet` utility then uses these interatomic response matrix elements to inform its Hubbard parameters.

You can generate these symmetries in a separate run, wherein you specify the atom upon which the perturbation is to be applied as a different species, in the same way described in this `lrUJ` tutorial. From the output, you read the number of symmetries (`nsym`), the symmetry operations (`symrel`), and the translation vectors (`tnons`).

In what follows, we assume that the input U and J values are zero. To do this, you can either set all values in `upawu` and `jpawu` to 0.0, or you can simply deactivate DFT+U by setting `usepawu = 0`. Crucially, make sure `prtwf` is set to 1 so that the `WFK` file is printed.

Once you have all aspects of your ground state run assembled, launch ABINIT with the input file to acquire your `WFK` file.

4.2.2. Perturbative calculations and generation of `LRUJ.nc` files

Once we have our reference wavefunctions, we can start the linear response procedure. We will take advantage of ABINIT's dataset functionality to iteratively apply perturbations of varying strength to our chosen subspace. For now, we describe the input variables needed to perform one such perturbation.

Building on top of the input file used in section 4.2.1, we further activate linear response with one input parameter: `macro_uj`. This parameter's integer value, in combination with the value `nsppol` (the number of independent spin channels), determines how the local potential perturbation is applied and the subsequent changes in occupancy harvested. These options are organized in table 2. The options `macro_uj = 1` and `nsppol = 1` represent the non-spin-polarized case, where total occupations are double those of one spin channel. Importantly, note that both `UJdet` and `lrUJ` are implemented exclusively for the non-collinear case as the linear response theory governing its implementation requires further consideration [45, 46].

It is worth highlighting that the J calculated here using `macro_uj = 3` and `nsppol = 2` is *not* the Hund's J parameter. For the purposes of calculating U, we rely primarily on `macro_uj = 1` and `nsppol = 2`. This setting will apply the same potential shift to both the up and down spin channels and monitor the occupancy response on the sum of occupancies on those same spin channels.

The strength of the perturbation is determined by `pawujv`. The default units for this variable are Hartree, but other units (notably eV) may also be specified. The variable `pawujat`, a single integer, specifies the atom number (the atom coordinate index listed under `xred` or `xcart`) on which the perturbation is to be applied. Make sure this is the same atom specified as a separate species in generating the `WFK` file in Step 1.

NOTE: Reading in the right WFK file

The default settings adopted when `macro_uj` is non-zero are `tolvrs = 1d-8` and `irdwfk = 1`. The former dictates the tolerance on the potential residual (i.e. the difference between the input and output potentials pertaining to a particular SCF iteration). With the latter, ABINIT is instructed to read in the `WFK` files for a prior run, given the files are named according to a specific convention. Alternatively, we can specify the `WFK` file and its path by name. To do this, set the variable `getwfk_filepath ="</path2file/filename_WFK>"`.

The input parameter named `dmatpuopt`, of which there are four options, selects the expression with which the density matrix elements for each subspace are calculated using PAW projectors. These options are discussed in section 4.3.2, and we refer the reader to [39] for a comprehensive evaluation of the influence of this variable on the Hubbard parameters.

Lastly, to have the `UJdet` internal functions print out a verbose level of information as it completes its routine, the variable `pawprtvol` should be set to -3. To further manage the print volume, set `prtvol` as needed.

In changing only these variables, we set up only one perturbative calculation. This is sufficient to avail of the `UJdet` utility functionalities, which require only two data points as discussed above. However, in many, if not all, cases, one perturbation is inadequate to compute a good regression of the linear response data, and no error analysis can be conducted thereof.

For this reason, we will need to conduct several (at minimum two, although the more, the better) perturbative calculations. We will take advantage of ABINIT's dataset function to get our system to iteratively undergo n perturbations by setting `ndtset` to n and then specifying which perturbation strengths `pawujv1`, `pawujv2`, ... , `pawujvn` we would like to apply. Once completed, launch the run.

Once the datasets have converged, your directory will have n files with the suffix `LRUJ.nc`. These files, which are NetCDF binaries, contain all the internal information pertaining to the perturbations undergone. The `lrUJ` utility will read in a series of these files and harvest the necessary information to calculate the selected Hubbard parameter.

Developer's Note: Info in LRUJ.nc file

For those considering development on the `lrUJ` utility, the internal variables stored in the `LRUJ.nc` files are as follows: `nnat`, `natom`, `ndtpawuj`, `nspden`, `nsppol`, `usepaw`, `macro_uj`, `pawujat`, `dmatpuopt`, `diemix`, `diemixmag`, `ph0phiint`, `uj_pert`, `luocc`.

In the ABINIT output file, all information related to the two-point calculation of the scalar Hubbard parameter and all information regarding the `UJdet` functionalities (completed once for every dataset) can be found between the '`calculate U, (J)`' flags. An annotated example of the standard, scalar Hubbard parameter output, in addition to the output of the supercell extrapolation scheme from `UJdet`, comprises figure 3.

4.2.3. Execution of the `lrUJ` post-processing utility

Once the `LRUJ.nc` files are printed, execute the `lrUJ` post-processing utility with the following command

```
lruj *_LRUJ.nc > lruj.out.
```

It should take less than a second to run. If the `lrUJ` utility runs successfully, the resulting output file, `lruj.out`, should resemble that shown in figure 4. The calculation shown looks at the Hund's J parameter (`macro_uj = 4`) using results from 6 perturbations, the strengths of which are listed in the first table alongside the corresponding subspace magnetizations, both unscreened (for χ_{0_M}) and screened (for χ_M).

The last table gives the values for χ_0 (χ_{0_M}), χ (χ_M), the Hubbard U (J), and their RMS errors in units of eV, for all polynomial regressions up to degree 3 (cubic), by default. One has the option to calculate higher-order polynomials, up to degree $n-2$ for n points. This is done by appending the degree option

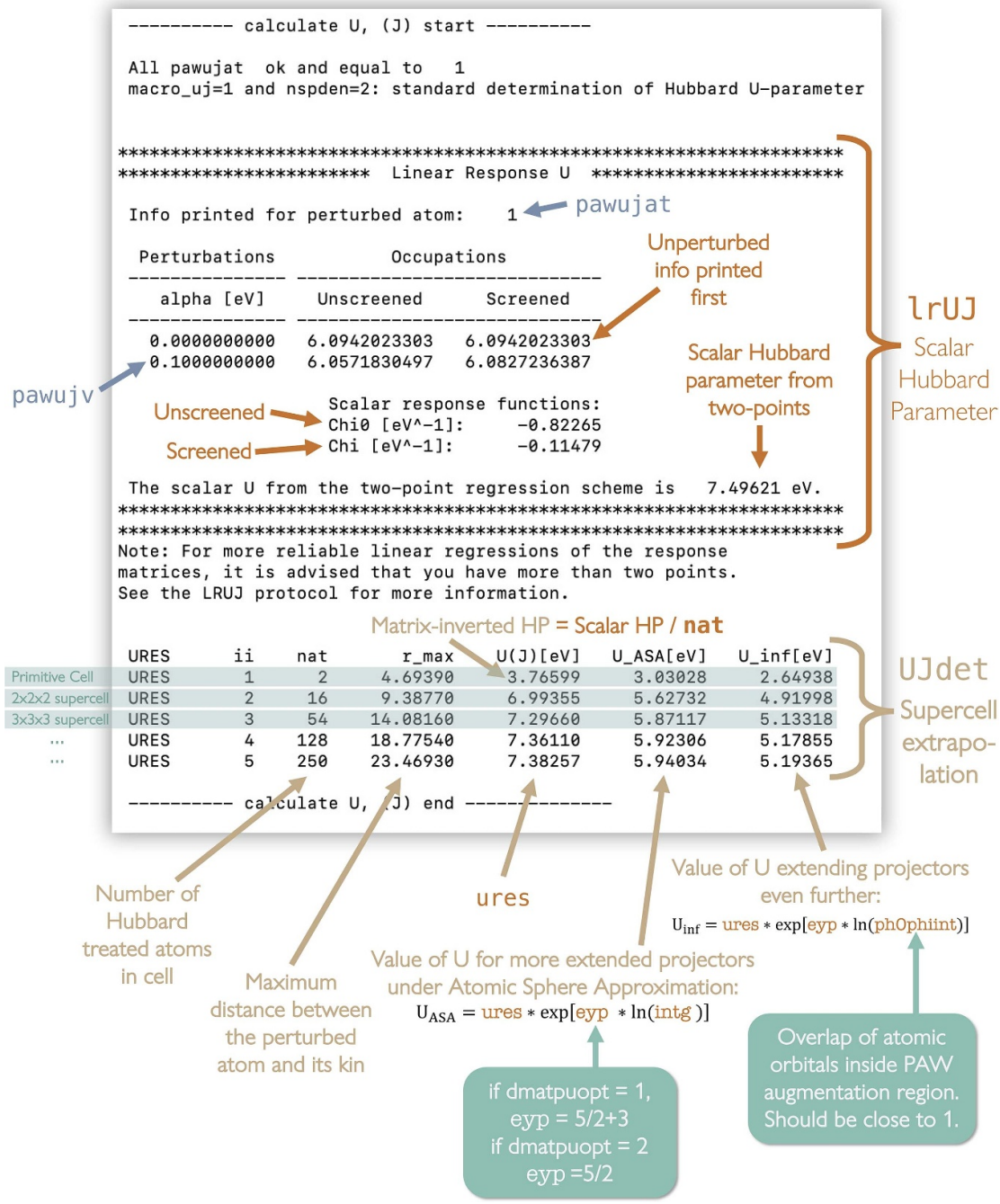

Figure 3. Generic output of the two-point Hubbard parameter determination functions, found in the ABINIT output file (typically with suffix `.abo`) for a linear response DFT run. Output for the Hubbard U is shown here. In the case of a Hund's J calculation (obtained with `macro_uj = 4`), the labels referencing `U`, `alpha`, and `Occupations` will be replaced by `J`, `beta`, and `Magnetizations`, respectively. The unscreened response `Chi0`, as printed, is treated with `diemix` (see appendix B and section 4.2.4 for clarification). Explanation of the `UJdet` supercell extrapolation output written to `.abo` file. See appendix D for more information regarding the intrinsic variables. Note that the small unit cell used here is only for illustrative purposes and will not yield converged (scalar or matrix) parameters.

`--d` <*maximum degree*> to the command line. For example, for the example calculation with 7 data points, one can bash

```
lruj *LRUJ.nc---d 5 > lruj_d5.out
```

to get parameters and errors corresponding to all polynomials of order 1 through 5, as shown in figure 4. Other command line options for the `lrUJ` utility include `--version` and `--help`.

The values in eV of the Hund's J parameter according to each regression are found in column four. To assess which one is best, you will want to use the RMS errors in column seven (more information on how the

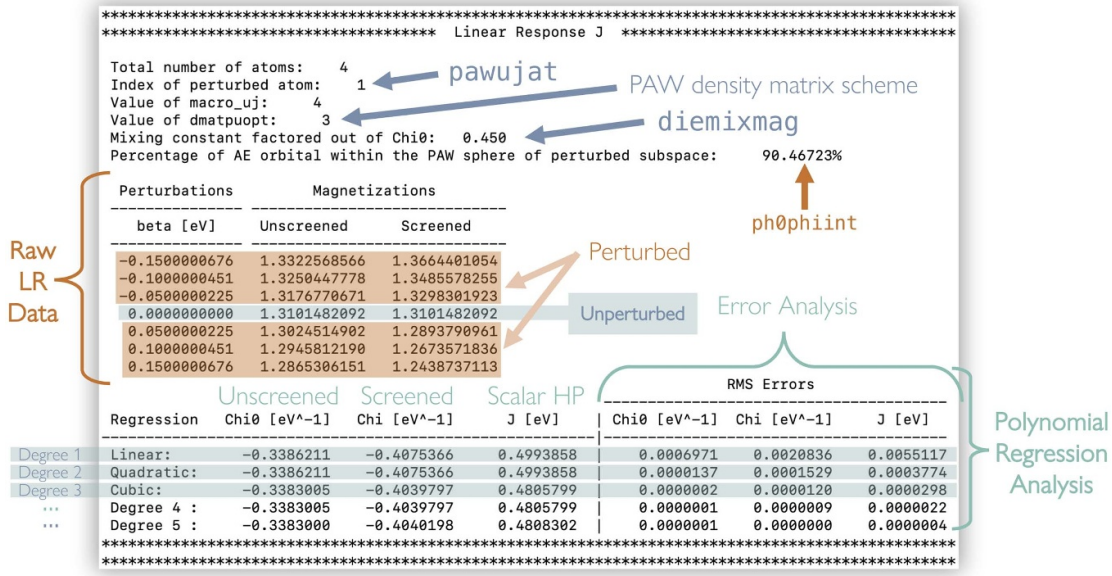

Figure 4. Output of the `lrUJ` post-processing utility for the Hund's J parameter, having requested a maximum polynomial degree of 5 via command line. Note that this output is from Abinit Version 9.10.5 and so contains erroneous HP RMS errors in column 7. See section 4.3.4 for more details.

error analysis is conducted in section 4.3.4) in addition to the visual behavior of the linear response, which can and should be plotted (see section 4.2.4), particularly if the RMS errors seem unusually large.

> **NOTE: Polynomial order in `lrUJ`**
>
> The `lrUJ` utility is programmed to only calculate polynomials up to degree $n - 2$ for n data points, a conservative measure implemented to avoid spurious overfitting effects. By default, then, if one reads in only two `LRUJ.nc` files, the maximum polynomial regression conducted will be linear; three `LRUJ.nc` files means maximum degree is quadratic, and so on, up to degree 3 (cubic). If higher polynomial order regressions are needed, and the number of data points suffices, then use the `--d` <*maximum degree*> command line option to maximum degree.

At the very end of the `lrUJ` output file, information handy for plotting, such as the coefficients of the polynomial regression formulae, is printed in YAML format.

4.2.4. Visualization of linear response data from ABINIT

Particular care must be taken when plotting the linear response data coming from ABINIT. The `lrUJ` and `UJdet` implementations both print out the raw data, meaning that the unscreened occupations (magnetizations) have not yet been scaled by the mixing constant `diemix` (`diemixmag`), as is necessary based on the conclusions of appendix B. If one were to directly plot this raw data, the plot would show a slope that does not match the χ_0 printed in the output file. To avoid this, we must perform a transformation on the data points, the form of which will be shown in the following proof. We assume a Hubbard U determination as an example proof, but the same deductions follow for the Hund's J parameter.

We will refer to the data set of unscreened occupations as N_0, to which some polynomial function of the perturbation strength α is fitted, producing a regression function $N_0(\alpha)$. The unscreened response function is defined as

$$\chi_0 = \frac{1}{\theta} \left. \frac{\mathrm{d}N_0}{\mathrm{d}\alpha} \right|_{\alpha=0} \tag{10}$$

where $\theta =$ diemix. In order to plot the unscreened response data with a function $p(\alpha)$ set such that χ_0 is shown with its θ-corrected slope at the zero-perturbation axis, we perform the following multiplicative transformation on $N_0(\alpha)$,

$$p(\alpha) = \gamma \cdot N_0(\alpha) + c \tag{11}$$

where γ and c are constants. The mandatory criterion governing the shape of $p(\alpha)$ is

$$\left.\frac{\mathrm{d}p}{\mathrm{d}\alpha}\right|_{\alpha=0} = \chi_0. \tag{12}$$

It follows from equations (10)–(12) that

$$\left.\frac{\mathrm{d}p}{\mathrm{d}\alpha}\right|_{\alpha=0} = \gamma \left.\frac{\mathrm{d}N_0}{\mathrm{d}\alpha}\right|_{\alpha=0} \tag{13}$$

$$= \chi_0 \tag{14}$$

$$\Rightarrow \gamma\,\theta\,\chi_0 = \chi_0 \tag{15}$$

$$\therefore \gamma = \frac{1}{\theta}. \tag{16}$$

In order to find the second constant c, we impose a secondary criterion on the shape of $p(\alpha)$ to ensure that $p(0) = N_0(0)$. This leads to

$$p(0) = \frac{N_0(0)}{\theta} + c = N_0(0) \tag{17}$$

$$\therefore c = \frac{N_0(0)\ (\theta - 1)}{\theta}. \tag{18}$$

Thus,

$$p(\alpha) = \frac{1}{\theta}\ \left(N_0(\alpha) + N_0(0)\ (\theta - 1)\right). \tag{19}$$

As mentioned earlier, the same conclusion can be reached assuming a β perturbation coupled with diemixmag as θ for the Hund's J parameter. Figure 5 demonstrates such a transformation for the raw Hund's J lrUJ data shown in figure 4.

One can customize the mixing constant-corrected linear response plot by importing the perturbation/occupation table into one's choice graphical utility. Plot the screened occupations as they are printed; for the unscreened occupations, plot the function in equation (19). Figure 5, for example, was generated with Mathematica.

4.3. Internal workings of the ABINIT U(J) determination procedures

In this section, we describe in detail the algorithm ABINIT undergoes to conduct linear response calculations. While sufficient as a launchpad for future developers of the program, this description is primarily intended to provide ABINIT users with a more transparent understanding of the linear response operations and their connection with the objects printed in the output files. We follow the internal variables as they undergo transformations and transfer relevant information, monitoring how the potential perturbations render occupancy responses render Hubbard parameters through the lrUJ post-processor. For ease of reference, we use the language intrinsic to ABINIT, referencing variables, functions, subroutines, modules and programs as they are in the ABINIT source code.

4.3.1. Application of perturbation

Say we launch an ABINIT run in which we seek to determine a Hubbard parameter for subspace $n\ell_U$ by perturbing the potential of atom pawujat by a strength pawujv.

After the variables are read from the input file and the ground state driver is activated, the non-zero macro_uj flag sets off the self-consistent cycle driven by pawuj_drive. Here, the strength of the perturbation pawujv is read and stored in a matrix called atvshift—a nsppol $\times$ $(2 * \ell_U + 1)$ array—according to the type of perturbation incited via the value of macro_uj. (For example, for the Hubbard U on a 3*d* subspace, atvshift will be a 2×5 matrix in which all elements are equal to +pawujv. Alternatively, for the Hund's J parameter, the second row of atvshift, representing the spin-down channel, will be set equal to −pawujv, keeping the first row equal to +pawujv.) Following this, the PAW density is initialized and the unperturbed occupancy matrix calculated, diagonalized, and printed. Here is where the

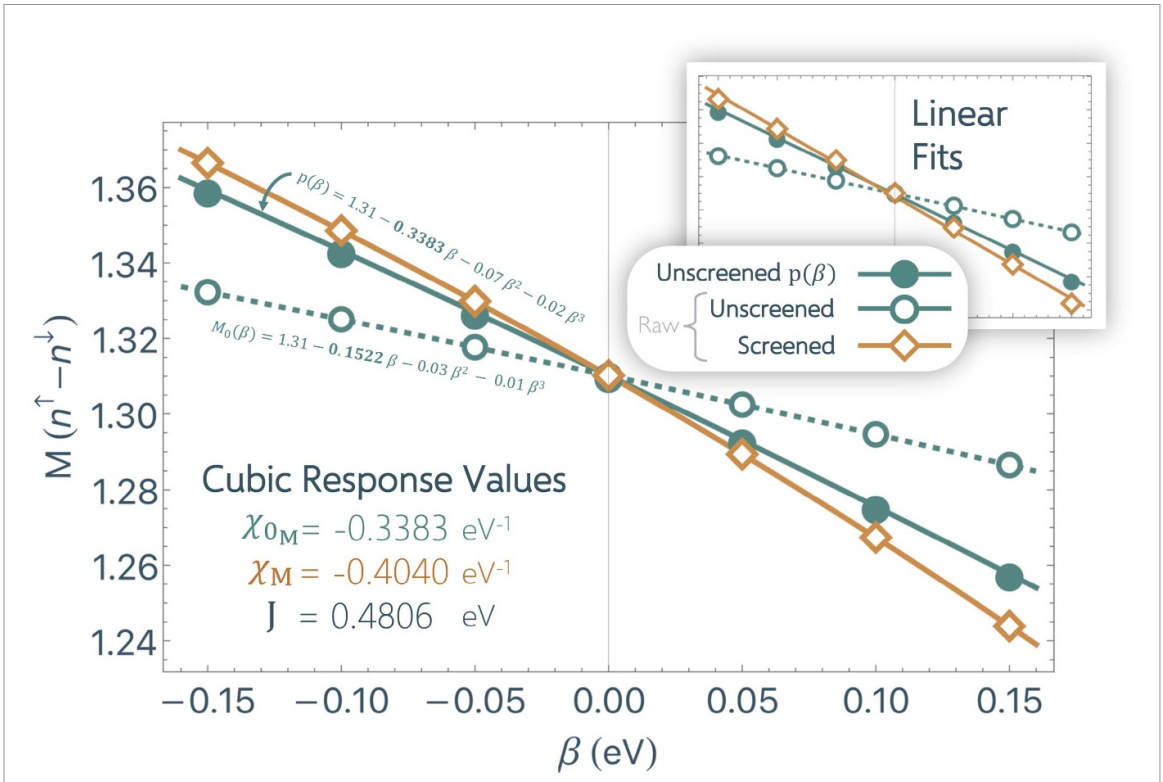

Figure 5. Linear response Hund's J plot demonstrating the necessary transformation of the raw ABINIT unscreened data to account for the mixing constant `diemixmag` (or `diemix` in the case of the Hubbard (U). The example data used here is the same as that of figure 4, which is more aptly fitted with a degree 3 (cubic) polynomial as opposed to a linear function. The cubic fits are shown in the main figure, and the linear fits for the same data are shown in the upper right inset for qualitative comparison. The $p(\beta)$ function takes the same form as equation (19), noting that we substitute M_0 for N_0 and β for α to accommodate the Hund's J as opposed to the Hubbard U.

first linear response data point, corresponding to the unperturbed ground state read in via the `WFK` file, is collected. The collection occurs in subroutine `pawuj_red`. More information on how and what information is collected is described in section 4.3.3.

It is important to note that at this point in the code, the density (and thus the potential) is mixed according to the mixing scheme outlined in appendix B. This means that the potential perturbation applied in the first iteration of the SCF cycle will be scaled by the value of `diemix`, which is equal to 0.45 by default when `macro_uj`>0. We must accordingly rescale the unscreened response function χ_0 when the time comes.

The program then calls subroutine `pawdij`. This is where the program computes the pseudopotential strengths D_{ij} of the non-local Hamiltonian operator. The potential

$$H^{\mathrm{PAW}} = -\frac{1}{2}\Delta + \tilde{\nu}_{\mathrm{eff}} + \sum_{ij} |\tilde{p}_i\rangle \left(\hat{D}_{ij} + D^1_{ij} - \tilde{D}^1_{ij}\right) \langle \tilde{p}_j|. \tag{20}$$

Here, $-\frac{1}{2}\Delta$ is the kinetic energy operator and $\tilde{\nu}_{\mathrm{eff}}$ is the effective one-electron potential written in the PAW formalism. As in section 2.1, the indices i,j refer to congruent but distinct sets of four indices: $\{t,\ell,m_\ell,n\}$. The non-local part of the Hamiltonian mirrors the PAW energy,

$$E = \hat{E} + E^1 - \tilde{E}^1 \tag{21}$$

such that $\hat{D}_{ij} = \partial\hat{E}/\partial\rho_{ij}$ is the derivative of the PAW pseudized energy with respect to the density. Similarly, $D^1_{ij} = \partial E^1/\partial\rho_{ij}$ and $\tilde{D}^1_{ij} = \partial\tilde{E}^1/\partial\rho_{ij}$, where E^1 and $\tilde{E}^1$ are, respectively, the AE and pseudized on-site energies.

Implementing this formalism in ABINIT requires categorization of these terms into those calculated outside the SCF loop and those calculated within the SCF loop. For a more detailed derivation of this, see [7]. For now, and for our purposes, it suffices to note that the pseudopotential strengths D_{ij} of the non-local Hamiltonian operator for each spin channel are calculated inside ABINIT as the following sum of terms,

$$D_{ij} = D^0_{ij} + \hat{D}_{ij} + D^{\mathrm{H}}_{ij} + D^{\mathrm{xc}}_{ij} + D^{\mathrm{U}}_{ij}, \tag{22}$$

where D_{ij}^{U} is the term to which the α perturbation is applied, and D_{ij}^{0}, D_{ij}^{H}, and D_{ij}^{xc} are, respectively, the atomic, Hartree, and exchange-correlation components, all of which are functionals of the density [7]. In the ABINIT source code, this matrix is named `dijpawu`.

Inside the `pawdij` driver, a loop over all atoms in the cell is induced, within which the subroutine `pawdiju` is summoned. Here, matrix `dijpawu` is defined as a function of the spin channel (either 1 or 2 for up or down, respectively) and of the matrix indices, which enumerate the non-core electrons by systematically combining the principle quantum number n, angular quantum number l, the magnetic quantum number m_l, the PAW projector index n, and the location in the matrix. In other words, the 2-dimensional matrix across all sub-indices pertaining to i and j is unfolded into a one-dimensional vector.

As an example of how this works, consider the case in which we apply a perturbation α to the spin-up $3d$ orbital of Ni. Say, we use a PAW dataset for Ni that has two partial waves to describe both the p and d orbitals, but only one plane wave for s orbitals, and freezes a core containing all orbitals with $n \leqslant 2$. For one spin channel on one atom, we are left with 18 combinations of quantum numbers $n > 2$, l, m_l and PAW projector n (the $4s$ orbital of Ni contributing 2 electrons $\times$ 1 partial wave $=$ 2 elements, and the $3d$ orbital contributing 8 electrons $\times$ 2 partial waves $=$ 16 elements). So, the matrix `dijpawu` for each spin channel is 18×18, yielding 324 matrix elements. However, the pseudopotential strengths are symmetric across the diagonal (i.e. element $ij =$ element ji); to save memory and time, ABINIT computes the upper right triangular matrix elements only (a total of 171 in our Ni example). The matrix elements are thus enumerated from 1 to 171 starting from the upper left and reading left to right, top to bottom.

The D_{ij}^{U} matrix elements themselves are found to be

$$\texttt{dijpawu} = \langle \phi_{n_i} | P_{mm'}^{t_i} | \phi_{n_j} \rangle * V_{m_i m_j}^{\sigma,\mathrm{U}}. \tag{23}$$

Here, $P_{mm'}^{t_i}$ is the AE projection operator acting on the radial parts of the PAW AE basis functions ϕ_n. This term is discussed in detail in section 4.3.2. Furthermore, the term $V_{m_i m_j}^{\sigma,\mathrm{U}}$ comprises a homogeneous potential $V^{\sigma,\mathrm{U}}$ across all m_i for a particular subspace and, if appropriate, the perturbation:

$$V_{m_i m_j}^{\sigma,U} = \begin{cases} V^{\sigma,\mathrm{U}} & m_i \neq m_j \\ V^{\sigma,\mathrm{U}} + \texttt{fatvshift} * \texttt{atvshift}(\sigma, m_i) & m_i = m_j. \end{cases} \tag{24}$$

The term `fatvshift` is vestigial from prior versions of the `UJdet` implementation, where a loop over values `fatvshift = 1` and `fatvshift = -1` corresponded to the positively and negatively valued perturbations of strength `pawujv`. (Now, `fatvshift = 1` only). If `pawprtvol = 3` in the input file, the entire `dijpawu` matrix will be printed in the `.log` file for all spin channels and all atoms, where one can verify that the perturbation is, indeed, being applied here. Here ends the subroutine `pawdiju`, which returns `dijpawu` for each spin channel. Back in `pawdij`, `dijpawu` is added via matrix addition to all other pseudopotential strength matrices to calculate the total D_{ij} in accordance with equation (22).

4.3.2. Calculation of orbital occupancies via `dmatpuopt`

Calculation of the occupancy matrix, or more precisely choice of the projection operator, is dictated by the input variable `dmatpuopt`, which may take on values one through four. More information on this topic can be found in [8, 39, 47]. Briefly, subspace occupancies in the PAW formalism may be calculated using the AE projection operator $P_{mm'}^{t_i}$ via

$$n_{mm'}^{t_i\sigma} = \sum_{ij} \rho_{ij}^{\sigma} \langle \phi_{n_i} | P_{mm'}^{t_i} | \phi_{n_j} \rangle, \tag{25}$$

where ϕ_n are the radial parts of the PAW AE basis functions, and the density matrix inside the PAW augmentation region is

$$\rho_{ij}^{\sigma} = \sum_{k,v} f_{kv}^{\sigma} \langle \widetilde{\Psi}_{kv}^{\sigma} | \widetilde{p}_i \rangle \langle \widetilde{p}_j | \widetilde{\Psi}_{kv}^{\sigma} \rangle. \tag{26}$$

When `dmatpuopt = 1`, occupations are projections on bound state atomic orbitals ϕ_0,

$$n_{m,m'}^{t_i,\sigma} = \sum_{ij} \rho_{ij}^{\sigma} \langle \phi_{n_i} | \phi_0 \rangle \langle \phi_0 | \phi_{n_j} \rangle. \tag{27}$$

The ABINIT documentation for this variable is clear that the `dmatpuopt = 1` option must be accompanied by a PAW dataset wherein the first atomic wavefunction of the correlated subspace (that which

is set to ϕ_0 in ABINIT) is a normalized atomic eigenfunction. To determine if a particular PAW dataset meets this criterion, one must refer to the documentation of its generator.

We are able to reasonably infer, based on the *atompaw* user guide in addition to [23], that PAW datasets generated by *atompaw* will always feature a normalized atomic eigenfunction as the first atomic wavefunction of an atomic dataset. Step 4 on Page 2 of [23] states clearly that atompaw mandates the use of 'atomic eigenfunctions related to valence electrons (bound states)' as the partial waves included in the PAW basis. Therefore, all PAW datasets generated by *atompaw*, including the JTH sets listed on PseudoDojo [29], list atomic eigenfunctions as the first atomic wavefunctions of the correlated subspace. The normalization, however, depends on the pseudo partial wave generation scheme. *atompaw* provides two options for this scheme: the Vanderbilt or the Blöchl. Based on the descriptions of these schemes in Sections 1.1 and 1.2 of [24] and an Abinit forum response in 2016, normalization of the pseudized basis functions and their corresponding projectors is guaranteed only under the Vanderbilt scheme. The JTH table of PAW datasets, available on the PseudoDojo website, therefore matches all criteria as a suitable dataset with which one may use `dmatpuopt = 1`.

Because the PAW datasets most readily available for widespread use do not necessarily fulfill these criterion, `dmatpuopt = 2` is established as the default setting. With `dmatpuopt = 2`, occupations are proportional to projections of atomic orbitals onto each other,

$$n^{t_i,\sigma}_{m,m'} = \sum_{ij} \rho^{\sigma}_{ij} \left\langle \phi_{n_i} | \phi_{n_j} \right\rangle. \tag{28}$$

Equation (28) corresponds to a projection operator of form

$$P^{t_i}_{mm'}\left(\mathbf{r},\mathbf{r}'\right) = 1_{\circ}\left(\mathbf{r}\right)\delta\left(|\mathbf{r}'-\mathbf{R_i}|-|\mathbf{r}-\mathbf{R_i}|\right)\times Y_{\ell m}\left(\hat{\mathbf{r}}\right)Y^{*}_{\ell m'}\left(\hat{\mathbf{r}}'\right),$$

where δ is the Dirac-Delta function that effectively 'counts' spatial overlap, $1_{\circ}\left(\mathbf{r}\right)$ is a step function equal to unity when $\mathbf{r}$ is inside the augmentation region and zero elsewhere, and $Y_{\ell m}$ are the spherical harmonics.

When `dmatpuopt = 3` or 4,

$$n^{t_i,\sigma}_{m,m'} = \mathcal{N}_0^{(2-\texttt{dmatpuopt})} \sum_{ij} \rho^{\sigma}_{ij} \langle \phi_{n_i} | \phi_0 \rangle \langle \phi_0 | \phi_{n_j} \rangle, \tag{29}$$

where $\mathcal{N}$ is a normalization constant representing the overlap between the bound state atomic eigenfunctions inside the augmentation sphere, delimited by cutoff radius r_c,

$$\mathcal{N} = \int_0^{r_c} \phi_0^2 \, \mathrm{d}r. \tag{30}$$

The value is computed in subroutine `pawpuxinit` and printed in the log file as `ph0phiint(1)`. When `dmatpuopt = 4`, $\mathcal{N}$ is squared in the denominator.

An evaluation of the effect of the choice of `dmatpuopt` on the magnitude of the Hubbard parameters can be found in [39].

4.3.3. Extraction of changes in occupation matrix

The outer SCF loop, declared in `pawuj_drive`, continues after the perturbation is applied; the loop symmetrizes and prints D_{ij}. If `macro_uj`>0, a subroutine labeled `pawuj_red` is called. The subroutine generates the mesh that directly associates the strength of the perturbation (translated from `atvshift` to a shorter variable called `vsh`), and the corresponding change in occupancy, called `occ`. This information, along with the atom and spin indices, are saved in a type called `dtpawuj`, which is made accessible to the internal `UJdet` and `lrUJ` functions after the SCF cycle. The occupancy is calculated as the trace of the occupancy matrix $n^{t_i,\sigma}_{m,m'}$ discussed in section 4.3.2,

$$\texttt{occ}\left(t_i,\sigma\right) = \mathrm{Tr}\left[n^{t_i,\sigma}_{m,m'}\right] * \left(3-\texttt{nspden}\right). \tag{31}$$

The factor $(3-\texttt{nspden})$ accounts for the occupation of two spin channels on a single atom if and only if we do not distinguish between the up and down spin channels. See table 2 for clarity.

The SCF iteration concludes by calling the subroutines associated with updating the density and the Hamiltonian. It then uses those updated quantities to find the updated potential, restarts a new SCF iteration, and so on and so forth until self-consistency is achieved.

The combinatory values of `macro_uj` and `nsppol` as outlined in table 2 determine how the elements of `occ` are combined and saved in a new array called `luocc`. If `nsppol = 1`, then `occ` and `luocc` are identical,

complementing the application of the perturbation to the entire atomic subspace by monitoring the response on the entire atomic subspace. In the case of the Hubbard U calculation, however, where `nsppol = 2` and `macro_uj = 1`, $\texttt{luocc}(t_i) = \texttt{occ}(t_i,\uparrow) + \texttt{occ}(t_i,\downarrow)$. In this way, the response is monitored on the total occupancy of the subspace. By contrast, when calculating the Hund's J by setting `nsppol = 2` and `macro_uj = 4`, one monitors the subspace magnetization: $\texttt{luocc}(t_i) = \texttt{occ}(t_i,\uparrow) - \texttt{occ}(t_i,\downarrow)$.

The type `dtpawuj` saves four (`vsh`, `luocc`) pairs, indexed by integers 1-4. (In the verbose `.log` file, these pairs are labeled (`vsh1`, `occ1`), (`vsh2`, `occ2`), etc; but the `occ` printed is actually the `luocc` value.) If the pair's referential index (called `iuj`) is an odd integer, that pair's occupation is harvested at the end of the first SCF cycle, immediately after the perturbation is applied to D_{ij}, but before the Hamiltonian and the density are updated to reflect that perturbation. These points will be used by `lrUJ` and `UJdet` to calculate the unscreened response matrix χ_0. Conversely, if `iuj` is an even integer, that occupation is harvested after self-consistency has been achieved. Following suit, these points will be used by the `UJdet` and `lrUJ` functions to calculate the screened response matrix χ.

> **Note: The unperturbed case**
>
> The pairs (`vsh1`, `occ1`) and (`vsh2`, `occ2`) correspond to the unperturbed ground state. Because no perturbation is applied, the unscreened and screened responses are identical. That is, $\texttt{vsh1} = \texttt{vsh2} = 0.0$ eV, and $\texttt{occ1} = \texttt{occ2}$, which are the ground state occupancies of the subspace in question.

All (`vsh`, `luocc`) pairs for all atoms and spin channels are printed out at the end of the SCF iteration in which they are determined.

4.3.4. The `lrUJ` post-processor

When the `lrUJ` post-processing utility is executed via command line, it reads in a user-specified series of NetCDF files with suffix `LRUJ.nc`. The program sorts the n files in order of perturbation strength, then reads in all necessary data related to those perturbations, including the unperturbed state (which is output in all perturbative calculations) in addition to the unscreened and screened occupations (or spin magnetization in the case of Hund's J).

> **NOTE: units**
>
> After the perturbation strengths are read from the NetCDF files, where they are reported in Ha, the `lrUJ` program converts them to eV. All calculations are then conducted in eV.

Compatibility tests are conducted on the input information, and then the linear response procedure begins.

The `lrUJ` utility was constructed with the occasional non-linearity of linear response in mind. Therefore, the program has an inbuilt subroutine that calculates the polynomial regression of any degree for any list of data points. This subroutine calculates the $n+1$ coefficients of a degree n polynomial by constructing a matrix using the fitted data points, then solving the resulting system with linear algebra. The mathematical specifics of this procedure are illustrated in figure 6 but outlined more formally on this website. Thus, as shown in figure 6, the `polynomial_regression` subroutine is dependent on the functionalities available in LAPACK.

We use polynomials only in this program because of their relative simplicity and reasonably predictable RMS error behavior. In case the user wants to fit another type of function to the data, the data points are printed out in an easily copy-pasted or parsed format for independent regression analysis.

To begin the regression procedure, the `lrUJ` program tests if the user has specified a maximum polynomial degree to calculate. If so, this degree has to be greater than or equal to the number of data points (i.e. the number of incoming `LRUJ.nc` files plus one unperturbed state) plus 2. If the user has left this information unspecified, then the maximum polynomial degree will default to cubic (degree 3) UNLESS the number of input files is equal to two or three, in which cases the maximum polynomial degree will be set to linear (degree 1) or quadratic (degree 2), respectively.

Once the max polynomial degree is set, the arrays storing the response information for each degree are allocated and the loop over polynomial degree begins. For every degree, the polynomial regression

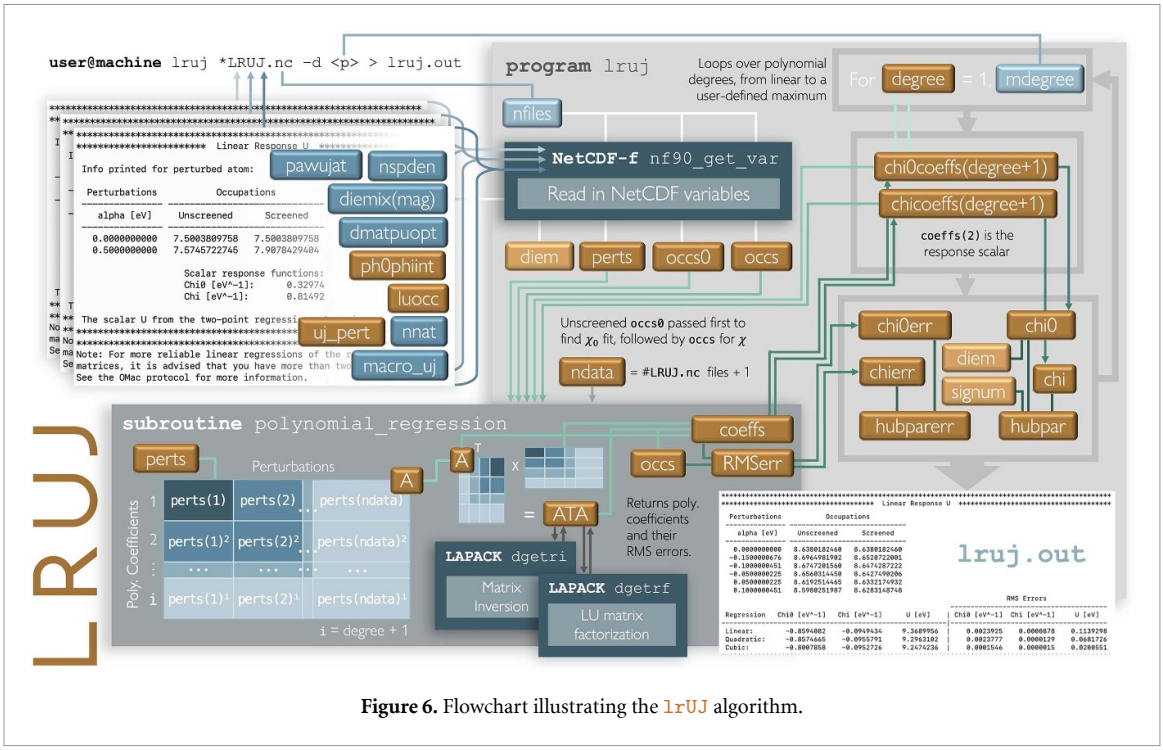

Figure 6. Flowchart illustrating the lrUJ algorithm.

subroutine is called twice: once to fit the unscreened occupancies (magnetizations) and record its unbiased RMS fit error, and the other to fit the screened occupancies (magnetizations) and record its unbiased RMS fit error. For an N-point linear regression $f(\alpha)$, where α_i is the perturbation strength corresponding to occupation (magnetization) μ_i, the equation for unbiased RMS fit error is

$$\sigma_{\text{RMS}} = \sqrt{\frac{\sum_{i=1}^{N} \left(f(\alpha_i) - \mu_i\right)^2}{N-1}}. \tag{32}$$

The RMS fit error, alongside the fit coefficients, are returned to the main program, where the utility uses that information to find χ and χ_0 as

$$\chi_0 = \frac{1}{\texttt{diemix}} \left.\frac{\mathrm{d}f_0(\alpha)}{\mathrm{d}\alpha}\right|_{\alpha=0} \tag{33}$$

$$\chi = \left.\frac{\mathrm{d}f(\alpha)}{\mathrm{d}\alpha}\right|_{\alpha=0}. \tag{34}$$

With one-dimensional polynomials as functions of α, the derivative at $\alpha = 0.0$ eV is simply the second coefficient pertaining to that polynomial function. The unscreened response χ_0 must be divided by the mixing parameter that was used in the preceding ABINIT run. This default mixing parameter is diemix and it is equal to, by default, 0.45. However, if the value of diemix is changed, or if a Hund's J calculation is conducted (at which point diemixmag instead of diemix is used for the mixing constant), χ_0 is divided by that value to get the true unscreened response.

The resulting scalar Hubbard parameter corresponding to these response functions is calculated as

$$\text{HP} = \texttt{signum} \cdot \left({\chi_0}^{-1} - \chi^{-1}\right) \tag{35}$$

where signum = 1 if calculating the Hubbard U or signum = -1 if calculating the Hund's J. The error on the Hubbard parameter, printed in column 7 of the lrUJ output file, is then

$$\sigma_{\text{HP}} = \sqrt{\left(\frac{\sigma_{\chi_0}}{\chi_0^2}\right)^2 + \left(\frac{\sigma_\chi}{\chi^2}\right)^2}. \tag{36}$$

Having executed its main function, the program concludes its operations by printing out the information in user-friendly format to the main output file (if specified in the command line; prints to terminal otherwise). An example of such an output is available in figure 4.

NOTE: Initial bug fixes

Although the lrUJ post-processor emerged with ABINIT version 9.10.1, important bug fixes for this utility were implemented in ABINIT version 9.10.5, and a further update using the correct form of equation (36) will go into effect in Abinit version 9.11. If using prior versions of lrUJ, please note that the reported RMS error for the Hubbard parameter is unreliable (i.e. these versions of lrUJ program, which were used for figures 4 and 6, erroneously do not square the response functions in the denominators of equation (36)).

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Acknowledgments

LM and DDO'R acknowledge the support of Trinity College Dublin through its Provost's PhD Project Awards and the School of Physics. Calculations were principally performed on the Boyle cluster at Trinity College Dublin, which was funded through grants from the European Research Council and Science Foundation Ireland and is maintained by Trinity Research IT.

Appendix A. The PAW pseudopotential

PAW dataset generators construct a local ionic pseudopotential using the chosen PAW basis functions via a method that is closely analogous to Vanderbilt's ultrasoft pseudopotential generation, described in [28]. We briefly describe the construction of a PAW pseudopotential here from the basis functions defined earlier [22, 26].

The first step is to construct a screened local atomic pseudopotential $\tilde{\nu}_{\rm loc}$ for an atom in some reference configuration (usually the isolated atom), which is to be equivalent to the AE atomic potential $\nu_{\rm loc}$ beyond some radius $r_{\rm pp}$. (The cutoff radius $r_{\rm c}$ is not necessarily the same as $r_{\rm pp}$.) As an example of such a construction, Vanderbilt proposed the use of a zero-order spherical Bessel function (equation (58) in [28]).

The pseudopotential $\tilde{\nu}_{\rm loc}$ comprises contributions from both the core and valence densities. To expand the contribution of the latter, we take the pseudo partial waves-which have been defined such that outside of the cutoff radius $r_{\rm c}$ they are equivalent to the AE basis functions-and use them to generate a different set of functions χ_i

$$|\chi_i\rangle = (\epsilon_i - T - \tilde{\nu}_{\rm loc}(r))\,|\tilde{\phi}_i\rangle \tag{A.1}$$

where ϵ_i are the orbital energy eigenvalues associated with the AE basis functions and T is the kinetic energy operator. The projectors $\tilde{p}_i$ are then represented in terms of χ_i,

$$|\tilde{p}_i\rangle = \sum_j P_{ij}^{-1}|\chi_i\rangle, \tag{A.2}$$

whereupon, owing to the orthogonality condition $\langle\tilde{p}_i|\tilde{\phi}_i\rangle = \delta_{ij}$,

$$P_{ij} = \langle\tilde{\phi}_i|\chi_i\rangle. \tag{A.3}$$

We take the operator P_{ij} and employ it in the definition of a non-local potential $D_{ij} = P_{ij} + \epsilon_j Q_{ij}$, where Q_{ij} is as defined in equation (5). In a final step, we effectively unscreen the potentials D_{ij} and $\tilde{\nu}_{\rm loc}$ to deduce a valence potential

$$D_{ij}^0 = P_{ij} + \epsilon_j Q_{ij} - \int \tilde{\nu}_{\rm loc}(r)\left(\phi_i^*(\mathbf{r})\,\phi_j(\mathbf{r}) - \tilde{\phi}_i^*(\mathbf{r})\,\tilde{\phi}_j(\mathbf{r})\right)\mathrm{d}\mathbf{r} \tag{A.4}$$

and a local ionic pseudopotential

$$\tilde{\nu}_{\rm H}[\tilde{n}_{Zc}] = \tilde{\nu}_{\rm loc} - \nu_{\rm H}\left[\tilde{n}^1 + \hat{n}\right] - \nu_{\rm xc}\left[\tilde{n}^1 + \hat{n} + \tilde{n}_c\right], \tag{A.5}$$

where ν_{H} and ν_{xc} are the Hartree and exchange-correlation potentials, respectively. Finally, the PAW pseudopotential representing the atom in its entirety is

$$V_{\mathrm{PP}} = \tilde{\nu}_{\mathrm{H}}\left[\tilde{n}_{Zc}\right] + \sum_{ij} D_{ij}^{0} |\widetilde{p}_i\rangle \langle \widetilde{p}_j|. \tag{A.6}$$

Appendix B. Mixing schemes

ABINIT provides two mixing schemes: one that mixes the potential and one that mixes the density. Both are available in the PAW implementation, and both prove to perform equally well in efficiently achieving self-consistency (density mixing slightly outperforms potential mixing). However, density mixing is preferable when using PAW because of the degrees of freedom added to the electronic density via the pseudovalence density and the compensation charge density, the latter of which is directly related to the PAW occupation matrix. From [3]:

'When potential mixing is activated, all parts of the total energy are computed at the same time; the total energy is thus variational with respect to the self-consistent cycle step. When density mixing is activated, parts of total energy are computed at various stages of the cycle which results in a behavior of total energy that is not variational.'

The new density is computed, mixed with previous densities, then used to update the energy total alongside other contributions that are not all updated at the same place in the SCF cycle. Inside PAW, the on-site density matrix ρ_{ij}, defined explicitly in [7], is updated at the same level as the electronic density $\tilde{n} + \hat{n}$, and is then mixed at that level. Therefore,the D_{ij} which is a potential term in PAW, is left unmixed by default [7].

Density mixing is the default for ABINIT under the PAW protocol (`iscf= 17`), specifically via the Pulay mixing algorithm [48], which was developed in 1980 as an efficient method of accelerating convergence of iterative sequences. Pulay mixing is used to mix ρ_j^{OUT} and the residual density $\rho_j^{\mathrm{OUT}} - \rho j^{\mathrm{IN}}$ in the following iterative update of the density,

$$\rho_{i+1}^{\mathrm{IN}} = \rho_i + \underset{j \leqslant i}{\mathrm{mix}} \left[\rho_j^{\mathrm{OUT}}, P(k) * \left(\rho_j^{\mathrm{OUT}} - \rho j^{\mathrm{IN}}\right)\right], \tag{B.7}$$

where $P(k)$ is a preconditioning factor corresponding to wavevector k, applied to the residual density $\rho_j^{\mathrm{OUT}} - \rho j^{\mathrm{IN}}$ of the prior iterations. This preconditioning factor is defined, by default, as the inverse of the model dielectric matrix

$$P(k) = \epsilon^{-1}(k) = \mathtt{diemix} * \frac{\mathtt{diemac}^{-1} + (\mathtt{dielng} * k)^2}{1 + (\mathtt{dielng} * k)^2} \tag{B.8}$$

where `diemix` is the dielectric mixing constant, set to 0.7 by default for PAW calculations and 0.45 for linear response calculations; `diemac` is the model dielectric macroscopic mixing constant, which is typically very large for metals and around 10 for insulators. (`dielng` is a fine-tuning parameter). The variable `iprcell` can select the function used to define the preconditioning factor.

The dielectric mixing constant `diemix`, and its magnetic analog `diemixmag`, is applied to the first SCF density after the α (β) perturbation is applied but before the on-site orbital occupations (magnetizations) are calculated. This means that `diemix` (`diemixmag`) inadvertently scales the potential perturbation of the unscreened response matrix χ_0 ($\chi_{0_{\mathrm{M}}}$) in the determination of the Hubbard U (Hund's J) parameter. To counteract this, therefore, we must use the value of `diemix` (`diemixmag`) to unscale χ_0 ($\chi_{0_{\mathrm{M}}}$) in the Hubbard U (Hund's J) data-processing step. Based on a series of tests, we can say conclusively that changing `diemixmag` does not influence the Hubbard U parameter, and analogously, changing `diemix` does not influence the Hund's J.

Appendix C. `UJdet` prior to ABINIT version 9.9

In the late 2000s, ABINIT developed a utility—the U(J) Determination (`UJdet`) protocol—designed to determine the Hubbard parameters based on Cococcioni and de Gironcoli's linear response method outlined in section 2.3. In ABINIT versions 5 to 9.9, when activated, the protocol would serially introduce two perturbations—one of strength `pawujv` and the other of strength -1.0 × `pawujv`—to the subspace-uniform potential and harvest the resulting occupancy responses at the beginning and end of the two ensuing self-consistent cycles. The utility then had two points with which it could calculate the screened and

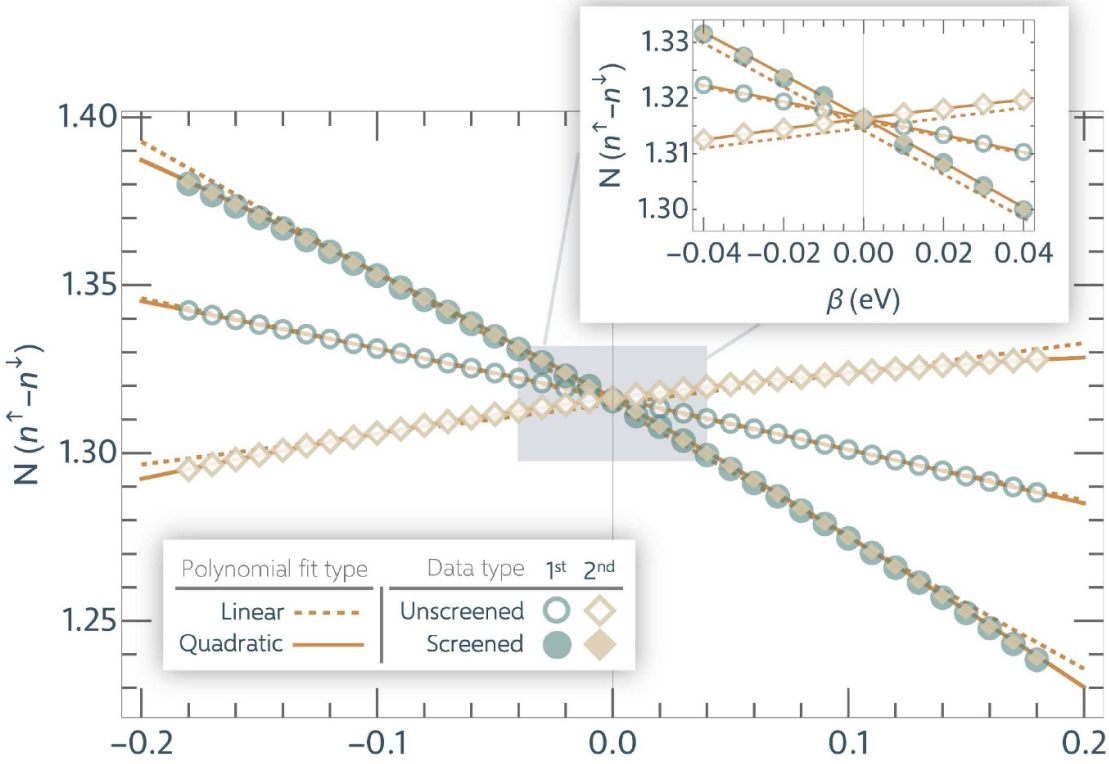

Figure C1. Example of the linear response procedure conducted using the now outdated rendition of `UJdet` in ABINIT version 9.6.2, to find Hund's J for Ni *3d* orbitals on ferromagnetically ordered NiO [49]. Data points are raw perturbation-occupancy pairs output by ABINIT categorized according to their place in the calculation queue; that is, the SCF calculation involving the first (1^{st}) β perturbation is represented by green circles, and the SCF calculation of the second (2^{nd}) β perturbation is represented by tan diamonds. Open markers indicate unscreened response and filled markers indicate screened response. The screened response χ relaxes to the same value regardless of its status as first or second calculation. The unscreened responses, however, are different depending on which perturbation is applied first. This contributes to J parameters differing, in this case, by several eV. Orange lines are polynomial regressions (dashed for first-order polynomial, solid for second-order polynomial) of each set of data.

unscreened response matrices, χ and χ_0, defined as derivatives of occupation with respect to perturbation strength.

It is important to note that the positive value of `pawujv` was applied first, followed by its negative image. One would expect, then, that in performing two separate runs with the positive and negative values of `pawujv` should yield the same linear response.

This was not the case in ABINIT versions prior to 9.6.2. To demonstrate the error, we produced figure C1 by performing β perturbations on a particular system (ferromagnetic NiO) and monitoring the response from both the first and second `UJdet` calculations, respectively. That is, we categorized perturbation-occupancy pairs according to their place in the queue in this double perturbation cycle. The screened response χ relaxes to a reasonably similar value regardless of its status as first or second calculation. The unscreened responses, however, are different depending on which perturbation is applied first. This discrepancy contributes to Hund's J parameters differing, in this case, by several eV. The same phenomenon was observed for the α perturbations contributing to the Hubbard U. The fact that the unscreened occupations differed depending on their place in the queue indicated that the perturbations were not being applied to the same initial ground state. Internal variables were not undergoing proper initialization, and so the second perturbation was inheriting information from the converged state of the preceding perturbative cycle.

There are a few methods available to test which χ_0 is the correct one. We know that, for the same system, where $\alpha = \beta$,

$$\frac{\mathrm{d}N_0}{\mathrm{d}\alpha} = \frac{\mathrm{d}M_0}{\mathrm{d}\beta}. \tag{C.9}$$

As was discussed in section 2.3. This requisite is fulfilled only for the first applied perturbation (i.e. only the value supplied in `pawujv`, not its negative counterpart). Furthermore, we know that when we apply a potential perturbation to the spin-up channel only, the unscreened occupancy on the spin-down channel should not change. Applying a perturbation exclusively to the spin-up channel can be achieved by setting

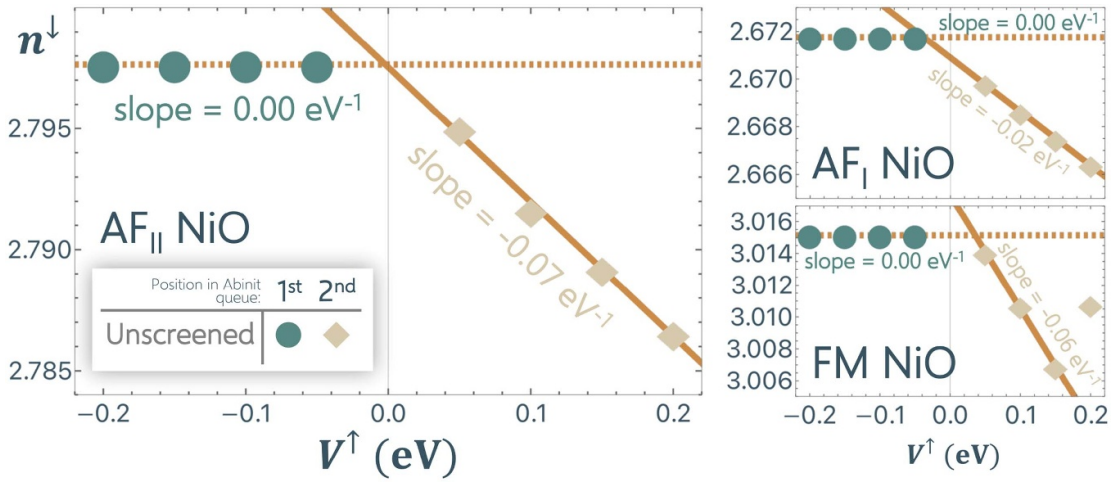

Figure C2. Comparison of unscreened spin-down occupation changes in response to a perturbation α applied exclusively to the spin-up 3*d* channel of a Ni atom on NiO of various magnetic orderings, where (`vsh`, `occ`) pairs are categorized according to their queue (first perturbation applied or second) in the full DFT cycle. We expect the slopes of these lines to be zero, but only the first ABINIT calculation fulfills this requirement.

`macro_uj = 2` (the `macro_uj` input parameter is explained in section 4.2.2). We ran perturbations under this setting, applying perturbations to the spin-up channel only of a Ni atom and monitoring the change in unscreened occupancy on the down spin channel of the same Ni atom. The results of this inquiry, displayed in figure C2, show that the unscreened occupancy on the spin-down channel remains constant only for the first applied perturbation, thereby corroborating the earlier conclusion that the second applied perturbation in the ABINIT cycle is unreliable. The silver lining for users of `UJdet` prior to ABINIT 9.10.1 is that the first perturbation-occupancy data point is still salvageable.

Appendix D. The Hubbard U parameter determination via `UJdet`

All subroutines constructing ABINIT's `UJdet` utility—an abbreviation of 'Hubbard U and J determination'—are housed inside module `65_paw/m_paw_uj.F90`. When the SCF cycle is complete, the same subroutine that launched the SCF cycle and allocated default variables for the `dtpawuj` type, `pawuj_drive`, calls the subroutine `pawuj_det`.

Once called, this subroutine calculates and prints the scalar Hubbard parameter for exclusively the perturbed atom using the two data points it has (i.e. the unperturbed occupancies and those of the one perturbation applied during its run). It is here that the program creates the NetCDF file with suffix `LRUJ.nc` for this perturbative run, filling it with all information that the `lrUJ` post-processor will need to determine the choice Hubbard parameter in tandem with other perturbations. Before ABINIT wraps up its DFT run, however, the `UJdet` algorithm switches to the matrix prescription for calculating the Hubbard parameters. In doing so, it proceeds to calculate all elements of the response matrices using the aforementioned (`vsh`, `luocc`) pairs in the following manner:

$$\chi 0_{t_i t_j} = \frac{\texttt{luocc3}_{t_i} - \texttt{luocc1}_{t_i}}{\texttt{diemix}\left(\texttt{vsh3}_{t_j} - \texttt{vsh1}_{t_j}\right)} \tag{D.10}$$

$$\chi_{t_i t_j} = \frac{\texttt{luocc4}_{t_i} - \texttt{luocc2}_{t_i}}{\texttt{vsh4}_{t_j} - \texttt{vsh2}_{t_j}} \tag{D.11}$$

where t_i, t_j are all atoms of the same species as the perturbed atom.

These matrices will typically feature only one element: the diagonal element for which the perturbed atom t_j is equal to the responding atom t_i. To extend this matrix and avail of the URES supercell facilities described below, one must tell ABINIT to monitor the occupancy responses on all atoms t_i in the cell (see note on `symrel` in section 4.2.1 for information on how to do this).

If going with the `symrel` procedure, it is furthermore recommended to set the input variable `pawprtvol = 3` in order to have the occupancies of all Hubbard atoms printed out in the `.log` file. There, the (`vsh`, `luocc`) pairs will be listed for every atom t_i after every SCF iteration. (The (`vsh4`, `luocc4`) pair is thus listed several times as the calculation converges; the relevant listing is, expectantly, the last one).

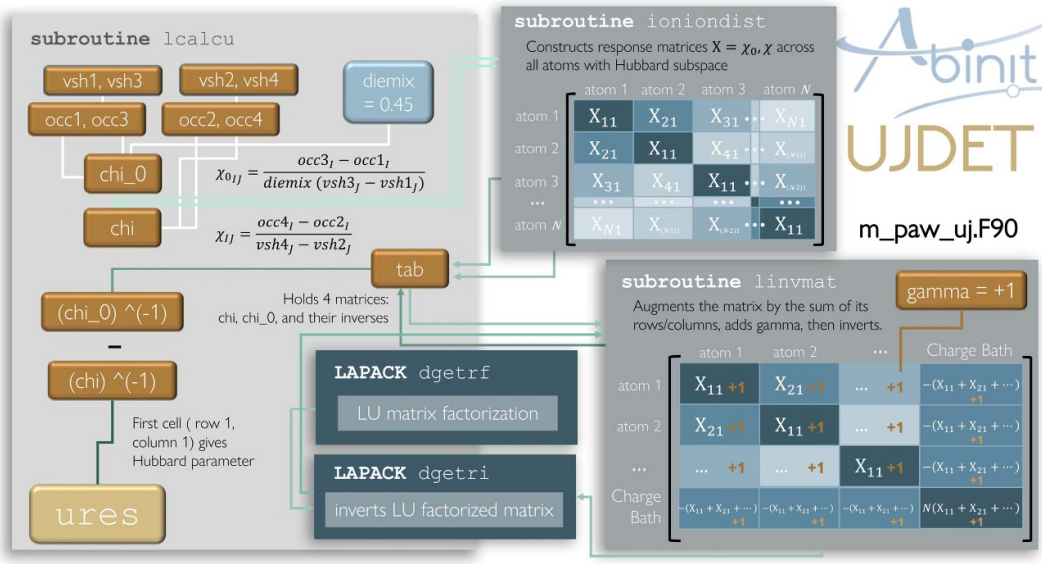

Figure D3. Flowchart illustrating manipulation of (`vsh`, `occ`) pairs to determine the final Hubbard parameter, `ures`, for a transition metal (TM) Hubbard subspace.

Even in this case, since perturbations are always be applied to atom $t_j = 1$, the only elements calculated via equations (D.10) and (D.11) are those pertaining to $(t_i, t_j) = \{\,(1,1),(2,1),(3,1),\ldots\,\}$. So when these matrices are subsequently funneled, via the mother Hubbard U subroutine `lcalcu`, to subroutine `ioniondist`, the lower triangular elements are completed via symmetry (i.e. $X_{JI} = X_{IJ}$, where $X = \chi_0, \chi$). The diagonal elements of the response matrices will all be set equal to X_{11}, as illustrated in figure D3. Once completed, the matrices are returned to subroutine `lcalcu` and saved into variable `tab`, which holds four matrices: χ_0, χ, and their matrix inverses. It follows that `tab` is shuffled over to subroutine `linvmat`, which calculates the inverses of not the response matrices themselves, but treated matrices designed to speed up the convergence of the Hubbard parameters with respect to supercell size.

These treatments are mentioned in the 'Further Considerations' section of [10], where it is posited that the perturbation on the Hubbard subspace would benefit from enhanced locality if charge neutrality in the response matrices was enforced, thereby isolating the perturbed atom from its periodic images, as one hopes to do using supercells. Following this understanding, ABINIT augments the response matrices with the negative of the sum of each row and each column, as illustrated in figure D3.

This augmented matrix is, by definition, singular and thus non-invertible. To render the matrix invertible, an all-ones matrix is added to it, breaking its singularity. Note that this matrix is no longer equivalent to the input response matrices. However, as demonstrated in appendix A.4 of [50], the difference of the inverses of two non-invertible matrices-which is not possible mathematically-may be calculated indirectly by adding the same non-zero constant to each matrix element. This renders these matrices invertible, and the added constant is canceled when taking the difference of the two matrices.

Once prepped, the response matrices are funneled into the LAPACK routines `dgetrf` and `dgetri`, which respectively, LU factorize the matrices then invert them. These inverted matrices are then saved into the last two positions of `tab` and returned back to subroutine `lcalcu`. At last, the inverted response matrices are subtracted, then scaled by a factor called `signum` (`= 1.0` for the Hubbard U and `= -1.0` for the Hund's J). The first element of that object then (row 1, column 1), in eV, is found to be the long-awaited Hubbard parameter.

The `UJdet` utility does not stop here, though. The 'Further Considerations' section of [10] considers a hypothetical extrapolation scheme speculatively designed to converge much more quickly the Hubbard parameter with respect to supercell size. The number of Hubbard subspaces in a supercell corresponds

linearly with the response matrix dimension. But intuition suggests that the occupancy effect of the perturbed subspace will attenuate with distance; that is, the matrix elements of the nearest neighbor atoms to that perturbed will feature most prominently in the determination of the Hubbard parameter, and those least neighborly to the perturbed atom will undergo small, even negligible, changes in occupancy, rendering their influence negligible. ABINIT developers took these further considerations to heart by incorporating an extrapolation scheme, wherein the response matrix elements of the primitive unit cell are used to fill out the response matrix elements of a supercell. Concisely, in ABINIT's `UJdet` utility, the off-diagonal elements of the primitive cell response matrices are multiplied by the number of next-nearest neighbor (NNN) Hubbard atoms in the primitive cell and divided by the number of Hubbard atoms in NNN shell in the supercell. These supercell response matrices are then inverted following the same procedure as above to approximate the Hubbard parameters for subspaces in supercells of increasing size.

Appendix E. What AbiPy can do with a `lrUJ` output file

This mixing constant-corrected linear response plot can be easily generated through the AbiPy python package; its version 0.9.7 is able to read in results from one's chosen `lruj.out` output file and visualize its results. To avail of this functionality, educe a python script in the same directory as that containing your `lruj.out` output. Begin the python script by importing the `LrujResults` function from the AbiPy package:

```
#/usr/bin/env python
from abipy.electrons.lruj import LrujResults
```

Import the `lruj.out` file using the following line.

```
lr = LrujResults.from_file("path_to_file/lruj.out")
```

The plot function may then be summoned with

```
lr.plot(ax, degrees, inset, insetdegree, insetlocale, ptcolor0,
  ptcolor, gradcolor1, gradcolor2, ptitle, fontsize)
```

where the arguments listed in blue are keywords to tailor particular characteristics of the ensuing plot. These options are described in more detail in table E1.

Other AbiPy tools to better accommodate the linear response process in ABINIT are currently in the works. For example, under development is a suite of functions that aim to facilitate visualization of the convergence of the Hubbard parameters with respect to supercell size. Keep an eye on forthcoming releases of AbiPy for such developments.

Table E1. Optional arguments and their possible values available in the `plot` function implemented in the `LrujResults` function of the AbiPy package.

`LrujResults plot` utility optional arguments			
Argument	Default	Other options	Description
`ax`	`None`	`ax`	Optional axes argument. If `None`, a new plot is generated. If `ax`, the figure without the axes is returned. Useful for generation of a grid of plots.
`degrees`	`"all"`	List of integers i such that $0 < i <$ `maximum degree`	Degrees of polynomial regressions to be included in the plot, provided as a Python list of integers (e.g. $[1, 2, 3]$). The maximum integer allowed is `maximum degree`, which is read in via the `lruj.out` output file
`inset`	`True`	`False`	Option to print inset with response information (i.e. values of χ_0, χ, the Hubbard parameter and their respective errors in units of eV). If `True`, information is printed for the linear regression case in the lower left corner of the plot (by default; see `insetdegree` and `insetlocale` options to tailor). If `False`, no Inset is included.
`insetdegree`	1	Any integer $\in$ `degrees`	Polynomial degree of printed response information appearing in inset.
`insetlocale`	`"lower left"`	`"upper right"`, `"center left"`, `"lower center"`, `"center"`, corresponding integers 0-10, etc.	Position of inset containing response information in standard format for `matplotlib` legend locations. See `matplotlib` documentation for all options.
`ptcolor0`	`"k"`	`"r"`, `"blue"`, `"FF6E42"`, `(0.1,0.9,0.54)`, `"0.75"`, etc.	Color of unscreened response data point markers in any standard `matplotlib` color format (see `matplotlib` documentation for all formatting options. Default color is black. Markers themselves are open circles ○ (immutably so, for now).
`ptcolor`	`"k"`	`"r"`, `"blue"`, `"FF6E42"`, `(0.1,0.9,0.54)`, `"0.75"`, etc.	Color of screened response data point markers in any standard `matplotlib` color format (see `matplotlib` documentation for all formatting options. Default color is black. Markers themselves are filled circles ● (immutably so, for now).
`gradcolor1`	`"#3575D5"`	Hexadecimal (HEX) code for any color	Line color of the lowest polynomial degree to be plotted (i.e. the smallest integer input via `degrees` argument). This color, in addition to that of `gradcolor2` will inform the line colors of the intermediate polynomial degrees in a linear gradient fashion. Must be entered as a HEX code (six characters preceded by a '#'). The default color is dark blue.
`gradcolor2`	`"#FDAE7B"`	Hexadecimal (HEX) code for any color	Line color of the highest polynomial degree to be plotted (i.e. the largest integer input via `degrees` argument). Must be entered as a Hex code (six characters preceded by a '#'). The default color is salmon pink.
`ptitle`	`"Linear Response for atom` *<pawujat>*`"`	Any string	Title of plot. Incorporates input value of `pawujat` by default. For no title, put `""`.
`fontsize`	12	Any integer > 0	Font size in point (pt) units of the plot legend.

ORCID iDs

Lórien MacEnulty https://orcid.org/0000-0001-8261-5524
Matteo Giantomassi https://orcid.org/0000-0002-7007-9813
Bernard Amadon https://orcid.org/0000-0002-9014-1725
Gian-Marco Rignanese https://orcid.org/0000-0002-1422-1205
David D O'Regan https://orcid.org/0000-0002-7802-0322

References

[1] Gonze X *et al* 2020 *Comput. Phys. Commun.* **248** 107042
[2] Romero A H *et al* 2020 *J. Chem. Phys.* **152** 124102
[3] Gonze X *et al* 2009 *Comput. Phys. Commun.* **180** 2582–615
[4] Gonze X, Almbladh C-O, Cucca A, Caliste D, Freysoldt C, Marques M, Olevano V, Pouillon Y and Verstraete M 2008 *Comput. Mater. Sci.* **43** 1056–65
[5] Gonze X *et al* 2002 *Comput. Mater. Sci.* **25** 478–92
[6] Blöchl P E 1994 *Phys. Rev.* B **50** 17953–79
[7] Torrent M, Jollet F, Bottin F, Zérah G and Gonze X 2008 *Comput. Mater. Sci.* **42** 337–51
[8] Amadon B, Jollet F and Torrent M 2008 *Phys. Rev.* B **77** 155104
[9] Pickett W E, Erwin S C and Ethridge E C 1998 *Phys. Rev.* B **58** 1201–9
[10] Cococcioni M, de Gironcoli S and de Gironcoli S 2005 *Phys. Rev.* B **71** 035105
[11] In March 2021, one of the authors posted a video lecture explaining the PAW formalism and its implementation in Abinit on YouTube, linked here for convenience. In the almost three years since the video was posted, it has accumulated just overunder 526000 views. (available at: https://www.youtube.com/watch?v=5WEdd78GDFw)
[12] Himmetoglu B, Wentzcovitch R M and Cococcioni M 2011 *Phys. Rev.* B **84** 115108
[13] Ryee S and Han M J 2018 *Sci. Rep.* **8** 9559
[14] Orhan O K and O'Regan D D 2020 *Phys. Rev.* B **101** 245137
[15] Albavera-Mata A, Trickey S B and Hennig R G 2022 *J. Phys. Chem. Lett.* **13** 12049–54
[16] Lambert D S and O'Regan D D 2023 *Phys. Rev. Res.* **5** 013160
[17] Burgess A C, Linscott E and O'Regan D D 2023 *Phys. Rev.* B **107** L121115
[18] Linscott E B, Cole D J, Payne M C and O'Regan D D 2018 *Phys. Rev.* B **98** 235157
[19] Moynihan G 2018 A self-contained ground-state approach for the correction of self-interaction error in approximate density-functional theory *Thesis* Physics Trinity College Dublin School of Physics, Trinity College Dublin
[20] Giannozzi P *et al* 2009 *J. Phys.: Condens. Matter* **21** 395502
[21] Giannozzi P *et al* 2017 *J. Phys.: Condens. Matter* **29** 465901
[22] Kresse G and Joubert D 1999 *Phys. Rev.* B **59** 1758–75
[23] Holzwarth N, Tackett A and Matthews G 2001 *Comput. Phys. Commun.* **135** 329–47
[24] Holzwarth N A W 2008 Notes for revised form of atompaw code (available at: http://users.wfu.edu/natalie/papers/pwpaw/notes/atompaw/atompawEqns.pdf)
[25] Jollet F, Torrent M and Holzwarth N 2014 *Comput. Phys. Commun.* **185** 1246–54
[26] Hine N D M 2017 *J. Phys.: Condens. Matter* **29** 024001
[27] Rappe A M, Rabe K M, Kaxiras E and Joannopoulos J D 1990 *Phys. Rev.* B **41** 1227–30
[28] Vanderbilt D 1990 *Phys. Rev.* B **41** 7892–5
[29] van Setten M J, Giantomassi M, Bousquet E, Verstraete M J, Hamann D R, Gonze X and Rignanese G-M 2018 *Comput. Phys. Commun.* **226** 39–54
[30] Hubbard J and Flowers B H 1963 *Proc. R. Soc.* A **276** 238–57
[31] Anisimov V I, Zaanen J and Andersen O K 1991 *Phys. Rev.* B **44** 943–54
[32] Anisimov V I and Gunnarsson O 1991 *Phys. Rev.* B **43** 7570–4
[33] Dudarev S L, Botton G A, Savrasov S Y, Humphreys C J and Sutton A P 1998 *Phys. Rev.* B **57** 1505–9
[34] Anisimov V I, Solovyev I V, Korotin M A, Czyżyk M T and Sawatzky G A 1993 *Phys. Rev.* B **48** 16929–34
[35] Himmetoglu B, Floris A, de Gironcoli S, Cococcioni M and de Gironcoli S 2014 *Int. J. Quantum Chem.* **114** 14–49
[36] O'Regan D D, Hine N D M, Payne M C and Mostofi A A 2012 *Phys. Rev.* B **85** 085107
[37] Liechtenstein A I, Anisimov V I and Zaanen J 1995 *Phys. Rev.* B **52** R5467–70
[38] Anisimov V I, Aryasetiawan F and Lichtenstein A I 1997 *J. Phys.: Condens. Matter* **9** 767–808
[39] MacEnulty L and O'Regan D D 2023 *Phys. Rev.* B **108** 245137
[40] Chen H and Millis A J 2016 *Phys. Rev.* B **93** 045133
[41] Czyżyk M T and Sawatzky G A 1994 *Phys. Rev.* B **49** 14211–28
[42] Amadon B, Applencourt T and Bruneval F 2014 *Phys. Rev.* B **89** 125110
[43] Aryasetiawan F, Imada M, Georges A, Kotliar G, Biermann S and Lichtenstein A I 2004 *Phys. Rev.* B **70** 195104
[44] Cococcioni M 2002 A LDA+U study of selected iron compounds Condensed matter theory Scuola Internazionale Superiore di Studi Ananzati (available at: https://iris.sissa.it/handle/20.500.11767/3939)
[45] Binci L and Marzari N 2023 *Phys. Rev.* B **108** 115157
[46] Moore G C, Horton M K, Ganose A M, Siron M, Linscott E, O'Regan D D and Persson K A 2022 High-throughput determination of hubbard u and hund j values for transition metal oxides via linear response formalism (arXiv:2201.04213)
[47] Geneste G, Amadon B, Torrent M and Dezanneau G 2017 *Phys. Rev.* B **96** 134123
[48] Pulay P 1980 *Chem. Phys. Lett.* **73** 393–8
[49] ABINIT 9.6.2 did not originally have functionality implemented to apply a β-perturbation as it is described in section refLinear Response to acquire the Hund's J. The version of ABINIT used for these calculations was modified locally to do so. These local modifications later served as the blueprint for the official Hund's J implementation in ABINIT Version 9.10.1.
[50] Linscott E B 2019 *Accounting for Strong Electronic Correlation in Metalloproteins* (Physics University of Cambridge Corpus Christi College)