ARC 3.0: An expanded Python toolbox for atomic physics calculations

E.J. Robertson, N. Šibalić, R.M. Potvliege, M.P.A. Jones

PII: S0010-4655(20)30413-6
DOI: https://doi.org/10.1016/j.cpc.2020.107814
Reference: COMPHY 107814

To appear in: Computer Physics Communications

Received date: 23 July 2020
Revised date: 26 November 2020
Accepted date: 5 December 2020

Please cite this article as: E.J. Robertson, N. Šibalić, R.M. Potvliege et al., ARC 3.0: An expanded Python toolbox for atomic physics calculations, Computer Physics Communications (2020), doi: https://doi.org/10.1016/j.cpc.2020.107814.

This is a PDF file of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability, but it is not yet the definitive version of record. This version will undergo additional copyediting, typesetting and review before it is published in its final form, but we are providing this version to give early visibility of the article. Please note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2020 Published by Elsevier B.V.
ARC 3.0: An expanded Python toolbox for atomic physics calculations

E. J. Robertson\textsuperscript{a}, N. Šibalić\textsuperscript{b,∗}, R. M. Potvliege\textsuperscript{a}, M. P. A. Jones\textsuperscript{a}

\textsuperscript{a}Joint Quantum Center (JQC) Durham-Newcastle, Department of Physics, Durham University, South Road, Durham, DH1 3LE, United Kingdom
\textsuperscript{b}Université Paris-Saclay, Institut d’Optique Graduate School, CNRS, Laboratoire Charles Fabry, 91127, Palaiseau, France

Abstract

ARC 3.0 is a modular, object-oriented Python library combining data and algorithms to enable the calculation of a range of properties of alkali and divalent atoms. Building on the initial version of the ARC library [N. Šibalić \textit{et al.}, Comput. Phys. Commun. 220, 319 (2017)], which focused on Rydberg states of alkali atoms, this major upgrade introduces support for divalent atoms. It also adds new methods for working with atom-surface interactions, for modelling ultracold atoms in optical lattices and for calculating valence electron wave functions and dynamic polarisabilities. Such calculations have applications in a variety of fields, e.g., in the quantum simulation of many-body physics, in atom-based sensing of DC and AC fields (including in microwave and THz metrology) and in the development of quantum gate protocols. ARC 3.0 comes with an extensive documentation including numerous examples. Its modular structure facilitates its application to a wide range of problems in atom-based quantum technologies.

Keywords: alkaline earth atoms, divalent atoms, alkali atoms, Rydberg states, dipole-dipole interactions, electron wave functions, optical lattices, Bloch bands, Wannier states, dynamic polarisability, magic wavelengths, atom-surface van der Waals interaction, Stark shift, Förster resonances, quantum technologies, neutral-atom quantum computing, atom-based sensors

1. Introduction

Neutral atoms are ideal building blocks for both fundamental research and technological applications exploiting quantum mechanics. The stability of their physical properties makes them an ideal choice for a sensing medium, virtually eliminating the need for recalibration [1, 2]. By promoting atoms to highly-excited, long-lived Rydberg states, the sensitivity to applied electric fields can be enhanced by many orders of magnitude compared to the ground state [3], which has applications to electric field metrology over a wide range of frequencies from DC to THz fields [2, 4, 5, 6, 7].

The reproducible properties of a given atomic species allows for the creation of hundreds of identical atomic qubits [8, 9]. Together with the dramatic increase in the interatomic interaction strength associated with Rydberg excitation, this enables applications in quantum simulation [8, 10, 11, 12] and quantum optics [13, 14, 15, 16, 17, 18, 19].

Having precise and accurate values of the relevant atomic parameters (e.g., transition dipole moments, Stark shifts, etc.) is crucial for developing new experiments, comparing to theory, or simply for interpreting the data. As the number of potentially relevant parameters vastly exceeds what can be tabulated, there...
is a need for an atom-calculator computing these parameters on

demand. ARC 1.0 [20] aimed to provide such a research tool

for alkali metals by combining the best available algorithms and

experimental data into a modular, object-oriented Python li-

brary. The open source nature of the package and the popularity

of this programming language facilitated its rapid adoption by

the community and the inclusion of newly developed methods.

Related efforts are those of JAC [21] and Pairinteraction [22]

programs.

However, a growing number of research groups are now

working on Rydberg states of atoms with two valence elec-

trons [23], such as the alkaline earth elements (e.g., Ca, Sr) and

some of the lanthanides (e.g., Yb, Ho [24]). While several

groups have performed calculations of Stark maps [25, 26, 27]

and interaction potentials [28, 29, 30], open-source codes like

those provided by ARC 1.0 [20] and Pairinteraction [22] for

the alkalis are not available for these species. This motivated

the major upgrade of the ARC library we are presenting here.

Specifically, we have extended ARC to divalent atoms, modify-
ing the algorithms as necessary and including supporting data for

Sr, Ca, and Yb. Methods for calculating long-range

van der Waals interactions have been extended to degenerate

perturbative calculations and pair-state calculations to arbitrary

interspecies calculations. This new version of ARC also adds

functionalities for atom-surface interactions, optical trapping

and visualisation of results. The miniaturisation of atom-based

sensors [31], and the strong long-wavelength transitions be-
tween Rydberg states [3], make atom-surface interaction effects

prominent [32]. We have therefore included non-retarded van
der Waals atom-surface potential calculations in ARC 3.0. Data

are currently included only for sapphire and perfectly reflective

surfaces, but other surfaces can easily be incorporated.

The importance of optical trapping stimulated the addition
of two modules. OpticalLattice1D is dedicated to optical lat-
tices and allows easy calculation of Bloch bands, Bloch states

and Wannier states. DynamicPolarizability calculates the

wavelength-dependent atomic polarizability relevant for optical

trapping, and enables searches for “magic-wavelengths” where
two different atomic states have the same polarizability.

Finally, to allow easy visualisation of different atomic states,
a module Wavefunction provides sectional views of the atomic

wave function for arbitrary atomic states. These are important
both for pedagogical and research purposes, especially since the

size of Rydberg electron orbitals can be large enough to encom-
pass other atoms, and may even approach the typical length-

scale over which external trapping potentials vary [33, 34].

How to start using the library is explained in Section 2. The
rest of the paper gives an overview of the newly implemented

calculations, with comments on restrictions and implementa-
tion details.

2. Installation and getting started

ARC 3.0 is available from the online repository Python Pack-
age Index (PyPI) and can be installed from the command line

simply by invoking

pip install ARC-Alkali-Rydberg-Calculator

This installs the package correct for the user’s environment, which
can be based on Python 3 for Windows, Mac OS or Linux. All the methods discussed in the following can be used after importing them from the arc library:

```python
from arc import *
# write your code
```

The examples of code given in the paper assume that this line was included at the beginning of the program to import the library.

The detailed documentation of the ARC library is available online [35] and is updated at each upgrade of the code. New users are recommended to consult the accompanying Jupyter IPython interactive notebooks listed on the “Getting started with ARC” page of the online documentation [35], which provide examples of calculations and describe the relevant physics. The interactive notebooks contain a number of examples that benchmark ARC results against the existing literature. Finally, we note that a selection of the functionality of the ARC library is available at https://atomcalc.jqc.org.uk as a part of the web-app Atom Calculator. This web page will also generate code that can be used as example-on-demand to help users start their own calculations. Bug reporting, questions and further code development are tracked on the projects GituHub page [36].

3. Overview of the new functions

A high-level view of the ARC 3.0 library is shown in Fig. 1. This upgrade generalises methods originally included in ARC 1.0, which were developed for AlkaliAtom with electron spin $S = 1/2$, to the singlet ($S = 0$) and triplet ($S = 1$) states of diva-

tent atoms. This approach is underpinned by the broad validity of the single active electron approximation for highly-excited states of divalent atoms, as discussed in Appendix A. Thus most of the divalent_atom_functions methods are directly inherited from alkali_atom_functions, upgraded as neces-
sary to support divalent atoms. The spin state is now specified through an optional keyword parameter $\alpha$, which defaults to 0.5 for alkali atoms and must be set to 0 or 1 for divalent atoms. To support these calculations, a semi-classical method for calculating dipole and quadrupole matrix elements has been implemented that does not require the use of a model potential.

The range of divalent atom data currently included in ARC 3.0 is discussed in Appendix B. New species/series may easily be added by generating the appropriate data files. Inter-species calculations for both PairStateInteractions and StarkMapResonances are possible.

In addition, PairStateInteractions.getC6perturbatively now supports degenerate perturbation theory for the calculation of $C_6$ coefficients. The possibility of taking into account a weak magnetic field directed along the quantization axis has also been introduced (only paramagnetic terms linear in field strength which shift energy levels but do not mix states are included in the calculation). Moreover, the new single-atom cal-
culations Wavefunction, AtomSurfaceVdW, OpticalLattice1D and DynamicPolarizability have been added in
calculations_atom_single. (The calculations of atom-surface interactions implemented in AtomSurfaceVdW require values of the refractive index for the surface of interest for a range of frequencies. Methods for general optical surface properties are implemented in materials.)

Finally, we have also created a group of advanced calculations which are likely to be too specialized for a core toolbox. The corresponding modules are intended for expert use and can be imported with

```python
# from arc.advanced.<mn> import *
# where <mn> is module name, for example:
from arc.advanced.population_lifetime import getPopulationLifetime
```

The modules in this advanced library will be built on top of the ARC core library and will provide solutions for specialised research questions. The first of these modules is population_lifetime, which has been contributed by the authors of Ref. [37]. It gives access to a getPopulationLifetime function calculating population lifetimes taking into account the redistribution of population within a Rydberg manifold driven by black-body radiation, including repopulation processes (such population lifetimes thus differ from state lifetimes).

In the following we outline the calculations newly implemented in ARC 3.0 and provide examples of code for each of them.

### 3.1. Divalent atoms dipole and quadrupole matrix elements

Dipole and quadrupole matrix elements factorize into products of radial matrix elements and angular factors. ARC provides access to both reduced (e.g. getReducedMatrixElementJ, getReducedMatrixElementJ_asymmetric, getReducedMatrixElementL, ...) and full matrix elements (getDipoleMatrixElement, getDipoleMatrixElementHFS, ...) as detailed in the online documentation [35].

| Atom     | Measured energy levels | Quantum defects | References |
|----------|------------------------|-----------------|------------|
| $^{88}$Sr | $^{1}S_0$, $^{1}P_1$, $^{1}D_2$, $^{3}S_1$, $^{3}P_0$, $^{3}P_1$, $^{3}D_1$, $^{3}D_3$, $^{3}D_4$, $^{3}F_4$ | $^{1}S_0$, $^{1}P_1$, $^{1}D_2$, $^{3}P_2$, $^{3}D_1$, $^{3}D_3$, $^{3}F_2$, $^{3}F_3$, $^{3}F_4$ | [38, 39, 40, 41, 42, 43, 44, 45, 46] |
| $^{40}$Ca | $^{1}F_3$ | $^{1}S_0$, $^{1}P_1$, $^{1}F_3$, $^{1}S_1$, $^{3}P_1$, $^{3}P_2$, $^{3}D_2$, $^{3}D_3$ | [38, 43, 47, 48, 49, 50] |
| $^{174}$Yb | $^{1}S_0$, $^{1}P_1$, $^{1}D_2$, $^{3}S_2$ | $^{1}S_0$, $^{1}P_1$, $^{1}D_2$, $^{3}D_2$, $^{3}D_3$, $^{3}D_4$, $^{3}D_5$, $^{3}F_2$, $^{3}F_3$, $^{3}F_4$, $^{3}F_5$ | [51, 52, 53] |
As noted above, the calculations are performed within the single active electron approximation. The radial matrix element for a dipole transition between states \(|n, L, J\rangle\) and \(|n', L', J'\rangle\) of hydrogen or an alkali atom are calculated as in ARC 1.0, i.e., by numerical integration between suitably chosen bounds \(r_{i}\) and \(r_{o}\):

\[
R_{nLJ, n'L'J'} = \int_{r_{i}}^{r_{o}} R_{nLJ}(r) r_{o}^{2} dr
\]

with \(S = S' = \frac{1}{2}\) and the wave functions \(R_{nLJ}(r)\) obtained as solutions of the Schrödinger equation for a model potential.

Since model potential methods are problematic for divergent atoms (Appendix A) the dipole radial matrix elements for these species are obtained in a semi-classical approximation [54].

This approach is not based on numerical wave functions. Instead, the dipole radial matrix elements take on the form

\[
R_{nLJ, n'L'J'} = \frac{3}{2} n_{e}^{2} \left[ 1 - \left( \frac{\ell_{c}}{n_{e}} \right)^{2} \right]^{1/2} \sum_{p=0}^{\infty} \gamma^{p} g_{p}(\Delta n_{e}),
\]

with \(S = S' = 0\) or 1. In this equation \(n'\) is the reduced principal quantum number \(n_{o} = n - \delta(n, L, S, J)\) with \(\delta(n, L, S, J)\) being the quantum defect for the \(|n, L, S, J\rangle\) state) and \(\ell_{c}, n_{e}, \gamma, \Delta L\) and \(\Delta n_{e}\) are defined as

\[
\ell_{c} \equiv \frac{L + L' + 1}{2}, \quad n_{e} \equiv \sqrt{n' n}, \quad \gamma \equiv \Delta L / n_{e},
\]

\[
\Delta L \equiv L - L', \quad \Delta n_{e} \equiv n_{e} - n'.
\]

Moreover,

\[
\begin{align*}
g_{0}(\Delta n_{e}) &= \frac{1}{3 \Delta n_{e}} \left[ J_{\Delta n_{e}, -1}(\Delta n_{e}) - J_{\Delta n_{e}, +1}(\Delta n_{e}) \right], \\
g_{1}(\Delta n_{e}) &= -\frac{1}{3 \Delta n_{e}} \left[ J_{\Delta n_{e}, -1}(\Delta n_{e}) + J_{\Delta n_{e}, +1}(\Delta n_{e}) \right], \\
g_{2}(\Delta n_{e}) &= g_{0}(\Delta n_{e}) - \frac{2 n_{e} \Delta n_{e}}{\pi \Delta n_{e}}, \\
g_{3}(\Delta n_{e}) &= \frac{\Delta n_{e}}{2} g_{0}(\Delta n_{e}) + g_{1}(\Delta n_{e}),
\end{align*}
\]

where the \(J_{\Delta n_{e}}(\Delta n_{e})\)'s are Anger functions:

\[
J_{\Delta n_{e}}(x) = \frac{1}{\pi} \int_{0}^{\pi} d\theta \cos[\Delta n_{e} \sin(\theta)].
\]

The radial quadrupole matrix elements are calculated as

\[
R_{nLJ, n'L'J'}^{Q} = \int_{r_{i}}^{r_{o}} R_{nLJ}(r) r_{o}^{2} R_{n'L'J'}(r) r^{2} dr,
\]

in the case of hydrogen and alkali atoms, and by using the corresponding semi-classical formulae for divergent atoms. The latter differ between different values of \(\Delta L\). For \(\Delta L = \pm 2\),

\[
R_{nLJ, n'L'J'}^{Q} = \frac{5}{2} n_{e}^{4} \left[ 1 - \left( \frac{\ell_{c} + 1}{n_{e}} \right)^{2} \right]^{3/2} \times \\
\sum_{p=0}^{\infty} \gamma^{p} Q_{p}(\Delta n_{e}),
\]

whereas for \(\Delta L = 0\),

\[
R_{nLJ, n'L'J'}^{Q} = \frac{5}{2} n_{e}^{4} \left[ 1 - \frac{3 c^2}{n_{e}^{2}} \sum_{p=0}^{\infty} \gamma^{p} Q_{2p}(\Delta n_{e}) \right].
\]

The expansion coefficients \(Q_{p}(\Delta n_{e})\) are the same in both cases:

\[
\begin{align*}
Q_{0}(\Delta n_{e}) &= -\frac{6}{5(\Delta n_{e})} R_{1}(\Delta n_{e}), \\
Q_{1}(\Delta n_{e}) &= -\frac{6}{5\Delta n_{e}} R_{0}(\Delta n_{e}) + \frac{5}{6} \sin(\pi \Delta n_{e}), \\
Q_{2}(\Delta n_{e}) &= -\frac{3}{4} \frac{6}{5\Delta n_{e}} R_{1}(\Delta n_{e}) + g_{0}(\Delta n_{e}), \\
Q_{3}(\Delta n_{e}) &= \frac{1}{2} \frac{\Delta n_{e}}{2} R_{0}(\Delta n_{e}) + Q_{1}(\Delta n_{e}).
\end{align*}
\]

We have compared the model potential and semi-classical methods for calculating dipole matrix elements using rubidium as a test case. The two methods give results in close agreement when \(|n - n'| \approx 0\), even for \(n, n'\) as low as 10 (below which multi-electron effects can be expected to become important).

While the agreement deteriorates when \(|n - n'|\) increases, the semi-classical results do not differ by more than 5% from the model potential results in the range 0.65 \(n \leq n' \leq 1.5 \Delta n_{e}\). Given that outside this range the dipole matrix elements are less than 1% of their values at \(|n - n'| \approx 0\) (except for very small values of \(n\)), the semi-classical approach is normally appropriate for any value of \(n - n'\) for which the dipole matrix element is large enough to be significant in calculations of Stark maps or dispersion coefficients. A multi-channel quantum defect approach would be more appropriate in the regions where perturbers mix states of different symmetries [55]; however, such calculations are beyond the scope of the current version of ARC.

All dipole and quadrupole matrix elements for calcium, strontium and ytterbium currently used or produced by ARC 3.0 are obtained as described above, as no literature values of these quantities are currently available for these atoms. Matrix elements obtained in the future from more accurate calculations or from measurements can be added to a literature file, as described in Section 4.3. The library will use the values found in this literature file, should there be any, rather than recalculate them.

3.2. Pair-state calculation of atom-atom \(C_{6}\) interactions in degenerate perturbation theory

In the single-active electron approximation, the interaction between two atoms at interatomic distance \(R\) arises from the interaction between the valence electrons and the interaction of each of these electrons with the screened nucleus of the other atom. Denoting the coordinates of the valence electrons relative to the respective nuclei by \(r_{1}\) and \(r_{2}\), this interaction is given by the following multipolar expansion (c.f. Eq. (19) in Ref. [20] for details):

\[
V(R) = \sum_{L_{1}, L_{2}=1}^{\infty} \frac{V_{L_{1}, L_{2}}(\mathbf{r}_{1}, \mathbf{r}_{2})}{K^{L_{1}+L_{2}+1}},
\]

where \(L_{1}, L_{2} \geq 0\), and \(V_{L_{1}, L_{2}}(\mathbf{r}_{1}, \mathbf{r}_{2}) \) is the multipole expansion coefficient for the interaction between the electrons with \(L_{1}\) and \(L_{2}\) as the lowest order.

\[
V_{L_{1}, L_{2}}(\mathbf{r}_{1}, \mathbf{r}_{2}) = \sum_{\ell_{1}, \ell_{2}=0}^{\infty} \sum_{m_{1}, m_{2}=\ell_{1}}^{\ell_{2}} \frac{V_{L_{1}, L_{2}}^{\ell_{1}, \ell_{2}}(\mathbf{r}_{1}, \mathbf{r}_{2})}{n_{e}^{\ell_{1}+\ell_{2}}},
\]

where \(n_{e}\) is the number of electrons, and \(V_{L_{1}, L_{2}}^{\ell_{1}, \ell_{2}}(\mathbf{r}_{1}, \mathbf{r}_{2})\) is the multipole expansion coefficient for the interaction between the electrons with \(L_{1}\) and \(L_{2}\) as the lowest order.
with

\[ V_{k_i,k_2}(r_1, r_2) = \frac{(-1)^{k_1-k_2} 4\pi}{\sqrt{(2k_1+1)(2k_2+1)}} \times \sum_p \sqrt{\frac{k_1+k_2}{k_1+p}} \frac{k_1+k_2}{k_1+p} r_1^{k_1} r_2^{k_2} Y_{k_i,p}(\hat{r}_1) Y_{k_i,-p}(\hat{r}_2). \]  \[ (7) \]

Here the \( \binom{n}{m} \) are binomial coefficients and the \( Y_{k,p}(\hat{r}) \) are spherical harmonics, and it is assumed that the quantization axis of both atoms is directed along the internuclear axis [as in Fig. 2(a) ]. Terms with \( k_1 + k_2 = 2, 3, 4, \ldots \) correspond respectively to dipole-dipole, dipole-quadrupole, quadrupole-quadrupole interactions and so on.

In the large \( R \) limit, the interaction between Rydberg atoms is typically dominated by dipole-dipole interactions. Such an interaction couples an initial pair-state \( |r_1 r_2\rangle \) to other states \( |r' r''\rangle \) whose energy differs by an energy defect \( \Delta_{r',r''} = E_{r'} + E_{r''} - E_{r_1} - E_{r_2} \). To second order in this interaction, the interaction energy of two atoms in the initial pair-state \( |r_1 r_2\rangle \) is given by a van der Waals interaction potential of the form \( -C_6/R^6 \), where \( C_6 \) is defined by the following equation if the pair-state energy \( E_{r_1} + E_{r_2} \) is non-degenerate:

\[ C_6 = \frac{\sum_{r' r''} |\langle r' r'' | V_{dd}(R) | r_1 r_2 \rangle|^2}{\Delta_{r',r''}}. \]  \[ (8) \]

Here the sum goes over all the pair states \( |r' r''\rangle \) that are coupled by electric dipole transitions to the original pair-state \( |r_1 r_2\rangle \), and \( V_{dd}(R) \) is the dipole-dipole part of \( V(R) \):

\[ V_{dd}(R) \equiv V_{11}(r_1, r_2)/R^3. \]  \[ (9) \]

However, there are situations in which the pair-state energy \( E_{r_1} + E_{r_2} \) is degenerate and the dipole-dipole interaction couples the initial pair-state \( |r_1 r_2\rangle \) to some other pair-states \( |r_3 r_4\rangle \) of the same energy — for instance, in the absence of external fields, to pair-states differing in magnetic quantum numbers only. Two examples of such off-diagonal couplings mixing pair-states are given in Fig. 2. In such situations, a perturbative calculation of the \( C_6 \) coefficients requires the diagonalization of the matrix \( C \) describing the second-order coupling between the energy-degenerate pair-states [28]. To this end, we work in the basis \( \{r_1 r_2\} \) of degenerate pair-states, i.e.,

\[ |r_1 r_2\rangle \in \{ n_1 L_1 S_1 J_1 m_{J_1} = -J_1, \ldots, n_1 L_1 S_1 J_1 m_{J_1} = +J_1 \} \]

\[ \otimes \{ n_2 L_2 S_2 J_2 m_{J_2} = -J_2, \ldots, n_2 L_2 S_2 J_2 m_{J_2} = +J_2 \}, \]  \[ (10) \]

where \( (n_1, L_1, S_1, J_1, m_{J_1}) \) and \( (n_2, L_2, S_2, J_2, m_{J_2}) \) are the quantum numbers associated with the states \( |r_1\rangle \) of atom 1 and \( |r_2\rangle \) of atom 2, respectively. The element of \( C \) corresponding to the second-order coupling between the \( q\)-th and the \( p\)-th basis states is

\[ R^6 \sum_{r' r''} \langle r_1 r_2 | V_{dd}(R) | r' r'' \rangle \langle r' r'' | V_{dd}(R) | r_1 r_2 \rangle / \Delta_{r',r''}. \]

Diagramalising this matrix yields a \( C_6 \) coefficient for each of the energy curves the degenerate pair-state energy splits into due to the dipole-dipole interaction, these coefficients being the eigenvalues \( C_6^{(i)} \) of \( C \) [see, e.g., Fig. 2(c)] . These eigenvalues are independent of the orientation of the inter-atomic axis relative to the quantization axis, in the absence of external fields, although the composition of the corresponding energy eigenstates in terms of the basis states defined above depends on the choice of the quantization axis [28].
Degenerate perturbation theory can be used within the
getC6perturbatively function by setting the flag degener-
atePerturbation=True. The function will then return the $C_6^{(i)}$
eigenvalues and the corresponding eigenvectors of the relevant
$C$ matrix. For example, the following fragment of code pro-
duces the results displayed in the third column of Fig. 2(c).

```python
# Sr 88 40 $^3P_1$ mj=0, 40 $^3P_1$ mj=0 state
calc = PairStateInteractions(
    Strontium88(),
    40, 1, 1,
    40, 1, 1,
    0, 0,
    s=1
)
theta = 0; phi = 0; deltaN = 5;
deltaE = 30e9; # in Hz
c6, eigenvectors = calc.getC6perturbatively(
    theta, phi, 5,
    deltaE, degeneratePerturbation=True)
# getC6perturbatively returns the C6 coefficients
# expressed in units of GHz mum^{-6}.
# Conversion to atomic units:
c6 = c6/1.445e-19
# These results should still be divided by n^{-11}5
# to be plotted as in Fig. 2(c).
```

Here theta and phi are, respectively, the polar angle ($\Theta$) and
azimuthal angle ($\Phi$) defining the orientation of the inter-atomic
axis in a reference whose z-axis is parallel to the axis of quan-
tisation of the angular momenta. The dependence on $\Theta$ and $\Phi$
of the elements of $C$ is taken into account by rotating the atomic
basis states using Wigner D-matrices, as in ARC 1.0 [20]. I.e.,
$V_{dd}(R)$ defined above is replaced by the angle-dependent $V_{dd}(R, \Theta, \Phi)$
with, in a simplified notation,

$$V_{dd}(R, \Theta, \Phi) = [D(J_1', \Theta, \Phi) \otimes D(J_2', \Theta, \Phi)] V_{dd}(R) [D(J_1, \Theta, \Phi) \otimes D(J_2, \Theta, \Phi)]\ \ (	ext{11})$$

where the $D(J m_j, \Theta, \Phi)$ represent the relevant rotation matrix-
es. Although the elements of the matrix $C$ and the composi-
tion of its eigenstates in terms of the basis states defined above
depend on $\Theta$ and $\Phi$, this is not the case for its eigenvalues $C_6^{(i)}$
(hence for the $C_6$ coefficients resulting from this calculation)
[28].

Invoking getC6perturbatively without the flag degeneratePerturbation=True or with degeneratePerturbation=False
will only return the individual element of the matrix $C$ corre-
sponding to the values of the quantum numbers specified in the
call, rather than the eigenvalues and eigenvectors of $C$. These
individual elements normally depend on $\Theta$ and $\Phi$ and can be
taken to be effective $C_6$ coefficients for particular combinations
of magnetic quantum numbers.

We note that the interaction energies can also be obtained
non-perturbatively by full diagonalisation of the Hamiltonian
using the function diagonalize. An example of results obtained
in this way is given by Fig. 3, which is produced by running the
following code:

```python
calc = PairStateInteractions(
    Strontium88(),
    60, 0, 1,
    60, 0, 1,
    1, 1,
    s=1
)
theta=0; phi=0; deltaN = 5;
deltaL = 5; deltaMax = 25e9 # [Hz]
# Generate pair-state interaction Hamiltonian
calc.defineBasis(theta, phi, deltaN, deltaL, deltaMax,
    progressOutput=True)
# Diagonalise
r = np.linspace(1.5, 4, 300)
nEig = 200 # Number of eigenstates to extract
calc.diagonalise(r, nEig, progressOutput=True)
# Plot
calc.plotLevelDiagram() calc.showPlot(interactive=False)
```

The selected range of values of $R$ shown on Fig. 3 is above the
LeRoy radius, estimated for low-$L$ states as $(r_1^n)^{1/2} + (r_2^n)^{1/2} \approx
3/2a_0(n_1^n + n_2^n)$, for electron coordinates $r_1, r_2$ measured relative
to nucleus, Bohr radius $a_0$, and principal quantum numbers $n_{1,2}$
of the respective states of two atoms.

We also note that the implementation of degenerate per-
turbation theory made in ARC does not take into account energy
degeneracies between states differing in $L$ or $S$. It is therefore
not appropriate for atomic hydrogen or for high angular mo-
mentum states. We recommend using the full diagonalisation
method for such cases.
3.3. Inter-species interaction calculations

PairStateInteractions supports inter-species calculations. Users can initialize such calculations using the keyword argument atom2 to explicitly state the species of the second atom. Note that to specify the spin state of the second atom, the keyword argument s2 should also be set. Setting s2 can also be used for calculations where the atoms are from the same atomic species but have different spin.

For example, pair-state calculations between rubidium atoms in the $|5s60s \ ^2S_{1/2} \ m_J = 1/2 \rangle$ state and ytterbium atoms in the $|6s54s \ ^1S_0 \ m_J = 0 \rangle$ state [Fig. 4] are initialized as follows:

```python
calc = PairStateInteractions(
    Rubidium(),
    60, 0, 0.5,
    54, 0, 0,
    0.5, 0,
    s=0.5,
    atom2=Ytterbium174(),
    s2=0
)
```

and transition frequency for pairs of states selected interactively by clicking on energy levels.

3.4. Single-atom properties for divalent atoms

ARC 3.0 extends most of the single atom methods available in ARC 1.0 to divalent atoms. For example, Stark maps can be obtained for divalent atoms by setting the additional key argument s to define the spin state. Note that in the single electron approximation, states of different total spin are not coupled. Example results from such a calculation are shown in Fig. 5. Static electric fields are often used to adjust pair-state energies. StarkMapResonances allows the user to search for electric field strengths where two pair-states have same energies (Forster resonances). Lastly LevelPlot allows the plotting and interactive exploration of energy level diagrams. These diagrams may be opened as interactive stand-alone plots (from a command line Python call or in a Jupyter notebook with `matplotlib` `qt`); then will then display the transition wavelength and transition frequency for pairs of states selected interactively by clicking on energy levels.

3.5. Electronic wave functions

Visualisations of atomic wave functions are a useful pedagogical tool enabling visual interpretation of effects such as dipole moments. In addition, the detailed shape of Rydberg electron wave functions plays an important role in a number of effects. For example, the wave functions can become so spread out that they encompass other atoms. The modulations of the electron probability density may then induce a significant variation in the potential energy of the encompassed atoms, which may lead to the formation of Rydberg molecules [56]. At the same time, the optical potentials used for atom trapping may vary substantially over the length-scale of the Rydberg electron wave functions, giving rise to energy shifts [33] and affecting the trap lifetime [34].

Wavefunction enables the calculation of atomic wave functions for arbitrary superposition states. Quick 2D and 3D visualisations are possible, with a choice of units (atomic or SI). For example, the following code can be used to obtain and plot the probability density function for the $10^2 F_{7/2} \ m_J = 7/2$ state of rubidium [Figs. 6(a and b)]:

```python
atom = Rubidium()
n = 10; l=3; j=3.5; mj=3.5;
stateBasis = [n, l, j, mj]
stateCoef = [1] # pure 10 F_7/2 m_j=7/2 state
wf = Wavefunction(atom, stateBasis, stateCoef)
wf.plot2D(plane="x-z", units="atomic"); plt.show()
wf.plot2D(plane="x-y", units="atomic"); plt.show()
```

Wavefunction can be integrated with other ARC functions. For example, one can find the electronic wave function for an atom perturbed by an electric field by getting the state from the corresponding StarkMap and using Wavefunction as per the following code to plot the result. An example is shown in Figs. 6(c-d).

```python
atom = Caesium()
```
Figure 6: Example wave function visualizations showing (a and b) cuts through the $^{85}$Rb $|10f\ 2F7/2 \ mJ>$ state unperturbed in zero electric field. The plot shows the eigenstate that is adiabatically connected to a $|S_{1/2} 1/2 \ mJ>$ perturbed state in zero electric field. Note the change of units (Å or nm), which can be easily specified using keyword arguments in the Wavefunction methods.

calc = StarkMap(atom)
states, coef, energy = calc.getState([28, 0, 0.5, 0.5],
-240000,23,32,20, accountForAmplitude=0.95,
degradOutput=True)
wf = Wavefunction(atom, states, coef)
wf.plot3D(plane="x-z", units="nm"); plt.show()
wf.plot2D(plane="x-z", units="nm", pointsPerAxis =400, axisLength=2800)
plt.show()

The density and scale of the mesh on which the wave function is calculated can be adjusted with optional keyword parameters. The probability density functions can be provided in Cartesian as well as in spherical coordinates (respectively through getRtimesPsi and getRtimesPsiSpherical), and it is also possible to obtain arrays of wave functions for all different possible spin states (using the getPsi method).

Note that Wavefunction is currently only supported for species in the alkali_atom_data class, as calculations based on model potentials are not currently supported for divalent atoms.

3.6. Atom-surface interactions: van der Waals potentials

In the vicinity of a surface, an atomic dipole interacts with its image in the surface, leading to shifts of the atomic energy levels. For small atom-surface distances $z < \lambda/(2\pi)$, where $\lambda$ is the wavelength associated with the strongest transition, the interaction potential $V(r)$ for an atom in the state $a$ is of the non-retarded, van der Waals form [57]

$$V_{\text{as}}(z) = -\frac{1}{4\pi\varepsilon_0} \sum_b \langle m(\omega_{ab}) - 1 [\mu_c^{ab}]^2 + [\mu_{\mu_c}^{ab}]^2 + 2[\mu_{\mu_c}^{ab}]^2 \rangle \frac{1}{16z^3} \equiv -C_3 \frac{1}{z^3}.\quad(12)$$

Here $n(\omega_{ab})$ is the frequency dependent refractive index associated with the surface. The summation covers all the states $b$ dipole coupled to state $a$. The corresponding transition frequencies and dipole matrix elements in the $x$, $y$, and $z$ directions are respectively $\omega_{ab}$, $\mu_x^{ab}$, $\mu_y^{ab}$, and $\mu_z^{ab}$. The $z$-axis is taken to be perpendicular to the surface. Note that different states will have different $C_3$ values, which leads to a modification of the atomic transition frequencies near the surface.

To specify the surface material, ARC provides an abstract class materials.OpticalMaterial with a method getN returning the refractive index $n$ for a specified wavelength. A subclass Sapphire is provided as an example. The AtomSurfaceVdW uses information on the optical properties of the surface and the atomic transition frequencies to calculate $C_3$. For example, the following code will return the energy shift with error for the $6s^2S_{1/2}$ state of caesium in the proximity of a sapphire surface (the result is 1.259(2) kHz $\mu$m$^3$):

from arc.materials import Sapphire
atom = Cesium()
surface = Sapphire()
calc = AtomSurfaceVdW(atom, surface)
# look at 6s $1/2$ state
n1 = 6; l1 = 0; j1 = 0.5
# take into account coupling to
# $6p_{1/2}$, $6p_{3/2}$, $7p_{1/2}$, $7p_{3/2}$
coupledStates = [[6, 1, 0.5],
[6, 1, 1.5],
[7, 1, 0.5],
[7, 1, 1.5]]
c3, c3_err = calc.getStateC3(n1, 11, j1, coupledStates,
degradOutput=True)

Such calculations are possible for all the atomic species supported by ARC, to the extent that the required dipole matrix elements are available.

3.7. Optical lattices: Bloch bands, Bloch states, Wannier states

Atoms in optical lattices are important in many areas of science and technology including atomic clocks and gravimeters, quantum gas microscopes and quantum simulators. A laser standing wave gives rise, through the AC Stark shift (Sec. 3.8), to a spatially periodic potential for the atoms. As is well-known from solid state physics, such a periodic potential can also be considered in reciprocal space. In momentum space ($k$-basis), a potential with spatial period $\lambda/2$ couples free particle states that are separated by an integer multiple of $\delta k = 4\pi/\lambda = 2k \equiv k_l$ where $\delta$ is the wavelength of the optical field, and $k_l = 2k$ is the lattice momentum. Therefore, from the Bloch theorem, the eigenfunctions of the Hamiltonian (called Bloch states or Bloch wave functions in this context) can be parametrized by a quasimomentum $q$ ($|q| < k$), and these eigenfunctions are of the form $\exp(irq)$ times a periodic function of $r$ whose period is the same as that of the lattice. The Hamiltonian can
thus be diagonalised on a discrete basis of free-particle states 
\{e^{iqr}, e^{i(q+kl)r}, e^{i(q+2kl)r}, \ldots \} separated by an integer 
multiple of the lattice momentum in momentum space. Plotting the resulting 
energy levels for different values of the quasimomentum 
gives Bloch bands [Fig. 7(a)].

Both Bloch states and Bloch bands can be calculated using 
OpticalLattice1D. For example, the following code produces 
a Bloch band diagram for rubidium atoms trapped in an optical 
lattice formed by a 1064 nm standing wave with a maximal 
depth of 5 $E_R$, using a basis which includes states up to \( q + 35k \) 
in momentum ($E_R = \hbar^2/(2mL^2)$ is the recoil energy):

```python
atom = Rubidium87()
trapWavelength = 1064e-9
lattice = OpticalLattice1D(atom, trapWavelength)
lattice.defineBasis(35)
qMomentum = np.linspace(-1,1,100)
lattice.diagonalise(Vlat, qMomentum,
    saveBandIndex=0)
fig = lattice.plotLevelDiagram()
plt.show()
```

We can enumerate Bloch bands in order of their increasing 
energies, starting with the index 0 for the ground state. The 
corresponding Bloch wave functions can be obtained by specifying 
the quasimomentum and the Bloch band index, as in the 
following example:

```python
# lattice depth = 40 recoil energies
# quasi momentum = + 0 k
# blochBandIndex = 0
f = lattice.BlochWavefunction(40, +0.0, 0)
```

The Bloch wave functions are delocalised across the lattice 
sites [Fig. 7(b) bottom panel]. For many calculations and a more intuitive mapping to atomic physics experiments, it is convenient to switch to a basis of localised functions, namely Wannier functions. For each Bloch band, the Wannier functions $w_i(x)$ are defined (up to a normalisation factor) as a complete orthogonal set of functions localised at lattice points defined by a lattice vector $R$:

$$w_{i,q}(x) \propto \sum_{q} e^{-i\mathbf{q}\cdot\mathbf{R}} b_{i,q}(x), \quad (13)$$

where the sum goes over all values of the quasimomentum $q$ and $b_{i,q}$ are the Bloch wave functions for a given Bloch band index $i$ and quasimomentum $q$. Values of the Wannier function can be obtained after diagonalisation of the interaction potential Hamiltonian for which we defined Bloch band index to be saved by setting `saveBandIndex` keyword argument in diagonalise method. We can then call `getWannierFunction` to obtain values of the Wannier function in a given Bloch band [Fig. 7(b) middle panel]. For example:

```python
atom = Rubidium87()
trapWavelength = 1064e-9
lattice = OpticalLattice1D(atom, trapWavelength)
lattice.defineBasis(35)
qMomentum = np.linspace(-1,1,100)
lattice.diagonalise(Vlat, qMomentum,
    saveBandIndex=0)
print(lattice.getWannierFunction(x, latticeIndex = 0))
```

Note that `saveBandIndex` selects the band index, and `latticeIndex` sets the localisation of the function at the site with the corresponding index. The Wannier functions returned by the program should be normalised on the relevant lattice by the user.

### 3.8. Dynamic polarisabilities and magic wavelengths

The dynamic (AC) polarisability $\alpha(\omega)$ of an atom exposed to an oscillating electric field of angular frequency $\omega$ can be expressed as the sum of a contribution from the polarisability of the valence electron(s) $\alpha_v$ and the core polarisability $\alpha_c$. The valence polarisability often dominates. For an electron in state $|a\rangle$ with total angular momentum $J$ and projection $m_J$ the valence polarisability for linearly polarised light can be written as [58]

$$\alpha_v(\omega) = \alpha_0(\omega) + \frac{3m_J^2 - J(J+1)}{J(2J-1)} \alpha_1(\omega) + \alpha_2(\omega) + \alpha_0^{cont}(\omega); \quad (14)$$

that is as the sum of a scalar polarisability

$$\alpha_0(\omega) = \frac{2}{3(2J+1)} \sum_{|b\rangle \in states} \langle b \mid e \mid a \rangle^2 (E_b - E_a), \quad (15)$$

Figure 7: Example calculations for Rb atoms trapped in an optical standing wave $V(x)$ with wavelength $J = 1064/2 \text{ nm}$ and depth $5 E_R$. (a) Bloch band diagram showing energy as a function of quasimomentum $q$. (b) Lower panel: Corresponding Bloch wave functions for $q = 0$ (green) and $q = \hbar k$ (red). Middle panel: Wannier functions for the ground state and first excited Bloch band at $R = 0$. Upper panel: optical lattice potential $V(x)$. 

and a tensor polarisability

\[ \alpha_2(\omega) = 4 \frac{5J(2J-1)}{(6J+1)(2J+1)(2J+3)}^{1/2} \times \sum_{\text{states } |b\rangle} (-1)^{J+J_b} \begin{pmatrix} J & J_b & 1 \end{pmatrix} \frac{|b\rangle \langle \sigma| |a\rangle^2 (E_b - E_a)}{(E_b - E_a)^2 - (\hbar \omega)^2}. \]  

(16)

Here \( |b\rangle \) are reduced dipole matrix elements and the summation runs over all the bound states \( |b\rangle \), with total angular momentum \( j_b \), dipole-coupled to the state \( |a\rangle \) of interest. Finally, there is also a term \( \alpha_0^{\text{(cont)}} \) contributed by the continuum of unbound states (this contribution will be discussed in more detail below).

For example, the scalar and tensor polarisabilities of the caesium 100p \( ^1P_{1/2} \) state in the AC field given by a 1100 nm wavelength optical trapping laser can be obtained using the following code (in this calculation, the sum over intermediate states \( b \) is limited to all the bound states with \( n \leq 115 \), including the ground state and the low lying excited states):

```python
atom = Caesium()
n = 100
calc = DynamicPolarizability(atom, n, 1, 1.5)
calc.defineBasis(atom.groundStateN, n+15)
alpha0, alpha1, alpha2, alphaC, alphaP, closestState =
calc.getPolarizability(1100e-9, units="SI")
```

closestState saves the state whose transition frequency is closest to that of the driving field.

In addition to alpha0 and alpha2 the method getPolarizability also returns the core polarizability, \( \alpha_0 \) (alphaC), and the ponderomotive polarizability, \( \alpha_p \) (alphaP). The core polarizability is approximated by its static value (saved in atom.alphaC) which is appropriate when the driving laser is far from resonance with core transitions. The ponderomotive polarizability can be linked to the contribution from the continuum of unbound states \( \alpha_0^{\text{(cont)}} \) (which is currently not calculated by ARC). Close to a bound-state resonance, the contributions from the bound intermediate states dominate \( \alpha_1(\omega) \). However, for Rydberg states one is often in the opposite limit far from bound-state resonances. Away from strong resonances, the total scalar polarisability \( \alpha_0 + \alpha_0^{\text{(cont)}} \) then approaches the free-electron ponderomotive polarizability \( \alpha_p \) associated with the time-averaged motion of a free electron in an oscillating electric field \( [\alpha_p = e^2/(2\hbar m_e \omega^2)] \), where \( \omega \) is frequency of driving field and \( e \) and \( m_e \) are electron charge and mass respectively. This result is obtained by applying the limit where the Kepler frequency of the electron orbit around the core is much smaller than \( \omega \) as is typical for Rydberg states [59], and applying a Born-Oppenheimer approximation to separate the fast quiver motion of the loosely bound electron driven by the electric field component and the slower relative motion of electron around the ionic core.

The code also returns alpha1 which is a vector polarisability relevant in driving with non-linearly polarised light, see, e.g., Ref. [60]. This feature is not exploited in the current implementation of DynamicPolarizability, which focuses on the simplest case of driving under linearly polarized light; however it provides a path for future support of other polarizations. Other future extensions could include adding bound-state resonances to the core polarizability and an extended treatment of the continuum contribution near the ionization threshold.

DynamicPolarizability calculations can involve dipole matrix elements between low-lying states or between low-lying states and highly excited states. For low-lying states dipole matrix elements calculated both with the semiclassical approximation and the model potential method become less accurate. That is why ARC uses literature values for these states, where available. Users should check the availability of literature values in the corresponding files (Sec. 4.3). If values for low-lying transitions are available, the expected accuracy can be of the order of ~ 1%, otherwise the accuracy is limited to ~ 10% for typical input parameters. Currently, there are significant compiled literature sources for dipole matrix elements in alkali atoms, but the data for divalent atoms is more scarce.

For quantum state control in atoms in optical traps, it is often desirable to find optical trap parameters at which the AC induced shifts of two states, i.e., the polarisabilities of these two states, is exactly the same. That happens at so called “magic wavelengths” for optical trapping field. For example, to find such wavelengths for the 15p \( ^1P_{1/2} \) \( m_I = 1/2 \) and 6s \( ^2S_{1/2} \) \( m_J = 1/2 \) states of caesium we plot the polarisability of these two states in the same plot (see Fig. 8). The magic wavelengths are then given by the intersections where the two polarisabilities are equal:

```python
atom = Caesium()
n = 15
calcP = DynamicPolarizability(atom, n, 1, 0.5)
calcP.defineBasis(atom.groundStateN, n+15)
calcS = DynamicPolarizability(atom, 6, 0, 0.5)
calcS.defineBasis(atom.groundStateN, 25)
```

```python
wavelengthList = np.linspace(800, 940,1000) * 1e-9 #
```

Figure 8: Example calculation of \( \alpha_0(\omega) \) for the caesium 6 \( ^2S_{1/2} \) (red dashed) and 15 \( ^1P_{1/2} \) (blue solid) states. Resonant transitions to other states are marked with vertical dotted lines. The inset show a region around 881 nm where the polarisabilities are equal, corresponding to a magic wavelength.
ax = calcP.plotPolarizability(wavelengthList,
debugOutput=True, units="au")
calcS.plotPolarizability(wavelengthList, debugOutput=True, units="au")
plt.show()

Resonances are indicated by vertical lines (as shown in Fig. 8). The states responsible for these resonances and the corresponding resonance wavelengths are also printed when the flag debugOutput=True is set on.

4. Implementation notes

4.1. Handling of the spin quantum number

To allow the use of both alkali atoms and divalent atoms, most of the methods have a new keyword argument s specifying the spin quantum number. The default value of this keyword argument, s=0.5, is appropriate for alkali atoms. As such, s does not have to be specified for these atoms, making the API of ARC 3.0 completely backwards compatible with ARC 1.0. For divalent atoms, users should specify whether they are working with singlet or triplet states by setting the corresponding spin quantum number to s=0 or s=1 respectively. To state that the second atom has a different spin in PairStateInteractions, the optional keyword argument s2 should be set explicitly.

4.2. Use of fitted quantum defects for divalent atoms

The available experimental energy level data is much sparser and less precise for divalent atoms than for the alkalis. Inaccuracies arising from experimental errors may be reduced by using energies derived from a fit of the data to the Rydberg-Ritz formula over a range of principal quantum numbers (see Appendix B), rather than using the experimental energies directly. However, in many cases the calculations will require energies outside the range of values of n used in the fitting to the Rydberg-Ritz formula. Using this formula is normally justified for values of n above this range, but might be invalid for values of n below this range (e.g., because the Rydberg-Ritz formula does not fit the ground state and low excited states, or because the energy levels do not vary smoothly and monotonically with n due to perturbers). Using the experimental energies is thus often preferable for small values of n.

For clarity, we have tabulated the range of values of n over which the quantum defects have been fitted using a dictionary indexed by terms. For each term the dictionary returns the smallest and largest principal quantum numbers defining the range of principal quantum numbers used in the calculation of the Rydberg-Ritz coefficients provided in ARC. For example, the following code will do this for the $^1F_3$ term in Calcium-40:

```python
atom = Calcium40()
term = '1F3'
if term in atom.defectFittingRange:
    fitRange = atom.defectFittingRange[term]
```

For principal quantum numbers above the fitted range, the energies are calculated by extrapolation of the Rydberg-Ritz formula to outside this range. No extrapolation is done for principal quantum numbers below the fitted range. Instead, ARC tries to use tabulated energies if they exist — either energies provided by the user or the energies provided by ARC, which are literature values as described in the online documentation [35]. In the case of S, P or D series, the calculation is aborted and a value error raised, explaining the problem, if the program requires a missing tabulated energy. For $L > 2$, the calculation is not aborted but a value of zero is assumed for the quantum defect. These tabulated energies must be stored in the local data directory (Sec. 4.3) under a file name defined for each atom in the levelDataFromNIST variable. For example, for strontium they are stored in sr_level_data.csv. The energyLevelsExtrapolated variable of the used atom will be set to True if the calculation uses energies obtained by extrapolation of the Rydberg-Ritz formula outside the range of values of n used in the fitting.

4.3. Local data directory and updates to literature values

Dipole matrix elements from the literature can be added to ARC by setting the literatureDMEfilename variable in the DivalentAtom class to the appropriate file name and ensuring the format of the file matches that specified in the documentation.

When ARC 3.0 is first used, a local hidden directory named .arc-data is created in the user’s home folder. The results of calculations of, e.g. dipole matrix elements are saved in human-readable files in this folder, forming a look-up table that can be used to speed up future calculations. There is an option to add dipole matrix elements from the literature (or obtained using other calculation methods) by modifying the corresponding files. For example, to add a new literature value for a $^{88}$Sr dipole matrix element, the user should modify the file strontium_literature_dme.csv as per the table header. This can be conveniently done by loading the appropriate .csv file in any spreadsheet program, as long as the file format is preserved in terms of separators and header comments. To change other atomic properties, like quantum defects, users should make their own subclasses which inherit classes defined in the ARC. An example is given in the following code, which could be used to update the quantum defects for calcium:

```python
class MyCalcium40(Calcium40):
    quantumDefect = ...
    # write updated quantum defects
    # list in order specified in documentation
dipoleMatrixElementFile = "my_ca_dme.npy"
```

Once defined, MyCalcium40 can be used instead of Calcium40 in all the ARC calculations. To ensure the program does not use old cached values, new names of caching files should be provided when redefining atomic species in this way. Also,
note that if a new version of arc-data is released (changing the
version number in version.txt in the data directory), the data
entered manually in the .arc-data folder may be overwritten
automatically if no new file names had been specified.

We encourage users to submit new experimental data and
parameters for use by the community via a pull request on the
ARC GitHub page [36].

4.4. Looking under the hood with debugOutput=True

As for ARC 1.0, setting the keyword argument debugOutput=True results in verbose output, which may be useful for
checking basis states and other intermediate results. Additionally, many methods have a progressOutput=True option which
can be used for tracking the progress of the calculations.

5. Outlook

In summary, we have presented ARC 3.0, a major new re-
lease of the ARC Python library that extends the library to di-
valent atoms and adds a number of new methods of general in-
terest in Rydberg physics and beyond. We believe this common
code base and consistent interface for many different atomic
properties can speed up the development of new applications
and lower knowledge barriers, e.g., in quantum technologies
based on neutral atoms. The library also offers rich possibili-
ties of advanced educational projects for students in atomic and
quantum physics.

Future improvements of ARC could include the addition of
the calculation of wave functions for divalent atoms and addi-
tional methods for the accurate calculation of low-lying wave
functions [61], support for multi-channel quantum defect the-
ory calculations [55], and the calculation of scattering proper-
ties. New experimental data can be straightforwardly added to
the existing base. ARC is a community-oriented open source
package, and the authors welcome contributions of new core
data or algorithms to the main library as well as contributions
of more specialized codes to arc_advanced.

Acknowledgements

We thank Tsu-Chun Tsui and Trey Porto for tabulating the
ytterbium data. We also thank Hei Yin Andrea Kam for the use
of her code to check the calculations of strontium dipole matrix
elements, Paul Huillery for help with Stark map calculations,
and Ian Hughes for suggesting the bootstrap method. N. Š. is
supported by the H2020 Marie Skłodowska-Curie Actions (CO-
QUIDDE, H2020-MSCA-IF-2017 grant agreement No.786702).
The project was supported by the EPSRC Platform grant EP/R002067/1
and Standard grant EP/R035482/1. We also acknowledge fund-
ing from the project EMPiR-USOQS (EMPiR projects are co-
funded by the European Union’s Horizon2020 research and in-
novation programme and the EMPiR Participating States).

Appendix A. The single active electron approximation

Atomic structure calculations for Rydberg states are simpli-
fied by the rapid scaling of the size of the wave function of the
Rydberg electron with principal quantum number \( n \). In atoms
with a single valence electron, the calculation of energy lev-
eels and wave functions is achieved by solving the Schrödinger
equation with a modified \("model\" potential that accounts for the
screening effects of the closed-shell core. Each Rydberg
series labelled by the quantum numbers \( L, S \) and \( J \) is charac-
terized by a quantum defect \( \delta \) that quantifies the deviation of
each energy level from its hydrogenic equivalent. This method
works well for alkali atoms and is used by alkali_atom_functions.

For divalent atoms, the situation is complicated by two ef-
fects. The first is that the ionic core is no longer a closed
shell, since it contains the remaining valence electron. The
core is therefore more strongly polarized by the Rydberg elec-
tron, leading to an additional \( n \) dependence in the quantum de-
fects. The second effect is the existence of compact states where
both electrons are excited (e.g., a 5p\(^2\) configuration), known as
perturbers. These perturber states exhibit strong interelectronic
correlation effects and lead to perturbations of nearby Rydberg
levels. Even away from the energy of a perturber, Rydberg
states may acquire a small admixture of doubly excited states
[62].

Nevertheless, under many circumstances the properties of
Rydberg states in divalent atoms can be described under a sin-
gle active electron approximation [28]. Under this approxima-
tion, Russell-Saunders \( L – S \) coupling is assumed to hold, and
\( L, S \) and \( J \) are regarded as good quantum numbers. The effects
of core polarizability and perturbers are partially included via
the energy dependence of the quantum defects. Calculations
can then be made in a similar way to those performed in alkali
atoms, with the appropriate generalization of the angular momen-
tum algebra to integer spin. Previous work has shown that
this treatment gives good agreement with experiment for quan-
tities that depend on the long-range part of the wave function,
such as the DC Stark effect [26, 63].

The interaction between two divalent Rydberg atoms is also
dominated by the long-range character of the wave function.
Therefore a single active electron approach may be used here
also [28]. A study of the effect of perturbers on the interactions
[64] demonstrated that the single-electron treatment is valid to
a high degree of accuracy (<2\%) except for Rydberg states in
the immediate vicinity of perturbers.

Note that other observables that depend on the short-range
properties of the wave function, such as the coupling to low-
lying states (e.g., radiative lifetimes), are significantly modified
by the presence of even small amounts of perturbing states, and
are not well treated in the single active electron approximation
[55]. Here other methods such as multi-channel quantum de-
fect theory (MQDT) [55] that explicitly include the effects of
perturbers must be used.
Appendix B. Atomic Data for Divalent Atoms

As outlined in section 3.1, calculations involving divalent atoms rely on fitted values for the quantum defects. In the case of Ca, no new data was available, and so values compiled in [28, 65] are used. For Sr and Yb, we provide a new analysis of the available data for Sr and Yb that takes into account recent improvements in the spectroscopic data.

Experimental energy levels were fitted to the modified Rydberg-Ritz formula

\[ E(n) = I_s - \frac{Ry}{n^2} \left( \frac{\delta_0}{(n - \delta_0)^2} + \frac{\delta_1}{(n - \delta_1)^4} \right) \]  

using \( \delta_0, \delta_2 \) and \( \delta_4 \) as free parameters. The ionization energy \( I_s \) was constrained to the values obtained from the analysis of the best available spectroscopic data. \( Ry \) is the atom-specific Rydberg constant: \( Ry = R_w m_a / (m_e + m_e) \), where \( R_w \) is the Rydberg constant, and \( m_a \) and \( m_e \) are the mass of the considered isotope and the electron respectively. The fine-structure splitting of the ionization threshold was neglected. A least-squares fitting method was used, implemented via the \texttt{curve_fit} function from the scipy Python package.

The results, along with references to the experimental dataset are provided in Table B.1.

The choice of the range of \( n \) used for each series was a compromise between maximising the number of data points, and reducing the effect of series perturbations not described by the Rydberg-Ritz formula. The uncertainty on the Rydberg-Ritz parameters are 68% confidence limits obtained using a "bootstrap" method based on resampling with replacement [66], such that each confidence interval includes 68% of the results falling above the quoted value of the corresponding parameter as well as 68% of the results falling below it. For each series the fitted range was resampled 150 times. The asymmetry of the confidence limits reflects the asymmetric dependence of the value of the energy on the quantum defects encapsulated in Eq. (B.1), as well as the limitations of the experimental data. Note that correlations in the uncertainties are expected to be strong and are not explicitly considered here; users seeking to set rigorous error bounds on derived quantities should take this into account.

Appendix C. ARC 3.0 API

The tables C.1 to C.14 list the APIs of ARC 3.0. All old methods have been amended to allow handling of divalent atoms with two possible spin states. Additionally, methods that are newly introduced, or have significant new functionality, are marked with a \( \diamond \). Since for divalent atoms the calculation of electronic wave functions in the model potential approach is not implemented, some methods are available only when working with \texttt{AlkaliAtom} instances, and these are marked by a \( \circ \).

References

[1] D. Budker, M. Romalis, Optical magnetometry, Nature Physics 3 (2007) 227. doi:10.1038/nphys566.
[2] D. H. Meyer, Z. A. Castillo, K. C. Cox, P. D. Kunz, Assessment of Rydberg atoms for wideband electric field sensing, Journal of Physics B: Atomic, Molecular and Optical Physics 53 (2020) 034001. doi:10.1088/1361-6455/ab9051.
[3] N. Šibač, C. S. Adams, Rydberg Physics, 2399-2891, IOP Publishing, 2018. doi:10.1088/978-0-7503-1638-4.
[4] J. A. Sedlacek, A. Schwettmann, H. Kübler, R. Lów, T. Pfau, J. P. Shaffer, Microwave electrometry with Rydberg atoms in a vapour cell using bright atomic resonances, Nature Physics 8 (2012) 819. doi:10.1038/nphys2423.
[5] C. G. Wade, N. Šibač, N. R. de Melo, J. M. Kondo, C. S. Adams, K. J. Weatherill, Real-time near-field terahertz imaging with atomic optical fluorescence, Nature Photonics 11 (2017) 40. doi:10.1038/nphoton.2016.214.
[6] K. C. Cox, D. H. Meyer, P. K. Fatemi, P. D. Kunz, Quantum-Limited Atomic Receiver in the Electrically Small Regime, Physical Review Letters 121 (2018) 110502. doi:10.1103/PhysRevLett.121.110502.
[7] L. A. Downes, A. R. MacKellar, D. J. Whiting, C. Bourgenot, C. S. Adams, K. J. Weatherill, Full-Field Terahertz Imaging at Kilohertz Frame Rates Using Atomic Vapor, Physical Review X 10 (2020) 011027. doi:10.1103/PhysRevX.10.011027.
[8] M. Saffman, Quantum computing with atomic qubits and Rydberg interactions: Progress and challenges, J. Phys. B: At. Mol. Opt. Phys. 49 (2016) 201001. doi:10.1088/0953-4075/49/20/201001.
[9] C. Gross, I. Bloch, Quantum simulations with ultracold atoms in optical lattices, Science 301 (2003) 995. doi:10.1126/science.1083387.
[10] H. Labuhn, D. Barredo, S. Ravets, S. de Léséleuc, T. Macrì, T. Lahaye, A. Browaeys, Tunable two-dimensional arrays of single Rydberg atoms for realizing quantum ising models, Nature 534 (2016) 667. doi:10.1038/nature18274.
[11] H.Bernien, S. Schwartz, A. Keesling, H. Levine, A. Omran, H. Picher, S. Choi, A. S. Zibrov, M. Endres, M. Greiner, V. Vuletić, M. D. Lukin, Probing many-body dynamics on a 51-atom quantum simulator, Nature 551 (2017) 579. doi:10.1038/nature24622.
[12] I. S. Madjarov, J. P. Covey, A. L. Shaw, J. Choi, A. Kale, A. Cooper, H. Picher, V. Schkolnik, J. R. Williams, M. Endres, High-fidelity entanglement and detection of alkaline-earth Rydberg atoms, Nature Physics (2020). doi:10.1038/s41567-020-0903-z.
[13] J. D. Pritchard, D. Maxwell, A. Gauguet, K. J. Weatherill, M. P. A. Jones, C. S. Adams, Cooperative atom-light interaction in a blockaded Rydberg ensemble, Phys. Rev. Lett. 105 (2010) 193603. doi:10.1103/PhysRevLett.105.193603.
[14] Y. O. Dudin, A. Kuzmich, Strongly Interacting Rydberg Excitations of a Cold Atomic Gas, Science (New York, N.Y.) 336 (2012) 887. doi:10.1126/science.1217901.
[15] T. Peyronel, O. Firstenberg, Q.-Y. Liang, S. Ho, C. S. Adams, J. D. Pritchard, J. P. Shaffer, Special Issue on Interacting Rydberg Atoms, Journal of Physics B: Special Issue on Interacting Rydberg Atoms.
[16] H. Pichler, V. Schkolnik, J. R. Williams, M. Endres, High-fidelity entanglement and detection of alkaline-earth Rydberg atoms, Nature Physics (2020). doi:10.1038/s41567-020-0903-z.
[17] H. Busche, P. Huillery, S. W. Ball, T. Ilieva, M. P. A. Jones, C. S. Adams, Contactless nonlinear optics mediated by long-range Rydberg interactions, Nature Communications (2017). doi:10.1038/ncomms14058.
[18] C. S. Adams, J. D. Pritchard, J. P. Shaffer, Rydberg atom quantum technologies, Journal of Physics B: Atomic and Molecular Physics 53 (2020) 012002. doi:10.1088/1361-6455/ab52ef.
[19] M. P. Jones, L. G. Marcassa, J. P. Shaffer, Special Issue on Rydberg atom physics, Journal of Physics B: Atomic, Molecular and Optical Physics 50 (2017) 060202. doi:10.1088/1361-6455/aa5d06.
[20] Special Issue on Interacting Rydberg Atoms, Journal of Physics B: Atomic, Molecular and Optical Physics 53 (2020). doi:10.1038/s41567-020-0903-z.
[21] N. Šibač, J. D. Pritchard, C. S. Adams, K. J. Weatherill, ARC: An open-source library for calculating properties of alkali Rydberg atoms, Computer Physics Communications 220 (2017) 319–331. doi:10.1016/j.cpc.2017.06.015.
[22] S. Fritzschke, A fresh computational approach to atomic structures, processes and cascades, Computer Physics Communications 240 (2019) 1–
Table B.1: Tabulated values of the Rydberg-Ritz parameters for $^{88}\text{Sr}$ and $^{174}\text{Yb}$. Here $I_{\text{Sr}} = 45\,932.2002\,\text{cm}^{-1}$ and $R_{\text{Yb}} = 109\,736.631\,\text{cm}^{-1}$ [46] and $I_{\text{Yb}} = 50443.07041\,\text{cm}^{-1}$ and $R_{\text{Yb}} = 109\,736.96595\,\text{cm}^{-1}$ [53]. Lower (cl.) and upper (cl.) confidence limits are defined such that 68% of the values obtained by bootstrap resampling and fitting procedure lie within the interval $[\delta_{l} - \delta_{u}, \delta_{l} + \delta_{u}]$.

| Series | $I_{\text{Sr}}$ | $I_{\text{Yb}}$ | Refs |
|--------|-----------------|-----------------|------|
| $^{1}\text{S}_{0}$ | $3.26912 \times 10^{-5}$ | $6 \times 10^{-5}$ | $[45, 39]$ |
| $^{3}\text{S}_{1}$ | $3.370773 \times 10^{-6}$ | $4 \times 10^{-6}$ | $[41, 46]$ |
| $^{1}\text{P}_{1}$ | $2.724 \times 0.002$ | $0.006$ | $[45, 42, 41]$ |
| $^{3}\text{P}_{0}$ | $2.8867 \times 10^{-4}$ | $9 \times 10^{-4}$ | $[43, 41]$ |
| $^{3}\text{P}_{1}$ | $2.8826 \times 10^{-4}$ | $9 \times 10^{-4}$ | $[38, 41]$ |
| $^{3}\text{P}_{2}$ | $2.882 \times 0.003$ | $0.002$ | $[38, 41]$ |
| $^{1}\text{D}_{2}$ | $2.3847 \times 10^{-4}$ | $6 \times 10^{-4}$ | $[44, 40]$ |
| $^{3}\text{D}_{3}$ | $2.6752 \times 10^{-5}$ | $7 \times 10^{-5}$ | $[44, 40]$ |
| $^{3}\text{D}_{4}$ | $2.66149 \times 10^{-5}$ | $10 \times 10^{-5}$ | $[44, 40]$ |
| $^{1}\text{F}_{2}$ | $2.66 \times 0.04$ | $0.03$ | $[44, 40]$ |
| $^{3}\text{F}_{3}$ | $0.0909 \times 10^{-4}$ | $18 \times 10^{-4}$ | $[44, 40]$ |
| $^{3}\text{F}_{4}$ | $0.120 \times 0.001$ | $0.004$ | $[44, 40]$ |
| $^{3}\text{F}_{5}$ | $0.119 \times 0.003$ | $0.004$ | $[44, 40]$ |
| $^{3}\text{F}_{6}$ | $0.120 \times 0.002$ | $0.003$ | $[44, 40]$ |

Table C.1: Class and function listing of the AlkalAtom module.

| Name (parameters) | Short description |
|-------------------|-------------------|
| AlkalAtoms(\text{preferQuantumDefects, cpp\_numerov}) | Implements general calculations for alkalii atoms (see Table C.2) |
| NumerovBack(\text{innerLimit, outerLimit, \text{Kfun}, \ldots}) | Full Python implementation of Numerov integration |
| saveCalculation(\text{calculation, fileName}) | Saves calculation for future use |
| loadSavedCalculation(\text{fileName}) | Loads previously saved calculation |
| printStateString(\text{i, \text{j}}) | Returns state spectroscopic label for \text{i, \text{j}} |

14. doi:10.1016/j.epc.2019.01.012. URL: https://github.com/OpenJAC/JAC.jl.

[22] S. Weber, C. Teep, H. Menke, A. Urvoy, O. Förstenberg, H. P. Bacher, F. Hofferberth, Tutorial: Calculation of Rydberg interaction potentials, J. Phys. B: At. Mol. Opt. Phys. 50 (2017) 133001. doi:10.1088/1361-6455/aa743a.

[29] B. J. DeSalvo, J. A. Mann, F. B. Dunning, T. C. Killian, H. R. Sadeghpour, S. Yoshida, J. Burgdörfer, Ultra-long-range Rydberg molecules in a divalent atomic system, Physical Review A - Atomic, Molecular, and Optical Physics 92 (2015) 031403(R). doi:10.1103/PhysRevA.92.031403.

[30] F. Robicheaux, Calculations of long range interactions for $^{87}\text{Sr}$ Rydberg states, J. Phys. B: At. Mol. Opt. Phys. 52 (2019) 244001. doi:10.1088/1361-6455/ab4c82.

[38] K. C. Younge, B. Kuhn, S. Yoshida, J. Burgdörfer, Tutorial: Calculation of Rydberg interaction potentials, Physical Review A 100 (2019) 022503. doi:10.1103/PhysRevA.100.022503.

[45] D. Barredo, V. Lienhard, P. Scholl, S. de Léséelée, T. Boulanger, A. Browaeys, T. Lahaye, Three-dimensional trapping of individual Rydberg atoms in ponderomotive bottle beam traps, Phys. Rev. Lett. 124 (2020) 023201. doi:10.1103/PhysRevLett.124.023201.

[43] N. Šibalić, E. J. Roberson, J. D. Pritchard, R. M. Potvliege, M. P. A. Jones, C. S. Adams, K. J. Weatherill, ARC documentation.
| Name (parameters) | Short description (units) | Typical rel. accuracy |
|-------------------|---------------------------|------------------------|
| getDipoleMatrixElement(n, l, j, ...) | Reduced dipole matrix element ($a_{ij}$) | $\sim 10^{-2}$ |
| getDipoleMatrixElementHFS(n, l, j, ...) | Hyperfine-structure resolved transitions ($a_{ij}$) | $\sim 10^{-2}$ |
| getTransitionWavelength(n, l, j, ...) | Calculated transition wavelength in vacuum ($\lambda$) | $\sim 10^{-6}$ |
| getTransitionFrequency(n, l, j, ...) | Calculated transition frequency ($\nu$) | $\sim 10^{-6}$ |
| getRabiFrequency(n, l, j, m, ...) | Returns a Rabi frequency (angular, i.e. $\Omega = 2\pi \nu$ for resonant excitation with a specified laser field beam in the center of TEM00 mode (rad s$^{-1}$)) | $\sim 10^{-2}$ |
| getRabiFrequency2(n, l, j, m, ...) | Returns a Rabi frequency (angular, i.e. $\Omega = 2\pi \nu$ for resonant excitation with a specified electric field driving amplitude (rad s$^{-1}$)) | $\sim 10^{-2}$ |
| getStateLifetime(n, l, j, ...) | Returns the lifetime of the state ($\tau$) | $\sim 10^{-2}$ |
| getReductionMatrixElementJ asymmetric(n, l, j, ...) | Reduced matrix element in J basis, defined in asymmetric notation ($a_{ij}$) | $\sim 10^{-2}$ |
| getReductionMatrixElementL(n, l, j, ...) | Reduced matrix element in L basis, symmetric notation ($a_{ij}$) | $\sim 10^{-2}$ |
| getRadialMatrixElementJ(n, l, j, ...) | Radial part of the dipole matrix element ($a_{ij}$) | $\sim 10^{-2}$ |
| getQuadropoleMatrixElementJ(n, l, j, ...) | Radial part of the quadrupole matrix element ($a_{ij}$) | $\sim 10^{-2}$ |
| getEnergy(temperature) | Vapour pressure at a given temperature (Pa) | $\sim 1 (5) \times 10^{-2}$ |
| getNumberDensity(temperature) | Atom number density at a given temperature (m$^{-3}$) | $\sim 1 (5) \times 10^{-2}$ |
| getAverageInteratomicSpacing(...)| Returns the average inter-atomic spacing in the atomic vapour (m) | $\sim 1 (5) \times 10^{-2}$ |
| corePotential(l, t) | $\bullet$ core potential felt by the valence electron (a.u) | $\sim 10^{-6}$ |
| effectiveCharge(l, t) | $\bullet$ effective charge of the core felt by the valence electron (a.u) | $\sim 10^{-6}$ |
| radialWavefunction(l, s, j, ...) | $\bullet$ radial part of the electron wave function | $\sim 10^{-6}$ |
| getEnergy(l, j) | Energy of the level relative to the ionisation level (eV) | $\sim 10^{-6}$ |
| getZeemanEnergyShift(l, j, ...) | $\bullet$ Returns the linear (paramagnetic) Zeeman shift (J) | $\sim 10^{-6}$ |
| getQuantumDefect(l, j) | Quantum defect of the level | $\sim 10^{-6}$ |
| getCation(n, l, j, n1, l1, j1, ...) | $C_{a}$ interaction term for the given two pair-states ($h \times Hz m^{2}$) | $\sim 10^{-6}$ |
| getCation2(n, l, j, n1, l1, j1, ...) | $C_{a}$ interaction term for the given two pair-states ($h \times Hz m^{2}$) | $\sim 10^{-6}$ |
| getEnergyDeficit(n, l, j, n1, l1, j1) | Energy defect for the given two pair-states, $E(|n, l, j\rangle) - E(|n1, l1, j1\rangle)$ (eV) | $\sim 10^{-6}$ |
| getEnergyDeficit2(n, l, j, n1, l1, j1) | Energy defect for the given two pair-states, $E(|n, l, j\rangle) - E(|n1, l1, j1\rangle)$ (eV) | $\sim 10^{-6}$ |
| updateDipoleMatrixElementsFile() | Updates the file with pre-calculated dipole matrix elements | $\sim 10^{-6}$ |
| getRadialCoupling(energy level) | Returns the radial part of the coupling between two states (dipole, quadrupole) (rad), (a.u) | $\sim 10^{-6}$ |
| getAverageSpeed(temperature) | Average one-dimensional speed at a given temperature (m/s) | $\sim 10^{-6}$ |
| getLiteratureDME(n, l, j, ...) | Returns literature information on requested transition | $\sim 10^{-6}$ |
Table C.3: Class listing of the alkali_atom_data module. All these classes inherit properties of alkali_atom_functions.AlkaliAtom from Table C.2.

| Name (parameters)                      | Short description |
|----------------------------------------|-------------------|
| Hydrogen([preferQuantumDefects, cpp_numerov]) | Properties of hydrogen atoms |
| Lithium6([preferQuantumDefects, cpp_numerov]) | Properties of lithium-6 atoms |
| Lithium7([preferQuantumDefects, cpp_numerov]) | Properties of lithium-7 atoms |
| Sodium([preferQuantumDefects, cpp_numerov]) | Properties of sodium-23 atoms |
| Potassium39([preferQuantumDefects, cpp_numerov]) | Properties of potassium-39 atoms; alias Potassium(...) |
| Potassium40([preferQuantumDefects, cpp_numerov]) | Properties of potassium-40 atoms |
| Potassium41([preferQuantumDefects, cpp_numerov]) | Properties of potassium-41 atoms; alias Rubidium(...) |
| Rubidium85([preferQuantumDefects, cpp_numerov]) | Properties of rubidium-85 atoms; alias Rubidium(...) |
| Rubidium87([preferQuantumDefects, cpp_numerov]) | Properties of rubidium-87 atoms |
| Caesium([preferQuantumDefects, cpp_numerov]) | Properties of caesium-133 atoms; alias Cesium(...) |

Table C.4: Class listing of the divalent_atom_data module. All these classes inherit the properties of divalent_atom_functions.DivalentAtom from Table C.2.

| Name (parameters)                      | Short description |
|----------------------------------------|-------------------|
| Strontium88([preferQuantumDefects])    | Properties of strontium-88 atoms |
| Calcium40([preferQuantumDefects])      | Properties of calcium-40 atoms |
| Ytterbium174([preferQuantumDefects])   | Properties of ytterbium-174 atoms |

[53] H. Lehec, A. Zaliani, W. Maineult, E. Luc-Koenig, P. Pillet, P. Cheinet, F. Niyazi, T. F. Gallagher, Laser and microwave spectroscopy of even-parity Rydberg states of neutral ytterbium and multichannel-quantum-defect-theory analysis, Physical Review A 98 (2018) 062506. doi: 10.1103/PhysRevA.98.062506.

[54] B. Oumarej, J. Picart, N. T. Minh, J. Chapelle, New and rapid method for calculation of electric dipole and quadrupole radial integrals between Rydberg states, Physical Review A 37 (1988) 1885. doi: 10.1103/physreva.37.1885.

[55] C. L. Vaillant, M. P. a. Jones, R. M. Potvliege, Multichannel quantum defect theory of strontium bound Rydberg states. Journal of Physics B: Atomic, Molecular and Optical Physics 47 (2014) 155001. doi: 10.1088/0953-4075/47/19/199601.

[56] A. Gaj, A. T. Knipp, J. B. Balewski, R. Löw, S. Hofferberth, T. Pfau, From molecular spectra to a density shift in dense Rydberg gases., Nature communications 5 (2014) 4546. doi: 10.1038/ncomms5546.

[57] M. Fichet, F. Schuller, D. Bloch, M. Duchoy, Van der Waals interactions between exited-state atoms and dispersive dielectric surfaces, Physical Review A 51 (2) (1995) 1553–1564. doi: 10.1103/PhysRevA.51.1553.

[58] J. Mitroy, M. S. Safronova, C. W. Clark, Theory and applications of atomic and ionic polarizabilities, Journal of Physics B: Atomic, Molecular and Optical Physics 43 (2010) 202001. doi: 10.1088/0953-4075/43/20/202001.

[59] S. K. Dutta, J. R. Guest, D. Feldbaum, A. Walz-Flannigan, G. Räthel, Ponderomotive optical lattice for rydberg atoms, Physical Review Letters 85 (26) (2000) 5551. doi: 10.1103/PhysRevLett.85.5551.

[60] F. Le Kien, P. Schneeweiss, A. Rauschenbeutel, Dynamical polarizability of atoms in arbitrary light fields: General theory and application to cesium, European Physical Journal D 67 (2013) 92. doi: 10.1140/epjd/e2013-30729-x.

[61] M. G. Kozlov, S. G. Porsey, M. S. Safronova, I. I. Tupitsyn, CI-MBPT: A package of programs for relativistic atomic calculations based on a method combining configuration interaction and many-body perturbation theory, Computer Physics Communications 195 (2015) 199. doi: 10.1016/j.cpc.2015.05.007.

[62] T. F. Gallagher, Rydberg atoms, Cambridge University Press, 1994. doi: 10.1017/CBO9780511524530.

[63] W. Bowden, R. Hobson, P. Gill, I. R. Hill, P. H. Huillery, M. P. A. Jones, Rydberg Spectroscopy for DC Stark Shift Characterisation in Optical Lattice Clocks, in: 2017 Joint Conference of the European Frequency and Time Forum and IEEE International Frequency Control Symposium (EFTF/IFCS), 2017, p. 154.

[64] C. L. Vaillant, R. M. Potvliege, M. P. Jones, Intercombination effects in resonant energy transfer, Physical Review A 92 (2015) 042705. doi: 10.1103/PhysRevA.92.042705.

[65] C. L. Vaillant, Long-range Interactions in One- and Two-Electron Rydberg Atoms, Ph.D. thesis, University of Durham (2014).

URL etheses.dur.ac.uk/10594
Table C.5: Method listing of the calculations_atom_single.LevelPlot(atomType) class.

| Name (parameters) | Short description          |
|-------------------|-----------------------------|
| makeLevel(nFrom, nTo, lFrom, lTo) | Constructs an energy level diagram in a given range |
| drawLevel()       | Draws a level diagram plot  |
| showPlot()        | Shows a level diagram plot  |

Table C.6: Method listing of the calculations_atom_single.StarkMap(atom) class.

| Name (parameters) | Short description          |
|-------------------|-----------------------------|
| defineBasis(n, l, j, mj, nMin, ...) | Initializes a basis of states around the state of interest |
| diagonalizeFieldList(..., ...) | Finds atom eigenstates in a given electric field |
| plotLevelDiagram(units, ...) | Makes a plot of a Stark map of energy levels |
| showPlot(interactive) | Shows plot made by plotLevelDiagram |
| savePlot(filename) | Saves plot made by plotLevelDiagram |
| exportData(fileBase[, exportFormat]) | Exports StarkMap calculation data |
| getPolarizability(maxField, ...) | Returns the polarizability of the state (MHz cm²/V²) |
| getState(state, electricField, ...) | ♦ Returns the state composition for the state with a largest contribution of a target state in given E-field |

Table C.7: Method listing of the calculations_atom_single.Wavefunction() class.

| Name (parameters) | Short description          |
|-------------------|-----------------------------|
| getC3contribution(n1, n2, j1, ...) | ♦ Contribution to C³ of the | (J-m⁴) |
| getStateC3(n, l, j, [s, ...]) | ♦ van der Waals atom-surface interaction coefficient for a given state (J-m⁻³) |

Table C.8: Method listing of the calculations_atom_single.AtomSurfaceVdW() class.

| Name (parameters) | Short description          |
|-------------------|-----------------------------|
| getC3contribution(n1, n2, j1, ...) | ♦ Contribution to C³ of the | (J-m⁴) |
| getStateC3(n, l, j, [s, ...]) | ♦ van der Waals atom-surface interaction coefficient for a given state (J-m⁻³) |

Table C.9: Method listing of the calculations_atom_single.OpticalLattice1D() class.

| Name (parameters) | Short description          |
|-------------------|-----------------------------|
| getRecoilEnergy() | ♦ Recoil energy for atoms in given optical lattice (J) |
| getTrappingFrequency(...) | ♦ Atom's trapping frequency for a given trap depth (Hz) |
| defineBasis(Limit) | ♦ Define the basis for a Bloch band calculation |
| diagonalize(...) | ♦ Calculates energy levels (Bloch bands) for a given list of quasimomenta |
| plotLevelDiagram() | ♦ Plots energy level diagram (Bloch bands) |
| BlochWavefunction(...) | ♦ Bloch wave function as a function of 1D coordinate |
| getWannierFunction(x1, ...) | ♦ Gives value of a Wannier function |

Table C.10: Method listing of the calculations_atom_single.DynamicPolarizability() class.

| Name (parameters) | Short description          |
|-------------------|-----------------------------|
| defineBasisMin(nMin, ...max) | ♦ Defines basis for the calculation of the dynamic polarizability |
| getPolarizability(...) | ♦ Calculates the scalar and tensor polarizabilities |
| plotPolarizability(...) | ♦ Plots the polarizability for a range of wavelengths |

Table C.11: Class listing of the material module. Each class inherits the abstract class OpticalMaterial() that implements the literature input of data, while getN in each specific class has arguments according to the requirements of the material (e.g. multiple axes of reflection etc.).

| Name (parameters) | Short description          |
|-------------------|-----------------------------|
| Air(...) | ♦ Air at normal conditions as an optical material |
| Sapphire(...) | ♦ Sapphire as an optical material |
Table C.12: Method listing of the `calculations_atom_pairstate.PairStateInteractions` class that calculates the Rydberg level diagram (a “spaghetti diagram”) for the given pair-state. The ♦ symbols for this class indicate significant additions of functionality: `PairStateInteractions` now supports arbitrary inter-species pair-state calculations, and `getC6perturbatively` supports degenerate perturbation calculations.

| Name (parameters) | Short description |
|-------------------|-------------------|
| `defineBasis`     | Finds the relevant states in the vicinity of the given pair-state |
| `getC6perturbatively` | ♦ Calculates $C_6$ coefficients from second order [degenerate] perturbation theory (GHz µm$^6$) |
| `getLeRoyRadius()` | ♦ Returns the Le Roy radius for the initial pair-state (µm) |
| `diagonalise(rangeR)` | Finds eigenstates in atom pair basis |
| `plotLevelDiagram(...)` | Plots a pair-state level diagram |
| `showPlot([interactive])` | Shows the level diagram printed by `plotLevelDiagram` |
| `exportData(fileBase[, ...])` | Exports `PairStateInteractions` calculation data |
| `getC6fromLevelDiagram(...)` | Finds the $C_6$ coefficient for the original pair-state (GHz µm$^6$) |
| `getC3fromLevelDiagram(...)` | Finds the $C_3$ coefficient for the original pair-state (GHz µm$^3$) |
| `getVdwFromLevelDiagram(...)` | Finds the $r_{vdW}$ coefficient for the original pair-state (µm) |

Table C.13: Method listing of the `calculations_atom_pairstate.StarkMapResonances` class that calculates pair-state Stark maps for finding resonances.

| Name (parameters) | Short description |
|-------------------|-------------------|
| `findResonances(nMin, ...)` | Finds near-resonant dipole-coupled pair-states |
| `showPlot([interactive])` | Plots a Stark map for the initial state and its dipole-coupled resonances |

Table C.14: Function and class listing of the `wigner` module providing support for angular element calculations

| Name (parameters) | Short description |
|-------------------|-------------------|
| `CG(j1,m1,...)` | returns a Clebsch-Gordan (CG) coefficient |
| `Wigner3j(j1,j2,...)` | returns a Wigner 3j-coefficient |
| `Wigner6j(j1,j2,...)` | returns a Wigner 6j-coefficient |
| `wignerDmatrix(theta,phi)` | Class for obtaining Wigner D-matrices |
Revised version of PROGRAM SUMMARY

Program Title: ARC 3.0
CPC Library link to program files: https://dx.doi.org/10.17632/c4z4n2cdf7.1
Licensing provisions: BSD-3-Clause
Programming language: Python
External Routines: NumPy [1], SciPy [1], Matplotlib [2], SymPy [3], LmFit [4]
Nature of problem: The calculation of atomic properties of alkali and divalent atoms including energies, Stark shifts and dipole-dipole interaction strengths using matrix elements evaluated through a variety of means.
Solution method: Dipole matrix elements are calculated using an analytical semiclassical approximation or wave functions obtained by numerical integration of the radial Schrödinger equation for a one-electron model potential. Interaction energies and shifts due to external fields are calculated using second order degenerate perturbation theory or exact diagonalisation of the interaction Hamiltonian, yielding results valid even at large external fields or small interatomic separation.
Additional comments including restrictions and unusual features: External electric and magnetic field must be parallel to the quantization axis. The accuracy of short range (∼1 µm) atom - atom interaction potentials is limited by the truncation of the basis. Only weak magnetic fields are supported as only linear Zeeman shifts are taken into account. Calculations for divalent atoms use a single-electron approximation and calculation of their wave functions is not supported.

References
[1] T.E. Oliphant, Comput. Sci. Eng. 9 (2007) 10. http://www.scipy.org/.
[2] J.D. Hunter, Comput. Sci. Eng. 9 (2007) 90. http://matplotlib.org/.
[3] A. Meurer et al., PeerJ Comput. Sci. 3 (2017) e103. https://doi.org/10.7717/peerj-cs.103
[4] M. Newville et al., LmFit/lmfit-py 1.0.0 (Version 1.0.0), Zenodo (2019). http://doi.org/10.5281/zenodo.3588521
Declaration of interests

X The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: