CamCASP **5.9**

Alston J. Misquitta[†] and Anthony J. Stone[††]

[†]*Department of Physics and Astronomy, Queen Mary, University of London, 327 Mile End Road, London E1 4NS*
[††] *University Chemical Laboratory, Lensfield Road, Cambridge CB2 1EW*

November 14, 2016

Abstract

CamCASP is a suite of programs designed to calculate molecular properties (multipoles and frequency-dependent polarizabilities) in single-site and distributed form, and interaction energies between pairs of molecules, and thence to construct atom–atom potentials. The CamCASP distribution also includes the programs Pfit, Casimir, Gdma 2.2, Cluster, and Process.

Contents

1 Introduction **1**
1.1 Authors . . . 1
1.2 Citations . . . 1

2 What's new? **2**

3 Outline of the capabilities of CamCASP and other programs **5**
3.1 CamCASP limits . . . 7

4 Installation **7**
4.1 Building CamCASP from source . . . 9

5 Using CamCASP **10**
5.1 Workflows . . . 10
5.2 High-level scripts . . . 11
5.3 The runcamcasp.py script . . . 12
5.4 Low-level scripts . . . 13

6 Data conventions **14**

7 CLUSTER: Detailed specification **14**
7.1 Prologue . . . 15
7.2 Molecule definitions . . . 15
7.3 Geometry manipulations and other transformations . . . 16
7.4 Job specification . . . 18
7.5 Energy . . . 23
7.6 Crystal . . . 25
7.7 `ORIENT` . . . 26
7.8 Finally, ... 29

8 Examples **29**
8.1 A SAPT(DFT) calculation . . . 29
8.2 An example properties calculation . . . 30
8.3 Dispersion coefficients . . . 35
8.4 Using CLUSTER to obtain the dimer geometry . . . 37

9 CamCASP program specification **39**
9.1 Global data . . . 39
9.2 Molecule definition . . . 40
9.3 The EDIT module: Modifying a molecular specification . . . 42
9.4 Density-fitting . . . 43
9.5 Propagator settings . . . 45
9.6 Quadrature settings . . . 49
9.7 The `ISA` module . . . 49
9.8 Multipole moments . . . 52
9.9 Display . . . 54
9.10 Polarizabilities . . . 55
9.11 Lattice . . . 58
9.12 Numerical integration grid . . . 59
9.13 Electrostatic interaction . . . 60
9.14 First-order exchange (exchange-repulsion) . . . 61
9.15 Second-order induction energy . . . 61
9.16 Second-order dispersion energy . . . 64
9.17 Energy scan . . . 64
9.18 Overlap model . . . 67
9.19 Integrals . . . 71

10 PROCESS: Syntax **73**

11 CASIMIR **77**

A Basis sets **79**

B Dispersion coefficients: in detail **80**

C Older Scripts **87**
C.1 High-level scripts 87
C.2 Low-level scripts 87

1 Introduction

CamCASP is a suite of programs for the calculation of interaction energies between pairs of molecules, and molecular properties (multipoles and frequency-dependent polarizabilities) in single-site and distributed form. The CamCASP distribution also includes the programs Pfit, Casimir, Gdma 2.2, Cluster, and Process, and together these form a package for the *ab initio* generation of site–site force fields between organic molecules containing up to about 60 atoms.

1.1 Authors

The CamCASP suite of programs, which includes Pfit, Casimir, Gdma 2.2, Process, and Cluster, has been written by Alston J. Misquitta and Anthony J. Stone with important contributions from Robert Bukowski, Wojciech Cencek, the GAMESS(US) team and the GAUSSINT team.

1.2 Citations

This code is provided as a service to the scientific community and our only recompense, such as it is, is in citations. Therefore, if you use any results from CamCASP in your publications we request that you cite the following papers. The choice of citations would depend on the parts of the code you have used for the published results.

- **SAPT(DFT) energies**
  - A. J. Misquitta and K. Szalewicz. Intermolecular forces from asymptotically corrected density functional description of monomers. *Chem. Phys. Lett.*, 357:301–306, 2002
  - A. J. Misquitta, B. Jeziorski, and K. Szalewicz. Dispersion energy from density-functional theory description of monomers. *Phys. Rev. Lett.*, 91:33201, 2003
  - A. J. Misquitta and K. Szalewicz. Symmetry-adapted perturbation-theory calculations of intermolecular forces employing density-functional description of monomers. *J. Chem. Phys.*, 122:214109, 2005
  - A. J. Misquitta, R. Podeszwa, B. Jeziorski, and K. Szalewicz. Intermolecular potentials based on symmetry-adapted perturbation theory with dispersion energies from time-dependent density-functional theory. *J. Chem. Phys.*, 123:214103, 2005
  - R. Bukowski, R. Podeszwa, and K. Szalewicz. Efficient generation of the coupled Kohn–Sham dynamic susceptibility functions and dispersion energy with density fitting. *Chem. Phys. Lett.*, 414:111–116, 2005
- **WSM polarizabilities**
  - A. J. Misquitta and A. J. Stone. Distributed polarizabilities obtained using a constrained density-fitting algorithm. *J. Chem. Phys.*, 124:024111, 2006
  - A. J. Misquitta and A. J. Stone. Accurate induction energies for small organic molecules: I. Theory. *J. Chem. Theory Comput.*, 4:7–18, 2008a
  - A. J. Misquitta, A. J. Stone, and S. L. Price. Accurate induction energies for small organic molecules. 2. Development and testing of distributed polarizability models against SAPT(DFT) energies. *J. Chem. Theory Comput.*, 4:19–32, 2008a. doi: 10.1021/ct700105f
- **WSM Dispersion models**
  - A. J. Misquitta and A. J. Stone. Dispersion energies for small organic molecules: first row atoms. *Molec. Phys.*, 106:1631 – 1643, 2008b
- **SRLO polarizabilities**
  - A. J. Misquitta and A. J. Stone. Distributed polarizabilities obtained using a constrained density-fitting algorithm. *J. Chem. Phys.*, 124:024111, 2006
  - Fazle Rob and Krzysztof Szalewicz. Asymptotic dispersion energies from distributed polarizabilities. *Chem. Phys. Lett.*, 572:146–149, 2013

- **GDMA multipole moments**
  - A. J. Stone. Distributed multipole analysis: Stability for large basis sets. *J. Chem. Theory Comput.*, 1: 1128–1132, 2005
- **Potentials & Overlap models**
  - A. J. Stone and A. J. Misquitta. Atom–atom potentials from *ab initio* calculations. *Int. Rev. Phys. Chem.*, 26:193–222, 2007
  - A. J. Misquitta, G. W. A. Welch, A. J. Stone, and S. L. Price. A first principles prediction of the crystal structure of $C_6Br_2ClFH_2$. *Chem. Phys. Lett.*, 456:105–109, 2008b
- **Charge-tranfer via regularisation**
  - A. J. Misquitta. Charge-transfer from regularized symmetry-adapted perturbation theory. *J. Chem. Theory Comput.*, 9:5313–5326, 2013. doi: 10.1021/ct400704a
- **Iterated stockholder atom (ISA)**
  - Alston J. Misquitta, Anthony J. Stone, and Farhang Fazeli. Distributed multipoles from a robust basis-space implementation of the iterated stockholder atoms procedure. *J. Chem. Theory Comput.*, 2014. doi: 10.1021/ct5008444

Furthermore, if you make any changes or additions to the code and would like to share them for inclusion in future releases, please submit the modifications to us with suitable documentation and examples.

2 What's new?

All changes are reported with respect to the previous version. The base version is 5.2.00.

Version 5.9.29

- Kernel integral evaluation about 2–3 times faster.
- ISA restarts are now possible. An ISA solution can be built from an ISA library. See §9.7 for details.
- Many low-level improvements to the code. See the ChangeLog file for details.

Version 5.8.23

There are only a few major changes in this version compared with the earlier release (5.7.00). These are

- The iterated stockholder atom (ISA) module, described in sec. 9.7, provides an implementation of the density-partitioning technique proposed by Lillestolen and Wheatley [Lillestolen and Wheatley, 2009]. The algorithm we have implemented in CamCASP is described in [Misquitta et al., 2014] and probably is one of the most stable and accurate implementations of this partitioning method currently available.
- The concept of atomic neighbourhoods is now used to make the calculation of certain types of integrals scale linearly with system size. This concept is normally inactive, but can be activated using the `EDIT` module described in sec. 9.3. At present, the only module that can benefit from this feature is the ISA module.
- The `DISPLAY` module allows the calculation of molecular and atomic iso-density surfaces. These can be visualised using the Orient program using suitable input files. This provides a powerful way of visualising the ISA atomic densities.
- The ALDA kernel used in the response calculations has been made significantly more numerically stable. Stringent tests now show that residual numerical noise arising from the numerical integration is of the order 0.01 kJ mol^{-1}.
- As always, a number of bugs have been fixed as part of this release, and, no doubt, new ones introduced.

Version 5.7.00

The numerical stability and accuracy of the code has been improved, and it is now considerably faster, particularly when calculating distributed polarizabilities. The ALDA+CHF kernel option is coded within CamCASP and no longer requires the hessian from Dalton. This is now the default, though the ALDAX+CHF option is still available.

Most of the scripts for job submission and related tasks have been rewritten in Python. The runcamcasp.py script can now be used to submit most forms of CamCASP calculation.

All CamCASP calculations can now be carried out in conjunction with the NWChem, Dalton or Gamess(US) programs.

The 'self-repulsion plus local orthogonality' (SRLO) density-fitting/distribution method of Rob & Szalewicz [Rob and Szalewicz, 2013] is available. See the `DENSITY-FITTING` module description for more details.

CamCASP now uses significantly less disk space for temporary files. As a consequence, multiple CamCASP jobs can be run on the same file system without incurring a significant performance penalty.

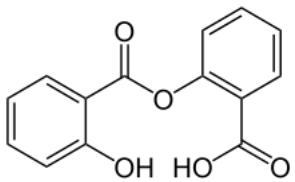
There are various bug corrections and other minor enhancements.

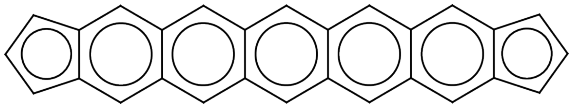
Detailed changes:

- The linear-response Kohn–Sham kernel is now calculated internally using the ALDA(+CHF) approximation with the LDA defined to contain the Slater exchange and PW91 correlation functionals. This kernel is now the default. As all integrals are evaluated internally, CamCASP no longer requires hessians from the DFT code, though this option is still available. Note that the Dalton program defines the LDA with the VWN correlation functional.
- We have made significant improvements to the numerical accuracy and stability of the response calculations. In earlier versions of CamCASP the ALDAX(+CHF) kernel could result in low levels of numerical noise in the induction and dispersion energies. This noise could be seen in weakly bound systems and could be as large as a few tenths of a $\mathrm{kJ\,mol^{-1}}$. In this version of the code this noise is at least an order of magnitude smaller.
- The computational efficiency of the distributed polarizability module has been dramatically improved. It is currently a few hundred times faster.
- The SRLO method can be used to calculate distributed, non-local polarizabilities. In this method, the site-self-repulsion constraint from Misquitta & Stone [Misquitta and Stone, 2006] is combined with an additional localization constraint by Rob & Szalewicz [Rob and Szalewicz, 2013] to result in a density-fitting-based distribution scheme that results in distributed, non-local polarizabilities with very small charge-flow terms.
- The file handling module in CamCASP has been replaced by one which is significantly more robust, simpler and faster.
- We have now provided a number of additional examples.
- Many of the high-level scripts that drive the calculations have now been replaced with Python scripts. These require Python 2.7 or later.

Version 5.6.00

The code has now been tested on fairly large systems and numerous changes have been made and bugs fixed to make these large calculations possible. Examples of recent work includes calculations of molecular properties of C_{60} and salsalate (2-(2-hydroxybenzoyl)oxybenzoic acid, structure on left), and energy scans (without exchange energies) of the pentapyralene ($C_{28}H_{16}$, structure on right) and C_{60} dimers.





While such calculations are possible with this version of the code, please see the discussion on the limitations of CamCASP in Sec. 3.1 below.

In earlier versions of CamCASP the second-order exchange-induction energy, $E^{(2)}_{\mathrm{ind,exch}}$, was calculated by scaling the uncoupled energy. We now calculate this energy without scaling. This is more accurate in strongly hydrogen-bonded systems. In practice, due to error cancellations in the second-order exchange-induction energy and the higher orders (calculated via the $\delta^{\mathrm{HF}}_{\mathrm{int}}$ approach), the two approaches lead to almost the same total exchange-induction energies. Nevertheless, we do not recommend scaling any more.

A major addition to this version is the option of regularized second-order induction energies. Through regularization we can define the second-order charge-transfer energy within SAPT or SAPT(DFT). A standard SAPT(DFT) interaction energy calculation will automatically include calculation of the regularized second-order induction and exchange-induction energies.

The Cluster program now has the capacity to perform simple energy calculations with isotropic dispersion models. While Orient remains the main program for energy evaluations, Cluster provides useful tools for creating and using dispersion models to be paired with density functionals, and allows calculations of dispersion energy and pressure contributions to crystals.

Summary of enhancements:

- Interfaces to Gamess(US) and NWChem. The interface to Dalton has been re-written and improved. All interfaces now use `ASCII` (plain text) files to store orbitals and energies and are consequently transferable across computational platforms.
- Molecular orbital space truncations are allowed. This is handled automatically in the NWChem, Dalton and Gamess(US) interfaces. Orbital-space truncations improve the stability of the SCF calculations and subsequent CamCASP calculations. They are particularly important in large basis calculations.
- `ENERGY-SCAN` is now more robust. First-order energies can be calculated in a dimer-centered auxiliary basis in the scan module. The second-order energies cannot be calculated if the scan involves molecular rotations.
- Coupled second-order exchange-induction energy, $E^{(2)}_{\mathrm{ind,exch}}$, without scaling.
- Second-order induction energies ($E^{(2)}_{\mathrm{ind,pol}}$ and $E^{(2)}_{\mathrm{ind,exch}}$) can be regularized using the R-SRS theory of Patkowski et al. [2001a].
- The `POLARIZABILITY` module allows calculations of the point responses to a point charge perturbation. This is useful for visualization of polarization in the molecule using the Orient program. See §9.10.1 for details.
- Additions to Cluster to aid the construction of dispersion models (using isotropic C_n coefficients ($n = 6 \cdots 12$)) and a variety of damping models. Dispersion energy calculations for clusters and crystals. See sections §7.5 and §7.6.
- More basis sets added. In particular, some of the NWChem basis sets are now included.
- More examples.

Version 5.5.00

Many improvements in the code. It is no longer necessary to use Dalton to do the coupled Kohn-Sham calculation — Dalton is only needed to calculate the eigenvalues and eigenvectors. The top-level scripts have been streamlined. Most calculations can be run or set up using the runcamcasp.py script, and all data affecting the calculation itself (such as ionization potentials) are provided in the cluster file. Details:

- Kernel integrals constructed in CamCASP.
- New propagator module: implements the adiabatic local density approximation ALDA(X) in the manner described by Bukowski et al. [2005].
  - There are more options in this module.
  - The PBE/AC ALDAX route is the fastest and consumes the least memory.

- PBE0/AC ALDAX+CHF is fast, but memory intensive.
- The old ALDA propagator module can still be accessed. This delivers the highest accuracy, but is slow and requires integrals from DALTON.

- 10-fold speed-up in calculating 2 and 3-centre Coulomb integrals.
- Latest version compiles with ifort. Binary is almost twice as fast as with pgf90.
- The speed of the distribution algorithm is improved by two orders of magnitude.
- Better scripts for easier use of the code.
- Energy scan for first order energies faster by a factor of 10.
- New program: res2disp added. Source code in src/res2disp/.
  Calculates dispersion energies of crystals using isotropic dispersion models.

Version 5.3.01

First-order interaction energy components can now be scanned using the `ENERGY-SCAN` module using auxiliary bases with either Cartesian or spherical GTOs. So can the overlap integrals. We will shortly be releasing a version that allows the second-order energies to be scanned using this module. We have also fixed some serious bugs in that module that affected the exchange energies.

The CLUSTER program has many modifications and a new command `MAKE-UNIQUE` that modifies all site labels so as to make them unique. The `JOIN` command now accepts multiple molecule names.

The other modules that have been modified are the `OVERLAP-MODEL` and the integral modules.

Some of the parameters of the program have been changed to allow larger systems to be used.

Version 5.3.00

There have been many enhancements to the previous version. We have also fixed a number of serious bugs. A large number of basis sets have been added to the basis set library and we encourage the user to experiment with these.

3 Outline of the capabilities of CAMCASP and other programs

The following types of calculation are possible with CAMCASP:

- **Dimer energies:**
  The first-order electrostatic and exchange energies, $E^{(1)}_{\rm elst}$ and $E^{(1)}_{\rm exch}$, and the second-order dispersion and induction energies, $E^{(2)}_{\rm ind,tot}$ and $E^{(2)}_{\rm disp,tot}$ (including their exchange terms). All are calculated using density-fitting, and the second-order dispersion can in principle be calculated without density-fitting, though this part of the code has not been tested in a long time. The $\delta^{\rm HF}_{\rm int}$ correction can also be calculated.
- **Molecular properties:**
  - *Multipole moments*: Total and distributed multipole moments can be calculated using a constrained density-fitting algorithm and the GDMA 2.2 code, which has been interfaced to CAMCASP.
  - *Frequency-dependent polarizabilities*: Total and distributed frequency-dependent polarizabilities are calculated using a constrained density-fitting algorithm. The Williams–Stone–Misquitta (WSM) method can be used to obtain the most accurate polarizability model within constraints imposed by the user.
  - *Point-to-point polarizabilities*: Responses to a frequency-dependent point-charge perturbation—called point-to-point polarizabilities—can be calculated. These are needed by the PFIT program when optimizing the distributed polarizabilities.

The ORIENT program is needed for some aspects of the property calculations. It is distributed separately but access to it is provided using the same credentials as for CAMCASP.

- **Energy scans:**
  Surfaces around a molecule can be defined (or supplied as a grid of points) and all dimer energies can be calculated on this surface, using a point charge (for the induction and electrostatics only) or a probe molecule. The total and distributed charge-density overlap is also scanned.

- **Atom–atom potentials:**
  Using the results of the energy scans and molecular properties, atom–atom potentials can be obtained using the density overlap model to model the short-range energies. The ORIENT program is used for some parts of this calculation.

- **Theory levels:**
  In all cases, calculations can be performed using either the coupled/uncoupled Kohn–Sham (CKS/UCKS) or coupled/uncoupled Hartree–Fock propagators. In practice, the CKS propagator will be used.

- **Density-fitted Integral Package:**
  The density-fitted (DF) integral package allows the calculation of 2-electron 4-index integrals in very efficiently, both in computation time and memory usage. With this package, the intermediate `TRAN` step is no longer needed for the Hessian calculation, thereby saving a lot of time. In principle, the SAPT2006 program can also be run using integrals obtained from this package, but this would involve some more effort.

  At present, the DF-integral package computes more than 30 kinds of integral.

The functionality of CAMCASP is greatly enhanced when used with the PFIT and ORIENT programs. Only some of the many features of ORIENT—those most useful for the calculation and analysis of intermolecular forces—are listed here.

- PFIT
  - *Fitting of polarizability models*: At present, polarizability models are not fitted from scratch, but polarizability models from the CAMCASP code can be optimised using PFIT and the point-to-point polarizabilities from CAMCASP.
  - *Dispersion coefficients*: These can be calculated by PFIT using frequency-dependent distributed polarizabilities from CAMCASP. This function is better performed by the CASIMIR program.

- CASIMIR
  - *Dispersion coefficients*: C_6, C_7, etc. to C_{12}. Dispersion coefficients between pairs of identical or dissimilar molecules can be calculated by CASIMIR using frequency-dependent distributed polarizabilities from CAMCASP that have been localized using ORIENT and possibly refined using PFIT. The input file for CASIMIR can be created with the PROCESS program from localized frequency-dependent polarizabilities. Minor changes to the input files allow the calculation of mixed-molecule dispersion coefficients.

- DISPERSION
  - *Dispersion energies using non-local polarizabilities*: This program allows two and three-body dispersion energy calculations using frequency-dependent non-local polarizabilities such as those calculated using the distributed polarizability module in CAMCASP. These calculations include contributions from the charge-flow polarizabilities.

- ORIENT
  - **Local polarizabilities**: The distributed polarizabilities from CAMCASP include non-local contributions. These can be transformed away using the localization module in ORIENT.
  - **Simplification of models**: The polarizability models can be modified or simplified using ORIENT.
  - **Displaying energies**: Energies can be displayed in 3-D using the OpenGL display module in ORIENT.
  - **Calculations of asymptotic energies**: Using the distributed multipoles and polarizabilities, interaction energies can be computed in the long-range (asymptotic) approximation.

Miscellaneous programs:

- PROCESS: This code is an interface code to process polarizabilities from CAMCASP or (localized/modified) polarizabilities from ORIENT into a form PFIT can use. It also performs transformations of the polarizabilities and is able to write out polarizabilities in LaTeX format for publications.
- CLUSTER: This is the program that handles the user's input data and generates the subsidiary data files needed by other parts of the package and by the auxiliary ab initio codes. It also enables the user to conduct elementary manipulations of the molecule geometry, build clusters, determine rotational and translation axes, and write output in a form suitable for CAMCASP, ORIENT, and any program that can read PDB files for imaging.

  An important function of CLUSTER is to create the input files for CAMCASP, DALTON (versions 2.0 and 2013 or later), NWCHEM and SAPT2006 programs. All the user need do is define the dimer within CLUSTER, perform any manipulations that might be needed, and then use the `RUN-TYPE` command to generate the files for the calculation. This greatly simplifies the file generation process and removes the chance of errors in the rather complicated input file structure of SAPT2006, DALTON and (to a lesser extent) NWCHEM and CAMCASP.

3.1 CAMCASP limits

The following limits apply only to those parts of the calculation that require integrals of the type `OVOV` or `OOVV`. These occur in the hybrid kernels and in the second-order exchange-dispersion and exchange-induction energies.

If only the non-exchange energies are required ($E^{(1)}_{\mathrm{elst}}$, $E^{(2)}_{\mathrm{ind,pol}}$ and $E^{(2)}_{\mathrm{disp,pol}}$) and the ALDAX kernel is used, the limits are significantly higher.

Sizes assume that individual arrays that need to be stored completely in memory in the present version of the code are limited in size to 16 GB.

- **Main basis size**: $N = n_o + n_v$ where $n_o n_v \leq 131072$
  Examples:
  - N-methylpropanamide: $n_o = 24$, therefore $N \leq 5461$.
  - Carbamazepine: $n_o = 62$, therefore $N \leq 2114$.
  - C_{60}: $n_o = 180$, so $N \leq 728$.
- **Auxiliary basis size**: $M \leq 131072$. In practice the limit will be lower for computational reasons.
- **Computational bottlenecks**:
  - *Hessians*: Calculation of the hybrid Hessians (`ALDA+CHF`) is an $O(n_o^3 n_v^3)$ process. This can be reduced using the methods recently developed by Bukowski et al. [2005]. This has been done for the `ALDAX` and `ALDA` kernels which can be calculated in $O(M^2 n_o n_v)$ effort.
  - *2-electron 4-index integrals*: These are required for the exchange energies. In particular the second-order exchange energies.
- **Other limitations**
  - Not parallelised.
  - While reasonably large calculations are possible with this version of the code (see the examples given above), bear in mind that this still is a serial code and that calculations of the second-order exchange energies for large systems will be difficult as they will typically require more resources and a re-programming of the relevant CAMCASP modules to make better use of memory and disk resources.

4 Installation

The CAMCASP package comprises several different programs, and also uses a number of third-party programs — specifically, an *ab initio* package for the DFT or wavefunction calculations and the ORIENT program. While we have interfaced CAMCASP to DALTON, NWCHEM, GAMESS(US) and (in part only) the GAUSSIAN programs, the scripts supplied with CAMCASP support DALTON and NWCHEM more fully than the others. Consequently, we recommend that the DALTON (2.0 or 2013 or later) or NWCHEM *ab initio* packages are used, and this is assumed below. Several Python scripts are provided to simplify the task of setting up the various files that are needed for the calculation,

and feeding them to the programs that do the work. The scripts require Python 2.7 or later. In order for this to work successfully, a few conventions need to be followed in the way that your computer is organised, and because of the use of third-party programs the installation procedure cannot be as fully automated as you might wish. The following instructions apply to Unix and Linux systems; they can be followed with little change for Mac OS X and Windows systems by working from a terminal window on the Mac or by using a Unix emulator such as Cygwin under Windows.

1. The first step is to unpack the CamCASP tarfile:
   ```
   tar xzf camcasp-5.9-base.tgz
   ```
   This will create a new directory `camcasp-5.9` containing the architecture-independent files, except for those needed for building the source. It doesn't include the program binaries. For these, you need to download the appropriate binary package for your architecture, and unpack that:
   ```
   tar xzf camcasp-5.9-<arch>-<compiler>.tgz
   ```
   This will unpack into the same directory, `camcasp-5.9`.

2. Next, you need to add the CamCASP `bin` directory to your path; for example:
   ```
   export PATH=/usr/local/camcasp-5.9/bin:$PATH
   ```
   This can be included in your initialization file. This would probably be your `.bashrc` file if you use the Bash shell.

3. Install Orient and either Dalton or NWChem if you don't already have them installed. Dalton-2013 and later require slightly different input from Dalton 2.0, but CamCASP can handle either.

4. You could also install the Gamess(US) program, but bear in mind that for the present, we do not include any examples of scripts to perform calculations with Gamess(US) and CamCASP.

5. Dalton-specific: The primary need for the Sapt2006 program is the patch to Dalton 2.0 included in the distribution. This allows calculations using the ALDA/ALDA+CHF kernels with xc-kernel integrals directly from Dalton and allows the Fermi–Amaldi (FA) asymptotic correction with the Tozer & Handy splicing scheme [Tozer and Handy, 1998]. The patch for Dalton needs to be applied, and Dalton recompiled if necessary. For sources for these programs see
   ```
   http://www.kjemi.uio.no/software/dalton/dalton.html
   http://www-stone.ch.cam.ac.uk/programs.html#Orient4
   http://www.physics.udel.edu/~szalewic/SAPT/SAPT.html
   ```
   Dalton 2013 and later don't require the patch, as they have the asymptotic correction already included, and CamCASP can now handle the ALDA and ALDA+CHF kernel itself.

6. NWChem-specific: The CS00 asymptotic correction [Casida and Salahub, 2000] will be used with NWChem. Ideally an energy shift should be provided for this method; it is the sum of the (positive) ionization energy and the (negative) HOMO eigenvalue. If this is not provided an empirical relationship between the HOMO eigenvalue and the IP is used instead. Neither is as good an approximation as the asymptotic correction scheme used with Dalton (above). The CS00 scheme is similar to the GRAC scheme of Gruning et al. [2001].

7. Now you need to define some environment variables.

   - `CAMCASP` should be set to the full pathname of the base directory for the CamCASP package.
   - `SCRATCH` should be set to the full pathname of a scratch directory that can be used for temporary files.
   - `DALTON` should be set to the full pathname of the directory containing the 2013 or later version of the Dalton program, if installed.
   - `DALTON2006` should be set to the full pathname of the directory containing the Dalton 2.0 program, if installed.
   - `NWCHEM` should be set to the full pathname of the directory containing the NWChem program, if installed.
   - `SAPT` should be set to the full pathname of the directory containing the Sapt2006 program, if installed.
   - `ORIENT` should be set to the full pathname of the directory containing the Orient program.

   These variables can all be set in your initialization script, so that you don't need to set them every session. For example:
   ```
   export CAMCASP=/usr/local/camcasp-5.9
   ```
   You may also also need to add the Sapt2006 `bin` directory to your path:
   ```
   export PATH=$SAPT/bin:$PATH
   ```

8. Finally you need to set up some symbolic links (aliases) in the CamCASP `bin` directory to the programs that are used by CamCASP. This only needs to be done once. To do so, run the command
   `setup.py`
   (after changing the `PATH` and setting the environment variables as explained above). The script checks that the given paths are correct. This script will also ask whether your system uses the PBS (Portable Batch System) or GE (Grid Engine) batch scheduling systems. If you intend to run all your calculations in the background, or via the Linux batch command, choose 'none'. If you use some other scheduler, you will need to modify one of the strings `header["PBS"]` or `header["GE"]` in the bin/camcasp.py script. The header will be prefixed to your job script before it is submitted. In any case, ensure that the environment variable `SCHEDULER` is either unset or set to the scheduler you require, as appropriate.

9. Testing the installation

   Once everything is set up, you can run some tests that will exercise all the programs. Go to the CamCASP subdirectory `examples` and follow the instructions in the README file. There are tests for single-configuration dimer energy calculations, for properties, and for energy scans over a set of dimer configurations.

   There are some further examples and tutorials in the Wiki at
   `https://app.ph.qmul.ac.uk/wiki/ajm:camcasp:start`

4.1 Building CamCASP from source

Executable binaries of the CamCASP, Process, Casimir, Pfit and Cluster programs are provided for PCs running Linux and for Mac OS X. Where possible these are static binaries and don't require additional libraries. If you need to compile the package yourself, you will need to obtain access to the source. This is not normally available, but may be permitted on request. The Makefile supplied with the CamCASP source files will build the CamCASP, Process, Casimir, Pfit and Cluster programs. Here are some brief instructions.

The build will happen in a directory `$CAMCASP`*/arch/compiler* where *arch* is the machine architecture and *compiler* is the name of the compiler chosen. The architecture and compiler must be specified, either on the `make` command line:
`make all ARCH=`*arch* `COMPILER=`*compiler*
by changing the beginning of the Makefile to set the default values, or by setting environment variables `ARCH` and `COMPILER`.

The compiler flags used are set in the file `$CAMCASP/`*arch*`/`*compiler*`/exe/Flags`. You will have to set machine-specific flags here. The `MACHINE` variable provides an alternative way to select flags for different platforms, using the same Flags file — see the examples provided. On a Linux machine the name of the machine should be determined automatically, but if this doesn't work you can specify it on the `make` command line. Probably the only thing that will need changing is the `LIBS` flag that sets the libraries to be linked to.

Architecture options are `x86-64` and `osx`.

Compiler options are: `pgf90`, `gfortran`, and `ifort`.

- The Portland `pgf90` compiler is reasonably fast, but doesn't always keep to the standard.
- The nagfor compiler unfortunately can't be used, as it won't accept the very old legacy routine gaussint.F, which is still used for some integral calculations.
- The `gfortran` compiler is under active development, and sometimes new bugs are introduced, but it generally works well, produces well optimized binaries and it's free.
- The `ifort` compiler generates fast code.

We recommend either `ifort` or `gfortran`.

For other compilers, you will need to make an *arch*`/`*compiler*`/exe` directory and an *arch*`/`*compiler*`/exe/Flags` file.

CamCASP needs a full installation of the Lapack and Blas libraries. It is crucial that you install a fast and reliable set of libraries. We recommend the Atlas libraries combined with the Lapack libraries from Netlib.org. The Atlas library contains a full Blas but by default only a partial Lapack installation that doesn't include all the routines used

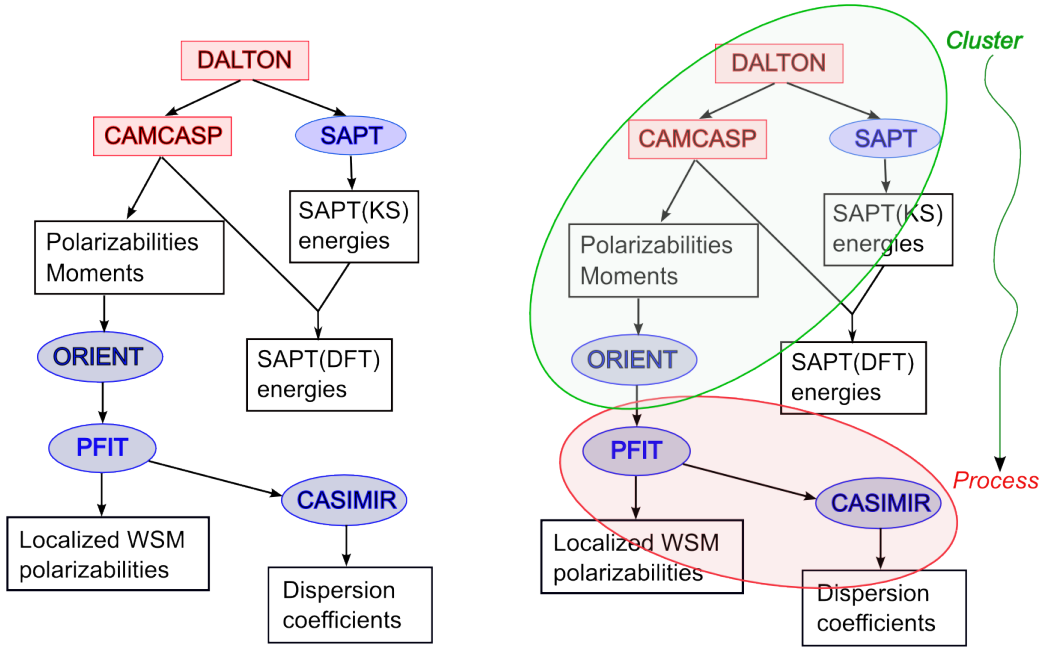


Figure 1: Workflow for CamCASP calculations. (a) Basic workflow. (b) The Cluster and Process programs, and associated scripts, generate most of the data files needed for the calculations.

by CamCASP. To include the full Lapack library, however, it is only necessary to download the source package from netlib.org and tell Atlas where it is. Instructions are provided on the Atlas webpage. It's quite easy and won't take much time. Under OS X, the Lapack and Blas libraries are available via the "-framework Accelerate" linker option, but you may wish to provide the Atlas libraries instead.

We used to recommended the GotoBLAS2 library of Kazushige Goto, available from http://www.tacc.utexas.edu/tacc-projects/, but this library is no longer under active development.

We do not recommend the Intel MKL library. This may work and could well result in faster execution times compared with Atlas, but we have encountered memory leaks associated with this library. If you use the MKL libraries and encounter errors please switch to Atlas and check to see if these errors persist.

To compile the whole package, use
`make all ARCH=`*arch* `COMPILER=`*compiler*
or just
`make all`
if you have specified `ARCH` and `COMPILER` in the Makefile or via environment variables. To compile individual components of the package, use
`make` *name* [ARCH=*arch* `COMPILER=`*compiler*]
where *name* is `camcasp`, `cluster`, `casimir`, `process` or `pfit`.

Cluster seems to pose a problem for the Portland compiler on the Opteron platform. We have not yet been able to sort this out.

5 Using CamCASP

5.1 Workflows

A properties or interaction energy calculation generally involves the use of several programs. Fig. 1(a) shows the workflow for a properties calculation. Each program requires its own input file, and these can be quite difficult to write without errors. The CamCASP package includes the Process and Cluster programs, and a number of other scripts, which can be used to generate the necessary data files and invoke the programs (Fig. 1(b)).

Examples will be provided in §8.

5.2 High-level scripts

Most CamCASP calculations can be set up and run using the high-level scripts in conjunction with one simple data file.

The following scripts are currently available. They can all be executed with the single argument `--help` to print brief usage details and exit.

The older `bash` scripts are still available and are described in Appendix C. While these are still maintained, they will probably be removed from subsequent distributions.

> The Python scripts require `python 2.7` with the `argparse` module installed. We have not tested the scripts with `python 3.x`. Since earlier versions of Python do not have the `argparse` module, these scripts will not work. In this case, consider either upgrading your Python installation, or use the older `bash` scripts as described in Appendix C. But bear in mind that the latter option is a stop-gap measure only as we will eliminate these scripts in a subsequent release.

- `runcamcasp.py`
  This python script supersedes the `runSAPT` script used previously to run a complete CamCASP calculation. It can be used to carry out calculations of the following types:

  properties: multipole moments, polarizabilities, etc. of single molecules.

  sapt-dft: symmetry-adapted perturbation theory of dimer interaction energies, using molecular wavefunctions obtained by density functional theory.

  sapt: symmetry-adapted perturbation theory using Hartree-Fock wavefunctions and perturbative correlation corrections.

  delta-hf: calculation of the δ_{HF} estimate of higher-order induction energy terms.

  supermolecule: standard supermolecule calculation of dimer energies, with counterpoise correction for BSSE.

  See below for more details.

- `execute.py`
  This script is invoked when the job is run, and carries out the Dalton or NWChem calculations, converts the resulting eigenvector files into a form convenient for CamCASP, and then runs the CamCASP calculation. All of this is done in a subdirectory of the `$SCRATCH` directory, and the output files are copied back to the specified job directory at the end of the calculation. Except in properties calculations, two or three DFT or HF calculations are needed, and `execute.py` starts them as separate processes so that they run in parallel.

- `extract_saptdft.py`
  The output of a SAPT(DFT) calculation can be analysed using the script `extract_saptdft.py`. It picks out the energy terms from the CamCASP output file and displays them in a compact form. The arguments are directories containing SAPT(DFT) calculations, as previously set up by `runcamcasp.py`. Often it is sufficient to run the command as `extract_saptdft.py *`, as the script ignores plain files and merely issues a warning for directories that don't contain completed SAPT(DFT) calculations.

- `localize.py`
  Using the polarizability files from a properties calculation, this script will generate local polarizabilities (static and dynamic). See §8.2.1 on p. 32 for details. The dynamic polarizabilities are then used to construct dispersion coefficients, if required.

- `batch_sapt.py`
  This is one of many batch scripts that can be used to schedule multiple calculations. Please see the `bin` directory for others. `batch_sapt.py` can be used to perform interaction energy calculations using a template Cluster input file and a geometry file. The other batch scripts work in a similar way. It is worth testing batch scripts before using them for a real calculation.

5.3 The runcamcasp.py script

The command in its simplest form is just
`runcamcasp.py` *JOB*
JOB is the name of the job, and is prefixed to files created for the job. By default it is also the name of the directory that is created to contain the job files, but that can be changed using the `--directory` or `-d` option. You need to provide one simple data file, normally *JOB*.clt, to define the system and specify the calculation. For details of this file see §7 and the example in §8.1 below.

The general form of this command is
`runcamcasp.py` *JOB options*
where *JOB* is the job name — that is, the file-prefix specified in the CLUSTER input file. The main options are

`--clt` *file* Specify the cluster file for the job. Default is *JOB*`.clt`.

`--directory` or `-d` *dir* Directory for the job files. If the `--restart` option is used it must exist already; otherwise it is newly created. If a directory of the same name already exists, it can be deleted with all its contents, or renamed, or the job may be aborted.

`--ifexists` `[delete | save | abort | ask]` Specify what to do if the job directory already exists. The default is to ask. If `save` is specified, the existing directory, *dir*, is renamed as *dir_nnn*, where *nnn* is the first suffix in the series 001, 002, etc., after any *dir_nnn* that already exists.

`--restart` This will start or restart the calculation in the specified directory, carrying out any DALTON or NWCHEM calculations that haven't been completed, and repeating the CAMCASP calculation regardless. This allows the calculation to be repeated with a modified CAMCASP data file *JOB*`.cks` if required.

`--queue` or `-q` *queue* The job is to be submitted to the specified queue (see below), or run in the background if "`-q bg`" is specified.

`--memory` or `-M` *nnnn* Memory required for the job in MB.

`--direct` Use direct integral management.

By default, this script uses a CLUSTER input file with the name *JOB*`.clt`. If this is not the case, use the "`--clt` *file*" option to specify the name of the CLUSTER file.

This carries out the following steps, unless `--restart` is specified:

- Create a new directory *dir* for the job. By default this is called *JOB*, but a different directory name can be given using the `-d` *dir* flag. If the directory already exists, you're given the option of either deleting that directory and its contents first, or renaming it as *dir*_`nnn`, where `nnn` is a digit string that doesn't clash with any existing file or directory. All the files for the job will now be created in the specified directory or a subdirectory of it.
- Copy the CLUSTER file into the new directory, chdir into it, and run the `cluster` program to generate the data files for the SCF calculations, and the CAMCASP input file *JOB*`.cks`.
- If the DALTON program is to be used, functions defined in `camcasp.py` are called to generate the DALTON `.mol` and `.dal` input files.

Then, whether or not `--restart` is specified:

- Call the `submit` function in `camcasp.py` to create the script to do the calculation, and to submit the job to the specified queue.

The main part of the calculation is carried out by the execute.py script. It begins by defining some directory names. The `MAINDIR` is the directory from which the job is run. The files in this directory are all copied to a scratch directory `$SCRATCH/`*JOB_str*, which is created if necessary, and the calculations are carried out there. where *str* is a unique identifier, usually the time at which the job starts. Many big files are generated in the course of the calculation, so there needs to be plenty of space available here. At the end of the calculation, the useful output files are copied back to `$MAINDIR/OUT`, and the scratch directory and its contents are deleted.

If `--restart` was specified, the `runcamcasp.py` script scans the copy of the `.clt` file in the job directory (not the original one, which may have been changed or deleted) in order to assemble information about the calculation. However no files are generated, so any changes made are retained. In particular, the `.cks` file may be modified as required before restarting. In a restarted job, completed DFT or HF calculations are not repeated, but incomplete ones are restarted. The CamCASP calculation itself is always repeated. Any files in the OUT subdirectory are normally overwritten, but "`--ifexists save`" can be used in a restarted job to rename the existing OUT subdirectory and use a new one.

5.3.1 Job queues

If your machine operates a queueing system for batch jobs, the `-q` or `--queue` flag may be used to specify the queue for the calculation. If the computer uses the PBS or GE batch queuing system, set the environment variable `SCHEDULER` to `PBS` or `GE`. Headers for these are defined in the `camcasp.py` Python module, but they may need to be changed to suit your system. The queues available will depend on your operating system. On a workstation running Linux, there is a `batch` queue, to which a series of jobs may be submitted; they will run in turn as the load on the computer permits. The queue can also be specified as "`-bg`", which causes the job to be run in the background — convenient for small jobs. If "`-q none`" is specified, the files are all set up, but the calculation isn't run immediately; instead a script `submit.bash` is left in the current directory, and you can execute this when you're ready to run the job; for example:

```
./submit.bash -q bg
```

If you regularly use a particular queue, you can set the environment variable `QUEUE` to that value, and that will become the default for the `--queue` option.

5.4 Low-level scripts

Some of the scripts and programs provided in the CamCASP package are low-level ones that just perform one step in setting up the job, for example by generating further files that are needed for the calculations. If you need to do something unusual you may have to use the lower-level scripts directly, but for most purposes the high-level scripts will suffice.

In all the following, *JOB* is the job-name, which identifies all the files belonging to the job — in particular, the cluster input file, usually called *JOB*`.clt`. Most jobs involve a large number of files, however, so it is not a good idea to rely on the job-name to distinguish files for different jobs from each other. Recommended practice is to run every job in its own private directory (folder).

The starting-point for all tasks is the 'cluster' file, conventionally given the `.clt` suffix, which contains information about the molecules involved, the basis sets to be used, and other details. Details are in §7. Once you have prepared the input file for Cluster, called for example `setup.clt`, the `cluster` command is the first step in setting up the job:

```
cluster < setup.clt
```

(It will usually be called by a script rather than directly.) If the Dalton program is to be used, it will produce a file called *JOB*`.DALtemplate`, where *JOB* is the file prefix specified (or implied) in the Cluster data file. It will also produce a number of other files — for example *JOB*`.cks` and *JOB*`_P.data` — depending on the information in the `.clt` file.

The `runcamcasp.py` script uses a number of functions defined in the `bin/camcasp.py` Python module. There are functions to generate the full data files for Dalton calculations from the *JOB*`.DALtemplate` file and to set up the shell script that is submitted to the appropriate queue to carry out the calculation. These will usually be called from the `runcamcasp.py` script, but they could be called from a Python interactive session with the `camcasp` module loaded.

For Dalton DFT calculations using the asymptotically-corrected PBE0 functional it is necessary to specify the molecular ionization potential in order to set up the asymptotic correction properly. The file `misc/IP.txt` contains a list of ionization potentials for some small molecules, and many others are available in the NIST Chemistry Webbook at webbook.nist.gov/chemistry/. If the IP cannot be found there or elsewhere in the literature, it is necessary to calculate it. The calculation requires the vertical ionization energy, i.e. the difference in energy between the neutral molecule and its cation at the same geometry.

Calculations using NWChem use a different asymptotic correction scheme. This ideally uses an energy shift which

is the sum of the (positive) ionization energy and the (negative) HOMO energy for the neutral molecule, but NWChem can estimate the shift using an empirical scheme if the shift is not specified. In our experience the asymptotic correction scheme used by Dalton is more satisfactory.

6 Data conventions

General notes on the conventions used for describing the form of input files:

- All data values are free format. Items may usually be given in any logical order.
- Comments may be included in the file. They start with "!␣" (note the space) and continue to the end of the line.
- Items in square brackets are optional. (The square brackets themselves should not be included.)
- Items separated by "|" are alternatives. If a list of such items is shown in braces, one of the alternatives must be chosen. Often several alternative keywords are recognized; for example `FREQUENCY` can be replaced by `FREQUENCIES` or `FREQ` or `FREQS`. In most cases the allowed alternatives are shown below.
- Keywords shown here in upper-case may be given in upper, lower or mixed case.
- Some keywords are shown in lower case. They too can be given in upper, lower or mixed case. These keywords are never required — they can be included in the data file to improve its human readability, but have no other purpose. They are checked for correctness but are otherwise ignored.
- Items shown in *italics* should be replaced by a suitable value.
- Long lines may be broken by replacing any space between items by "␣+++\n", where "␣" denotes space and "\n" denotes newline. This notation is also used occasionally in this manual to break lines that are too long for the page.
- It is sometimes convenient to include data from other files:
  `#include` *filename*
  If such a line occurs anywhere in the input file, the contents of the specified file are read exactly as if they had been pasted into the data file at this point.
- For files in the CamCASP installation directory, the following command may be used:
  `#include-camcasp` *path relative to* CamCASP
  This allows files to be specified *relative* to the CamCASP directory. For example, instead of specifying a basis file as
  `#include /home/ajm/CAMCASP/basis/gamess_us/sadlej/H`
  you could use
  `#include-camcasp basis/gamess_us/sadlej/H`
  For this to work correctly, the `CAMCASP` environment variable needs to have been defined.
- The SKIP command skips execution of following lines of the input file. Normal execution resumes after an UNSKIP or ENDSKIP command.
- The FLUSH command can be used to force flushing of the standard output buffer.

7 Cluster: Detailed specification

The cluster file is normally all that is needed to define a calculation, and in essence it is very simple, although it provides very versatile commands for manipulating the configuration of the system to be studied. It comprises four sections:

(i) Prologue

(ii) Molecule definitions

(iii) Geometry manipulations etc.

(iv) Job specification

We describe these sections in turn. See above for the conventions used to describe the commands.

7.1 Prologue

TITLE "*title*"
Title for the job. Optional.

```
GLOBAL
  UNITS { DEG | DEGREE | DEGREES | RAD | RADIAN | RADIANS }
  UNITS { BOHR | AU | ANG | ANGSTROM | ANGSTROMS }
  [CAMCASP path]
END
```

The UNITS command sets default units for lengths and angles, but they can be over-ridden locally if necessary. The CAMCASP entry specifies the full file path to the CamCASP base directory, but it will not usually be needed — if Cluster has been compiled with a modern (Fortran 2003) compiler it will read the path from the CAMCASP environment variable.

7.2 Molecule definitions

You can define as many molecules as you want here, and later select which of them are to be used in the actual calculation.

```
MOLECULE molecule-name
  [UNITS { BOHR | AU | ANG | ANGSTROM | ANGSTROMS }]
  [CHARGE molecular-charge]
  [I.P. ionization potential [eV] ]
  [AC-SHIFT \emph{asymptotic correction shift value}]   [ATOM-CHARGES [AUTO | USER]]
    label  Z  key <data> [TYPE type]
    label  Z  key <data> [TYPE type]
    ⋮
END
```

The UNITS line is only needed if the units for atom positions are different from the global units. CHARGE specifies the total molecular charge, default zero. I.P. specifies the ionization potential, normally in Hartree, but it can be given in *e*V if the eV unit is specified. The I.P. is needed if asymptotic correction is to be applied to the exchange-correlation potential, which is recommended. If the I.P. is omitted and Dalton is used to perform the DFT calculation, no asymptotic correction is applied. At present, no I.P. is needed to use the asymptotic correction in NWChem as the CS00 scheme used by NWChem can estimate it, but the shift (I.P.+ HOMO energy) can be supplied.

The molecular geometry follows, one line for each atom. The label is an atom identifier, up to 8 characters in length. *Z* is the nuclear charge. Positions may be given in a variety of ways; see below. The site type can be optionally specified. It is used to impose symmetry and this information is passed to other programs. Sites can be given the same type if they are related by a molecular symmetry operation that is a proper rotation. Sites that are related only by improper rotations have to be treated specially, but this situation is uncommon. The type name can be up to 8 characters long.

The nuclear charges *Z* can be skipped in the geometry specification if ATOM-CHARGES AUTO is used. In this case, the code attempts to work out the nuclear charges from the name of the atom. Sites which are not recognised will be assigned a zero nuclear charge.

Atom positions may be specified in any of the following ways. The default is to use Cartesian coordinates, and this will be the most convenient if the geometry is obtained from an *ab initio* geometry optimization. If the geometry

is given in terms of bond lengths and bond angles the other methods may be more helpful. s_1, s_2 and s_3 are labels or sequence numbers of previously-defined atoms in this molecule. Negative sequence numbers can be used, and count back from the new atom, so the most recently defined atom is -1, the previous one -2, and so on. The new atom is notionally atom 0. If names are used to identify atoms, it is best if the names are unique, but if two or more atoms have the same name, the one most recently defined will be assumed. Dummy sites, with $Z < 0$, may be included to help define the geometry. They are discarded once the molecule definition is complete.

- Absolute Cartesian coordinates:
  `[AT]` $x\ y\ z$
- Absolute polar coordinates:
  `POLAR` $r\ \theta\ \phi$
- Linear Connexion:
  `LC` $s_1\ s_2\ b$
  Place the new atom at distance b from s_1, along the extension of the line from s_2 to s_1.
- Planar Connexion:
  `PC` $s_1\ s_2\ s_3\ b\ \theta$
  Place the new atom at distance b from s_1, in the 1–2–3 plane, with bond angle 2–1–0 equal to θ, *cis* to s_3 if $\theta > 0$, *trans* if $\theta < 0$.

For the remaining position definitions it is useful to define a local coordinate system with origin at s_1, z in the direction from s_1 to s_2 and x in the 1–2–3 plane with s_3 at positive x.

- General Connexion:
  `GC` $s_1\ s_2\ s_3\ b\ \theta\ \phi$
  Place the new atom at distance b from s_1, with bond angle 2–1–0 equal to θ and torsion angle 3–2–1–0 equal to ϕ. θ and ϕ can be viewed as standard spherical polar angles in the local coordinate system.
  Alternatively, ϕ can be regarded as torsion angle 3–2–1–0, taken counter-clockwise looking from 2 to 1 with atom 0 *cis* to 3 when $\phi = 0°$. θ should be positive, but a negative value is equivalent to increasing ϕ by 180°.
- Central Connexion
  `CC` $s_1\ s_2\ s_3\ b\ \theta\ \beta$
  θ = 2–1–0 bond angle, $|\beta|$ = 3–1–0 angle. y is positive or negative in the local frame according to the sign of β.
- Ring Closure
  `RC` $s_1\ s_2\ s_3\ b\ b_2\ \phi$
  The new atom is placed at distance b from s_1 and b_2 from s_2, and ϕ is the angle between the 2–0–1 plane and the 2–3–1 plane. ϕ is again the spherical polar angle in the local frame, so if $\phi = 180°$ the new site is in the 2–3–1 plane, on the opposite side of the s_1–s_2 line from s_3.

7.3 Geometry manipulations and other transformations

None of the commands in this section are required, and indeed it is common for none of them to be needed. However they do provide powerful ways to modify the geometry of the system and other details.

For example, `FIND ROTATION` and `FIND TRANSLATION` find the rotation (axis and angle) and translation needed to go from one molecular geometry, defined by nuclear coordinates, to another. This feature has proved very useful in constructing dimers or small clusters that have the same relative positions as in the crystal.

`JOIN`, `COPY`, `INVERT`, `ROTATE`, and `TRANSLATE` operate on molecules and facilitate the assembly of clusters and macro-molecules that can then be operated on as units.

The `BOND` command is very handy for "bonding" together molecules along a specific axis (defined using coordinates or sites within a molecule).

The full list of available commands is as follows.

`UNIQUE-SITES [` *list of molecule names*`]`
The site labels need to be unique when calculating polarizabilities and dispersion coefficients. This command

modifies the site labels in the specified molecules to make them unique (within a given molecule). If no molecules are specified, this command processes all the molecules currently defined.

`{ TOL | TOLERANCE }` *tolerance*
Specifies the tolerance used in comparing geometries. Default is 10^{-6}, which is probably too tight for most purposes.

`COMPARE` *name of molecule 1* `[and]` *name of molecule 2*
If the data for a molecule have come from different sources (e.g. X-ray diffraction and ab initio optimization) it is useful to be able to check that they describe the same geometry. This command sets up, for each molecule, a distance matrix between the atoms, and compares the two, reporting any significant differences. The atoms must be listed in the same order in both cases. Note that enantiomorphs will be reported as matching.

`CENTRE` *molecule* `[ on | at ] { [SITE]` *label* `| COM | XYZ` *x y z* `}`
The centre of the specified molecule is defined to be at the position given. Initially the 'centre' is the origin of global coordinates.

`{ WRITE | PRINT }` *molecule* `[[in] [ PDB | ORIENT ] [format]]`
With the `PDB` option, this writes the molecular geometry to a PDB-format file ⟨molecule-name⟩.pdb. Otherwise the atom names and positions are printed on standard output; if `ORIENT` is specified, they are in a form suitable for pasting directly into an Orient data file. See also the `ORIENT` option of the `RUN-TYPE` command, below.

`FIND ROTATION`
`[TOLERANCE` *tol*`]`
`FROM` *triad*
`TO` *triad*
`END`
Find the rotation that would take the first triad to the position of the second. Each triad is a set of three points specified as Cartesian coordinates, typically the coordinates of three atoms:
$x_1\ y_1\ z_1 \quad x_2\ y_2\ z_2 \quad x_3\ y_3\ z_3$
or as site labels:
`SITES` $s_1\ s_2\ s_3$ `[in] { MOLECULE | MOL }` *molecule*
The triad definition may run over more than one line if necessary. The triangles defined by the two triads must be congruent to within the tolerance specified, default 0.001.

`FIND { TRANS | TRANSLATION }`
`FROM` *position*
`TO` *position*
`END`
Find the translation vector from the first position to the second. Each position may be given in Cartesian coordinates:
`XYZ` *x y z*
or as a site label:
`{ SITE` *site* `| COM } [ in ] { MOLECULE | MOL }` *molecule*

`FIND COM [ of ]` *molecule*
Find and print out the position of the centre of mass of the specified molecule. The atom masses are assumed to be those of the most abundant isotope.

`JOIN` *molecule name* `[ and | & ]` *molecule name* `...` `INTO` *new molecule name*
Combine any number of specified molecules, at their current positions, into a single entity. This is useful for creating a dimer or, more generally, an *N*-mer out of pre-defined molecules. Site types are preserved.

`INVERT` *molecule*
Invert the specified molecule w.r.t. to its centre. Note that the 'centre' is the origin of global coordinates if no centre has been defined — see the `CENTRE` command above.

`COPY` *moleculeA* `[ to ]` *moleculeB*
Define a new molecule, initially at the same position as the original.

`BOND` *moleculeA* `[ [ -> | TO ] { ` *moleculeB* ` | XYZ } ]`
`[FROM | SITES]` $a_3\ a_2\ a_1$ `{ -> | TO }` $b_1\ b_2\ b_3$
`{ DISTANCE | LENGTH | SEPARATION }` *distance* `[ ANGSTROM | ANG | BOHR ]`
`END`

This moves molecule A so that sites a_2, a_1, b_1 and b_2 are collinear and arranged in that order, and so that sites a_1 and b_1 are the specified distance apart. a_3 is a third site lying off the a_1–a_2 axis, and b_3 a third site lying off the b_1–b_2 axis. If molecule A is non-linear, it is rotated about the a_2–a_1 axis so that all six sites are coplanar, with sites a_3 and b_3 *cis* to each other. If A is linear, then a_3 is irrelevant, but at present it must still be specified and must still lie off the a_1–a_2 axis. It can be specified as a position in global coordinates using the syntax `XYZ` *x y z*. Molecule B is not moved, and indeed it need not be a molecule at all. Any or all of the B sites may be specified as cartesian positions using the syntax `XYZ` *x y z*, and if 'molecule B' is not specified or is specified as `XYZ`, then *all* of the sites of 'B' must be specified using the syntax `XYZ` *x y z*.

`ROTATE` *molecule*
Rotate the specified molecule, using the rotation found by the most recent `FIND ROTATION` command. The rotation axis runs through the centre of the molecule. Note that the 'centre' is the origin of global coordinates if no centre has been defined — see the `CENTRE` command above.

`ROTATE` *molecule* `BY` *angle* `[ DEGREES | RADIANS ] ABOUT` n_x n_y n_z
The axis of rotation need not be normalized. The rotation axis runs through the centre of the molecule.

`PLACE` *molecule* `AT` *x y z*
Position the specified molecule so that its centre is at (x, y, z) in global coordinates.

`TRANSLATE` *molecule*
Move the specified molecule by the vector translation found in a previous `FIND TRANSLATION` command.

`TRANSLATE` *molecule* `BY` *distance* `[ ANGSTROM | BOHR ] ALONG` *x y z*
The distance is assumed to be in the global units unless otherwise specified. The vector direction need not be a unit vector, but is normalized by the program.

`OVERWRITE`
Overwrite existing files that have the same names as those generated by this run of the program.

`OVERWRITE { OFF | FALSE | NO }`
Don't overwrite existing files, but use alternative file names if files with the same names as those generated by the program are found to exist already. If any such file exists, the modification is applied to all files generated by this run. This action is the default. However it is recommended to start the calculation in a clean directory to avoid any confusion – a typical calculation will generate many files.

`INTERPOLATE` *molecule1* `[AND | &]` *molecule2* `INTO` *new molecule name* `[FACTOR` ξ `]`
Given two conformers *molecule1* and *molecule2* with sites in the same order, create a third molecule with a structure that linearly interpolates between the two. Site coordinates in the interpolated molecule are defined by:

$$\mathbf{r} = \mathbf{r}_1 + \xi(\mathbf{r}_2 - \mathbf{r}_1).$$

7.4 Job specification

This section specifies the input files and scripts that are to be constructed for the calculation, as well as basis-set details and some other options.

```
RUN-TYPE
   MOLS/MOLECULE/MOL/MOLECULES <moleculeA> [[and] <moleculeB>]
   FILE-PREFIX <prefix of interface files: CASE SENSITIVE>

   [SAPT(DFT) | SAPT | {DELTAHF | DELTA-HF} | PROPERTIES | SUPERMOL | CAMCASP]
   SCFCODE [DALTON | GAMESS | NWCHEM ] [NO-SYMM]

   METHOD <Electronic structure method | DFT>
   FUNCTIONAL <functional | PBE0>  (applicable only for DFT)
   {KERNEL | KERNEL-TYPE} [ALDA | ALDA+CHF | CHF | ALDAX+CHF | ALDAX]
         To use Hessians from DALTON use: ALDA+CHF-DALTON or ALDA-DALTON
   {AC | ASYMPTOTIC-CORRECTION} [TH | LINEAR | GRAC] [CS00 | MULTPOLE | LB94]

   BASIS <basis> [of] TYPE <type of basis>  [SPHERICAL | CARTESIAN]
   TYPE <type of basis>
```

```
   {AUX-BASIS | AUX} <auxiliary basis> [TYPE <type of aux basis>]  +++
       [SPHERICAL | CARTESIAN]
   [{AUX-TYPE | AUX-BASIS-TYPE} <type of auxiliary basis>]
   ISA-AUX [SET1 | SET2 | SET3 | GISA | USR]          +++
       [MIN-S-EXP-H [=] {\em real} (default 0.0)]     +++
       [MIN-S-EXP-DUMMY [=] {\em real} (default=0.2)] +++
       [DUMMY-SYMM-LIMIT [=] {\em char} (default = S)]
   MIDBOND/MID-BOND <name of midbond basis>/NONE +++
      [and/of] TYPE WEIGHTED/COM

   ! Use these to override the default MO-file names and formats
   {MO-FILE-A | MO-FILE-B | MO-FILE} <name of MO file for molecule> +++
        [[FORMAT] {ASCII | BINARY}]

   OPTIONS [TESTS] [ISOTROPIC-POL] [NO-3RD-ORDER] [SOLVER {LU | GELSS}]
   MEMORY <max memory> BYTES/KB/MB/GB  <---default MB
   SITE-ORDER [{OLD | DALTON2006} | {NEW | DALTON2013}]
   INTEGRAL-SWITCH <Integral switch (integer)>

   CAMCASP-COMMANDS
      Use this block to insert User-defined commands for the CamCASP command
      file. These will be used (below the MOLECULE blocks) in lieu of the
      default commands. Useful if you wish to do something different. Use
      with caution, preferably by *editing* a default CamCASP command file
      rather than starting from scratch.
   END-CAMCASP-COMMANDS

   ORIENT [file | files] [for] [DISPLAY] [ENERGY] [LOCALIZATION]
   PROCESS [file | files] [for] [CASIMIR] [and] [PFIT]
   GAUSSIAN [file]
   INTERFACE [file | files]
   RANK [LOCAL <rank>] [WSM <rank>] [NONLOCAL <rank>] +++
        [HYDROGEN <rank>] [DMA <rank>] [DMA-H <rank>]
   SITES [file]
END
```

The subcommands are:

`{ MOL | MOLS | MOLECULE | MOLECULES }` *moleculeA* `[[and]` *moleculeB* `]`
Specify one or two molecules involved in the calculation. Either or both may be supermolecules or clusters constructed by joining fragments.

This command can appear any number of times. However, at present, any molecules mentioned after the first two will be ignored.

`FILE-PREFIX` *prefix*
Prefix for all the file-names to be created for this job. Default is "*molA_molB*" for a calculation involving molecules *molA* and *molB*, or "*molA_basis name*" for a single-molecule calculation. Note however that the prefix may be modified, depending on the `OVERWRITE` setting.

`{ SAPT(DFT) | DFT-SAPT | SAPT-DFT | SAPTDFT }`
Perform a SAPT(DFT) calculation. Default if two molecules have been specified.

`SAPT`
Perform a SAPT calculation. Default kernel type is set to `CHF`.

`{ DELTA-HF | DELTAHF }`
Perform a calculation of the $\delta_{\rm int}^{\rm HF}$ correction. Default kernel type is set to `CHF`.

`{ PROPERTIES | PROPERTY }`
Perform a property calculation. Default if only one molecule has been specified.

`{ SUPERMOL | SUPERMOLECULE | SUPERMOLECULAR }`
Perform a supermolecule calculation.

`METHOD` *electronic structure method*
The electronic structure method to be used in the calculation. This option is needed only for a `SUPERMOLECULE` calculation of the interaction energy. By default the method is set to `DFT`. Options are `HF`, `MP2` and `CC`.

`FUNCTIONAL` *functional*
Functional to be used in the DFT calculation of molecular orbitals. `PBE0` (default) or `PBE` is recommended, usually with asymptotic correction, which is applied automatically if non-zero ionization potentials are provided in the molecule definitions.

The type of functional is used to determine the fraction of exact exchange to be included. Consequently this is important only for hybrid functionals. At present allowed density functionals are: `PBE0 PBE HCTH HCTH147 PW91`. If the functional you require is not in this list we suggest you first create the data files for one of the allowed functionals using the standard command (`runcamcasp.py` with the `-q none` option to stop after the files have been created) and then edit the CamCASP data file to set the fraction of exact exchange include the `FRACTION-HFX` command in the `SET GLOBAL-DATA` block.

`KERNEL` *kernel*
The recommended choice, and normally the default, for the kernel is `ALDA+CHF` with the `PBE0` functional. This option is still memory intensive and is not recommended for large jobs. If you can tolerate a small reduction in accuracy, then we recommend the `ALDA` kernel with the asymptotically corrected `PBE` functional.

While the kernel integrals are normally calculated within CamCASP, it is also possible to use Dalton for these calculations, in which case more advanced kernels including gradient contructions can be used. To to calculate the kernel integrals with Dalton use `ALDA-DALTON` or `ALDA+CHF-DALTON`.

The `ALDA` kernel differs from `ALDAX` in that it includes a contribution from the LDA correlation functional (see sec. 9.5.1 for details). This could be the `VWN` or `PW91c` correlation functionals. Tests on numerous systems have shown that the lack of correlation contribution to the kernel results in a loss in accuracy of about 1% to the dipole-dipole polarizability and 1-2% to $E^{(2)}_{\mathrm{disp,tot}}$. Depending on the balance of energy components, the error made in the total interaction energy could be larger than this.

See the CamCASP program specification (p. 45) for other possibilities. If you need to use these other options, you can generate the CamCASP input file (suffix `.cks`) using the standard `runcamcasp.py` script with "`-q none`", and modify the propagator specification before running the job.

`BASIS` *basis* `[TYPE { MC | MC+ | DC | DC+ }]`
`[SPHERICAL | CARTESIAN]`
Specify the basis to use. Default `Sadlej`; other possibilities are `aug-cc-pVxZ` (or `aVxZ`) and `cc-pVxZ` (or `VxZ`), x =D, T or Q.
The basis type should be specified here (the separate command `BASIS-TYPE` is still allowed but is not compatible with the scripts). Default is `MC` (monomer-centred), meaning that the monomer basis set has basis functions only on the atom sites of the monomer. In the `MC+` case, the 'monomer' basis also has basis functions on the sites of the partner molecule (so-called 'far-bond' functions) and may also include 'mid-bond' functions in the region between the two molecules. (See the `MIDBOND` sub-command, below.) The `DC` (dimer-centred) basis has basis functions on both molecules, and `DC+` may also have mid-bond functions.

By default spherical GTOs are used in the basis set, but this can be changed using the argument `CARTESIAN`.

`{TYPE | BASIS-TYPE} { MC | MC+ | DC | DC+ }`
Specify the type of basis set to be used for a dimer calculation. We do not recommend using this command as it is incompatible with the `runcamcasp.py` and related scripts.

`MIDBOND { NONE |` *midbond* `[TYPE { COM | WEIGHTED }] }`
Specify the mid-bond basis functions to be used, if any, and their position. Default for basis-types `MC+` and `DC+` is `3s2p1d`; the only other possibilities are `3s2p1d1f` and `NONE`. `COM` specifies that the mid-bond functions are to be placed at the mid-point of the line joining the centres of mass. The `WEIGHTED` placing is more suitable for large molecules and is the default; this calculates the position of the midbond functions as [Akin-Ojo et al., 2003]

$$\mathbf{R}_{\mathrm{MB}} = \frac{\sum_{a\in A, b\in B} w_{ab}\boldsymbol{p}_{ab}}{\sum_{a\in A, b\in B} w_{ab}}, \tag{1}$$

where $\boldsymbol{p}_{ab}$ is the midpoint of the vector $\boldsymbol{r}_{ab}$ from atom a to atom b, and $w_{ab} = |\boldsymbol{r}_{ab}|^{-6}$. The inclusion of mid-bond functions has been found to improve the description of the dispersion interaction [Williams et al., 1995].

`AUX-BASIS {` *auxiliary basis* `[TYPE { MC | MC+ | DC | DC+ }] +++`
`[SPHERICAL | CARTESIAN]`
Specify the auxiliary basis to use. This command is optional. If absent, the auxiliary basis is chosen based on the main basis. The default is to use the RIMP2 auxiliary basis sets from the Turbomole program, which are specified in the same way as the Dunning basis sets, that is, `aug-cc-pVxZ` (or `aVxZ`) and `cc-pVxZ` (or `VxZ`), x =D, T or Q. The Sadlej basis has no optimized auxiliary basis associated with it, so the default here is the RIMP2 `aug-cc-pVTZ` auxiliary basis. As a rule, the RIMP2 auxiliary basis sets should be used for the calculation of the second-order energies.

It is also possible to specify the auxiliary basis set type as `MC` etc. We recommend the use of the `DC` or `DC+` auxiliary basis sets with `MC` and `MC+` main basis sets, respectively. This is the default.

Other choices are `Dgauss-A1-c`, `Dgauss-A1-x`, `Dgauss-A2-c`, `Dgauss-A2-x`, and the `TZVPP` and `SVP` RI-J bases from the Turbomole program. We have not yet tested the other basis sets, so use them with caution.

By default Cartesian GTOs are used in the auxiliary basis set, but this can be changed using the argument `SPHERICAL`.

```
ISA-AUX [SET1 | SET2 | SET3 | GISA | USR]      +++
    [MIN-S-EXP-H [=] <real> (default 0.0)]     +++
    [MIN-S-EXP-DUMMY [=] <real> (default=0.2)] +++
    [DUMMY-SYMM-LIMIT [=] <char> (default=S)]
```
Define the s-basis set for the iterated stockholder atoms (ISA) approach. We recommend `SET1` and `SET2`. The `GISA` basis set is taken from [Verstraelen et al., 2012]. User-defined basis sets can be supplied using the `USR` option. In this case, the basis sets should be placed in `$CAMCASP/basis/auxiliary/ISA/USR/`. The other commands provide some ability to control the basis sets: The minimum value of the exponents on hydrogen atoms can be set using `MIN-S-EXP-H` and the minimum exponent on dummy (ghost) sites can be set using `MIN-S-EXP-DUMMY`. Further, the maximum symmetry of basis functions on dummy sites can be set using `DUMMY-SYMM-LIMIT`.

`{MO-FILE-A | MO-FILE-B | MO-FILE} <MO file name> [[FORMAT] {ASCII | BINARY}]`
Normally the names of the molecular orbital (MO) files are decided by the Cluster program. These names are based on the molecule names. This command allows the user to specify the name and format of these MO files.

`MEMORY` *n* `{ BYTES | KB | MB | GB }`
The specified amount of memory will be used unless more is needed, in which case a warning will be printed.

`OPTIONS [TEST | TESTS] [ISOTROPIC-POLS] [NO-3RD-ORDER] [DF-SOLVER {LU | GELSS}]`
The `OPTIONS` command allows the user to set various special options.

- `TESTS`: Use this command to create a test job. This will typically reduce the size of the run so the job finishes in a reasonable time.
- `ISOTROPIC-POLS`: This command will cause the Process input file to include the `ISOTROPIC` command, thereby limiting the WSM polarizability description to include isotropic terms only.
- `NO-3RD-ORDER`: Suppress the calculation of 3$^{\text{rd}}$-order SAPT(DFT) energy terms when running the Sapt2006 program.
- `DF-SOLVER`: Use this option to set the type of solver to be used to solve the density-fitting equations in CamCASP. Options are `LU` and `GELSS`. The default in CamCASP is currently `LU`. Use `GELSS` for higher accuracies, but bear in mind that it uses considerably more resources than `LU`.

`SITE-ORDER [{OLD | DALTON2006} | {NEW | DALTON2013}]`
Specify which ordering of the sites is to be used. This is present for backward-compatibility only.

`INTEGRAL-SWITCH` *switch value*
Specify which integral switch to use in the CamCASP code. The default is 1. This choice affects the SAPT(DFT) energy components. The choices are:

- `INTEGRAL-SWITCH = 0` : Use density-fitting for the overlap and nuclear integrals. This is a low-accuracy option, but is needed for certain types of calculations. For example, regularization requires this option.

- `INTEGRAL-SWITCH = 1` : (default) Use the non-fitted forms of the overlap and nuclear integrals. This is the higher accuracy option.

`CAMCASP-COMMANDS`
commands go here
`END-CAMCASP-COMMANDS`
Use this to insert user-defined commands for the CamCASP input file. These will be used (below the `MOLECULE` blocks) in lieu of the default commands. This is useful if you wish to do something different but do now want to write the entire CamCASP input file by hand or wish to perform a batch of calculations. Use this with caution, preferably by *editing* a default CamCASP command file rather than starting from scratch.

When this block is present, Cluster will create the CamCASP file using the usual preamble and the `MOLECULE` specifications, after which the commands specified within this block will be appended. Consequently the last command should be `FINISH` to signify the normal end of the CamCASP file.

The files in directory `$CAMCASP/data/camcasp-cmnds` contain such blocks. For example, the file sapt-dft contains the CamCASP comands for a standard sapt-dft calculation. It refers to molecules `A` and `B` rather than using the molecule names. These are respectively the first and second molecules specified in the `MOLECULES` line of the job description section of the cluster file. It isn't necessary to change `A` and `B` to the actual names. So instead of specifying `sapt-dft` in the job description, it would be possible to specify
`#include-camcasp data/camcasp-cmnds/sapt-dft`
There would be little point in doing this, but a modified version could be filed in `$CAMCASP/data/camcasp-cmnds` and used in this way, or filed elsewhere and used with `#include`, or with the alternative syntax:
`CAMCASP-COMMANDS [FILE] \emph{name}`
If the `FILE` keyword is present, the *name* should be the full pathname of the file containing the commands. If it is omitted, the *name* should be the name of one of the files in the `$CAMCASP/data/camcasp-cmnds` directory. In either case, the `END-CAMCASP-COMMANDS` line is not needed.

`ORIENT [file] [for] [DISPLAY] [ENERGY] [LOCALIZATION]`
File suffix: `.ornt`
Construct an input file for a later Orient calculation to localize polarizabilities. This may need a bit of editing – see §8.2 for details. '`[file]`' is not the name of the file — it's redundant syntax, but can be included to clarify the function of the data line for the human reader. The file name is constructed automatically using the specified file-prefix, possibly modified, and a standard suffix, '`.ornt`' in this case.

If two or more molecules have been listed using the `MOLECULE` command, the Orient command file for the dimer consisting of the first two molecules will be constructed. If only one molecule has been specified, this command will write out the Orient command file containing commands for localizing its polarizabilities.

Normally all you need is the `ORIENT` command and the types of files created is determined as described above. This behaviour can be overriden using the `DISPLAY ENERGY LOCALIZATION` options which, if present, will generate the appropriate files.

The energy file is generated with a number of comments and options. It will require editing and for this you may need to consult the Orient manual.

`PROCESS [[file | files] [for] [CASIMIR] [and] [PFIT]]`
File suffix: `.prss` and `_casimir.prss`
Construct input files for the Process program for use in a later Pfit calculation to refine the localized polarizabilities and/or to create the Casimir input file for a later dispersion coefficient calculation. If no arguments are present, both files are created. The Process input file with commands to create the Casimir input file are put in a file ending with `_casimir.prss`.

The Process input file with commands for a later Pfit calculation is written for the first in the list of molecules, but the commands for a later Casimir calculation can use one or two molecules. If only one molecule is present in the list, the Process input file is written out for a calculation of the dispersion coefficients between pairs of the same molecule. If two or more molecules are present, the same is done for the first two molecules in the list.

These files may need a bit of editing – see §8.2 for details.

`GAUSSIAN [file]`
File suffix: `.com`
Construct the input file for a Gaussian calculation. The Gaussian file is written for the first molecule in the list.

`INTERFACE [file | files]`
This command writes out the CamCASP command file ('`.cks`' suffix), the `*.template` file, and, if needed, the `_P.data` file for a SAPT calculation.

If only one molecule has been specified using the `MOLECULE` command, only the CamCASP command file and `*.template` file are created, as it is assumed the interface files for a properties calculation are needed.

If two or more molecules have been specified, it is assumed that this is an interaction energy calculation using SAPT(DFT) and all three command files are created.

`RANK [NONLOCAL` l_{NL}`] [LOCAL` l_{L} `] [WSM` l_{WSM} `] [HYDROGEN` l_{H} `]`
`[DMA` l_{DMA} `] [DMA-H` $l_{\mathrm{DMA-H}}$ `]`
By default, the files produced by this module assume a polarizability and multipole moment description with rank 1 terms on the hydrogens and rank 4 terms on all other atoms. This can be changed using this command. There are many stages in the calculation of localized polarizabilities and the rank can be set for each of these, but as each stage uses the results of the previous one the rank cannot increase between stages. Also, at present, polarizabilities and multipole moments are limited to rank 4 (hexadecapole). Consequently we need to ensure that $4 \geq l_{\mathrm{NL}} \geq l_{\mathrm{L}} \geq l_{\mathrm{WSM}} \geq l_{\mathrm{H}}$ and $4 \geq l_{\mathrm{DMA}} \geq l_{\mathrm{DMA-H}}$. Cluster will stop with an error message if these requirements are not satisfied.

`SITES [file | files]`
File suffix: `.sites`
This command writes out the molecular geometry for each molecule in the list. The geometry is written in the file *prefix*`.sites`, or in files *prefix_nnn*`.sites` if more than one molecule has been listed. The format is that used by the Pfit program.

7.5 Energy

This module to some extent duplicates a small part of the functionality of Orient. There are a few reasons for this: (1) to read/write/alter a potential defined in Orient format, and (2) to calculate dispersion energies of clusters of molecules and crystals (see §7.6) using *isotropic* dispersion models (in this case, the anisotropy will be ignored), and hence (3) to aid the development of dispersion models to be paired with density functionals.

```
ENERGY
   ...see commands below...
END
```

The commands that can be used in the `ENERGY` block are:

```
POTENTIAL
   UNITS [BOHR | ANGSTROM | NM | HARTREE | KJ/MOL | KCAL/MOL | CM-1 | EV | K ]
   SCALE-C6 <scaling for C6>
   SKIP-SITES <list of sites to be skipped>
   NAME <name>
   [MOLECULE-PAIR <MolName1> <MolName2>]
   [ISOTROPIC-ONLY]
   <atom name 1>  <atom name 2> [rho] [alpha] [C6] [C7] [C8] [C9] [C10] [C11] [C12]
      00   00   0     [<rho value> <alpha value> <C6 value> <...>]
      10   00   1     [<rho value> <alpha value> <C6 value> <...>]
      ...
   END
END
```

The potential is defined and modified using this block. `UNITS` specifies the units used for the potential parameters. `SCALE-C6` scales the C_6 parameters by the specified factor. This is useful when *effective* C_6 models are needed. `SKIP-SITES` allows a set of sites to be removed from the potential decsription. `NAME` defines the name of the potential. And `MOLECULE-PAIR` is used to define the pair of molecules associated with the potential.

What follows this preamble is the actual potential definition. The format used is identical to that used by Orient, so please consult the Orient manual for details of the specification. Briefly, parameters for each atom pair are specified with a header line:

<atom name 1> <atom name 2> [rho] [alpha] [C6] [C7] [C8] [C9] [C10] [C11] [C12]
that starts with the atom pair labels and is followed by names of all parameters included. The order doesn't matter as long as it is consistent with the order of the parameter values that appear on the following lines. The parameters are specified along with their angular momenta as follows:

$l_a k_a$ $l_b k_b$ j [<rho value> <alpha value> <C6 value> <...>]

The pair-block is terminated with END. As many pairs can be specified as needed.

If an atom pair appears more than once the parameters in subsequent specifications are used to augment or override the prior sets. This is useful when the short range parameters are specified in one potential file but the long-range (dispersion) parameters are in another. Both files can be read in and concatenated into a single potential. Also, as input units can be altered using UNITS, unit conversions are possible.

If ISOTROPIC-ONLY is specified, all anisotropy parameters will be ignored.

```
WRITE-POTENTIAL
   [SKIP-DUPLICATES | NO-SKIP-DUPLICATES ]
END
```

Writes out the potential in the ORIENT format. The SKIP-DUPLICATES command can be used to prevent equivalent pairs from being written. Note that the pair H1 N is equivalent to N H1. This command would write only one of these.

Taken together with the commands in the POTENTIAL block, these provide a convenient way to manipulate the potential file: discard sites, alter units, merge potentials, etc.

```
ANALYSE
   TABLE [ANISO [10 | 20]]
END
```

Perform an analysis of the potential model. Please see the source code to figure out what kind of analysis has been programmed.

```
{C6 | C6-MODEL | CN-MODEL | DISP-MODEL | DISPERSION-MODEL }
    DAMPING
       GRIMME
          FUNCTIONAL [REVPBE | PBE | BLYP | B-P86 | TPSS | B3LYP | B97-D]
          DAMPING [04 | 06] [d [=] <value>]
          S6 [=] <value>
          POWER [=] <power>
          RADIUS-SCALING [=] <value>
       END
       TANG-TOENNIES [BETA [=] <beta>]
       GRIMME+T-T
    END
    {GRIMME06 | USER-DEF}
    SCALE-C6 [=] <multiplier>
    TYPES
       UNITS [BOHR | ANGSTROM] [HARTREE | KJ/MOL | CM-1 | EV | KELVIN | KCAL/MOL]
       [NAME [RADIUS] [C6] [C7]...]   <---Strings
       <TYPE> [<radius>] [C6] [C7]... <---Values
    END
    DEBUG
    [QUIET | VERBOSE]
END
```

Use this block to define the dispersion model used for a dispersion energy calculation. There are two main sub-blocks:

- **The potential block**
  If a potential has already been defined using the POTENTIAL block then this potential could be used by specifying USER-DEF. Else, the 2006 isotropic C_6 model defined by Grimme [Grimme, 2006] can be used with the GRIMME06 command. These two commands are mutually exclusive. Grimme specifies atomic properties from which the pair potential parameters are developed, typically using combination rules. These atomic properties can be altered using the TYPES block. The units used in this block is specified using the UNITS

command. Next appears a header that starts with the string NAME and is followed by the list of parameter labels as the appear in subsequent lines. You can use the TYPES block to alter atomic parameters from the Grimme 2006 values.

- DAMPING
  The following damping models can be used:
  - GRIMME: Either the 2004 or 2006 models. These take the form:

$$f^{\mathrm{G}}(r) = s_6 \left(\frac{1}{1 + \exp -d(\frac{r}{r_0} - 1)} \right)^{\alpha} \tag{2}$$

    The two models differ in the choice of the d parameter which can be set using DAMPING [04 | 06] to get the 2004/06 values, or it can be set explicitly using ... d [=] <value>. The s_6 parameter is determined using the functional. Or it can be specified using S6 [=] <value>. The power α is 1 in the Grimme models. You can alter it using POWER [=] <power>. Finally, the atomic radii that are used to determine r_0 can be scaled using RADIUS-SCALING [=] <value>.
  - TANG-TOENNIES [BETA [=] <beta>]: The Tang–Toennies [Tang and Toennies, 1992] can be used with a specified damping parameter β. These functions take the form:

$$f_n(\beta R_{ab}) = 1 - \exp(-\beta R_{ab}) \sum_{k=0}^{n} \frac{(\beta R_{ab})^k}{k!} \tag{3}$$

  - GRIMME+T-T specifies that both damping functions are used.

DISPERSION
Calculate the dispersion energy between all molecules defined in the CLUSTER input file using the energy model defined.

NO-INTRA
Do not calculate the intra-molecular energy. In many dispersion models no distinction is made between the inter- and intra-molecular contributions. This command causes the intermolecular energy only to be calculated.

7.6 Crystal

Once an interaction energy model has been defined, the dispersion energy of a molecular crystal can be calculated using this module.

```
CRYSTAL
   ...see commands below...
END
```

Allows calculations of the dispersion energy in crystals using the following commands:

```
{CRYSTAL | CRYSTAL-CELL | CRYSTALCELL | CELL}
    UNITS <units for length, energy, angles, pressure in the RES file>
    RES-FILE <file name> [READ-TITLE-LINE] [{FIXED | VARIABLE} CELL]
    CELL-FILE <file name>
    RADIUS [=] <radius> [Bohr | Angstrom]
    SHELL-TYPE [CUBIC]
    DEBUG
    [QUIET | VERBOSE]
END
```

Define the crystal cell using either a ShelX RES file or a CELL file. These formats are not defined here. Please see the CASTEP manual for details. One caveat: CLUSTER does not know about molecular point groups. Consequently, the crystal must be specified in P_1 symmetry, that is, you must specify all ions in the unit cell, and not just ions in the asymmetric unit. Normally, CASTEP will write out a CELL/RES file with P_1 symmetry, so you should be able to use these files here.

The dispersion energy of the crystal is calculated using real-space summations by including shells to a specified radius. The radius is set using RADIUS and has a default value of 15 Å. At present, only one algorithm for constructing shells has been provided so SHELL-TYPE must be CUBIC, which is the default. Here, the first shell is just the unit cell. The second contains all cells with a face in contact with the first shell, and so on. Pictorially, the construction looks like this in 2-D:

```
      2
    2 1 2
  2 1 0 1 2
    2 1 2
      2
```

```
DISPERSION    MIN-SEP [=] <minimum site-site separation>
    EPSILON [=] <epsilon value for finite difference>
    [QUIET | VERBOSE]
END
```

This calculates the dispersion energy for the crystal. The command MIN-SEP allows control of the minimum distance between sites. Dispersion energies will not be evaluated for sites with smaller separations. The default value is 2 Å.

The code also estimates the van der Waals pressure on the cell using the method of finite differences:

$$P_{\mathrm{vdW}} = \frac{dE_{\mathrm{disp}}}{dV} \approx \frac{E_{\mathrm{disp}}(V_2) - E_{\mathrm{disp}}(V_1)}{V_2 - V_1}, \tag{4}$$

where, if V_0 is the cell volume, $V_1 = (1 - \epsilon)V_0$ and $V_2 = (1 + \epsilon)V_0$. This equation scales all coordinates in the cell to calculate the pressure. The value of ϵ can be set using EPSILON. The default is 10^{-3}.

7.7 ORIENT

CLUSTER provides functions to use the output from an ORIENT BASIN-HOPPING search. ORIENT will write out the stable structures in a *.geom file in the format:

```
Minimum 1, energy =   -14.149952 kJ/mol
Moments of inertia:    1063.1800       2635.1297       2762.3835      amu bohr^2
pyridine1 at     0.128741    0.029517    1.422383 rotated by   36.672 deg
            about  -0.209496  0.602668 -0.770002
pyridine2 at    -0.128471   -0.029088    8.577618 rotated by   70.480 deg
            about  -0.005891  0.354489  0.935042
Minimum 2, energy =   -15.520807 kJ/mol
Moments of inertia:    664.76089       4988.6802       5176.5432      amu bohr^2
pyridine1 at     0.046180   -0.253926   -5.155193 rotated by  113.477 deg
            about   0.937866  0.028060  0.345861
pyridine2 at    -0.046161    0.253801    5.154207 rotated by   79.420 deg
            about   0.456964 -0.717933 -0.525125
...
...
```

This module can select particular minima from this file and create dimer geometry files in the standard format (with one molecule fixed in a reference position and the other rotated. Such a file can subsequently be used in all the CAMCASP and ORIENT calculations possible with scripts included in this CAMCASP distribution.

The syntax is as follows:

```
ORIENT
  UNITS ...
  [QUIET | VERBOSE]
  MAXCFGS = <maximum number of configurations| par_maxcfgs>
  CONFIG-FILE <name> [SKIP-MOMENTS] [MAXCFGS = <maximum num of configs>] +++
```

```
    [OFFSET <value>]  [APPEND] [NAME <name>]
  REORIENT [[molecule] <molecule name>]
  R-SCALINGS <scalings to be applied to R vectors for
              molecules in configurations>
  WRITE [FORMAT [ORIENT [[&] GEOM]]] [ALL | CONFIG <index>] [LABELS]
  SORT
  ANALYSE
    [CFG-LIST-INDEX <index>]
    DETAILS
    SORT
    SIMILARITY [ENERGY] [MOMENTS]
    ENERGY-SIGMA <sigma> [<units>]
    MOMENT-SIGMA <sigma> [<units>]
    SIMILARITY-CUTOFF <value in [0,1]>
  END

  ! Subsequent calls to ANALYSE can be of the form:
  ANALYSE CFG-LIST-INDEX <index>

  CORRELATE
    {ALL | LIST list of config-list indices}
    DETAILS
    SIMILARITY [ENERGY] [MOMENTS]
    ENERGY-VAR <variance> [UNITS <units>]
    MOMENT-VAR <variance> [UNITS <units>]
    SIMILARITY-CUTOFF <value in [0,1]>
  END
  TABULATE {ALL | LIST list of config-list indices}
END
```

`UNITS ...`
This is the standard units specification with will typically specify length and angle units.

`MAXCFGS = <maximum number of configurations| par_maxcfgs>`
Specify the maximum number of configurations. This should be larger than the number of minima included in the `*.geom` file. You should not have to specify this as it takes on a sensible default set in parameter `par_maxcfgs`.

`CONFIG-FILE <name> [SKIP-MOMENTS] [MAXCFGS = <maximum num of configs>] +++`
`[OFFSET <value>] [APPEND] [NAME <name>]`
Specify the name of the ORIENT`*.geom` file here. The option `SKIP-MOMENTS` will suppress reading the moments of inertia. And the maximum number of configurations cab be set here too using `MAXCFGS`. This command can be used as often to read configurations from multiple structure searches. By default each set of configurations will be placed in a new list. The `APPEND` command causes configurations to be added to the previous list. In this case, to avoid a conflict of the indices of the minima, use the `OFFSET` command to offset the indices of the minima by the specified value. Optionally, the configuration list can be given a name using `NAME`.

`REORIENT [[molecule] <molecule name>]`
As shown in the example above, the `*.geom` file will usually have both molecules rotated and translated. This is a problem as the convention for all scripts included in CAMCASP is to have one molecule fixed in its reference orientation and the other translated and rotated. The `REORIENT` command re-orients the dimer to have the specified molecule re-oriented back into its reference orientation.

`R-SCALINGS <values>`
Having obtained the re-oriented dimer, generate a set of configurations by scaling the translation vector by the specified values. An example set is
`R-SCALINGS 0.80 0.85 0.90 0.95 1.00 1.05 1.10 etc.`.

`WRITE [FORMAT [ORIENT [[&] GEOM]]] [ALL | CONFIG <index>] [LABELS]`
Write out the geometries in a suitable format. `FORMAT ORIENT` writes out all geometries in the format used in `*.geom`. For the example quoted above with `pyridine1` re-oriented, this would result in:

```
UNITS  BOHR DEGREES HARTREE
Minimum     1, energy =        -0.005389 HARTREE
Moments of inertia:    1063.180000    2635.129700    2762.383500 AMU BOHR^2
    pyridine1 at     0.000000     0.000000     0.000000  rotated by
        0.000000 DEGREES about     1.000000     0.000000     0.000000
    pyridine2 at    -2.527069    -1.715543     6.475941  rotated by
        93.750732 DEGREES about    -0.138703     0.004010     0.990326
Minimum     2, energy =        -0.005912 HARTREE
Moments of inertia:     664.760890    4988.680200    5176.543200 AMU BOHR^2
    pyridine1 at     0.000000     0.000000     0.000000  rotated by
        0.000000 DEGREES about     1.000000     0.000000     0.000000
    pyridine2 at     4.513912     8.832546    -2.856706  rotated by
        114.211262 DEGREES about    -0.676391    -0.735007    -0.047549
...
...
```

The FORMAT GEOM for minimum 1 and the above R-SCALINGS results in the following:

```
  Reorient  pyridine1
  R-scalings 0.80 0.85 0.90 0.95 1.00 1.05 1.10 1.20 1.40
  Write  Format ORIENT & GEOM Config 1
! CGF =     1
! UNITS  BOHR DEGREES HARTREE
    1 -2.02165515  -1.37243434   5.18075303  93.75073208  -0.13870331   0.00401048   0.99032586
    2 -2.14800859  -1.45821149   5.50455009  93.75073208  -0.13870331   0.00401048   0.99032586
    3 -2.27436204  -1.54398863   5.82834716  93.75073208  -0.13870331   0.00401048   0.99032586
...
...
```

Here we have selected the first configuration using the CONFIG <index> option. By default the geometries are written out with integer indices; the command LABELS optionally used the scaling factors to label the geometries.

SORT
Sort the list of structures based on their energies.

```
ANALYSE
  [CFG-LIST-INDEX <index>]
  SORT
  SIMILARITY [ENERGY] [MOMENTS]
  ENERGY-VAR <variance> [<units>]
  MOMENT-VAR <variance> [<units>]
  SIMILARITY-CUTOFF <value: [0,1]>
  DETAILS
END

! Or the shorter form that uses the setting from
! the previous call to ANALYSE:

ANALYSE CFG-LIST-INDEX <index>
```

Analysing more than a few structures by hand can be tedious and is prone to error. This module provides a set of tools to make structure analysis easier.

By default, the analysis is done on the last list of configurations read using CONFIG-FILE. A specific configuration list can be selected using CFG-LIST-INDEX. Structures can be sorted based on energies using SORT. SIMILARITY tests can be performed based on the ENERGY and/or MOMENTS of intertia. As a rule of thumb, use the ENERGY criterion only if all structures have been obtained from the same potential energy surface. Rather than implement hard cutoffs to judge similarity Gaussian similarity probabilities are assigned based on supplied variances. Variances can be set using the ENERGY-VAR and MOMENT-VAR commands in specifed units or current units. The only hard cutoff

used is the SIMILARITY-CUTOFF which should be between 0 and 1. Strcutures with similarity probabilities greater than this will be deemed similar. Normally, this sub-module will perform the analysis and write out a truncated GEOM file that consits of unique structures only. If details of the analysis are desired use the DETAILS option. Of use is the similarity probability matrix printed out.

Once analysis parameters have been set using the above block, the analysis of other configuration lists can be performed using the same settings with the following command:
ANALYSE CFG-LIST-INDEX <index>

```
CORRELATE
  {ALL | LIST list of config-list indices}
  DETAILS
  SIMILARITY [ENERGY] [MOMENTS]
  ENERGY-VAR <variance> [UNITS <units>]
  MOMENT-VAR <variance> [UNITS <units>]
  SIMILARITY-CUTOFF <value in [0,1]>
END
```

Once a set of configurations lists has been analysed, we may want to establish correlations between the structures in the lists. This is very tedious to do by hand and is likely to be error-prone. The CORRELATE module can be used to establish correlations. Configuration lists can be selected using ALL for all lists, or LIST with a selection of list indices. The other commands are analogous with the ANALYSE module. As a rule, the comparison tolerances need to be loosened when attempting to make comparisons between structures found on different energy surfaces.

The results of the correlation will be printed out as a table of configuration indices and energies. Configurations in a row are deemed to be similar within the tolerances set. This table can be visualised using the script utilities/plot-levels.py.

```
TABULATE {ALL | LIST list of config-list indices}
```

Creates a table of configurations in either ALL or specific LISTs.

7.8 Finally, . . .

```
FINISH
```
End of data.

8 Examples

8.1 A SAPT(DFT) calculation

```
!  Cluster file for H3N...HF

!  These are the global units. Can be overridden locally in some modules.
Global data
  Units bohr deg
  Camcasp-path /usr/local/camcasp-5.7   ! Full path to the CAMCASP installation
                                        ! (not usually needed)
End

Molecule H3N
  Units bohr
  I.P.  0.3701
  !  Geometry from Herzberg
  N      7.0    0.0000000000     0.0000000000     0.0000000000
  H1     1.0    1.79359084       0.0000000000    -0.69190123
  H2     1.0   -0.89679542       1.5532952280    -0.69190123
```

```
  H3     1.0   -0.89679542      -1.5532952280    -0.69190123
END

Molecule HF
  Units angstrom
  I.P. 0.5891
  !  Geometry from Herzberg
  F      9.0    0.00000000     0.00000000      0.00000000
  H      1.0    0.00000000     0.00000000      0.9171
END

!  As defined, the F of HF coincides with the N of NH3, and the H
!  of HF points away from N. Rotate the HF (of course we could
!  have placed the H on the -z axis in the first place):
Rotate HF by 180 about 1.0 0.0 0.0
!  Now move the HF so that the N...F distance is 6 bohr:
Translate HF by 6.0 bohr along 0.0 0.0 1.0

!  Specify the calculation
Run-Type
  SAPT(DFT)
  Molecules H3N and HF
  File-prefix H3N_HF
  !
  Basis Sadlej  Type MC
  Aux-Basis aTZ Type DC
  !
  SCFcode DALTON     ! These are all defaults.
  Functional PBE0
  Kernel ALDAX+CHF
  !
  Interface files    ! Needed to generate the Camcasp, *.template and
                     ! *_P.data files.
End

Finish
```

This file creates an H3N...HF cluster. The monomer basis set will be used. The auxiliary basis is usually set by default, but here we have defined it explicitly. Note that even though the main basis is type `MC`, the auxiliary basis is set to type `DC`. This is the default and results in far higher accuracies than the `MC` basis type.

The command `File-prefix` allows you to specify your own file prefix. The default is to use the concatenation of the molecule names, which can be ugly if they're spelt out.

We are going to run a standard SAPT(DFT) calculation, which requires a DALTON calculation to obtain wavefunctions for each monomer, with and without ghost functions on the other monomer, followed by a CAMCASP calculation to obtain the induction and dispersion energies using coupled Kohn–Sham perturbation theory. All of this can be done using a single high-level script:
`runcamcasp.py H3N_HF -q small`
where `H3N_HF` is the job-name, and is the same as the file prefix specified in the CLUSTER input file.

8.2 An example properties calculation

We start with the simple input file for CLUSTER. For formamide it would look like Fig. 2 on the next page. Let's call it `HCONH2.clt`. The '!' character starts a comment, which continues to the end of the line. Notice that the atoms must all have different names — the properties will be different for the different H atoms, for example, and we need to be able to tell which is which.

Now the `runcamcasp.py` script will set up and run the main calculation; for example
`runcamcasp.py HCONH2 -q small`
This would be a properties calculation even without the `Properties` line, because we have only specified one molecule. The `-q` flag specifies how the job is to be run, as for a SAPT(DFT) calculation; see p.13. By default,

```
TITLE Formamide geometry file

Global data
  UNITS Bohr Degree  !  These are the defaults
  Camcasp /usr/local/camcasp-5.7 ! This is the full path to the \Camcasp installation
                                 ! (probably unnecessary)
End

!  Define the molecular geometry
Molecule formamide
  I.P. 0.3745
  Charge 0
   C      6.0        0.0000000000        0.0000000000        0.0000000000
   O      8.0        2.2906315140        0.0000000000        0.0000000000
   H1     1.0       -1.1390824747        1.7724683628        0.0000000000
   N      7.0       -1.4598529795       -2.1020505657        0.0000000000
   H2     1.0       -3.3620904774       -1.9841855358        0.0000000000
   H3     1.0       -0.6271017798       -3.8218918498        0.0000000000
End

!  Now specify the calculation to be done
Run-Type
  Properties
  Molecule formamide
  file-prefix HCONH2  !  This will identify files for this job
  Basis Sadlej        !  The default is type MC.
  SCFcode DALTON
  Func PBE0           !  Specify the functional to use. See below.
  Kernel ALDAX+CHF    !  Specify the kernel for the coupled Kohn-Sham
                      !  perturbation propagator. See below.
  Orient file         !  These lines produce input files for
  Process file        !  later stages of the calculation
  Interface file      !  Needed to create the CamCASP and *.template files
  Sites file          !  Creates the HCONH2.sites file containing the
                      !  molecular geometry.
End

Print formamide in Orient format  !  This just prints the geometry

Finish
```

Figure 2: Cluster file for formamide

this script uses a CLUSTER input file with the name *JOB*`.clt`, i.e. `HCONH2.clt`. If this is not the case, use the "`--clt` *cluster file*" option to specify the name of the CLUSTER file.

The initial steps in the calculation are the same as for a dimer energy calculation, but the files created are different:

- Create a new directory for the job, as above.
- Run the `cluster` program to generate the necessary files. If NWCHEM were to be used, `cluster` would generate the input file directly. In the case of DALTON, `cluster` generates a template file that is used to construct the DALTON `.dal` and `.mol` files. This step uses the ionization potential for formamide.
- Once the files are set up, the job submission proceeds as for a SAPT(DFT) job. The submitted job runs DALTON and then the CAMCASP program to calculate the properties specified in the *JOB*`.cks` file.

A note on the choice of functional: PBE0 is normally recommended, and is the default, but it is possible to use any functional provided by Dalton. Also the propagator kernel is `ALDA+CHF` by default, but `ALDAX+CHF` may be specified for a faster calculation with a small loss in accuracy.

Now you've got your moments, distributed polarizabilities at 11 frequencies (static and 10 imaginary, all in a single `.pol` file) and point-to-point polarizability responses on a grid of 2000 points at 11 frequencies (all in a single `p2p` file).

Next we'd like to localize the polarizabilities and refine them, and perhaps calculate dispersion coefficients.

8.2.1 Localization

Localization is carried out by changing to the directory for the job, and executing the simple command
`localize.py` *JOB options*
This carries out the following steps:

- Find the `.pol` file containing the calculated polarizabilities, and the `.p2p` file containing the point-to-point responses. They will usually be in the OUT subdirectory. If for some reason there is more than one possible file, you will be asked to choose the right one.
- The `.pol` file is split into separate files for each frequency.
- The polarizabilities at each frequency are localized, using the Orient program. Normally localized polarizabilities are calculated only up to rank 2, but this can be changed using the `--limit` option. The localization can be carried out using either the Le Sueur–Stone procedure [Le Sueur and Stone, 1994] or the Lillestolen–Wheatley procedure [Lillestolen and Wheatley, 2007]. The latter is recommended and is the default.
- Unless the `--norefine` option has been specified, the local polarizabilities are refined by fitting to the point-to-point responses. This is done separately at each frequency.
- Unless the `--nodisp` option has been specified, the refined polarizabilities are used to evaluate site–site dispersion coefficients.

Further options for the `localize.py` command can be listed by invoking the command with the single argument '`--help`'. It is also possible to modify the behaviour of the command by editing the data files that were created automatically by the `runcamcasp.py` command, or by providing additional ones, as explained below. The full list of options is as follows:

`--sites`*file* The molecular sites are defined in the specified file. Default file is *JOB*.sites, and this is constructed for you if specified in the `.clt` file. It can be modified if necessary, in particular to add a definition of site types at the beginning and to specify the type of each site if not done already.

`--axes`*file* The local axes are defined in the specified file. The default file is *JOB*`.axes`. If the specified file isn't found in the current directory, it is searched for in the directory above (the one in which you issued the original `runcamcasp.py` command). If one of these exists but is empty, or if neither exists, global axes are used. If there is symmetry, it is important to define local axes that correspond to the symmetry, and to ensure that symmetry-related sites are of the same type (and that symmetry-unrelated sites are of different types). For more about this, see §8.3 on p. 35.

`--limit` n Generate localized polarizabilities up to rank n. Default 2. Higher ranks may be specified, but 2 is usually enough.

`--wsmlimit` n_w Limit the rank of refined polarizabilities to n_w. $n_w < n$. Default $n_w = n$.

`--hlimit` n_h Limit the rank of refined polarizabilities on hydrogen atoms to n_h. $n_h \leq n_w$, default $n_h = n_w$. For most purposes it is sufficient to set $n_h = 1$.

`--weight` w Use weighting scheme w for refining the polarizabilities. See p. 74 for details. Default is scheme 4.

`--isotropic` When refining, fit only isotropic polarizabilities. The default is to fit anisotropic terms as well.

`--norefine` Skip the refinement step and the calculation of dispersion coefficients.

`--nodisp` Skip the calculation of dispersion coefficients.

`--keep` Don't delete the intermediate files generated by the script. Shouldn't be necessary unless bugs occur.

`--loc { LW | LS }` Use the specified localization procedure — LW (Lillestolen–Wheatley) or LS (Le Sueur–Stone). LW is recommmended and is the default.

```
Molecule  formamide at  0.0 0.0 0.0
  ! Units BOHR
  C           0.00000000    0.00000000    0.00000000   Type C
  O           2.29063151    0.00000000    0.00000000   Type O
  H1         -1.13908247    1.77246836    0.00000000   Type H1
  N          -1.45985298   -2.10205057    0.00000000   Type N
  H2         -3.36209048   -1.98418554    0.00000000   Type H2
  H3         -0.62710178   -3.82189185    0.00000000   Type H3
End

Polarizabilities for formamide
   Read rank  4 sites +++
      C        O        H1       N        H2       H3
   #include HCONH2_NL4_<INDEX!000>.pol
   Limit rank <LIMIT! 2> for +++
      C        O        H1       N        H2       H3
   Sum-rule test 1e-7
   Localize <LOC!LW> test 1e-7 Limit <LIMIT! 2> sites +++
      C        O        H1       N        H2       H3
End

Edit formamide
   #include <AXES!axes>
End
```

Figure 3: Part of the Orient file for formamide

8.2.2 Further details

The data files used by the localization process are mostly constructed by Cluster at the start of the `runprops` procedure. If necessary they can be modified before carrying out the localization.

Orient is used to obtain the local polarizabilities. Fig. 3 shows part of the `HCONH2.ornt` input file, as constructed by Cluster. In this data file, constructs like `<INDEX!000>` are replaced by appropriate values during the execution of the `localize.py` script.

If necessary, the file can be edited before carrying out the localization. Another possibility is to include another `EDIT` section immediately after the molecule definition to define the bonds explicitly. The bonding is used by the LW procedure, and normally the bonds are assigned automatically by reference to standard atom radii. It might be necessary to define the bonds explicitly if the automatic procedure doesn't work, for example if one of the 'bonds' is unusually long. See the Orient manual for how to do this.

It is possible, as mentioned above, to provide a file defining local axes for each atom. By default, the file is called *JOB*`.axes`, and it is automatically used if provided, either in the job directory or the directory above. In the present example we might choose local axes for each H atom, with x along the bond and z perpendicular to the molecular plane. Instructions for specifying this are in the Orient manual. It is not necessary in this case, and if no local-axis file is provided, an empty file is supplied. Local axes become important when there is molecular symmetry, and an example is discussed in §8.3.

8.2.3 Refining the local polarizabilities

We have found that the localization procedure used by Orient results in local polarizabilities that are not quite optimum. They can be refined by using the Pfit program to reproduce the point-to-point polarizabilities. The localization script does this automatically, unless the `--norefine` flag is specified. The necessary input files for Pfit are generated automatically by Process from the `HCONH2.prss` input file, shown in Fig. 4. Constructions like `<LIMIT! 2>` are replaced by default values or by values specified on the command line.

Pfit has other important uses. The standard procedure does not impose full symmetry, though it does require sites of the same type to have the same polarizabilities. The grid of points used for the point-to-point polarizabilities does not have any symmetry, so fitting to the values may result in polarizabilities that should be zero having small non-zero values. This can provide a guide to the accuracy of the fitted values, but usually it is better to obtain values that have the correct symmetry. Also the standard procedure fits all possible polarizability components up to the

```
TITLE  PROCESS file for formamide
! Usage: process < HCONH2.prss


Set Global-data
   Units  BOHR  DEGREE
   Overwrite
   Echo Off
End

Molecule formamide
  Units BOHR
  C           6.00    0.00000000    0.00000000    0.00000000   Type C
  O           8.00    2.29063151    0.00000000    0.00000000   Type O
  H1          1.00   -1.13908247    1.77246836    0.00000000   Type H1
  N           7.00   -1.45985298   -2.10205057    0.00000000   Type N
  H2          1.00   -3.36209048   -1.98418554    0.00000000   Type H2
  H3          1.00   -0.62710178   -3.82189185    0.00000000   Type H3
End

Read local pols for formamide
   Use binary file HCONH2_L<LIMIT!2>_0f10.pol
   Maximum rank  <LIMIT! 2>
   Limit rank to <WSMLIMIT! 2>
   Limit rank to <HLIMIT! 1> for sites +++
      H1       H2       H3
   Frequencies STATIC + 10
   ! Use this command to split the P-2-P pol file into 11 parts:
   ! P2P-Pols <point-2-point pol file> SPLIT
End

Write
   File-prefix HCONH2
   PFIT file for formamide +++
      with cutoff 0.0001 and freq <INDEX!0> +++
      with penalties and weight <WEIGHT! 4> +++
      and <ISOTROPIC!> polarizabilities
End

Finish
```

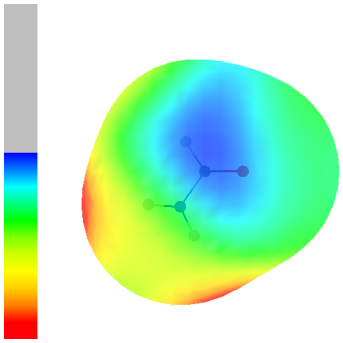
Figure 4: The `HCONH2.prss` input file

specified maximum rank, but a simpler model may be needed. These needs can be met by editing the polarizability definition file, *JOB*`.pdef`. This file is generated automatically if it doesn't already exist, either in the job directory or the one above, so the usual procedure is to carry out the localization and refinement, and check the *JOB*`.pdef` to make sure that the polarizability model is correct and suitable. If it needs to be changed, just run the localization command again with the revised file in place.

The *JOB*`.pdef` file as automatically generated includes all polarizability components up to the limits specified in the `localize.py` command, and the polarizabilities for sites of the same type are assumed to be identical. This means that sites should not be assigned the same type unless they are related by symmetry, or they are to be treated as equivalent. Also the axes for the symmetry-related sites must be related by *proper* symmetry operations (pure rotations), as sites related only by improper operations may have unavoidable sign differences between some of the components.

Also some components may be zero as a consequence of the local site symmetry. Such components should be deleted from the *JOB*`.pdef` file. If left, they will usually be fitted with values close to zero, but not exactly zero because the data grid used for the localization is not symmetric. Also including components that must be zero adds unnecessarily to the complexity of the model and may lead to numerical problems.

The ORIENT program can be used to map the induction energy arising from the response of the local or non-local polarizabilities to a point charge on the molecular surface. The example shows it for the non-local polarizabilities of formamide, up to rank 4, on the vdW×2 surface. The energy zero is at the top of the grey section of the energy scale. Such maps can be generated for a variety of polarizability models, which might be obtained for example by simplifying the model defined by the *JOB*`.pdef` file, and they give a clear guide to the quality of the polarizability description [Misquitta et al., 2008a].



8.3 Dispersion coefficients

Distributed (i.e. atom–atom) dispersion coefficients will be calculated by the `localize.py` procedure unless the `-nodisp` option (or the `-norefine` option) is specified. Coefficients from C_6 up to C_{10} or C_{12} can be calculated, but note that some contributions to the higher coefficients may be absent. For example, C_8 coefficients should include contributions from dipole–octopole polarizabilities, but these will be absent if only polarizabilities up to quadrupole–quadrupole are provided.

Let's use 1-chloro-2,6-dibromo-4-fluoro-benzene as an example. This molecule was one of those included in a blind test of crystal structure prediction by the Cambridge Crystallographic Database (CCDC) and was referred to as molecule XIII. We will use this name below. The cluster file is shown in Fig. 5. Note that the type names assigned to the atoms reflect the C_{2v} symmetry of the molecule —for example the two Br atoms, which are symmetry-related, are both of type Br. We need to specify an axis system appropriate for the symmetry. The local axis file is shown in Fig. 6.

We could use the global axis system, by not providing a local-axis file and by leaving out the type definitions in the cluster file. However this leads to inconvenient sign differences between symmetry-related polarizabilities on different atoms. The cluster file also imposes symmetry by specifying that, for example, the two Br atoms are of the same type. This means that their polarizabilities and dispersion coefficients are constrained to be identical. They should be identical anyway, with the local axes as defined in Fig. 6, but the grid of points used for the point-to-point polarizabilities is not constrained to have the C_{2v} symmetry of the molecule, because that would lead to unnecessary duplication of computational effort.

Note that the local axes defined here are all right-handed. This means that the reflection taking C2 to C6 takes x into x and z into z but y to $-y$. However y is perpendicular to the molecular plane, and any property that changes sign under reflection in the molecular plane is zero by symmetry.

Now we can run the `localize.py` script, as for formamide:

```
localize.py XIII
```

And we've finally got our dispersion coefficients. Here's part of the output (p. 35):

```
$ cat XIII_ref_wt4_L2_casimir.out

Title XIII ... XIII
XIII
Frequencies   0.5    10
Skip  0
Print nonzero

Molecule  XIII
Site  C2
10 10
             5.72200000      5.70700000      5.61200000      5.27400000 +++
             4.44000000      3.07700000      1.62100000      0.58700000 +++
             0.11800000      0.00600000
   ...
   ...
   ...
Dispersion coefficients for XIII and XIII
  C2  C2               C6             C7              C8      ...
    00   00   0    88.39684131     0.0         4820.75066614 ...
```

```
Global
  Units Bohr degrees
  Overwrite yes
End

Molecule XIII
  I.P. 0.3342
  Units Bohr
  C1     6.0    0.00000    0.74117    0.00000  Type C1
  C2     6.0    2.27488   -0.62887    0.00000  Type C2
  C3     6.0    2.29640   -3.26453    0.00000  Type C3
  C4     6.0    0.00000   -4.52851    0.00000  Type C4
  C5     6.0   -2.29640   -3.26453    0.00000  Type C3
  C6     6.0   -2.27488   -0.62887    0.00000  Type C2
  Cl    17.0    0.00000    3.99305    0.00000  Type Cl
  Br2   35.0    5.43284    1.05478    0.00000  Type Br
  H3     1.0    4.05277   -4.30400    0.00000  Type H
  F      9.0    0.00000   -7.08125    0.00000  Type F
  H5     1.0   -4.05277   -4.30400    0.00000  Type H
  Br6   35.0   -5.43284    1.05478    0.00000  Type Br
End

Run-Type
  Properties
  Molecule XIII
  Basis Sadlej
  File-prefix XIII
  Orient file
  Process file
  Sites file
  Interface files
End

Finish
```

Figure 5: The `XIII.clt` cluster file

```
Axes
  C1   z from C1 to Cl     x from C1 to C6
  Cl   z from C1 to Cl     x from C1 to C6
  C2   z from C2 to Br2    x from C2 to C3
  Br2  z from C2 to Br2    x from C2 to C3
  C6   z from C6 to Br6    x from C6 to C5
  Br6  z from C6 to Br6    x from C6 to C5
  C3   z from C3 to H3     x from C3 to C4
  H3   z from C3 to H3     x from C3 to C4
  C5   z from C5 to H5     x from C5 to C4
  H5   z from C5 to H5     x from C5 to C4
  C4   z from C4 to F      x from C4 to C3
  F    z from C4 to F      x from C4 to C3
End
```

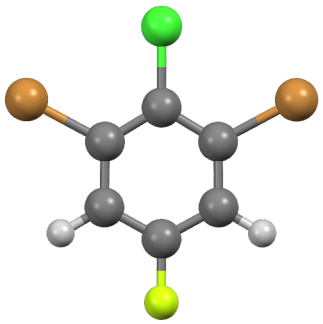
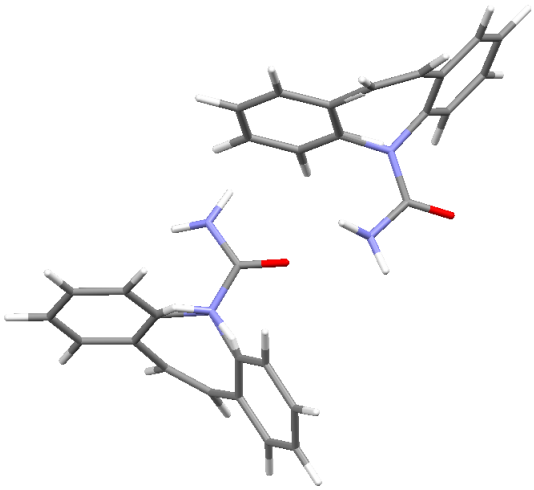


Figure 6: Local axis definitions for molecule XIII.

```
    00   11c  1     0.0          809.88012198                ...
    00   11s  1     0.0          477.13620553                ...
    00   20   2   -24.31832249     0.0        -3056.70651103 ...
   ...
   ...
   ...
Finish
```

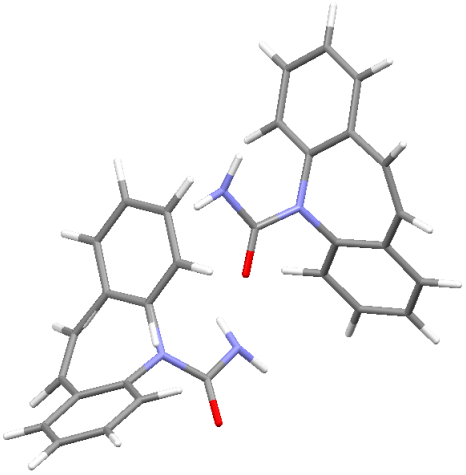
8.4 Using CLUSTER to obtain the dimer geometry



The problem: We wished to study the interaction between carbamazepine molecules in the crystal, but this molecule **was at the time** too big for the calculation to be carried out between whole molecules. *With the current version of* CAMCASP *this dimer is not a problem,* but similar fragment methods are still useful in other, larger, systems. In this case, we modelled the close interaction using a urea fragment to represent part of one molecule and a benzene fragment for part of the other. We needed to place the fragments in the same relative geometry as in the carbamazepine molecules. We have the CBMZ crystal geometry from the X-ray structure.

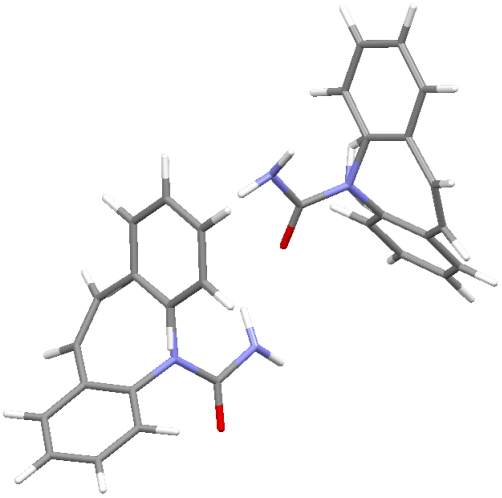
```
TITLE CBMZ fragments
UNITS Bohr Degrees

!  Define the molecules.
!  We need the benzene and urea molecules, and the CBMZ (carbamazepine)
!  dimer to form a template onto which we will place the benzene and urea.
!  We define the benzene to lie in the xy plane
Molecule benzene
 ...
End

Molecule urea
 ...
End

Units Angstrom
Molecule CBMZcat
 !  Two molecules here; coordinates from the X-ray structure.
 ...
End

!  We keep benzene nicely placed and rotate CBMZcat instead.
!  Sites C9, C10 and C11 in CBMZcat are in one of the benzene rings.
Find rotation
  Tol 0.03
  From sites C9 C10 C11 in CBMZcat
  To   sites C1 C2  C3  in benzene
End
Rotate CBMZcat

Find translation
  from site C9 in CBMZcat
  to   site C1 in benzene
End
Translate CBMZcat

!  Print out the new CBMZcat geometry to confirm
!  that its ring is superimposed on the benzene:
```

```
Show CBMZcat

!  Alternatively, here's a nice way of seeing
!  whether all is well:
Join CBMZcat and benzene into tmp1
Write tmp1 in PDB format
```



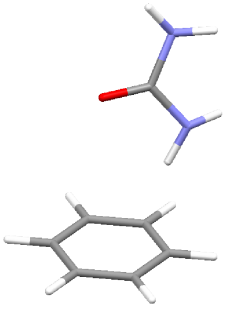
We can see the benzene middle left, superimposed on the lower CMBZ. (The view has been rotated – the benzene is actually still in the *xy* plane.) All's well, so let's go on ...

```
!  Now place the urea in place
!  It should align as follows:
!  urea:  C1u O1u N1u N2u H1u H2u H3u H4u
!  CBMZcat    :  C30 O2  N4  N3  4H2 3H2 --  --

Find rotation
  Tol 0.01
  From sites C1u O1u N2u in urea
  To   sites C30 O2  N4  in CBMZcat
End

Rotate urea


Find translation
  From site C1u in urea
  To   site C30 in CBMZcat
End
Translate urea

!  Check that this is right:
Join urea and tmp1 into tmp2
Write tmp2 in PDB format
```



All's well again, so let's finish the job ...

```
!  Make a PDB file of the interacting fragments
!  (not necessary for the calculation)
Join benzene and urea into benzene_urea
Write benzene_urea in PDB format

!  Construct the files for the SAPT(DFT) job
Run-Type
  SAPT(DFT)
  Molecules benzene and urea
  Basis Sadlej
  Type MC+
  Midbond 3s2p1d Position weighted
  Interface files
End

Finish
```



Now use the `runcamcasp.py` script as in §8.1 to run the calculation.

9 CamCASP program specification

Usually the input dataset for CamCASP will be generated automatically with the Cluster command, and this is usually a good starting-point in any case. Sometimes, however, it becomes necessary to modify the generated file, so full details of the program input are given here.

See §6 for general notes on syntax descriptions. In the CamCASP input, the form SET *keyword* is used to start a block which just sets parameters for later use, while BEGIN *keyword* is used for blocks that carry out a calculation. In the latter case, the block is read to the end before the calculation is done.

The CamCASP input file begins with a title and memory declaration:

TITLE *title*
MEMORY *memory* [BYTES | KB | MB | GB]

The memory can be set in BYTES, KB, MB or GB.

It then continues with a series of data blocks, which are each described here in turn. In addition to the options mentioned, most blocks have QUIET and VERBOSE options, to reduce or increase the amount of output, and a DEBUG option, which is for development purposes and should not normally be needed.

In addition to numerical and string arguments in commands, some commands expect a *switch* (Boolean or logical) value, which can be YES, Y, ON, TRUE or T, which are all equivalent, or NO, N, OFF, FALSE or F. If the switch value is omitted it is assumed to be TRUE, which however may not be the default if the command is omitted altogether. Thus the default if the ECHO command is omitted is TRUE but the default if the DEBUG command is omitted is FALSE.

9.1 Global data

SET { GLOBAL | GLOBAL-DATA | DATA }
  settings
END

SET is optional. The settings are as follows. Most of them can be over-ridden by commands in individual modules:

CAMCASP-PATH *absolute* path to the CamCASP installation.
If set, the #include-camcasp command can be used to include files present in the CamCASP directory using only the *relative* file paths. It will usually be set automatically from the CAMCASP environment variable.

UNITS *unit unit* ...
where the available units are
ANGSTROM, ANG, BOHR or NM;
HARTREE, KJ/MOL, CM-1, eV or K/KELVIN;
DEG, DEGREE, DEGREES, RAD, RADIAN, or RADIANS.
Defaults are BOHR, HARTREE and DEGREES. The units set in this command are used for all output. They are also used for input unless over-ridden by a local UNITS declaration in a module.

ECHO *switch*
If ON (the default) input is echoed to the output.

OVERWRITE [files] *switch*
OVERWRITE instructs the program to overwrite any existing files with the same names as those that the program would normally use. If OFF (the default) the names are modified, by inserting a 3-digit code just before the file suffix. The 3-digit code may be different for different files, depending on what files are already present. The file names are printed in the program output, but to avoid confusion it is better to carry out each run in a different directory.

{SCF-CODE | DFT-CODE} [{DALTON | DALTON2013 } | DALTON2006 | CADPAC | GAMESS(US) | NWCHEM]
Specifies the SCF code to use. Default DALTON, i.e., the Dalton program. (Cadpac is unlikely to be available for most users.)

{XC-FUNC | XC-FUNCTIONAL | FUNCTIONAL} *functional*
Specify the exchange-correlation functional to be used. Available functionals are PBE PW91 PBE0 HCTH HCTH407.
Default PBE0.

{C-X | FRACTION-HF | EXACT-EXCHANGE} *Fraction of exact-exchange*
The default fraction is set according to the chosen functional and should not need to be changed. Use this command if the exchange-correlation functional you have used is not included above.

DEBUG *switch* [MEM | MEMORY] [TIMER] [TYPES] [IO]
The DEBUG option sets debugging on or off for the whole calculation. However the setting can be over-ridden for any module by a DEBUG option in that module. The MEMORY sub-option tracks memory allocation, the TIMER sub-option prints timing information for subroutine entry and exit, the TYPES sub-option prints information about the user-defined types used in the program, and IO prints information about input and output.

Setting any of these debugging options on does not mean information will be printed out. This would be overwhelming. Rather, these options provide specific debugging information about only those modules in which the DEBUG command has been used.

9.2 Molecule definition

Each molecule needs a main basis set and an auxiliary (density-fitting) basis set. For a property calculation there will be just one molecule; for a calculation of intermolecular interactions there will be two. The molecules are identified by name in most modules; the names are case-sensitive. The first two molecules to be defined may also be referred to as A and B, and interaction energies will be calculated between these two by default.

Notes:

- Basis functions can be in SPHERICAL or CARTESIAN form and this needs to be specified.
- CamCASP can read basis sets in three formats: GAMESS (actually, GAMESS(US)), NWChem and TURBOMOLE. This is specified by the FORMAT option.
- For the main basis, the sites must be specified in the same order as was used in the CADPAC, DALTON, NWchem or GAMESS(US) runs. *This is important*. The site name, nuclear charge and position are followed by the basis set specification, which can be typed in or #INCLUDEd from a file. Explicit paths are needed. The site definition is terminated by a line containing three dashes: -.
- The auxiliary basis is defined in the same way, but it need not have the same number of sites, nor do they need to be in the same order as the main basis. They are completely independent. However, for sensible results you'd be better off keeping all sites.

MOLECULE *name* [AT $x\ y\ z$] [ROTATED BY *angle* ABOUT $n_x\ n_y\ n_z$]
  options
  data
  main basis specification
  auxiliary basis specification
END

The molecule is taken to be at the global origin and not rotated unless otherwise specified.

The options are:

CHARGE q
The total charge on the molecule, in units of the proton charge.

ECHO *switch*
Switch echoing of input data on or off. This may appear in several places to switch the echoing of particular items.

If the molecular orbitals and Hessians are not needed, use the command
SKIP MOS

Otherwise, the molecular orbitals and Hessians should have been calculated at a previous stage, and they are attached to the molecule description by the following commands:
{ MO | MOS | MO-FILE } [in] [file] *file-name* [ASCII | BINARY | ASCII-1]
{ H1 | H1-FILE } [in] [file] *file-name*
{ H2 | H2-FILE } [in] [file] *file-name*
HESSIAN-FORMAT {SAPT2002 | SAPT2006}
H1 and H2 refer to the electric and magnetic Hessians used in the polarizability calculation. The orbitals and

Hessians must have been calculated for the basis set and molecular geometry defined in the BASIS blocks. If a rotation is applied to the molecule the orbitals and Hessians are rotated too and need not be recalculated, but this only works at present if SPHERICAL basis sets are used. The HESSIAN-FORMAT command is used to specify the format the Hessians have been written in. Use SAPT2002 or SAPT2006 depending on the patch you have used to build DALTON. The default is SAPT2006.

The MO file is expected to be in ASCII format by default.

A special ascii format, ASCII-1 is also available. In this case the file format is

MO N
$C(1,1)\ C(1,2)\ C(1,3)\ \ldots\ C(1,N)$
$C(2,1)\ C(2,2)\ \ldots\ C(N,N)$
ENERGY N
$E(1)\ E(2)\ \ldots$

Here $C(m,i)$ is the coefficient of the m^{th} atomic orbital in the i^{th} molecular orbital, and $E(i)$ is the energy of the i^{th} molecular orbital. The keywords MO and ENERGY must appear at the start of a new line, but otherwise the numbers are read in free format, separated by space or newline. N is the number of molecular orbitals.

Each basis set specification takes the form

```
BASIS { MAIN | AUX }
  [ CARTESIAN | SPHERICAL ]
  [ FORMAT { GAMESS | TURBOMOLE | NWCHEM} ]
  [ ECHO [ ON | OFF] ]
  [UNITS length units]
  label   charge   x   y   z   [TYPE  site-type  ]
  [LIMIT limit]
  basis specification for this atom
  ---
  label   charge   x   y   z
  [TYPE  site-type  ]
  [LIMIT limit]
  basis specification
  ---
  ⋮
END
```

At present it is necessary to give the atom positions twice, once for the main basis specification and once for the auxiliary basis. This makes it possible to use different sites for the main and auxiliary basis sets. However the entire molecule definition is normally provided automatically for you by the CLUSTER program.

The limit option applies to individual atoms, and takes the form
LIMIT { S | P | D | F | G }
Limit the basis to this symmetry type, i.e. omit basis functions of higher angular momentum. The basis set is usually specified by including it from a suitable file; for example:

```
#include /usr/local/camcasp/basis/gamess_us/sadlej/C
```

and the LIMIT command allows just (say) the *s*, *p* and *d* functions from the standard basis to be used. The default is currently LIMIT G, as CAMCASP can't currently handle *h* or higher functions.

FORMAT { GAMESS | TURBOMOLE }
The basis function definitions that follow are in the format used by GAMESS(US) or TURBOMOLE, as specified. Default is GAMESS(US). The auxiliary basis library supplied with the CAMCASP program uses basis sets in the TURBOMOLE program format. Note that the NWCHEM format is not available for this auxiliary basis set.

[CARTESIAN | SPHERICAL]
Use Cartesian basis functions (6-component *d* functions) or spherical (5-component *d* functions). Default is Cartesian as this results in a perceptible increase in accuracy of the density-fitting procedure. However the resulting increased linear dependencies could cause problems in the density-fitting procedure.

UNITS { A | ANG | ANGSTROM | BOHR | AU | A.U.]}
Atom positions will be given in the specified unit. Only needed if this is different from the global unit.

9.3 The EDIT module: Modifying a molecular specification

Once defined, the molecule can be manipulated in much the same way as in the CLUSTER program. There is one caveat: The commands involving a rotation will work only with main basis sets of spherical Gaussian type orbitals (GTOs). The command takes the form

```
EDIT molecule-name
  options
END
```

and the options are as follows:

`{ TOL | TOLERANCE }` *tolerance*
Specify tolerance used in finding rotations and comparisons.

`RESTORE`
Restore the molecule to its original configuration, as set up in the `MOLECULE` declaration, and restore the original molecular orbitals.

Note that in the present version of CAMCASP the Hessians are not rotated. This has consequences when an `ENERGY-SCAN` is performed.

`CENTRE [on/at] [[SITE]` *site* `| COM | XYZ` *x y z* `]` Define the molecular centre to be at the position given. If Cartesian coordinates are given, they are taken to specify a position within the molecule as originally defined, i.e. in local molecular axes which rotate and translate with it.

`{ WRITE | PRINT | SHOW } [ {MO | MOS} {GEOM | GEOMETRY} ]`
Print the molecular orbitals or the molecular geometry on standard output.

`BOND { -> | TO } { POINTS | XYZ |` *molecule* `}`
`  SITES` a_3 a_2 a_1 `{ -> | TO }` b_1 b_2 b_3
`  { DISTANCE | LENGTH | SEPARATION }` *distance* `[ ANGSTROM | ANG | BOHR ]`
`END`
(Implemented but not fully tested.) Moves this molecule (molecule A) so that sites a_2, a_1, b_1 and b_2 are collinear and arranged in that order, and so that sites a_1 and b_1 are the specified distance apart. If molecule A is non-linear, a_3 is a third site lying off the $a_1 - a_2$ axis, and molecules A is rotated about the $a_2 - a_1$ axis so that all six sites are coplanar, with sites a_3 and b_3 *cis* to each other. If A is linear, then a_3 is irrelevant, but at present it must still be specified and must still lie off the $a_1 - a_2$ axis. It can be specified as a position in global coordinates using the syntax `XYZ` *x y z*. Sites b_1, b_2 and b_3 may be in another molecule B, or may be specified as cartesian positions in global coordinates using the syntax `XYZ` *x y z*.

`INVERT`
Invert the site positions with respect to the molecular origin. (Not yet implemented.)

`ROTATE BY` *angle* `[ DEGREES | RADIANS] ABOUT` n_x n_y n_z
Rotate the molecule about an axis in the specified direction, through the molecular centre. The rotation is not carried out until an `UPDATE` command is encountered.

`TRANSLATE BY` *distance* `[ ANGSTROM | ANG | A | BOHR | AU ] ALONG` *x y z*
The distance is assumed to be in the global units unless otherwise specified. The vector direction need not be a unit vector — it is normalized by the program. The translation is not carried out until an `UPDATE` command is encountered.

`{ PLACE | POSITION | PUT ] [ at ] [ XYZ | POINT ]` *x y z*
Position (translate) the molecule so that its defined centre coincides with the specified point. The translation is not carried out until an `UPDATE` command is encountered.

`UPDATE [MOS | ORBITALS]`
Carry out any translation and rotation that has been specified. Note that the rotation is about the molecular centre, which moves with the molecule, so translation and rotation commute. If `ORBITALS` is specified, the molecular orbitals and density matrix are rotated with the molecule.

`NEIGHBOURS [TYPE [=] [ONE | ALL | OVR]] [EPS [=] <value|0.0>] [PRINT]`
Every atom in a molecule has a list of neighbours. By default, this list contains every other atom, but it can be changed using this command. There are three neighbourhood types that can be defined using the `TYPE` option. Type `ONE` defines only one 'neighbour' for every atom: the atom itself. Type `ALL` is the default and defines all sites

as the neighbourhood. Finally, type OVR uses an overlap criterion and the value of EPS to decide which sites are to be included in the neighbourhood. The latter is a basis and geometry-dependent criterion. We have not tested this feature adequately, but it seems like a reasonable choice for EPS is 0.01. The list of neighbours for every atom in the molecule can be printed out using the PRINT option.

The reason for implementing a concept of neighbourhoods is to speed up the evaluation of integrals. Using
NEIGHBOURS TYPE = OVR EPS = 0.01
we can significantly speed-up the evaluation of the numerical integrals needed for the ISA module, without a significant reduction in the overall accuracy of the calculation. But it must be borne in mind that the largest speed-ups are obtained for linear systems.

9.4 Density-fitting

ssec:df

Significant computational gains are achieved by expanding orbital products $\rho_{ij} = \phi_i\phi_j$ as a single index expansion:

$$\rho_{ij}(\mathbf{r}) \approx \tilde{\rho}_{ij}(\mathbf{r}) = \sum_k d_{ij,k}\chi_k(\mathbf{r}).$$

This is usually achieved by the minimization of the functional:

$$\Delta_{ij} = \iint [\rho_{ij}(\mathbf{r}_1) - \tilde{\rho}_{ij}(\mathbf{r}_1)]\frac{1}{r_{12}}[\rho_{ij}(\mathbf{r}_2) - \tilde{\rho}_{ij}(\mathbf{r}_2)]d\mathbf{r}_1 d\mathbf{r}_2$$

In CamCASP we additionally include constraints to impose the orthonormality of ρ_{ij}:

$$\int \rho_{ij}(\mathbf{r})d\mathbf{r} = \delta_{ij},$$

and localization constraints that are used to obtain distributed polarizabilities [Misquitta and Stone, 2006]. The latter are optional and should not be used in interaction energy calculations. This gives us two types of functional that we minimize:

- In the first we include a term that minimizes the inter-site repulsion (the default):

$$\Xi^{\mathrm{A}}_{ij} = \Delta_{ij} - \eta \sum_{a,b\neq a} E^{ab}_{ij} + \lambda\left(\int \tilde{\rho}_{ij}(\mathbf{r})d\mathbf{r} - \delta_{ij}\right)^2.$$

- In the second we include a term that maximizes the site self-repulsion:

$$\Xi^{\mathrm{B}}_{ij} = \Delta_{ij} + \eta \sum_{a} E^{aa}_{ij} + \lambda\left(\int \tilde{\rho}_{ij}(\mathbf{r})d\mathbf{r} - \delta_{ij}\right)^2.$$

where

$$E^{ab}_{ij} = \iint \frac{\tilde{\rho}^a_{ij}(\mathbf{r}_1)\tilde{\rho}^b_{ij}(\mathbf{r}_2)}{r_{12}}d\mathbf{r}_1 d\mathbf{r}_2.$$

and $\tilde{\rho}^a_{ij}(\mathbf{r})$ is the (transition-)density associated with site a which is defined as

$$\tilde{\rho}^a_{ij}(\mathbf{r}) = \sum_{k\in a} d_{ij,k}\chi_k(\mathbf{r}). \tag{5}$$

For details and numerical results see Misquitta & Stone (2006) [Misquitta and Stone, 2006].

In addition to the above, the following constraint can be optionally included:

$$C_{ij} = \gamma \sum_a \left[\int \rho^a_{ij}(\mathbf{r})d\mathbf{r}\right]^2. \tag{6}$$

This constraint forces the (transition-)density 'charge' on each site to be close to zero. The larger γ is, the more strongly this condition is enforced. This constraint should be used with the transition densities only, and even then with caution. When used together with a distributed polarizability calculation, this constraint has the effect of suppressing the charge-flow polarizabilities.

This gives us a number of choices for the density-fitting. We recommend the following in everyday usage:

- **For energy calculations** use only the orthonormality constraint, that is, either Ξ^{A}_{ij} or Ξ^{B}_{ij} with $\eta = 0.0$ and $\lambda = 1000.0$.
- Do the same for **total molecular properties** such as the molecular density, quantities that depend on the total density, and total polarizabilities.
- For **distributed polarizabilities** perform the transition-density fitting using either Ξ^{A}_{ij} or Ξ^{B}_{ij} with $\eta = +0.0002$ and $\lambda = 1000.0$.
- For the **SRLO** method of Rob & Szalewicz [Rob and Szalewicz, 2013] use Ξ^{B}_{ij} with the additional constraint C_{ij} and $\gamma = 1.0$.

Many of the modules require density-fitting and while some will perform the density-fitting automatically if it hasn't already been done, it is generally a good idea to perform it explicitly.

The (ortho)normality constraints are checked if `PRINT...NORMALIZATION...` is used. The localization constraint can be selected using the `CONSTRAINT` command.

`BEGIN { DF | DENSITYFITTING | DENSITY-FITTING}`
options
`END`

Or, to set parameters without performing the density-fitting:

`SET { DF | DENSITYFITTING | DENSITY-FITTING}`
options
`END`

The options are:

`[MOLECULE` *name* `| DIMER]`
Apply the fitting procedure to the specified molecule if `MOLECULE` is specified, or, if `DIMER` is specified, perform the density-fitting for the dimer consisting of the first two molecules in the input file. That is, the dimer is assumed to comprise molecules `A` and `B`. At present, there is no way to change the molecules in the dimer.

`TYPE [OO] [OV | VO] [FULL | NN] [RHO] [RHO-W]`
Specifies the type of density-fitting to perform. `OO` applies density-fitting only to occupied–occupied orbital pairs, needed for the density. `OV` does the occupied–virtual pairs, which are needed for all second-order terms. `FULL` does all pairs, and is the default.

For the dimer, only types `OO` and `OV` are available. Selecting type `FULL` will result in an error.

There are two special DF types:

- `RHO` fits the density directly.
- `RHO-W` is an experimental option that fits the difference of the density and (spherical) pro-atom densities. These would typically be obtained from an iterated stockholder atoms calculation. The site-charge constraint C_{ij} can be used in this context to enforce zero site charges of the `RHO-W` fit. This is an experimental option. Please contact the authors if you are interested in this.

`SOLVER { LU [[with] ITERATIONS` *iterations* `]`
`| GELSS [[with] CONDITION` *condition-number*`] }`
Specifies the linear-equation solver to use. The LU solver is fastest but gets unstable for larger basis sets. In principle the solution can be improved by iterating if the density-fitting equations are ill-conditioned, but we have not experimented with this much and the default number of iterations is 0.

Usually the auxiliary basis set will be roughly three times the size of the main basis. But this may not be the case if, say, you are using a `Sadlej-pVTZ` main basis together with a R12 `aug-cc-pVTZ` auxiliary basis which can be four to six times as large as the Sadlej main basis. In this case, the density-fitting linear equations will be near singular and `LU` will not result in reliable results. These equations can also be ill-conditioned for large, dense systems. For these cases we suggest the `GELSS` solver which uses singular value decomposition (SVD) to solve the density-fitting equations. The effective rank of the problem is determined by the condition number set using

CONDITION which has a default value of 10^{-8}. Bear in mind that GELSS requires more memory and processing resources than LU.

RESTART
Restart the density-fitting, using matrices calculated previously.

Note: Use this with care as the matrices will not be correct if the density-fitting options have changed.

ETA [=] η
η is the coefficient in front of the self-repulsion constraint term. Default = 0.0, i.e., no constraint. For distributed polarizabilities use η = +0.0005 with the INTER-SITE constraint. We have no recommendations for the SITE self-repulsion constraint.

LAMBDA [=] λ
λ applies to the (ortho)normality Lagrange multiplier. Default value 1000.0.

GAMMA [=] γ
γ applies to the SITE self-repulsion constraint only. It is new and still being experimented with. Default value 0.0. For the SRLO distribution method use γ = 1.0 together with CONSTRAINT SITE self-repulsion with η = 0.0002 and λ = 1000.0.

CONSTRAINT [type] [SITE | INTER-SITE] [repulsion | self-repulsion] [SRLO]
Implement one of the two localization constraints described above. The default is the INTER-SITE constraint.

If SRLO is chosen, then the parameters for the SRLO method described by Rob & Szalewicz [Rob and Szalewicz, 2013] are chosen: the SITE self-repulsion constraint is used with λ = 1000.0, η = 0.002 and γ = 1.0. These parameters can be altered by subsequent ETA, LAMBDA and GAMMA declarations.

TOL [=] *tolerance*
Tolerance used in DF tests. Default 0.001.

PRINT { NOTHING | EVERYTHING | ALL }
PRINT [ONLY] REPULSION and NORMALIZATION [constraint/s] and SOLUTION }
Default is to perform the density-fitting tests and print the results. This is a good idea as (ortho-)normality violations are usually indicative of problems either with the density-fitting or with an inconsistency in the molecular geometry or orbitals.

REDO-DF-ON-ROTATION [TRUE | FALSE]
Repeat the density-fitting if the molecule has been rotated. The default is TRUE.

RESET
Parameter changes are normally saved. To reset to the default parameters, use this command.

DEBUGGING { *switch* | EXTREME [[and] MATRIX | MATRICES *switch*}
Default is OFF, and with debugging on, the default action is not to print the matrices. (They are large.)

[QUIET | VERBOSE]
Default is QUIET.

9.4.1 The DF-OLD and DF-FULL modules

These are earlier versions of the density-fitting code, which are still available but are not recommended for normal use.

9.5 Propagator settings

This module is the heart of CamCASP. The propagator is used for calculations of the polarizability, induction energies and dispersion energies. This module is also in a state of change and this version of the CamCASP program contain two propagator modules. The newer module uses the method of Bukowski et al. [2005] to reduce the computational effort needed to calculate the propagators. But it is still incomplete and only some types of propagator can be calculated using this approach. Consequently we have still made available the older module. We will describe both below.

A few levels of theory can be used for constructing the propagator. These are:

CKS : The coupled Kohn–Sham propagator, which is the most accurate.

CHF : Coupled Hartree–Fock.

UCKS : Uncoupled Kohn–Sham.

UCHF : Uncoupled Hartree–Fock.

The CKS propagator has additional possibilities, as you can add some fraction of the CHF propagator to it. This is usually the case when a hybrid exchange-correlation functional, such as PBE0, is used. In this case you have the option of constructing the propagator using the PBE kernel only, or the (faster) adiabatic local density approximation (ALDA) kernel, or part PBE/ALDA kernel and part CHF kernel.

Within linear-response Hartree–Fock/Kohn–Sham theory, the frequency-dependent density susceptibility (FDDS) function is defined as

$$\alpha(\mathbf{r},\mathbf{r}'|\omega) = \sum_{n\neq 0} \frac{2\omega_{no}}{\hbar(\omega_{no}^2-\omega^2)} \langle 0|\hat{\rho}(\mathbf{r})|n\rangle\langle n|\hat{\rho}(\mathbf{r}')|0\rangle, \tag{7}$$

and $\hat{\rho}(\mathbf{r})$ is the electron density operator $\sum_i \delta(\mathbf{r}-\mathbf{r}_i)$, where i runs over the electrons in the system. The FDDS describes the linear response of the electron density to a frequency-dependent perturbation. Two practical ways of evaluating the FDDS are coupled Hartree–Fock (CHF) theory and coupled Kohn–Sham (CKS) theory. Of these, the latter is in principle exact, while the former is, by definition, an approximation and will not be used in this work. The FDDS evaluated in both CHF and CKS theories can be written in the form

$$\alpha(\mathbf{r},\mathbf{r}'|\omega) = \sum_{iv,i'v'} C_{iv,i'v'}(\omega)\phi_i(\mathbf{r})\phi_v(\mathbf{r})\phi_{i'}(\mathbf{r}')\phi_{v'}(\mathbf{r}'), \tag{8}$$

where the subscripts i and i' (v and v') denote occupied (virtual) molecular orbitals, $C_{iv,i'v'}(\omega)$ is related to the polarization propagator, and ϕ_l is a molecular orbital. The coefficients $C_{iv,i'v'}(\omega)$ are conveniently written as

$$C_{iv,i'v'}(\omega) = 4[(\mathbf{H}^{(2)}\mathbf{H}^{(1)} - \hbar^2\omega^2\mathbf{I})^{-1}\mathbf{H}^{(1)}]_{iv,i'v'}, \tag{9}$$

where $\mathbf{I}$ is the unit matrix and the $\mathbf{H}^{(1)}$ and $\mathbf{H}^{(2)}$ matrices, called the electric and magnetic Hessians, respectively, are defined in CKS theory (in the adiabatic approximation [Casida, 1995, Colwell et al., 1995]) as follows:

$$\mathbf{H}^{(1)}_{iv,i'v'} = (e_v-e_i)\delta_{iv,i'v'} + 4(iv|i'v') - \xi[(ii'|vv') + (iv'|i'v)] + 4\int \phi_i\phi_v\phi_{i'}\phi_{v'}\frac{\delta(v_{\rm xc}-\xi v_{\rm x})}{\delta\rho}\mathrm{d}^3\mathbf{r}, \tag{10}$$

and

$$\mathbf{H}^{(2)}_{iv,i'v'} = (e_v-e_i)\delta_{iv,i'v'} - \xi[(ii'|vv') - (iv'|i'v)], \tag{11}$$

where ξ is the fraction of the Hartree-Fock exchange included in the exchange-correlation (XC) functional ($\xi = 0$ for a continuum functional), $v_{\rm x}$ is the exchange part of $v_{\rm xc}$, and the two-electron coulomb integral $(ij|kl) = \langle\phi_i(1)\phi_j(1)|r_{12}^{-1}|\phi_k(2)\phi_l(2)\rangle$. This module calculates the FDDS expansion coefficients $C_{iv,i'v'}(\omega)$.

The direct evaluation of these coeffcients through eq. (9) incurrs a computational cost of $O(o^3v^3)$ and a memory cost of $O(o^2v^2)$. This can be prohibitive and is usually not feasible for large systems. However this is not necessary for as shown by Misquitta et al. [2003] and by Podeszwa et al. [2006], we use the density-fitted form of the FDDS in all SAPT(DFT) calculations. In this case, for local and semi-local density functionals (for which $\xi = 0.0$), Hesselmann et al. [2005] have shown that the computational cost of evaluating the DF-FDDS coefficients is only $O(ovM^2)$ where M is the size of the auxiliary basis set. Furthermore, only $O(M^2)$ memory is used in this approach.

For the more general case when $\xi \neq 0.0$, that is, for a hybrid functional, Bukowski et al. [2005] have shown that the FDDS can be computed with a computational cost that scales as the fifth power of the system size. At present we do not have this more general method coded in CamCASP.

With this background we are now in a position to understand the way the propagator is calculated in CamCASP. The PROPAGATOR module calculates the most general form of the propagator. The syntax for this module is:

```
SET PROP/PROPAGATOR
  TYPE cks/ucks/chf/uchf +++
      [WITH/PLUS/AND FRACTION-CHF-KERNEL <fraction of CHF kernel>]
  {FRACTION-CHF-KERNEL | C-X-KERNEL} <fraction of CHF kernel to be included>
  {FRACTION-CHF-FUNCTIONAL | C-X-FUNCTIONAL} <fraction of HFX in functional>
```

```
  HESSIAN/S [from] [DALTON | CADPAC | INTERNAL]
  DENSITY-FITTING/DF/DENSITYFITTING [T TRUE YES | NO F FALSE] +++
         [WITH | WITHOUT] [CONSTRAINTS]
  DIRECT/DF-INTS/DF-INTEGRAL/s [integral/s]
  SOLVER [LU [ITERATIONS [=] <number of iterations|0> ] +++
         [GELSS [CONDITION [=] <condition number>]
  KERNEL [ALDA/ALDAX]/[ALDA+CHF/FULL]
  DEBUGGING ON/OFF/EXTREME
END
```

These options are described in sec. 9.5.1. Use this module for linear-response calculations with **hybrid density functionals only**. While it will also work for standard GGAs like PBE, as explained above, this module is computationally expensive and is not suited for systems with more than 20 atoms.

For all non-hybrid kernels (i.e., $\xi = 0.0$ in eqns. (10) and (11)) use the NEW-PROP module:

```
SET NEW-PROP
  RESET
  PROPAGATOR-TYPE [UCKS | UCHF | CKS | ALDAX | ALDA ]
  {FRACTION-HF-FUNCTIONAL | CX-FUNC | CX-FUNCTIONAL}
                                  <fraction of HFX in functional>
  {FRACTION-CHF-KERNEL | CX-KERNEL } <fraction of CHF in kernel>
  [{CONSTRAINED-DF | C-DF} | DF]
  KERNEL-INTEGRAL-PARAMETERS
    INFINITY-CONTROL-METHOD [ZERO | CONSTANT | FD (default)]
    F-MAX    [=] <value| default=1000.0>
    RHO-EPS  [=] <value| default=1e-8>
    FD-DELTA [=] <value| default=0.01>
    FD-ALPHA [=] <value| default=1.0>
    KERNEL-INTEGRAL-CUTOFF [=] <value|default=par_kernel_integral_cutoff>
  END
  SOLVER [LU [ITERATIONS [=] <number of iterations|0> ] +++
         [GELSS [CONDITION [=] <condition number>]
  [QUIET | VERBOSE]
  DEBUG [ON | OFF | EXTREME]
END
```

These options are as follows:

`RESET`
Reset all parameters to the defaults.

`PROPAGATOR-TYPE [UCKS | UCHF | CKS | ALDAX | ALDA ]`
Defines the type of propagator. Options are:

- `ALDA`: The adiabatic LDA kernel with the Slater exchange and PW91 correlation kernel. `CKS` is currently synonymous with the `ALDA` kernel.
- `ALDAX`: The adiabatic Slater exchange kernel. This would typically result in errors of about 2-3% compared with the `ALDA` kernel.
- `UCKS/UCHF`: The un-coupled Kohn–Sham and Hartree–Fock kernel.

`FRACTION-HF-FUNCTIONAL` *fraction*
Fraction of Hartree–Fock exchange. While not applicable to this module, it can be used to change settings in the `PROPAGATOR` module.

`[{CONSTRAINED-DF | C-DF} | DF]`
Choose the type of density-fitting used in calculating the propagator. `C-DF` sets the density-fitting solution with constraints and `DF` uses the density-fitting solution without constraints.

SOLVER [LU [ITERATIONS [=] <number of iterations|0>] +++
[GELSS [CONDITION [=] <condition number>]
Set the solver used for solving the propagator equations. Normally LU will suffice, but for higher accuracies use GELSS. We do not recommend iterations with the LU solver. The condition number for the GELSS solver should be around 10^{-15}. The default is the value of par_condition_number in parameters.F90 and is currently 10^{-15}.

```
KERNEL-INTEGRAL-PARAMETERS
  INFINITY-CONTROL-METHOD [ZERO | CONSTANT | FD (default)]
  F-MAX    [=] <value| default=1000.0>
  RHO-EPS  [=] <value| default=1e-8>
  FD-DELTA [=] <value| default=0.01>
  FD-ALPHA [=] <value| default=1.0>
  KERNEL-INTEGRAL-CUTOFF [=] <value|default=par_kernel_integral_cutoff>
END
```

The kernel integrals used to define the Hessian elements are susceptible to numerical noise as the ALDA kernel contains terms involving negative powers of the density which diverge for large-**r** where the density tends to zero. Three ways of controlling this divergence are provided:

- ZERO: Where the kernel integrand is greater than F-MAX, set the kernel to 0.0.
- CONSTANT: Here, where the kernel integrand is greater than F-MAX we instead set it equal to F-MAX.
- FD: Here we make the kernel integrand tend to zero smoothly using a Fermi–Dirac-type switching function with parameters set by FD-DELTA and FD-ALPHA. The former sets the width of the FD switching and the latter sets the asymmetry in the switching by raising the FD function to the power specified. It seems like this parameter should not deviate from 1.0 by much so we have set it to 1.0.

For details of how this works see dft_Sx_PW92c.F90. The defaults should normally be sufficient and should be modified only if absolutely necessary.

9.5.1 The PROPAGATOR module

This is the older and more computationally demanding propagator module, but it is still used when the ALDA+CHF propagator is desired.

```
SET { PROP | PROPAGATOR }
  settings
END
```

The possible settings are:

TYPE { CKS | CHF | UCKS | UCHF }
KS is Kohn–Sham; HF is Hartree–Fock; C is coupled, UC uncoupled. Default is CKS.

{ FRACTION-CHF | C-X } *fraction*
Fraction of exact exchange to include in the CKS. Must be in the range [0.0, 1.0]. Default 0.25.

{ HESSIAN | HESSIANS } [from] { DALTON | CADPAC }]
See below. Default is DALTON.

[DF | DENSITY-FITTING | DENSITYFITTING *switch*] [{ WITH | WITHOUT } CONSTRAINTS]
Default is OFF. Constraints will be needed if distributed polarizabilities are to be calculated.

[DF-INTEGRALS | DF-INTS]
The 2-electron integrals for the Hessians are to be constructed using a density-fitting algorithm. If this is set, make sure you have made a call to module DF_FULL beforehand. This option is very useful for large problem sizes or when the TRAN code in Sapt2006 cannot be conveniently called. It will be made the default option once tests are complete.

KERNEL { { ALDA | ALDAX } | { ALDA+CHF | FULL } }
Default is FULL. See below.

`[ DEBUG | DEBUGGING] [ switch | EXTREME ]]`
Default is `OFF`. Set `ON` for minor debugging printout. `EXTREME` prints out Hessian, integrals, etc.; use with caution.

Note on Hessians: If the propagator type is `CKS`, the Hessians are not completely constructed within this module, and at least part needs to be read from file.

- Cadpac 6.5: The H_1 and H_2 Hessians are read from files `h1A.data` and `h2A.data` (or `h1B.data` and `h2B.data`). The diagonal energy term $(e_i - e_j)$ is not included in these files and so must be added internally.
- Dalton: Only the term $\int \phi_a\phi_i\phi'_a\phi'_i(d(v_{xc} - c_x v_x)/d\rho)dr$ is included in the file `h1A.data` (or `h1B.data`). The rest of H_1 and all of H_2 are constructed within this module. Also, when Dalton is used, the fraction of exact exchange to be included needs to be explicitly specified, or else the code will stop (see `FRACTION-CHF`).

`KERNEL` setting: If set to `ALDA` or `ALDAX` the adiabatic local density approximation is to be used. `ALDA` includes the exchange–correlation term, while `ALDAX` includes exchange only. It is assumed that the integral included in `h1A.data` (or `h1B.data`) uses ALDA, but possibly with $c_x \neq 0$, so that the term in `h1A.data` is $(1 - c_x) * \int \phi_a\phi_i\phi'_a\phi'_i d(v_x)/d\rho dr$. (Put $v_{xc} = v_x$ in the expression given earlier.) This is experimental and is not the default.

9.6 Quadrature settings

The quadrature procedure may need to be specified if dispersion energies or coefficients are required. Here too the entrire section may be omitted if the defaults are acceptable.

`SET { QUADRATURE | QUAD }`
settings
`END`

The possible settings are:

`TYPE` *type* `[`*option* `]`
The type may be `GAUSS-LEGENDRE` (or `GAUSSLEGENDRE` or `GAULEG`), or `GAUSS-LAGUERRE` (or `GAUSSLAGUERRE` or `GAULAG`). The *option* is relevant only for Gauss–Legendre, where it specifies the transformation between the $0 < \omega < \infty$ range of (imaginary) frequency and the $-1 < t < 1$ or $0 < t < \pi/2$ range of the Gauss–Legendre quadrature. Allowed values are:

1: use the $\beta \times (1 + t)/(1 - t)$ transformation.

2: use the $\beta \tan(t)$ transformation (not recommended).

The default is Gauss–Legendre with option 1.

`BETA` β
Set the base frequency β $(= \omega_0)$ for the Gauss–Legendre transformation. Default 0.5 Hartree.

`ALPHA` α
Value of α for the Gauss–Laguerre transformation. Default 0.

9.7 The `ISA` module

The ISA (iterated stockholder atom or iterated spherical atom) method is a procedure for separating the molecular electron density into atomic components[Lillestolen and Wheatley, 2009, Misquitta et al., 2014]. In brief, each atom a is associated with a spherically-symmetrical weight function $w_a(\mathbf{r})$, and the electron density is allocated between atoms in proportion to the weight function at each point:

$$\rho_a(\mathbf{r}) = \rho(\mathbf{r}) \frac{w_a(\mathbf{r})}{\sum_b w_b(\mathbf{r})}.$$

In the ISA method, the weight functions are required to satisfy

$$w_a(\mathbf{r}) = \langle \rho_a(\mathbf{r}) \rangle;$$

where the angle brackets denote a spherical average. That is, each weight function is the spherical average of the corresponding atom density. When both of these equations are satisfied for every atom, the sum of the weight functions (the *promolecule* density) is the best possible approximation to the electron density as a sum of spherical densities, and the ρ_a provide an exact partition of the density into atom densities that are as nearly as possible spherical.

The equations have to be solved iteratively, and achieving convergence rapidly and reliably is difficult. See Misquitta et al. [2014] for details of the method. Various options are provided for adjusting the parameters of the method to improve convergence, though the suggested values will usually be satisfactory.

The ISA data section in CamCASP takes the form

```
BEGIN ISA
   DF-TYPE [=] { DOO | DOO-C | DRHO (default) | DRHO-C }
   ISA-TYPE [DENSITY (default) | TRANSITION-DENSITY | OV]
   ISA-METHOD [A | B1 | B2 | DF+ISA | A+DF (default) ] [ZETA [=] ζ_ISA (default=0.9)]
   W-INIT [=] { ONE-GTO ALPHA0 [=] alpha0 | ALL-GTOS | DF | DRHO-C }
   DF-PARAMETERS LAMBDA [=] λ
   SOLVER { LU (default) | GELSS CONDITION [=] condition-number }
   CONVERGENCE
     options
   END
   TAIL-FIX
     options for tail control
   END
   RESTART
     options for restart
   END
END ISA
```

- The `DF-TYPE` option specifies the density-fitting method to be used. Options are the occupied–occupied solution without constraints, `DOO`, or with constrains, `DOO-C`, and the analogous density expansions `DRHO` and `DRHO-C`. If a density-fitting calculation of `TYPE OO` has been done, then both `DOO/-C` and `DRHO/-C` will be present.
- `ISA-TYPE` specifies whether the method is to be applied to the total electron density or to the occupied-virtual transition densities. At present, only the `DENSITY` option is available.
- `ISA-METHOD` specifies the algorithm to be used. Of the options available, we recommend one of `A`, `A+DF` and `DF+ISA`. The `A` algorithm (this corresponds to functional *stock(A)* in [Misquitta et al., 2014]) is appropriate for large systems, as with an appropriate definition of the atomic `NEIGHBOURS` (see sec. 9.3), this algorithm scales linearly with system size for sufficiently large molecules. However, the `A` algorithm results in a small (10^{-3} electrons or so) charge violation with the supplied ISA *s*-function basis sets. The `A+DF` algorithm fixes this by first converging the `A` algorithm, and then modifying the functional minimised to include some fraction of the density-fitting functional. This fraction is controlled using the $\zeta_{\rm ISA}$ parameter. We should have $0 \le \zeta_{\rm ISA} \le 1$. $\zeta_{\rm ISA} = 0$ gives just the density-fitting functional and $\zeta_{\rm ISA} = 1$ gives just the ISA `A` functional. We recommend values between 0.1 and 0.9. An alternative to `A+DF` is the `DF+ISA` algorithm which uses the mixed DF and ISA functional at each iteration.
- `W-INIT` specifies the initial guess for the weight function. The choice is not critical and this option can normally be omitted.
- The `DF-PARAMETER` λ (see §9.4) should usually be set to 1000.0, and this is the default.
- The linear-equation solver is normally `LU`, but it may be necessary to use the `GELSS` method, which uses singular value decomposition.

The convergence options are

```
CONVERGENCE-TYPE [ W (default) | RHO | CHARGE | Q ]
W-DAMPING [=] η_damp (default 0.0)
EPS-NORM [=] ε_norm (default 1.0e−9)
EPS-Q [=] ε_Q (default 1.0e−3)
MAX-ITERATIONS [=] integer (default 120)
W-MIX-FRACTION [=] real (default 0.0) [SKIP-ITERATIONS integer (default 0)]
```

```
W-EPS [=] value [DECOUPLE | COUPLE (default)] +++
  [S-BLOCK-ONLY (default) | ALL-BLOCKS] [ACTIVATE-AT ϵtail (default 1.0e−5)]
TAIL-ITERATIONS [=] integer (default 30)
POSITIVE-W [LAMBDA [=] λ (default 0.001)] for [MAX-ALPHA [=] α (default 0.2)] +++
  [AUTO (default) [with] [ACTIVATE-AT [=] eps norm (default 1.0e−5)]]
[SELF-CONSISTENT-TAIL]
[SKIP-DUMMY-SITES [YES (default) | NO]]
```

Only the important commands will be described here.

- The convergence test is based on quantities Δ_a for each atom a. Δ_a is the difference between the normalized overlap between successive iterations of the weight function for atom a, and 1. ($|1 - d_a|$, in the notation of the paper.) Convergence is usually considered to have been reached when Δ_a is less than ϵ_{norm} for every atom. (See Misquitta et al. [2014].) This is `CONVERGENCE-TYPE W`. It is the default and is recommended.

- The convergence threshold is set by `EPS-NORM`. The default is 10^{-9} which is appropriate for most systems.

- The algorithm can be forced to stop after a maximum number of iterations set by `MAX-ITERATIONS`. The default is 120 which is usually sufficient.

- As described in [Misquitta et al., 2014], the tail region is given an additional weight by using a tail-weighted norm:

$$\|f\|_{\text{tail}} = \int f(\mathbf{r})\ \exp(+\epsilon|\mathbf{r} - \mathbf{R}_a|^2)\ d\mathbf{r}, \tag{12}$$

  where ϵ is a positive number that must be less than twice the smallest exponent in the basis set so as to ensure integrability. We apply this weight to the s-function block only (`S-BLOCK-ONLY`). ϵ is set using `W-EPS`. The default is $\epsilon = 0.17$ which is appropriate for the ISA basis sets supplied with CamCASP, but it may need to be made smaller if more diffuse basis sets are used. This additional weight is applied only after a certain degree of convergence has been attained; specifically, when all the Δ_a are smaller than the value ϵ_{tail} set by the `ACTIVATE-AT` option. The default is 10^{-5}.

- If the `A+DF` algorithm is used, the maximum number of iterations for the combined A and DF functionals needs to be set. This is set using `TAIL-ITERATIONS`. The default is a maximum of 30 extra iterations. They are called 'tail iterations' as the prime function of these extra iterations is to tweak the atomic density tails.

- The `POSITIVE-W` option allows the algorithm to attempt to dynamically suppress negative coefficients in the ISA expansion if these are associated with *s*-functions with exponent less than the value specified by `MAX-ALPHA`. This is normally done automatically (`AUTO`) after convergence is deemed sufficient (`ACTIVATE-AT`). This is an experimental option that can be disabled by setting `POSITIVE-W LAMBDA = 0.0`.

- Normally, as explained in [Misquitta et al., 2014], tail fixes are imposed during the ISA iterations. If `SELF-CONSISTENT-TAIL` is present, then one additional iteration is conducted without a tail fix. In a sense, this achieves a self-consistency in the atomic density tails.

- If the auxiliary basis set contains dummy sites (sites with basis functions but no nuclear charge) then the usual algorithm may fail. The `SKIP-DUMMY-SITES` option allows the algorithm to skip dummy sites when determining the convergence criterion.

An important aspect of the method is the need to control the behaviour of the weight-function tails. They can be, and usually are, constrained to decay exponentially at long distance. The options for this procedure are:

```
[NO-FIX]
ACTIVATE[-AT] [EPS-NORM [=] real (default 1.0e−6)] [MAX-ITER [=] integer (default 20)]
[FIX-DUMMY-SITES | NO-FIX-DUMMY-SITES (default)]
R1-MULTIPLIER [=] real (default 1.5)
R2-MULTIPLIER [=] real (default 3.5)
FUNC [=] [±1 | ±2 (default = +1)]
[TEST]
FIT-TYPE [=] [1 | 2 | 3 (default)]
TAG [=] string (default null)
```

- The NO-FIX option just suppresses the tail control. This option is not recommended; divergence will usually ensue.
- The ACTIVATE-AT option allows the tail corrections to be applied only after convergence is deemed sufficient (EPS-NORM) or if a maximum number of iterations (MAX-ITER) is reached. This is useful in ensuring that the tail fix is consistent with the ISA solution.
- Dummy sites are usually omitted from the promolecule, and if they are included the tail-fix should normally omitted for such sites. NO-FIX-DUMMY-SITES is the default.
- The R1-MULTIPLIER multiplies the Slater covalent radius of each atom, giving a distance from the nucleus beyond which the weight-function tail needs to be controlled by forcing exponential decay. Larger values give faster but less reliable convergence. Increasing the default to 2.0 or 2.5 may be satisfactory, particularly for larger atoms (or those which are electronegative).
- Some of the fitting methods used to determine the parameters in the tail-fix functions require a second radius which is set by R2-MULTIPLIER.
- There are two kinds of tail function that can be used: $w_1(\mathbf{r}) = A\exp(-\alpha|\mathbf{r}-\mathbf{R}|)$ and $w_2(\mathbf{r}) = Ar^{\beta}\exp(-\alpha|\mathbf{r}-\mathbf{R}|)$. The FUNC command chooses between these two. If negative values are set, then the tail fix is fitted but not applied.
- There are various ways of determining the parameters in w_1 and w_2. FIT-TYPE controls the method used and defaults to 3 which is the charge-conservation procedure described in [Misquitta et al., 2014]. This option does not require R2-MULTIPLIER.
- The ISA atom pro-atomic data is written to file and TAG can be used to modify the file name.

After an ISA calculation, the code writes out a file containing information about the ISA shape-functions, basis set, and tail-fix for each of the atoms. Restart is possible from this kind of file. It is an ASCII file with keyword-based fields, so the user may be able to edit it.

The RESTART commands take the form:

FILE *filename*
ITERATE *YES (default) or NO*
TEST *YES (default) or NO*

- The restart file is specified with FILE. This command can appear as many times as is needed, so a restart may involve the outputs of multiple ISA calculations. Atoms are matched first using the LABEL field, then using the TYPE field, and finally, if no unique match is possible using those, a match is attempted using the coordinates in the COORDS field. If no unique match is possible, none is made. If multiple files are provided, matches are made in the order the files are listed: i.e., first file comes first.
- By default, the code assumes that iterations will be resumed after the ISA solution has been read. If no iterations are desired set ITER NO.
- Likewise, after the ISA solution has been read, a few tests will be performed. These are consistency checks. The can be disabled using TEST NO.

Output from this module comprises the weight functions, tabulated in a separate file for each atom type. The CamCASP output also includes expressions for each weight function as an expansion in Gaussian functions. The ISA atom densities ρ_a are saved in the program for use in other CamCASP modules.

9.8 Multipole moments

Distributed moments can be calculated from a partition of the molecular charge density into atom densities. We recommend the use of the ISA atom densities for this purpose, but other density partitions can be used. The GDMA 2.2 program can also be used to calculate distributed multipole moments; see §9.8.2 below.

9.8.1 Distributed moments from atom densities

This module is used to calculate distributed moments using a partition of the electron density betwen atoms. Syntax:

```
BEGIN MULTIPOLES | MOMENTS | DISTMOMENTS
   [MOLECULE name (default is first molecule)]
   RANK n
   {DF | DENSITY-FITTING} [WITH | WITHOUT] [CONSTRAINTS] [TYPE [OO | RHO | RHO-W | ISA (default)]]
   PRINT [ONLY] TOTAL [and] DISTRIBUTED [moments/multipoles]
END
```

The options are:

- The MOLECULE option is only needed if more than one molecule has been defined and the required molecule is not the first one.
- RANK: Calculate multipoles up to the rank specified. Default 2, maximum 4.
- {DF | DENSITY-FITTING} [WITH | WITHOUT (default)] [CONSTRAINTS]
  [TYPE [OO | RHO (default) | RHO-W | ISA]]
  Specifies which type of density to use. The ISA expansion is recommended and is the default.

  For types ISA and RHO-W the expansions should be created before calling this module.

  Density-fitting for other types is done by this module if it has not been done explicitly in an earlier step, but bear in mind that if the density-fitting solution with constraints has been requested, the actual values used for the constraints are those set in the last call to the DF module. This may not be what you expect as many calls to to the DF module are made internally. We strongly recommend that an explicit density-fitting call be made if constraints are needed.
- PRINT *items*
  Default is to print both TOTAL and DISTRIBUTED moments.

9.8.2 GDMA 2.2 module

The GDMA 2.2 module, written by Anthony J. Stone, has been interfaced with CAMCASP and can be accessed using the commands described below. This interface allows the densities obtained from the DALTON program to be used by GDMA 2.2, albeit with some limitations. In particular, the Pople (6-31G, etc.) basis sets are not allowed. Additionally, check-point files from the GAUSSIAN program can still be used as before, without any limitations.

The GDMA 2.2 module is used as follows:

```
BEGIN GDMA
  options
END
```

If a check-point file from GAUSSIAN is to be used, enter GDMA 2.2 commands as you would normally (see the GDMA manual). If DALTON or CADPAC 6.5 is to be used, omit the FILE command which is used to specify the name of the check-point file and use the following command instead:

```
BEGIN GDMA
  ! FILE ... omit this command and use ...
  MOLECULE <molecule name>
  ! Rest of GDMA commands follow as before
  ...
  ...
END
```

Here's a sample set of commands:

```
BEGIN GDMA
  MOLECULE <molecule name>
```

```
  Bohr
  Multipoles
    Limit 4
    Limit 1 H
    Radius H 0.661
    Punch H2O_dma_L4.punch
  Start
END
```

For a description of the DMA method and options of the GDMA 2.2 module please see the manual, which should be in the same directory as this file.

9.9 Display

It is possible to construct a display file containing a surface grid suitable for display in ORIENT. The possibilities at present are to generate an isodensity surface for the whole molecule, with values of the molecular electrostatic potential on the surface, or to generate isodensity surfaces for individual atoms. The electrostatic potential is provided for an atomic surface too, but is less useful since atomic surfaces may pass close to other nuclei where the potential becomes very high. Usually the grid will be used in ORIENT with a potential generated from the multipole moments of just the atom in question.

For a molecular surface, the syntax is:

```
  BEGIN DISPLAY
     [MOLECULE name]
     [MAXPOINTS N_pts | MAXTRIANGLES N_tri]
     MOLECULAR-DENSITY
       SLATER-MULTIPLIER [=] M_slater (default 10.0)
       ISODENSITY ρ
       STEP [=] step [BOHR | ANGSTROM]
     END
  END
```

The MOLECULE line is not required if the display is wanted for molecule 1. The procedure initially sets up a rectangular grid of points at interval *step*, large enough to contain spheres round each atom of radius M_{slater} times the atom's Slater covalent radius. The electron density is evaluated at each point, and interpolated to obtain a triangulated surface grid at the specified isodensity. The electrostatic potential is then evaluated at each surface grid point. MAXPOINTS or MAXTRIANGLES specify the maximum size of the grid; only one of these needs to be specified as the other is obtained from the Euler formula. In most cases the default of 100,000 triangles should be adequate.

For atomic surfaces, the procedure is similar, except that the initial grid only needs to be large enough to enclose the individual atom sphere. The syntax is:

```
  BEGIN DISPLAY
     [MOLECULE name]
     [MAXPOINTS N_pts | MAXTRIANGLES N_tri]
     AIM-ATOMS
       ATOMS [=] [ALL | list of atom names]
       W-MIN [=] W_min
       R-MAX [=] R_max
       SLATER-MULTIPLIER [=] M_slater (default 10.0)
       STEP [=] step [BOHR | ANGSTROM]
       [ANISO-ONLY]
       AIM-METHOD [DF | ISA ] [FUNC-EXPANSION | GRID-BASED ] [with] [RHO | RHO-C]
       ISODENSITY [=] ρ (default 0.001)
       [PREFIX file prefix]
     END
  END
```

Here SLATER-MULTIPLIER, R-MAX and W-MIN are alternative ways of specifying the size of the grid. R-MAX specifies the radius of the atom sphere explicitly, and W-MIN specifies it as the distance where the weight function falls below the specified value. Any or all of these may be used; the radius taken is the largest of those specified.

If the display is required for some atoms only, the atom names can be specified on the ATOMS line. In this case,

make sure that all atoms have distinct names so the correct ones can be identified.

The file names used for the atom grids are of the form *prefix_atom*.grid, where *atom* is the name of the atom.

By default, the full atomic density (in the atom-in-a-molecule sense) is used to create the isosurface. This would include the pro-atomic density and the anisotropy. The ANISO-ONLY command causes the code to use the anisotropy only. That is, the pro-atomic (spherical) density is subtracted off.

The atomic densities can be calculated in two main ways: using a density-fitting (DF) based partitioning scheme or using the ISA scheme. These options can be set using the AIM-METHOD command which also accepts a few qualifiers. For the ISA scheme, the partitioning can be achieved using either the atomic density expansions in a basis set (this is the FUNC-EXPANSION option), or can be achieved using the ISA shape-functions through a grid-based scheme (GRID-BASED). The former is fast as it works entirely in basis-space, but it is susceptible to small negative terms that can affect the quality of the low-density isosurfaces. The latter is slower, but generally yields better isosurfaces. Finally, the type of density used can be set using RHO (no constraints) or RHO-C (with constraints). The density option affects the DF partitioning and the GRID-BASED ISA option.

9.10 Polarizabilities

```
BEGIN {POLARIZABILITY | POLARISABILITY}
  MOLECULE  <name>
  INVERT    [YES | NO]
  SQ_FREQ           omega2_1, omega2_2,...,omega2_n
  SQ_FREQ           omega2_n+1, omega2_n+2,...etc.
  QUAD              <number_of_points>
  PROPAGATOR   <prop_type> with FRACTION-CHF/C-X <fraction> [and]
       HESSIAN/S [from] DALTON/CADPAC
  {SPHERICAL | CARTESIAN}
  RANK <rank>
  CALCULATE [ONLY] TOTAL [and] DIST/DISTRIBUTED +++
            [pols/polarizabilities][and] PERTURBATIONS
  P2P-RESPONSE
     ORIENT-LATTICE
       #include <Orient lattice file>
     END
     PERTURBATION-LATTICE
       x   y   z   [q]
       ...
     END
     FILE-PREFIX <prefix of output ORIENT grid files>
     FILE-SUFFIX <suffix of output ORIENT grid files>
  END
  PRINT NOTHING
  PRINT [ONLY] TOTAL [and] DIST/DISTRIBUTED ONLY_STATIC/STATIC
       (also accepts the use of POLS/POLARIZABILITIES)
  PRINT [pols] [for] ORIENT [and] PFIT
  PRINT [upto] RANK <n>
  {CENTRE | CENTER} COM/[XYZ x,y,z] [[ANG/ANGSTROM]/[BOHR]]
  POL-FILE    pol_file_prefix
  PERT-FILE   pert_file_prefix
  PFIT-FILE   pfit_file_prefix
  PFIT-CUTOFF <pfit_cutoff>
  DEBUGGING  ON/OFF
END
```

Options for the polarizability module are:

MOLECULE *name*
The default molecule is A.

INVERT *switch*
Invert the molecule. Inversion gives polarizabilities for the enantiomorph. The changes are trivial, involving some changes of sign, but inversion can be used to check the sign changes.

SQ-FREQ $\omega_1^2, \omega_2^2, \ldots, \omega_n^2$
SQ-FREQ $\omega_{n+1}^2, \omega_{n+2}^2, \ldots$
Specify frequencies explicitly. Several lines may be used if necessary. The values given are the squared frequencies, and are negative for imaginary frequencies. Default is to calculate just the static polarizability. Non-zero frequencies are normally set in the QUADRATURE section and need not be given here, but if they are given here they over-ride any values set previously. This command cannot be used together with QUAD (below).

QUAD *number of quadrature points*
This is the second method for setting the complex frequencies at which the polarizabilities are to be calculated. Real frequencies are not possible using this command. The quadrature points are calculated using the settings in the QUADRATURE section. This command cannot be used together with SQ-FREQ (above).

The static polarizabilities will be calculated in addition to frequency-dependent polarizabilities calculated at the specified number of non-zero complex frequencies.

Use this option for calculations of polarizabilities to be used by the Pfit and Casimir programs to calculate dispersion coefficients. Ten quadrature points are enough to ensure converged dispersion coefficients.

{ SPHERICAL | CARTESIAN } [tensor form] [to] [RANK] *rank*
Polarizabilities are calculated up to the specified rank, in Cartesian or spherical-tensor form. $0 \leq rank \leq 4$. Default 2.

CALCULATE *item* [and] *item* [and] ...
Specify the quantities that are to be calculated. They can be any or all of
TOTAL [pols | polarizabilities]
[DIST | DISTRIBUTED] [pols | polarizabilities]
PERTURBATIONS
The last of these specifies 'point-to-point' polarizabilities; that is, the change in potential at a point P resulting from the change in charge distribution induced by a unit point charge at Q. These quantities are used to refine the distributed polarizabilities. It is necessary in this case to define the grid of points for which point-to-point polarizabilities are required; this is done in the LATTICE section. Default is to calculate total and distributed polarizabilities.

PRINT [[to] RANK *n*] [NOTHING | *item* [and] *item* ...
Results are always sent to suitable files, but can also be printed on standard output if required. Any or all of the following items can be specified, or NOTHING can be used to suppress this printing:

```
TOTAL [ pols | polarizabilities ]
[ DIST | DISTRIBUTED] [ pols | polarizabilities ]
[ONLY_STATIC | STATIC ] [ pols | polarizabilities ]
[polarizabilities for] ORIENT
[polarizabilities for] PFIT
```

The default is to print results up to the maximum calculated rank both in a form suitable for Orient, and in a form suitable for Pfit. Specifying a rank will limit the printing on standard output to that rank. This may be useful if the molecule has many sites and you are computing high rank polarizabilities. It does not limit the printing of polarizabilities in Orient format.

[CENTRE | CENTER] [COM | XYZ x, y, z [ANG | ANGSTROM] | BOHR]]
Take the molecular origin as specified. (Note that polarizabilities above dipole depend on the choice of origin.) The default centre is the origin of global coordinates. (This usage is deprecated. Please use the EDIT block to set the molecule centre.)

POL_FILE *pol_file_prefix*
PERT_FILE *pert_file_prefix*
PFIT_FILE *pfit_file_prefix*
If the global OVERWRITE option is NO, output in Orient form is written to file *pol_file_prefix*_nnn.dat, where nnn is a 3 digit integer chosen to avoid over-writing existing files. If the global OVERWRITE option is YES, the file-name used is *pol_file_prefix*.dat, whether it already exists or not. The default *pol_file_prefix* is pol_out.

Similarly, the perturbations (point-to-point polarizabilities) are written out in a form suitable for the Pfit pro-

gram in file *pert_file_prefix*_nnn.dat or *pert_file_prefix*.dat, and the output in PFIT form is written to file: *pfit_file_prefix*_nnn.dat. The default *pert_file_prefix* is alphaPP, and the default *pfit_file_prefix* is pfit.

PFIT-CUTOFF *cutoff* Omit polarizabilities smaller in magnitude than the specified cutoff from the file of polarizabilities in PFIT form.

9.10.1 Visualising the point-to-point responses

```
P2P-RESPONSE
   ORIENT-LATTICE
     #include <Orient lattice file>
   END
   PERTURBATION-LATTICE
     x   y   z   [q]
     ...
   END
   FILE-PREFIX <prefix of output ORIENT grid files>
   FILE-SUFFIX <suffix of output ORIENT grid files>
END
```

The point-to-point polarizabilities can be visualised on a surface (using ORIENT) when calculated using this set of commands. There are two main blocks:

- ORIENT-LATTICE...END: Specify the lattice of points on which the calculations are to be preformed. The lattice is in the form output by ORIENT in its DISPLAY module. For a complete description see the ORIENT manual. Here is an example of how this might be done:

```
Display energy
  Title "pentapyralene...Q 1.0 induction "
  Molecule pentapyralene
  ! Specify the grid:
  Radii scale 1.0
  Step 0.75 B
  Grid exp
  ! And the colour map:
  Colour-map
    0   210  0.25  1
    6   240  0.75  1
   12   300  1.0   0
   18   360  0.75  1
   24   390  0.25  1
  End
  Viewport 25
  Colour-scale min -20 max +10 top +10
  Probe X
  Ball-and-stick
  Write pentapyralene_1.0vdW.grid no values
End
```

  This example would write out the grid (points and triangles) into file pentapyralene_1.0vdW.grid. This file could be included into CAMCASP using

```
ORIENT-LATTICE
  #include pentapyralene_1.0vdW.grid
END
```

- PERTURBATION-LATTICE...END specifies the set of points at which the perturbation charges are placed. At least one point should be defined. The point-to-point perturbations are then calculated by placing the point charge perturbation at each of these points in turn, with every one of the ORIENT-LATTICE points. Each set is saved to an individual file than will contain the lattice in ORIENT format but will additionally include the value of the response. The responses can be read and displayed by ORIENT using a set of commands similar to those given above:

```
Display energy
  Title "pentapyralene...Q 1.0 induction "
  Molecule pentapyralene
  Import pentapyralene-dz-1.5vdW-1.resp with values
  Colour-map
    0   210  0.25  1
    6   240  0.75  1
   12   300  1.0   0
   18   360  0.75  1
   24   390  0.25  1
  End
  Viewport 25
  Colour-scale min -0.07 max +0.07 top +0.05
  Probe X
  Ball-and-stick
End
```

  Where the file pentapyralene-dz-1.5vdW-1.resp contains the responses for perturbation point 1.

The responses from the point charge perturbation are saved in separate files. These take the form PREFIX-<number>.SUFFIX. By default, the prefix is P2Presp and suffix resp. These can be altered using the commands FILE-PREFIX and FILE-SUFFIX.

9.11 Lattice

For point-to-point polarizabilities and energy scans it is necessary to specify the grid of points to be used. This section sets parameters for the grid but it is not constructed until needed.

```
SET LATTICE
  CHARGE   LatticeCharge
  UNITS [BOHR | ANGSTROM]
  POINTS
    ...
  {END | ---}
  RANDOM   nlat
  SEED     seed
  STEP     step
  LOWLIM   lowlim
  HIGHLIM  highlim
  GRID     <generate a grid using highlim and lowlim (see below)>
  NEWGRID  icosahedron grid.
  SPHERE   lebedev grid of radius lebrad and type nscheme.
  ORIGIN   origin(1),origin(2),origin(3)
  RADIUS   lebrad
  NSCHEME  nscheme
  DEBUG
END
```

The options are:

CHARGE *q*
Specify the charge to be associated with each lattice point. Default −1 a.u. The charge is formally arbitrary since we are dealing with linear response, but the choice may have numerical implications.

RANDOM *n*
Construct a random grid (subject to constraints below) with a total of *n* points. (Default 2000.)

SEED *seed*
Specify the seed for the random number generator. Default 1.

STEP *step*
Target grid-point spacing for non-random grids.

[LOWLIM | LOLIM] *l*
[HIGHLIM | HILIM] *h*
Confine the grid to the region between the vdW×*l* and vdW×*h* surfaces. Defaults: $l = 2.0$, $h = 4.0$.

NEWGRID
Generate a grid using a Goldberg icosahedron scheme.

ORIGIN *x y z*
Specify the centre of a spherical-shell grid.

RADIUS *lebrad*
NSCHEME *nscheme*
SPHERE
Construct a Lebedev grid of radius *lebrad* and type *nscheme*.

We recommend using a RANDOM grid for most calculations. The form of the input would be typically

```
SET LATTICE
  Charge -1
  HighLim 2.0
  LowLim  4.0
  Random
  Seed integer
END
```

Keep in mind that the SEED will need to be changed if you want a different set of random points.

9.12 Numerical integration grid

Numerical atom-grids are used by many modules including the GDMA and ISA modules and also the kernel integral module. These grids are normally automatically set by the CamCASP program, but can be adjusted using the GRID module commands:

```
BEGIN GRID | INTEGRATION-GRID
  MOLECULE name
  GRID-TYPE Lebedev (default) | Gauss-Legendre
  RADIAL-POINTS integer (default = 80)
  ANGULAR-POINTS integer (default = 590)
  BECKE-SMOOTHING-PARAMETER integer (default = 3)
  RELATIVE-RADII EQUAL | SLATER (default) | name radius name radius ...
  RADIUS-SCALING [=] real (default = 1.0)
  WRITE | FILE file name
  [DEBUG]
END
```

Atom-grids are generated using an angular grid with either Lebedev or Gauss-Legendre quadrature and number of points greater than or equalt to that specified with ANGULAR-POINTS. This is combined with an atom-centered radial grid with points specified with RADIAL-POINTS. The BECKE-SMOOTHING-PARAMETER determines how the integration weights are allocated to atoms. The default value of 3 results in a fairly smooth transition of the weights between atomic domains. The smaller the parameter is, the smoother this transition will be.

The relative sizes of the atoms are normally taken from their Slater radii (RELATIVE-RADII SLATER). This is the default. The radii govern the sizes of the atomic domains (where the weights allocated to the atom are large). The radii can be uniformly scaled using RADIUS-SCALING, or can be made equal using RELATIVE-RADII EQUAL. Or

can be set for each atom using `RELATIVE-RADII` *name radius name radius ...*. The atom names should be the atom labels used in the molecule specification. The units of the relative radii should be BOHR to make them consistent with the default radii.

Credits
The original C-code on which the module that generates the atom-grids is based was provided by Dr. Dmitri N. Laikov and translated into fortran by Dr. Christoph van Wuellen. The routines may be obtained in original form from
http://www.ccl.net/cca/software/SOURCES/FORTRAN/Lebedev-Laikov-Grids/index.shtml.

9.13 Electrostatic interaction $E^{(1)}_{\text{elst}}$

The first-order electrostatic energy $E^{(1)}_{\text{elst}}$ is calculated using integrals obtained via density-fitting. This results in errors of about 0.2% when the R12-optimized auxiliary basis sets are used. Perhaps basis sets optimized for the density should be used. This has yet to be tested.

The `PROBE` command can be used to calculate the interaction with a point-charge probe at a specified location.

```
[BEGIN | SET] {ELST | E1ELST}
  UNITS  [{A | ANG | ANGSTROM} |  {BOHR | AU | A.U.}  <must come before PROBE>
  PROBE [molecule | atom] [A | B] [with] CHARGE <charge> AT <x> <y> <z>
  COULOMB [energy] [for] [molecule | atom] {A | B}
  {REG | REGULARISE | REGULARIZE} ETA [=] <reg_eta>
  INTEGRAL [SWITCH [=] <switch>]
  [USE-DRHO {with} [CONSTRAINTS | NO-CONSTRAINTS]]
  [RESET | DEFAULTS | DEFAULT]
  QUIET
  VERBOSE
  DEBUG
END
```

Use `BEGIN` to calculate an energy and `SET` to define parameters which will be used in a subsequent energy calculation. `SET` is useful when performing `ENERGY-SCAN`s.

Options:

`UNITS [ A | ANG | ANGSTROM | BOHR | AU | A.U. ]`
This must come before `PROBE` if required.

`PROBE [ molecule | atom ] { A | B |` *name* `} [with] CHARGE` q `AT` $x\ y\ z$
Probe the electrostatic potential around the specified molecule by calculating its interaction with the specified point charge. (The option of denoting the molecule by name rather than by `A` or `B` is not yet implemented in this module.)

`{ REG | REGULARIZE | REGULARISE } ETA [=]` η
Regularize the intermolecular potential — that is, remove the Coulomb singularities, replacing $1/r_{ki}$ by $(1 - \exp(-\eta r_{ki}^2))/r_{ki}$ in the attraction between nucleus k and electron i. See the Theory section (currently in preparation) for details. Default is no regularization.

`COULOMB [energy] [for] [molecule | atom] { A | B |` *name* `}`
The `COULOMB` command calculates the electron–electron, electron–nuclear and nuclear–nuclear *intramonomer* energy of a specified molecule. This was required for a particular problem and will probably not be generally useful.

`INTEGRAL-SWITCH [=]` *switch value*
The `INTEGRAL-SWITCH` option can be used to control the way some integrals are computed. The default and recommended value is 1, which specifies that certain integrals are to be computed without density-fitting. A value of 0 specifies that density-fitting is to be used weherever possible.

`[USE-DRHO {with} [CONSTRAINTS | NO-CONSTRAINTS]]`
Evaluate the electron–electron repulsive energy using an expansion fitted directly to the density. By default, the code assembles the total electronic density from expansions of the squares of the occupied molecular orbitals. This command allows a more direct evaluation of the density, and, at least in principle, a more accurate one.

RESET
Parameter changes are normally saved. To reset to the default parameters, use this command.

9.14 First-order exchange (exchange-repulsion) $E^{(1)}_{\rm exch}$

Calculate the first-order exchange energy between molecules A and B using the single-exhange (S^2) approximation. Integrals are obtained using density-fitting.

```
[BEGIN | SET] E1EXCH
  REG/REGULARISE/REGULARIZE [with] ETA [=] <reg_eta>
  OVERLAP [integrals] [ALGORITHM [DF [WITH | WITHOUT] [CONSTRAINTS]] [{ISA | STOCKHOLDER}]]
  INTEGRAL [SWITCH [=] <switch>]
  NO-E1EXCH
  QUIET
  VERBOSE
  [RESET | DEFAULTS | DEFAULT]
  DEBUG
END
```

Use BEGIN to calculate an energy and SET to define parameters which will be used in a subsequent energy calculation. SET is useful when performing ENERGY-SCANs.

Options:

{ REGULARISE | REGULARIZE | REG } [with] ETA [=] η
Use the regularized Coulomb potential. That is, remove the Coulomb singularities, replacing $1/r_{ki}$ by $(1-\exp(-\eta r_{ki}^2))/r_{ki}$ in the attraction between nucleus k and electron i. See the Theory section (currently in preparation) for details. Default is no regularization.

OVERLAP [integrals] [ALGORITHM [DF [WITH | WITHOUT] [CONSTRAINTS]] [{ISA | STOCKHOLDER}]]
Calculate the overlap integrals. This can use the DF solution with or without constraints. Default is without constraints. By default the overlap integrals are not calculated at all. The ISA algorithm is experimental and not accessible in this version of the program.

INTEGRAL-SWITCH [=] *switch value*
The INTEGRAL-SWITCH option can be used to control the way some integrals are computed. The default and recommended value is 1, which specifies that certain integrals are to be computed without density-fitting. A value of 0 specifies that density-fitting is to be used weherever possible.

NO-E1EXCH
Skip the calculation of $E^{(1)}_{\rm exch}$. This only makes sense if OVERLAP is used to calculate the overlap integrals.

RESET
Parameter changes are normally saved. To reset to the default parameters, use this command.

9.15 Second-order induction energy $E^{(2)}_{\rm ind,tot}$

The second-order induction energy $E^{(2)}_{\rm ind,tot}$ is calculated using the propagator defined in sec. 9.5. At present, only the density-fitted expression can be calculated using this module (though the non-DF induction can be calculated using an older module). By default, the induction energy of molecules A and B is calculated. However, using PROBE, the induction energy of a molecule with a point charge at a specified position can also be calculated.

If the second-order dispersion energy has been calculated before a call to this module, computational effort can be reduced by using the FORCE command to stop the initialisation of the J-integrals (JINT).

This module can use the R-SRS theory described by Patkowski et al. [2012] to evaluate the induction and exchange-induction energies. Here, the amplitudes are computed in the field from the regularized nuclear potentials of the partners, but the energies evaluated using the full potential. Regularization is a convenient and robust way to define a basis-independent charge-transfer energy. Full details of how the regularized induction energy is defined are given by Misquitta [2013].

The idea here is to write the singular electron–nuclear potential as a short-ranged, singular part and a long-ranged part that is well-behaved. In the notation used by Patkowski et al. this is expressed as

$$\frac{1}{r} = v_p(r) + v_t(r), \tag{13}$$

where v_t is the singular, short-ranged part and v_p the long-ranged, well-behaved part of the nuclear potential. Various schemes can be used to achieve this splitting, CamCASP uses the Gaussian-based scheme [Patkowski et al., 2001b]:

$$\begin{aligned} v_p(r) &= \frac{1}{r}\left(1 - e^{-\eta r^2}\right), \\ v_t(r) &= \frac{1}{r} e^{-\eta r^2}. \end{aligned} \tag{14}$$

In the version of regularized SRS theory (R-SRS) derived by Patkowski et al. [2012] the dimer wavefunction corrections are obtained in response to the interaction operator with regularized nuclear potentials, but the interaction energy corrections are calculated using the original, un-regularized interaction operator. For the second-order induction energy this means that we calculate the first-order induction wavefunction correction in response to the *regularized* electrostatic potential $\omega^{\mathrm{B}}_{\mathrm{Reg}}$:

$$\omega^{\mathrm{B}}_{\mathrm{Reg}}(\mathbf{r}) = -\sum_{\beta} Z_\beta v_p(\mathbf{r} - \mathbf{R}_\beta) + \int \frac{\rho^{\mathrm{B}}(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}', \tag{15}$$

to give

$$\Phi^{\mathrm{A}}_0(1)[\mathrm{Reg}] = \sum_{r\neq 0} \frac{\Phi^{\mathrm{A}}_r \langle \Phi^{\mathrm{A}}_r | \hat{\Omega}^{\mathrm{B}}_{\mathrm{Reg}} | \Phi^{\mathrm{A}}_0 \rangle}{E^{\mathrm{A}}_0 - E^{\mathrm{A}}_r}, \tag{16}$$

where many-electron electrostatic operator is defined as $\hat{\Omega} = \sum_{i\in \mathrm{B}} \omega^{\mathrm{B}}_{\mathrm{Reg}}(\mathbf{r}_i)$. The regularised second-order polarization component of the induction energy is then defined as:

$$E^{(2)}_{\mathrm{ind,pol}}(\mathrm{Reg}) = \langle \Phi^{\mathrm{A}}_0 | \hat{\Omega}^{\mathrm{B}} | \Phi^{\mathrm{A}}_0(1)[\mathrm{Reg}] \rangle. \tag{17}$$

To this, as always, we need to add the similarly regularized exchange-induction energy [Jeziorski et al., 1994, Patkowski et al., 2004].

The SAPT(DFT) expression for the polarization part of the induction energy of monomer A in response to the field of B is

$$E^{(2)}_{\mathrm{ind,tot}}(\mathrm{A} \leftarrow \mathrm{B}) = 2 s^i_v (\omega^{\mathrm{B}})^v_i \tag{18}$$

where i and v label occupied and virtual states, $(\omega^{\mathrm{B}})^v_i$ are matrix elements of the unperturbed potential of monomer B and the amplitudes s^i_v are obtained, in the case of SAPT(DFT), by solving the coupled Kohn–Sham equations

$$\mathbf{H}^{(1)}_{\mathrm{A}} \mathbf{s}^A = -\boldsymbol{\omega}^{\mathrm{B}} \tag{19}$$

where $\mathbf{H}^{(1)}_{\mathrm{A}}$ is the electric Hessian [Casida, 1995, Colwell et al., 1995] of Kohn–Sham linear-response theory that is given in full form for hybrid functionals by Misquitta and Stone [2008a]. The expression for $E^{(2)}_{\mathrm{ind,exch}}(\mathrm{A} \leftarrow \mathrm{B})$ also involves the amplitudes defined above, though the matrix elements multiplying it are more complex and are given in full form by Jeziorski et al. [1993], Patkowski et al. [2012]. Analogous expressions exist for the induction energy of monomer B due to the field of A.

We now define the second-order charge-transfer energy as

$$\mathrm{CT}^{(2)}(\mathrm{Reg}) = E^{(2)}_{\mathrm{ind,tot}} - E^{(2)}_{\mathrm{ind,tot}}(\mathrm{Reg}). \tag{20}$$

Here $E^{(2)}_{\mathrm{ind,tot}}(\mathrm{Reg})$ is the regularized second-order induction energy that may be identified with the true polarization energy. There is no basis restriction on the above definition, except that the basis set used needs to be large enough to converge the total induction energy, which is the usual requirement for any energy calculation.

There is one free parameter in this model for the charge-tranfer: η in eq. (14) determines the length-scale for the regularisation. Misquitta [2013] has shown that for a variety of systems the choice $\eta = 3.0$ a.u. is appropriate. This is the default here, but may be altered using the commands listed below.

```
{SET | BEGIN} {IND | INDUCTION | E2IND}
  USE [the] {DF | DENSITYFITTING | DENSITY-FITTING} [algorithm]
  NEW-PROP
  UNITS [{A | ANG | ANGSTROM} | {BOHR | AU | A.U.}]  (must come before PROBE)
  PROBE [molecule | atom] [A | B] [with] CHARGE <charge> AT <x> <y> <z>
  FORCE [NO] {INITIALIZE | INIT}
  {REG | REGULARISE | REGULARIZE} ETA [=] <reg_eta>
  INTEGRAL [SWITCH [=] <switch>]
  NO-EXCHANGE
  AMPLITUDE-TYPE [COUPLED | UN-COUPLED]
  [RESET | DEFAULTS | DEFAULT]
  QUIET
  VERBOSE
  DEBUG/DEBUGGING [T/TRUE/ON/YES or F/FALSE/OFF/NO] [EXTREME]
END
```

Use BEGIN to calculate an energy and SET to define parameters which will be used in a subsequent energy calculation. SET is useful when performing ENERGY-SCANs.

Options:

USE [DF | DENSITYFITTING | DENSITY-FITTING]
This is always ON and need not be specified.

NEW-PROP
Uses the new propagator module. This is applicable only for the ALDAX, ALDA and UCKS | UCHF propagator types.

UNITS [A | ANG | ANGSTROM | BOHR | AU | A.U.]
This must come before PROBE if required.

PROBE [molecule | atom | *name*] [with] CHARGE q AT $x\,y\,z$
Probe the induction energy of the specified molecule interacting with the specified point charge.

{ REG | REGULARIZE | REGULARISE } ETA [=] η
See above.

QUIET
VERBOSE
DEBUG
Default is VERBOSE, debugging off.

NO-EXCHANGE
Skip the calculation of the exchange-induction energy.

INTEGRAL-SWITCH [=] *switch value*
Sets the algorithm used to compute some of the two-index integrals (nuclear and overlap). Possible switch values are:

- 0: Use density-fitting. Lower accuracy.
- 1: Higher accuracy. Recommended and the default.

AMPLITUDE-TYPE [COUPLED | UN-COUPLED]
Set the method used to compute the second-order induction amplitudes. This affects the exchange-induction energy calculation only. The default is to use coupled amplitudes as this results in the highest accuracy. At the present, the coupled amplitude calculation uses an algorithm that is memory intensive. Consequently, for large systems we recommend using the more approximate un-coupled amplitudes, in which case the code *estimates* the coupled result using scaling:

$$E^{(2)}_{\mathrm{ind,exch}}[\mathrm{C}] \approx E^{(2)}_{\mathrm{ind,exch}}[\mathrm{UC}] \times \frac{E^{(2)}_{\mathrm{ind,pol}}[\mathrm{C}]}{E^{(2)}_{\mathrm{ind,pol}}[\mathrm{UC}]}. \tag{21}$$

In practice, the above approximation does not lead to large errors in the total interaction energy.

RESET
Parameter changes are normally saved. To reset to the default parameters, use this command.

9.16 Second-order dispersion energy $E^{(2)}_{\text{disp,tot}}$

The second-order dispersion energy $E^{(2)}_{\text{disp,tot}}$ is calculated using the propagator defined in section PROPAGATOR. Both the density-fitted and non-density-fitted propagators can be used. Only the former (default) should be used for large calculations.

```
{SET | BEGIN} {DISP | DISPERSION | E2DISP}
  USE [the] [{DF | DENSITYFITTING | DENSITY-FITTING} | NODF ] [algorithm]
  FORCE-INITIALIZE
  NEW-PROP
  {QUAD | QUADRATURE} [grid] [with] <n> [point/s]
  NO-EXCHANGE
  INTEGRAL [SWITCH [=] <switch>]
  [RESET | DEFAULTS | DEFAULT]
  DEBUG
  {QUIET | VERBOSE}
END
```

Use BEGIN to calculate an energy and SET to define parameters which will be used in a subsequent energy calculation. SET is useful when performing ENERGY-SCANs.

Options:

USE [DF | DENSITYFITTING | DENSITY-FITTING | NODF]
Density-fitting is used by default in the calculation of the dispersion energy. Specifying NODF here causes a different algorithm to be used, without density-fitting.

FORCE-INITIALIZE
Forces a re-initialization of the module.

NEW-PROP
Uses the new propagator module. This is applicable only for the ALDAX, ALDA and UCKS | UCHF propagator types.

NO-EXCHANGE
Skip the calculation of the exchange-dispersion energy.

INTEGRAL-SWITCH [=] *switch value*
Sets the algorithm used to compute some of the two-index integrals (nuclear and overlap). Possible switch values are:

- 0: Use density-fitting. Lower accuracy.
- 1: Higher accuracy. Recommended and the default.

{QUAD | QUADRATURE} [grid] [with] <n> [point/s]
Set the number of quadrature points used in the (imaginary) frequency integration grid. The details of this integration are set in the QUAD block described in sec. 9.6. The default is 10 points.

RESET
Parameter changes are normally saved. To reset to the default parameters, use this command.

9.17 Energy scan

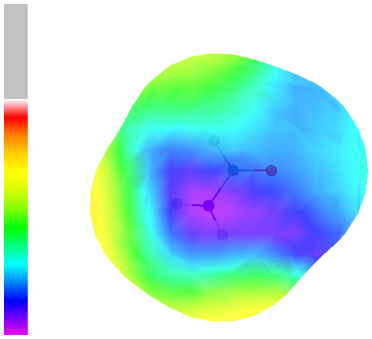
Energy scans using one molecule or a charge as a probe can be performed using this module. There are some limitations in the module:

- There is no limitation if the energy scans are to be performed *without* molecular rotations.

- If molecules are to be rotated, bear in mind the following:
  - For molecule...atom scans, if the molecule is kept fixed and only the atom moved around, there is no problem at all.
  - For molecule...molecule scans, only the overlap integrals and first-order energies can be correctly scanned at present. This is because the Hessians needed for the second-order energies are not yet rotated correctly. Also, such a scan limits you to use auxiliary basis sets with *spherical* GTOs. Bear in mind that this leads to lower accuracies in the density-fitting. We are working to resolve this problem.

The list of points at which the scan is to be performed will usually be read in from file (if `USE POINTS` is used) or defined in the `LATTICE` module (if `USE LATTICE` is used).

The data generated by the energy scan module can be used by the ORIENT program to produce a graphical display of the energy over a molecular surface. In this case the grid of points would be generated by ORIENT, exported to a file, and read into CAMCASP using the `POINTS` option. The Figure shows the map of dispersion energy between formamide and a neon atom, calculated in this way.



```
BEGIN SCAN/ENERGY_SCAN
  {PROBE | SCAN} [molecule | atom] <molname> +++
      WITH [<molname> [[and] CHARGE <charge>]]
  UNITS [A | ANG | ANGSTROM | BOHR | AU | A.U.]  +++
      [DEG | DEGREE | DEGREES | RAD | RADIAN | RADIANS]
      NOTE: UNITS must come before POINTS/RANDOM
  CENTRE/CENTER PROBE [on] [COM]/[XYZ <x> <y> <z>]
  {DC-AUX | DCAUX}
  REGULARIZE [E1] [[and] E2] [with] ETA [=] <reg_eta>
  {ENERGIES | ENERGY} [DISP/DISPERSION/E2DISP] +++
   [&/and/,] [IND/INDUCTION/E2IND] [&/and/,] [ELST/E1ELST/ELECTROSTATICS] +++
   [&/and/,] [E1EXCH] [&/and/,] [E2EXIND] & [E2EXDISP] & [OVERLAP] [NOTHING]
  LATTICE
  POINTS
    [TRANSLATIONS | TRANSLATIONS-ONLY]
    [SKIP-FIRST-COLUMN]
    x1 y1 z1    [angle1 nx1 ny1 nz1]
    x2 y2 z2    [angle2 nx2 ny2 nz2]
    etc
  ---  <----end POINTS block with three hyphens.
  RANDOM
    POINTS [=] <number of points>
    DRMIN  [=] <dRmin>
    DRMAX  [=] <dRmax>
    RADIAL [points] [=] <number of radial points per angular configuration>
    SKIP   [=] <number to skip> [angular] [configurations]
    WRITE  <filename> [APPEND]
  ---
  ENERGY-FILE [PREFIX] <file prefix> [[in | using] STYLE <style index>]
  OVERLAP-FILE [PREFIX] <file prefix> [[in | using] STYLE <style index>]
  DEBUGGING T/TRUE/ON or F/FALSE/OFF
  {QUIET | VERBOSE}
END
```

The options are:

```
{ PROBE | SCAN } [ molecule | atom ] { name } +++
        [with] [ [ molecule | atom ] { name  } ] [ [and] CHARGE q ]
```

Here the probe can be either a point charge or another molecule or both. Default is to use molecule *B* as the probe; that is, molecule *A* remains fixed while molecule *B* is moved so that its origin is successively at the positions

specified by the grid. If a charge probe is used, the first-order electrostatic and second-order induction energies will be scanned with this charge.

`UNITS` *unit specification*
Default, as usual, is to use the global units. For this to apply to the coordinates read in using the `POINTS` command it should appear *before* `POINTS`.

`CENTRE PROBE [on] [COM | XYZ x y z]`
Default `XYZ 0.0 0.0 0.0`. The atom positions in a molecule are specified relative to a molecular centre, which is initially the global origin of coordinates. This command may be used to redefine the molecular centre to be the centre of mass, or some other point in global Cartesian coordinates. When the molecule is used as a probe this centre is repositioned successively at the points of the grid.

`{ ENERGIES | ENERGY }` *terms*
The energy terms to be calculated at each point of the scan can be any or all of the following, separated by space, and or &.

```
DISP | DISPERSION | E2DISP
IND | INDUCTION | E2IND
ELST | E1ELST | ELECTROSTATICS
E1EXCH
E2EX-IND
E2EX-DISP
OVERLAP
NOTHING
```

Default is to calculate all the energies. Use `NOTHING` if all you need is the grid that would be used. This might be useful if the `RANDOM` grid is chosen (see below). Since the Hessians are not rotated at present, $E^{(2)}_{\mathrm{disp,tot}}$ and $E^{(2)}_{\mathrm{ind,tot}}$ should not be calculated in scans that involve molecular rotations.

NOTE: `E2EX-IND` implies `E2IND` and `E2EX-DISP` implies `E2DISP`. This is because the second-order exchange energies use their polarization counterparts to obtain an empirical scaling factor. See Misquitta and Szalewicz [2005] and Misquitta et al. [2005] for details.

`{ REG | REGULARIZE | REGULARISE }` *terms* `ETA [=]` η
Use a regularized form of the perturbation operator for the terms specified, which may be `[E1ELST | ELST | E1]` and/or `[ E2IND | IND | E2]`. Default is to use the unregularized operator. The dispersion is never regularized.

`ENERGY-FILE` *file-prefix* `[[in | using ] STYLE` *index*
`OVERLAP-FILE` *file-prefix* `[[in | using ] STYLE` *index*
Defaults file prefixes are `energy_file` and `overlap_file`. The file suffix used is `.dat` in both cases. Each file will contain a description of the dimer, the units used, and the translation and rotation parameters used to position the probe molecule. The rotation is in angle–axis form. The molecule being probed stays in the same position throughout the scan. The energy file will contain the energies including total overlap, and the overlap file will contain the distributed overlap.

There are two file styles. The second (`STYLE 2`) is recommended and is the default. This style is both human-readable and parsable by CamCASP and can be read in for later use.

These are plain text files, and are in a `KEYWORD` *value/s* format that is readable by CamCASP. They are in free format and can be edited quite easily. For example, multiple energy files can be joined into one large one. If you need to edit them, do not remove any fields, since the parser is used to read them back.

The configurations for which the energy is to be calculated are specified by one of the options `LATTICE`, `POINTS` or `RANDOM`.

`LATTICE`
Create a grid of points using the `LATTICE` module. This can only be used if you are probing a molecule with an *atom* as no rotations can be created using the `LATTICE` module.

```
POINTS
[SKIP-FIRST-COLUMN]
[ TRANSLATIONS | TRANSLATIONS-ONLY ]
  points, one per line
{ --- | END }
```

This option requires the points to be listed immediately following. The most convenient way to do this is to read

them from a file:
`READ` *file-name*
or to include the file in the data:
`#include` *file-name*
`SKIP-FIRST-COLUMN` is used if the first column of the file contains index numbers for the listed points, which is the case if the file was generated by a previous energy scan. `TRANSLATIONS` or `TRANSLATIONS-ONLY` specifies that the probe is not to be rotated. The points themselves should be given, one per line, as Cartesian coordinates in whatever units have been specified, each followed by an angle and axis of rotation unless `TRANSLATIONS-ONLY` was specified. That is, the layout is

$x_1\ y_1\ z_1\ [\psi_1\ n_{x1}\ n_{y1}\ n_{z1}]$
$x_2\ y_2\ z_2\ [\psi_2\ n_{x2}\ n_{y2}\ n_{z2}]$
$\vdots$

If `TRANSLATIONS-ONLY` was specified, angle-axis data may be present but will be ignored provided they are on the same line as the translation data. Blank lines and lines beginning "`!`" are ignored. The list of points is terminated by a line containing just '`---`' or '`END`'. This line should occur in the main data stream, not in an included file.

`RANDOM`
  `POINTS` *N*
  `DRMIN` $dR_{\min}$
  `DRMAX` $dR_{\max}$
  `RADIAL [points] [=]` *number of radial distances per orientation*
  `SKIP` *skip* `[angular] [configurations]`
  `WRITE \emph{filename} [APPEND]`
`{ --- | END }`

This option generates a set of N configurations (default 500) randomly, by the following scheme. First an orientation of the dimer is generated using a Sobol pseudo-random scheme, and the distance of van der Waals contact R_0 for this orientation is determined, using standard van der Waals radii. The energy is calculated at one or more random distances not less than $R_0 + dR_{\min}$ and not greater than $R_0 + dR_{\max}$. The defaults for $dR_{\min}$ and $dR_{\max}$ are -1.5 bohr and $+1.2$ bohr.

The `RADIAL` and `SKIP` commands allow some control over the kind of grid. With `RADIAL` you can specify the number of radial points (randomly chosen to lie in $[R_0 + dR_{\min}, R_0 + dR_{\max}]$) per dimer orientation. The default is 5 points per orientation. The `SKIP` command allows you to skip some number of orientations; for example, if you have done a scan using 500 angular orientations and want to add a further 1500, you would specify 2000 orientations and skip the first 500. The orientations are not obtained randomly, but using a Sobol sequence, which is a non-random scheme designed to cover configuration space as uniformly as possible, however many points are used.

The `WRITE` command writes out the list of configurations to the specified file. If `APPEND` is present, the data will be appended to the file that should be already present.

9.18 Overlap model

Calculates the density overlap for use with the density-overlap model of the exchange–repulsion energy. This is done in two stages. First, the exchange-repulsion energies are fitted to the total density overlap:

$$e \equiv E^{(1)}_{\text{exch}} \approx K_0 S = \sum_{a\in A, b\in B} K_{ab} S^{ab},$$

where the total density overlap S is partitioned via the density-fitting procedure into a sum of atom–atom terms. In this fit all the K_{ab} have the same value K_0. The quantity minimized is

$$\chi^2 = \sum_p w(e_p)\left[\frac{K_0 S_p}{e_p} - 1\right]^2,$$

where $w(e)$ is a weight that depends on the energy e (see below).

In the second stage, the coefficients K_{ab} are permitted to have different values depending on the atom types. However they are mildly constrained by imposing a penalty function that discourages values that are too different from

K_0. The quantity minimized is now

$$\frac{\chi^2}{\sum_p w(e_p)} + \sum_{ab} \lambda (K_{ab} - K_0)^2,$$

with

$$\chi^2 = \sum_p w(e_p) \left[\frac{\sum_{a\in A, b\in B} K_{ab} S_p^{ab}}{e_p} - 1 \right]^2.$$

The configurations and S_p^{ab} values, and the fitted K_{ab} values, are written to files suitable for input to ORIENT.

Molecules will often have symmetry. In CAMCASP symmetry is handled using the TYPE specification (see Sec. 9.2) that sets the types of each site. The overlap matrix is symmetrized between site types *within each molecule* as follows:

$$S_p^{ab} = \sum_{i=a} \sum_{j=b} S_p^{ij}(\text{NoSymm}) \tag{22}$$

Here S_p^{ij}(NoSymm) is the matrix of overlap integrals for sites i in molecule A with type a and sites j in molecule B with type j, and S_p^{ab} is the symmetrized overlap matrix for *types* a and b.

Additionally, sites between molecules may have the same type. This would be the case for a symmetric dimer A...A, but it could also be that an approximate symmetry has been imposed to make, say, all hydrogen atoms have the same type. If this is the case, an additional symmetrization of the overlap matrix may be performed using the SYMMETRIZE-BETWEEN-MOLECULES command described below.

The syntax for this module is

```
BEGIN OVERLAP-MODEL
  UNITS [HARTREE | HARTREES] [KJMOL | KJ/MOL] [CM-1 | 1/CM]
  WEIGHT [NONE] [TYPE <number>] [and] [{E0 | E_0} [=] <E_0>] +++
     [and] [ALPHA [=] <alpha>]

  ENERGY [=] [sign] <component> [ [sign] <component> ] ...
  where
  sign = + or -
  and
  component = E1ELST, E1ELST(AS), E1EXCH, E2IND, E2IND(AS), E2EXIND,
              E2DISP, E2DISP(AS), E2EXDISP, DELTA
  or a composite energy such as
          E1PEN = E1ELST - E1ELST(AS)
          E2PENIND = E2IND - E2IND(AS)
          E2PENDISP = E2DISP - E2DISP(AS)
          E2PEN = (E2IND - E2IND(AS)) + (E2DISP - E2DISP(AS))
          E2EXCH = E2EXIND + E2EXDISP
          E2INT = E1ELST + E1EXCH + E2IND + E2EXIND + E2DISP + E2EXDISP

  ENERGY-OPTIONS [CHECK-ASYMP] [ABS-MAX-ENERGY [=] <value>]

  {EMAX | E-MAX} [=] <maximum energy>
  {EMIN | E-MIN} [=] <minimum energy>
  ENERGY-FILE <full file name of energy file>
  OVERLAP-FILE <full file name of distributed overlap file>
  CONDITION [number] [=] <condition number for solver>
  CONSTRAINT [=] <lambda>
  [ SYMMETRIZE-BETWEEN-MOLECULES | SYMMETRIZE ]
  ORIENT [files] [YES | NO | TRUE | FALSE] [ISOTROPIC-SITES] [ANCHOR-LIST] +++
      [SHAPE-SYMMETRY]
  ONLY-TOTAL-ORIENT [C6] [NO-WEIGHTS]
  DEBUGGING T/TRUE/ON or F/FALSE/OFF
  QUIET
  VERBOSE
```

```
  [RESET | DEFAULTS | DEFAULT]
END
```

The options are:

`UNITS [HARTREE | HARTREES] [KJMOL | KJ/MOL] [CM-1 | 1/CM]`
Specify the units to be used for energies, if different from the global default.

`WEIGHT [TYPE` *index* `] [and] [{E0 | E_0} [=]` e_0 `] [and] [ALPHA [=]` α `]`
Specify the weighting scheme for the data points. The weight $w(e)$ for a point with energy e is, according to the index,

0: 1

1: $(1 + (e/e_0))^{-1}$

2: $(1 + (e/e_0)^2)^{-1}$

3: $(e_0/e) + 1$

4: $\exp[-\alpha(\ln(e/e_0))^2]$

The last of these is the scheme used by Hodges and Wheatley [2000]. It is recommended and is the default. The default parameters are $e_0 = 20.0\,\mathrm{kJ\,mol^{-1}}$ and $\alpha = 1/\ln 10$.

`EMIN` $E_{\min}$
`EMAX` $E_{\max}$
Energy values outside the range specified are ignored (given zero weight). Defaults: $E_{\min} = 0.01\,\mathrm{kJ\,mol^{-1}}$, $E_{\max} = 100\,\mathrm{kJ\,mol^{-1}}$.

```
ENERGY [=] E1EXCH [+ | -] E1PEN [+ | -] +++
  [+ | -] [E2EX-IND | E2EXIND] [+ | -] [E2EX-DISP | E2EXDISP] +++
  [+ | -] [E2PEN-IND | E2PENIND] [+ | -] [E2PEN-DISP | E2PENDISP] +++
  [+ | -] DELTA
```
The energy to be fitted to can be defined using this statement. By default, the energy used is `E1EXCH + E2EXIND + E2EXDISP + E1PEN + E2PENIND + E2PENDISP + DELTA`.
These energies are:

- `E1EXCH`: The first-order exchange energy, $E^{(1)}_{\mathrm{exch}}$.
- `E2EX-IND`: The second-order exchange-induction energy, $E^{(2)}_{\mathrm{ind,exch}}$.
- `E2EX-DISP`: The second-order exchange-dispersion energy, $E^{(2)}_{\mathrm{disp,exch}}$.
- `E1PEN`: The first-order penetration energy defined as $E^{(1)}_{\mathrm{elst}} - E^{\mathrm{Asymp}}_{\mathrm{elst}}$.
- `E2PEN-IND`: The induction contribution to the second-order penetration energy defined as $E^{(2)}_{\mathrm{ind,tot}} - E^{\mathrm{Asymp}}_{\mathrm{ind}}$.
- `E2PEN-DISP`: The dispersion contribution to the second-order penetration energy defined as $E^{(2)}_{\mathrm{disp,tot}} - E^{\mathrm{Asymp}}_{\mathrm{disp}}$.
- `DELTA`: A correction energy. This energy can be anything, and is left to the user. For example, it could be defined to be the difference between SAPT(DFT) and CCSD(T) energies.

Alternatively, `E2EXCH` can be used as a synonym for `E2EX-IND + E2EX-DISP` and, likewise, `E2PEN` can be used for `E2PEN-IND + E2PEN-DISP`. These energies are all in the energy file (see below). We will usually use only + signs in the energy expression.

`ENERGY-OPTIONS [CHECK-ASYMP] [ABS-MAX-ENERGY [=]` *value*`]`
The option `CHECK-ASYMP` will check the energies calculated with the multipole models (the asymptotic forms of the energy components) for missing values. If an energy is identically zero, that dimer is removed from the list.

The energy bounds set using `EMIN` and `EMAX` are useful for selecting configurations but these are not foolproof. There are instances where, because of energy cancellations with the multipole-expanded energies, repulsive configurations are included when they should not, to the detriment of the fit. This can be avoided using the option

ABS-MAX-ENERGY to set an upper limit to the absolute values of the individual interaction energy components. By default this upper limit is set to be $100|E_{\max}|$.

SYMMETRIZE-BETWEEN-MOLECULES
As described above, the overlap matrix is symmetrized between site types within each molecule.
If SYMMETRIZE-BETWEEN-MOLECULES is set, the site type information is used to symmetrize the overlap matrix, S_p^{ab}, for site types *between molecules A and B* as follows:

$$S_p^{ab}(\text{symm}) = S_p^{ab} \quad + \quad \{\, S_p^{ba}, \;\; \text{if} \;\; b \in A \;\; \text{and} \;\; a \in B \;\} \tag{23}$$

where a and b are unique site types on molecules A and B, resp.

This command would be used for a dimer of the same molecule, or when you have assumed approximate symmetries between molecules.

CONSTRAINT [=] λ
Specifies the parameter to be used in the constrained fit. Default 10^{-8}.

CONDITION [NUMBER] [=] c
If the least-squares problem leads to an ill-conditioned matrix, the problem is reduced to one with estimated condition number less than $1/c$. The default value, $c = 10^{-7}$, should be satisfactory.

ENERGY-FILE *full file name of energy file*
The energy file will have been created from the ENERGY-SCAN module.

OVERLAP-FILE *full file name of overlap file*
This is the file containing the distributed overlap. If the OVERLAP-MODEL block follows the ENERGY-SCAN block in a CamCASP run, then the file name need not be specified here as it will be passed internally.

The ENERGY-FILE and OVERLAP-FILE commands can appear as many times as needed. This is particularly useful if data points from multiple runs are needed for an overlap model calculation. Make sure that the sequence of files is the same for both of them. That is, if energies and overlaps have been computed in three runs, resulting in three sets of files, the sequence should be:

```
ENERGY-FILE energy_file_1.dat
ENERGY-FILE energy_file_2.dat
ENERGY-FILE energy_file_3.dat
OVERLAP-FILE overlap_file_1.dat
OVERLAP-FILE overlap_file_2.dat
OVERLAP-FILE overlap_file_3.dat
```

The recommended practice however is to generate the energy files using an MC basis but a DC auxiliary basis, while the overlap-mode calculation has to be done using an MC auxiliary basis and so requires a separate CamCASP job. This second job needs an energy-scan section just to read in the data points and to generate the overlap file. The overlap-model section will then use this overlap file, which can be passed internally, and the energy file from the previous job.

ORIENT [files] [YES | NO] [ISOTROPIC-SITES]
By default, the results of the constrained overlap model will be used to construct input files for Orient. These input files are needed to obtain the fit to the Born–Mayer term that describes the short-range interaction. It is done in a site–site form, so a file is generated for every pair of site types. Using this command the creation of these files can be suppressed. Use ISOTROPIC-SITES to write out the Orient command files for a potential with all sites isotropic.

The generated Orient files normally include separate data files for optimizing the individual site-site potentials, as well as a data file for optimizing the total potential. The site–site fits should be carried out first, and the resulting parameters edited into the data file for the total potential fit. It will often be necessary to restrict the full optimization by anchoring the parameters to their initial values by harmonic constraints, so that they don't drift to physically unreasonable values. The anchor definition can be set up using the the anchors.pl script. Details of the procedure can be found in the CamCASP Wiki at https://jasper.ph.qmul.ac.uk/wiki/ajm:camcasp:potentials.

ONLY-TOTAL-ORIENT [C6] [NO-WEIGHTS]
Use the ONLY-TOTAL-ORIENT command to write out the Orient file for the entire potential. Use this when fine-tuning the potential fit by relaxing all or some parameters using anchors. By default we is assumed that the C_6

coefficients will not be relaxed. Use the C6 option to generate an ORIENT file to relax these coefficients. Likewise, by default weights will be included in the ORIENT command file. The NO-WEIGHTS option will suppress this.

RESET
Parameter changes are normally saved. To reset to the default parameters, use this command.

9.19 Integrals

The integral routines are not meant to be accessed directly, but there may well be situations which demand it, so CAMCASP allows direct control over the integrals. While almost all integrals are computed using density-fitting techniques, some can be computed without this approximation.

```
BEGIN/SET { DF-INT | DF-INTEGRALS | DF-INTS | INTEGRALS }
  INT {TYPE <type>} {DESC <description>} +++
      [and] [SWITCH [=] <switch>] [and] [PRINT] [and] [ROTATE]
  GEN-INT {TYPE <type>} {DESC <description>} +++
      [and] [SWITCH [=] <switch>] [and] [PRINT] [and] [ROTATE]
  [DF-CONSTRAINTS | CONSTRAINTS ] [ETA [=] <eta>] +++
       [and] [LAMBDA [=] <lambda>] +++
       [and] [TYPE [=] [SITE | INTER-SITE] [repulsion | self-repulsion]]
  DF-TYPE-DIMER [OO | {OV | VO}]
  DF-TYPE-MONOMER [OO | {OV | VO} | {NN | FULL}]
  QUIET
  VERBOSE
  DEBUG/DEBUGGING [T/TRUE/ON/YES or F/FALSE/OFF/NO]
  RESET
END
```

Options:

INT TYPE *type* DESC *description* +++
[and] [SWITCH [=] *switch*] [and] [PRINT]
The INT command will cause a particular integral to be computed. Integrals are referenced by two four character codes: the integral type (TYPE) and the description (DESC).

The integral types describe which operators are used. At present, the following integral types are available (chemical notation is used for all integrals):

- **2-electron integral types**: OVOV, VVOO, OOOO
  O refers to an occupied orbital and V to a virtual orbital.
- **Nuclear integrals**: NUCA, NUCB
  NUCA use the nuclei of molecule with label A and NUCB use nuclei from the molecule with label B.
- **Integrals with a point-charge probe**: PROB
  These are a special case of the nuclear integrals where the 'nucleus' is a point charge.
- **Overlap integrals**: OVRL
- **Miscellaneous integrals**: See module df_integrals for details.

The integral description sets the kinds of basis functions or molecular orbitals that the operator operates on. There are too many of these to describe here, but the main ones are:

- **4-index descriptions**: AAAA, BBBB, AABB, ABAB, ABBB, AAAB
  These are used for the 2-electron integrals only, though not all combintations are possible.
- **2-index descriptions**: OOAA, OOBB, OOAB
  These are used for the nuclear, probe and overlap integral types.

- **Integrals of the auxiliary bases**: `__XA, __XB, XAXB, _XAB`
  Here the `X` denotes an auxiliary basis function and `A, B, AB` denote molecule A, B and the dimer (AB). Once again, these are used mainly with the nuclear and probe integrals.

- **Miscellaneous descriptions**: See `module df_integrals` for details.

The `SWITCH` option allows us to control the method used to obtain particular integrals. Here `SWITCH = 0` is the default, which uses density-fitting wherever possible. `SWITCH = 1` results in some integrals being computed without density-fitting. This may seem a good idea at first sight, but since we frequently see a cancellation of errors between integrals of different kinds, calculating some accurately could result in even larger errors in the end. This option should therefore be used with caution.

However there is also an `INTEGRAL-SWITCH` option under each of the `E1ELST`, `E1EXCH`, `E2IND` and `E2DISDP` blocks, which allows the setting to be changed independently for each of these. Here the recommended value, and the default, is 1, i.e. to use density-fitting wherever possible

At present, only the nuclear and overlap integrals can be computed without density-fitting.

The integrals are not printed unless the `PRINT` option is included in this line.

`{REG | REGULARIZE} [with] ETA [=]` *regularization constant*
Regularize the intermolecular potential — that is, remove the Coulomb singularities, replacing $1/r_{ki}$ by $(1 - \exp(-\eta r_{ki}^2))/r_{ki}$ in the attraction between nucleus k and electron i. See the Theory section (currently in preparation) for details. Default is no regularization.

`GEN-INT TYPE` *type* `DESC` *description* `+++`
`[and] [SWITCH [=]` *switch*`] [and] [PRINT]`
The generalized 2-electron integrals can be computed using this command. These integrals are defined as (Chemical notation):

$$v(ij|kl) = (ij|kl) + \frac{(i|j)(k|v_A|l)}{N_A} + \frac{(i|v_B|j)(k|l)}{N_B} + \frac{(i|j)(k|l)}{N_A N_B} E_{nn}, \tag{24}$$

where $(ij|kl)$ is the standard 2-electron integral, $(i|j)$ is the overlap integral, $(k|v_A|l)$ is the nuclear integral with the potential v_A of the nuclei of molecule A, E_{nn} is the nuclear–nuclear Coulomb energy, and N_A is the number of electrons of molecule A.

We assume canonical ortho-normal orbitals when calculating the generalized integrals.

```
{DF-CONSTRAINTS | CONSTRAINTS} [ETA [=] <eta>] +++
    [and] [LAMBDA [=] <lambda>] [and] [GAMMA [=] <gamma>] +++
    [and] [TYPE [=] [SITE | INTER-SITE] [repulsion | self-repulsion]]
```
By default, the integral routine uses the density-fitting solution without any constraints. If needed, the DF solution with constraints can be used. Use this with caution. See the Density-fitting section for details on the options.

```
QUIET
VERBOSE
DEBUG
```

The parameters used in evaluating the primitive integrals can be changed using:

```
SET {INTEGRAL | INTEGRAL-PARAMETERS}
  DUMMY-S <dummy S-exponent>
  CUTOFF  <integral cutoff>
  KERNEL-CUTOFF <kernel integral cutoff>
  OVERLAP-CUTOFF <overlap integral cutoff>
END
```

Options:

`CUTOFF` *integral cutoff*
This sets the cutoff used in the pre-screening of the 2-electron integrals calculated for the density-fitting. The default cutoff is 10^{-12}.

DUMMY-S *dummy S-exponent*
We evaluate our 2-index and 3-index, 2-electron integrals using the GAMINT integral module which forms part of the GAMESS(US) quantum chemistry code. This module computes 4-index integrals. So, in order to get the 2-index and 3-index integrals, we use dummy functions in calls to gamint. The dummy functions are Gaussian orbitals of *s*-symmetry and a very small exponent — ideally 0.0. The value of this exponent is set using this command. The default value is 10^{-18}. Values between 10^{-12} and 10^{-18} have been found to result in identical interaction energies for a variety of systems.

We do not recommend changing this value.

KERNEL-CUTOFF <kernel integral cutoff>
This is the cutoff used to pre-screen the kernel integrals.

OVERLAP-CUTOFF <overlap integral cutoff>
This cutoff is used in the overlap pre-screening of certain real-space two and three index integrals.

10 PROCESS: Syntax

The PROCESS program is used to read distributed polarizabilities from either CAMCASP or ORIENT and process them in some way. It can perform tasks such as calculating average polarizabilities and anisotropic polarizabilities, or writing local polarizabilities to file in LaTeX table format, or writing an input file for the PFIT program with either local or non-local polarizabilities. The program can also handle polarizabilities at multiple frequencies. The INVERT command will apply the inversion operator on the distributed polarizabilities, to obtain the polarizabilities for the enantiomorph of the current molecule. (This just involves some changes of sign, but it is easy to make mistakes when doing it by hand.)

The CASIMIR command will create the input file for the CASIMIR program. The SKIP option to CASIMIR tells the program to skip some of the initial frequencies. This is useful when the first frequency is 0.0 (i.e., static polarizabilities) as this one is not needed by CASIMIR.

The PFIT code can refine polarizabilities (as yet, only for one frequency at a time). An input file for such a refinement can be created using the PFIT...PENALTIES command. PENALTIES tells PROCESS that the PFIT file should include the penalties section. This works for one molecule at a time so the second molecule (if defined) is ignored. Also, only local polarizabilities can be refined. If the polarizabilities have been computed with multiple frequencies, then a frequency can be chosen using the FREQ <index> sub-command. Thus, to create a PFIT input file for the static polarizabilities, which are (typically) read in as the *zeroth* index in the polarizability file, use:

```
Molecule name
  molecule specification
End
READ LOCAL polarizabilities for molecule name
  USE BINARY file file-name
  MAX rank maximum rank of polarizabilities on file
  LIMIT rank to rank-limit
  FREQUENCIES STATIC + number of freqs
END


WRITE
  PFIT file for name with CUTOFF cutoff +++
     and FREQ 0 use PENALTIES with WEIGHT 4
End
```

The default is to use the static, i.e., 0^{th} frequency, so the FREQ command could be omitted. The default number of non-zero frequencies is 10.

If the polarizability model needs to be simplified to include isotropic terms only, the following command can be used:

```
WRITE
  PFIT file for name with CUTOFF cutoff +++
     and FREQ 0 use PENALTIES with WEIGHT 4 +++
     and ISOTROPIC polarizabilities
End
```

The ISOTROPIC command does the following: from polarizabilities localized using the method of [Le Sueur and

Stone, 1994], isotropic polarizabilities are calculated for each site. The Pfit input file for the refinement step is then constructed using these isotropic terms (limited to the specified rank). Pfit then refines the polarizabilities as in the standard WSM procedure. This option can be very useful in creating a polarizability and dispersion model that is suitable for use in most standard simulation programs.

If a rank-limit is given, polarizabilities of higher rank are ignored. The Pfit program adjusts the input polarizabilities to improve the agreement between the point-to-point polarizabilities given by the local polarizability model and those calculated by CamCASP. The `WEIGHT` item specifies the weights attached to the input polarizabilities. A weight of zero means that the input polarizability value is ignored, so the fitted value is determined purely by least-squares fit to the point-to-point polarizabilities. A non-zero value means that the least-squares function is modified by a penalty that depends on the square of the difference between the fitted value and the input value, so it discourages excessive deviations from the input value. This is useful to prevent poorly-determined values from becoming non-physical. This example uses `WEIGHT 4`, which is the default and the recommended weighting. The possible weighting schemes are as follows, where t and u label components of the polarizability tensor and α_{tu} is its value in a.u.:

`WEIGHT 0`: $w(t,u) = 0.0, \quad \forall t,u.$

`WEIGHT 1`: $w(t,u) = 10^{-5}, \quad \forall t,u.$

`WEIGHT 2`: $w(t,u) = 10^{-5}/(|\alpha_{tu}|+1), \quad \forall t,u.$

`WEIGHT 3`: $w(t,u) = 10^{-4}/(\alpha_{tu}^2+1), \quad \forall t,u.$

`WEIGHT 4`: $w(t,u) = \begin{cases} 10^{-5} & \text{if } t \in 10, 10c, 10s \text{ and } u \in 10, 10c, 10s \\ 0.0 & \text{otherwise.} \end{cases}$

`WEIGHT 5`: $w(t,u) = \begin{cases} 10^{-5} & \text{if } t \in 10, 10c, 10s \text{ and } u \in 10, 10c, 10s \\ 10^{-8} & \text{otherwise.} \end{cases}$

`WEIGHT 6`: $w(t,u) = \begin{cases} 10^{-5} & \text{if } t \in 10, 10c, 10s \text{ and } u \in 10, 10c, 10s \\ 10^{-7} & \text{otherwise.} \end{cases}$

For frequency-dependent polarizabilities at frequency ω, the weight is modified to be

$$w(t,u) \to \frac{w(t,u)}{1+|\omega|^2}. \tag{25}$$

It is often useful to limit the maximum rank of polarizabilities on some atoms, particularly H atoms. In order to limit the rank of polarizabilities on non-H atoms to 2 and on H atoms to 1, use something like:

...define global data and molecule...

```
READ LOCAL polarizabilities for mol name
  USE BINARY file file-name
  MAX rank maximum rank of polarizabilities on file
  LIMIT rank to 2
  LIMIT rank to 1 for SITES H1 H2
  FREQUENCIES STATIC + number of frequencies in file
END
```

Here the first `LIMIT` limits all sites to rank 2, and the second limits just sites `H1` and `H2` to rank 1. At present, this can be done only for local polarizabilities.

If you wish to remove certain sites from the polarizability model altogether, use, for example:

```
READ LOCAL polarizabilities for name
  USE BINARY file file-name
  MAX rank maximum rank of polarizabilities on file
  LIMIT rank to 0
  LIMIT rank to 2 for SITES C N O
  FREQUENCIES STATIC + number of frequencies in file
END
```

Assuming there are no charge-flow polarizabilities (i.e., they are all zero), this will retain only sites `C N O`.

The local polarizabilities refined by the Pfit program are written in ASCII format. They can be read in using:

```
READ LOCAL [polarizabilities] [for] [molecule] name
```

```
  USE ASCII file file-name
  MAX rank maximum rank of polarizabilities on file
  FREQUENCIES STATIC + number of frequencies in file
END
```
This read (unlike the others) is done in a safe way. That is, the sites listed here can be in a different order from the sites used to create the polarizability file. They must all be present — this allows a cross-check and hence reduces the chance for errors.

The refinement using Pfit is done one frequency at a time. This means you get as many polarizability files as you have frequencies — typically 10. To get Process to read them all in, concatenate them into one file using the UNIX `cat` command. Just make sure you don't mess up the order of the frequencies. *Use the `localize.py` script to do this (see sec. 8.3).*

The point-to-point polarizabilities needed for the refinement will normally be calculated by CamCASP at 11 frequencies — static plus 10 non-zero frequencies. These polarizabilities will be written into one file, but when used by Pfit in the refinement procedure, they are needed in separate files: one for each frequency. A simple `split` command (Linux or Unix) will not work due to the format of the file. Process can be used for this using the following commands:
```
READ LOCAL [polarizabilities] [for] [molecule] name
  FREQUENCIES STATIC + number of frequencies in file
  P2P-POLS point-to-point polarizability file name [SPLIT]
END
```
The point-to-point polarizabilities will be put into files indexed by the frequency index. *The `localize.py` script takes care of this (see sec. 8.3).*

To calculate dispersion coefficients we need to use the Casimir program. To get Process to write out an input file for the Casimir program use, for example:
```
Write
  Casimir file for molecule-1 and molecule-2 with cutoff cutoff skip n frequencies
End
```
This command will use polarizabilities for molecules 1 and 2 to create the input file for the Casimir program. (Molecules 1 and 2 may be the same.) Only polarizabilities greater in magnitude than the cutoff specified will be used; this helps to avoid what could be numerical noise and results in a smaller file. A suitable value for the cutoff is 0.0001. The default is to skip the static (i.e., zeroth) frequency, so the `SKIP` command could be omitted. *The `localize.py` script takes care of this (see sec. 8.3).*

Frequency-dependent polarizabilities are needed for the dispersion coefficient calculation. By default, the Process program assumes that it should use the polarizabilities it has read in. There are some assumptions here:

- The Casimir program assumes frequencies on a certain quadrature grid. The default is a 10 point Gauss–Legendre grid with $\beta = 0.5$a.u. (β is sometimes denoted ω_0). The Process program assumes that you have used this grid. As this is the default for CamCASP and Casimir, this should not be a problem. If you've used another grid, edit the Casimir input file by hand. See the commands in the section on Casimir below.
- By default, CamCASP will calculate static polarizabilities in addition to the frequency-dependent polarizabilities. The static polarizabilities are put first in the polarizability file. The `SKIP` sub-command in the above example can be used to tell the Process program to skip the first frequency, as follows:
```
WRITE
  CASIMIR file for molecule-1 and molecule-2 with cutoff 0.0001 skip 1 frequency
END
```

Process now uses symmetry when creating the command files for Casimir and Pfit. Symmetry is imposed using the `TYPE` command in the `MOLECULE` block. We highly recommend imposing symmetry, but bear in mind that sites become equivalent only when the local axes are suitably oriented. The transformation to a local axis system are not done by Process, but by the Pfit and Orient programs.

To write out a sub-set of the polarizability matrix use:
```
WRITE
  POLS for for molecule :: freq {site1 site2 | : : } {comp1 comp2 | : : }
END
```
Where `freq` is the frequency index, `site1,site2` are the site indices and `comp1,comp2` are the component indices. The first `::` is needed. The optional colon-pairs `: :` are indicate that all indices will be used. So to write out the static charge-flow terms for all pairs of sites use
```
WRITE
  POLS for for molecule :: 1 : : 1 1
```

END The charge-flow polarizabilities appear first in the list of polarizabilities. So you do need to know the order of terms. This is not a very user-friendly format, but it works.

Here's the full syntax of this program:

```
  TITLE title string

  { SET GLOBAL-DATA | GLOBAL DATA | GLOBAL }
    See commands described in sec. 9.1.
  END

MOLECULE molecule-name
  [UNITS { BOHR | AU | ANG | ANGSTROM | ANGSTROMS }]
  [CHARGE] molecular-charge
    label charge x y z [TYPE type]
    label charge x y z [TYPE type]
    ⋮
END

  READ [ LOC | LOCAL | NONLOCAL | NON-LOCAL | P2P] [polarizabilities] [for] [atom | mol] name
    USE [ ASCII | BINARY ] [file] file-name
    Here max-rank is the maximum rank of polarizabilities in the file.
    [ MAXL | MAX | MAXIMUM ] [rank] max-rank
    The SITES command is deprecated. Use MOLECULE instead.
    { SITES | SITE } list of sites
    LIMIT [rank] [to] limit [[for] { SITE | SITES } s1 s2 ...
    { FREQUENCIES } [STATIC] [ + | PLUS ] number of frequencies
    P2P-POLS point-to-point polarizability file name [SPLIT]
  END


  [ INVERT [molecule] name

  WRITE
    This prefix will be used for all data files. Optional.
    FILE-PREFIX prefix for data files
    CUTOFF cutoff
    PFIT [file] [for] molecule-1 [ [and] molecule-2 ] ] [[and] FREQUENCY n] +++
        [[use] PENALTIES] [[with] WEIGHT scheme ] +++
        [[and] [ISOTROPIC | ISO] [polarizabilities]]
    CASIMIR [file] [for]molecule-1 [and] molecule-2 ] [SKIP n [frequencies]]
    TABLE [for] [ MOLECULE ] name [and] [FREQUENCY index ]
    ORIENT [file] [for] [ MOLECULE ] name [with] [[ALL] [ frequencies ] | [ FREQUENCY index]
    POLS [for] [ATOM/MOL] <name> [CUTOFF <cutoff>] +++
      :: freq {site1 site2 | : : } {comp1 comp2 | : : }
  END

  ***This command performs an analysis of the polarizabilities***
  ANALYSE [polarizabilities for] [ MOLECULE ] name [at] [FREQ index ]

  ENERGY
    {FILE | READ} energy file (1)
    {FILE | READ} energy file (2)
    ***Use as many READ commands as you need.***
    WRITE new energy file
    ***The energy files will be combined into a single file.
       The UNITS used in this file will be those specified in GLOBAL-DATA.
       So this is a handy way not only to concatenate energy files,
       but to convert units. ***
  END


  FINISH
```

Notes:

- Everything in the WRITE command is optional. The PFIT, CASIMIR and ORIENT options send data files suitable for those programs to standard output. The TABLE option produces a tabular output suitable for processing with LaTeX.
- When data for several frequencies are read in, they are numbered from 1 and can be referred to individually by this number, i.e. *index* in the syntax summaries above.
- The BINARY file comprises sections of the form
  s_1 s_2 *rank size*
  $\alpha(s_1, s_2, 1, 1{:}size)$
  $\alpha(s_1, s_2, 2, 1{:}size)$
  ⋮
  $\alpha(s_1, s_2, size, 1{:}size)$

  where s_1 and s_2 are site numbers, *rank* the maximum rank and *size* the dimension of the following polarizability matrix $\alpha(s_1, s_2)$, which is written one row to a record. If size is zero or no polarizability matrix exists for a particular pair of sites, nothing is written for that pair. *size*= $(rank{+}1)^2$ but is included in the file for convenience.

11 Casimir

Author: Anthony J. Stone

The Casimir program reads uncoupled local polarizabilities (i.e. of the form α^{aa}_{tu}) for one or more molecules, recouples them (i.e. to the form $\alpha^{a}_{v(jj')}$), and computes dispersion coefficients up to $C_1 0$. The code for $C_1 1$ and $C_1 2$ is included in the distribution, but is not activated as it hasn't been adequately tested.

The input file for this program can be quite complicated for all but the smallest of molecules. This file will normally be created by the Process program.

Input commands:

`FREQUENCIES` ω_0 $n_{\texttt{freq}}$
Specify the base frequency and the number of frequencies to be used for the quadrature, in a.u. Defaults: 0.5 and 10, respectively.

`SKIP [ 0 | 1 ]`
By default, the static polarizability is not expected in the data file. If however it is present, `SKIP` or `SKIP 1` will skip the first of each set of values, assuming it to be the static polarizability. `SKIP 0` cancels a previous `SKIP 1`.

`MOLECULE` *name*
Subsequent polarizability data refers to the specified molecule. The end of the data for this molecule is specified by an `END` statement.

`SITE` *name*
Introduces polarizability data for the site specified, in the current molecule. It comprises a number of sets of the form
t u
[$\alpha(0)$] $\alpha(\omega_1)$ $\alpha(\omega_2)$ ... $\alpha(\omega_n)$
terminated by "End" on a line by itself. Here *t* and *u* are spherical tensor indices ('00', '10', '11c', etc.). The polarizabilities, in a.u., follow on one or more lines. If the static polarizability is present, `SKIP` must be specified. (See above.)

`PRINT ALL | NONZERO`
Printing option for the recoupled polarizabilities. The default is to print only the nonzero ones.

`CGDIR` *path*
Specify the directory where the coupling coefficients are to be found, if not in the current directory. These are coupling coefficients for the real spherical tensor components. These coefficients are provided in `$CAMCASP/data`.

`RECOUPLE [ALL | site site ...]`
Calculate recoupled polarizabilities for all sites, or for the sites specified. This is not needed if a `DISPERSION` command follows.

`DISPERSION` *n molecule-1 molecule-2*
Recouple the polarizabilities, if not already done, and calculate and print the dispersion coefficients to C_n, where n is $6, 7, 8, 9, 10, 11$ or 12. Coefficients up to C_{12} can be calculated, using (at present) only polarizabilities up to octopole–octopole.

Appendices

A Basis sets

The basis set requirements for intermolecular interactions are quite different from those for standard energy calculations. We need basis sets that get the molecular properties correct. These tend to be bases augmented with diffuse functions and off-bond functions. The augmented Dunning basis sets of triple-ζ quality or higher are generally adequate for molecular properties and first-order interaction energies, but the off-bond functions are needed for accurate second-order interaction energies.

We will give more details and references later, but for now, here's a brief description of the basis sets that can be used in CamCASP through Cluster. Bear in mind that CamCASP can use just about any basis set if you are prepared to set up the input files. This setup is done for you by the Cluster program, at the expense of limiting you to a few important types of basis set. These are:

- **Monomer basis**: Used in a properties calculation on a single molecule. This is the usual type of basis. It is obtained by using a Cluster file with the block:
  ```
  RUN-TYPE
    Molecule name
    Basis name
    ...
  END
  ```
  No `MID-BOND` or `BASIS-TYPE` command should be present.

  The scripts supplied with CamCASP use a command-line option `-mono` to specify this type of basis.

- **SAPT(DFT) basis sets**: A SAPT(DFT) calculation of the interaction energy between two molecules (a dimer) can be done using four kinds of basis set. These fall into two categories:

  - **Dimer-centred basis**: In these basis types, the molecular orbitals and Hessians of each monomer is constructed in the entire dimer basis.

    There are two dimer-centred basis types: `DC` and `DC+`. The '+' in the latter denotes the presence of basis functions located between the interacting monomers. These are the 'mid-bond' functions that are very important for obtaining basis-set converged dispersion energies.

    A supermolecular calculation of the interaction energy would use this type of basis set (`DC` or `DC+`) so as to correct for the basis set superposition error (BSSE). SAPT(DFT) is free from the BSSE so we are free to use other kinds of basis set.

  - **Monomer-centred basis**: In these basis types, the molecular orbitals and Hessians of each monomer are constructed using the basis of the monomer, and possibly some basis functions located between the two monomers (the 'mid-bond' set) and a subset of the basis set of the interacting partner (the 'far-bond' set).

    Like the dimer-centred bases, the monomer-centred bases come in two varieties: `MC` and `MC+`. The latter differs from the former by the presence of the 'mid-bond' or 'far-bond' functions. These extra basis functions contribute very significantly to the convergence of the second-order energies and we strongly recommend the `MC+` type of basis.

  These basis sets are obtained using a Cluster command file with the lines:
  ```
  RUN-TYPE
    Molecules name of A  and name of B
    Basis name
    Basis-Type { MC | MC+ | DC | DC+ }
    Mid-bond { 3s2p1d | 3s2p1d1f } and Type { weighted | COM }
    ...
  END
  ```
  The `Mid-bond` command will be ignored for types `MC` and `DC`. The type of mid-bond set determines where it gets placed. The `weighted` option is recommended for large molecules. There are only two types of mid-bond set supplied with CamCASP. This is because interaction energies are largely insensitive to the kinds of functions used. We recommend the `3s2p1d` set for most calculations.

B Dispersion coefficients: in detail

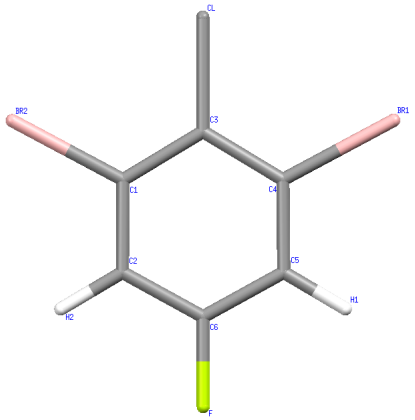
The distributed C_6, C_7 and C_8 dispersion coefficients can be obtained using the following steps. We have written a few simple scripts to perform some of the steps in this procedure, but you will still have to do some small tasks by hand.

Before getting to this, we will have had to perform a calculation of the non-local polarizabilities (say, ranks 0 to 4) at 10 frequencies, and the point-to-point polarizabilities (on a grid of 1000 to 2000 points) at the same 10 frequencies. These calculation are done in a standard properties calculation with the CamCASP program.

Let's use $C_6Br_2ClFH_2$ (XIII) as an example.

First of all we decide which axis system would be most appropriate for this molecule. The C_{2v} symmetry of XIII makes this an easy choice. Here is the local axis file for XIII:

```
$ cat local_axes
Sites
  ! Units used: BOHR
  C1      2.27448932   -0.63093982    0.00000000
  C2      2.29369649   -3.26609468    0.00000000
  C3      0.00000000    0.74116754    0.00000000
  C4     -2.27526222   -0.62679565    0.00000000
  C5     -2.29946016   -3.26296530    0.00000000
  C6     -0.00413850   -4.52851101    0.00000000
  BR1    -5.43183783    1.06069375    0.00000000
  BR2     5.43383527    1.05079536    0.00000000
  CL      0.00318230    3.99305149    0.00000000
  F      -0.00577500   -7.08124548    0.00000000
  H1     -4.05673525   -4.30060061    0.00000000
  H2      4.04880407   -4.30740551    0.00000000
End

Axes
  C1   z from C1 to BR2    x from C1 to C2
  BR2  z from C1 to BR2    x from C1 to C2
  C2   z from C2 to H2     x from C2 to C6
  H2   z from C2 to H2     x from C2 to C6
  C3   z from C3 to CL     x from C3 to C4
  CL   z from C3 to CL     x from C3 to C4
  C4   z from C4 to BR1    x from C4 to C5
  BR1  z from C4 to BR1    x from C4 to C5
  C5   z from C5 to H1     x from C5 to C6
  H1   z from C5 to H1     x from C5 to C6
  C6   z from C6 to F      x from C6 to C2
  F    z from C6 to F      x from C6 to C2
End
```



This choice of axes imposes the C_{2v} symmetry. If we had wished to use the global axis system, the `Axes...End` block would not be needed.

First split the non-local polarizability file which contains polarizabilities at all computed frequencies (1+10 in this example) into 11 parts; one for each frequency. This can be done with a `split` command on a Linux system (see the man page for `split` for details), or using Process with the following input file (at present, this file is created by hand):

```
$ cat split_NL4.prss
!
! execute with: process < split_NL4.prss
!

read non-local pols for XIII
  Use ascii file XIII_0.0005_1000_f11_NL4.pol
  Maximum rank 4
  ! Use LIMIT to limit the rank, but it's not essential here.
  ! Limit rank to 2
```

```
  Sites C1 C2 C3 C4 C5 C6 BR1 BR2 CL F H1 H2
  Frequencies Static + 10
End

Write
  ORIENT file for XIII with Freq 0
  ORIENT file for XIII with Freq 1
  ORIENT file for XIII with Freq 2
  ORIENT file for XIII with Freq 3
  ORIENT file for XIII with Freq 4
  ORIENT file for XIII with Freq 5
  ORIENT file for XIII with Freq 6
  ORIENT file for XIII with Freq 7
  ORIENT file for XIII with Freq 8
  ORIENT file for XIII with Freq 9
  ORIENT file for XIII with Freq 10
End

Finish
```

On execution, PROCESS creates the files `XIII_NL4_000.pol, XIII_NL4_001.pol` etc. As always, the file with index `0` or `000` contain the static polarizabilities.

Now we create input files for ORIENT and PROCESS input file using CLUSTER. Here's our CLUSTER input file

```
$ cat XIII.clt
! XIII : Cluster input file
!
UNITS bohr
Overwrite Yes

Molecule XIII
  ! XIII MP2/6-31G(d,p) optimised using Gaussian
  UNITS  Angstrom
   C1   6.0    1.203608   -0.333879    0.000000
   C2   6.0    1.213772   -1.728343    0.000000
   C3   6.0    0.000000    0.392209    0.000000
   C4   6.0   -1.204017   -0.331686    0.000000
   C5   6.0   -1.216822   -1.726687    0.000000
   C6   6.0   -0.002190   -2.396385    0.000000
   Br1 35.0   -2.874405    0.561295    0.000000
   Br2 35.0    2.875462    0.556057    0.000000
   Cl  17.0    0.001684    2.113032    0.000000
   F    9.0   -0.003056   -3.747234    0.000000
   H1   1.0   -2.146732   -2.275780    0.000000
   H2   1.0    2.142535   -2.279381    0.000000
End

! Write XIII in ORIENT format and PROCESS command file

Run-Type
  Properties
  Molecule XIII
  Camcasp-path usr/local/camcasp
  ORIENT file
  PROCESS file
  SITES file
  INTERFACE files
End

Finish
```

Notice that we have used the `CAMCASP-PATH` command to tell CLUSTER where the CAMCASP installation is. This is the only place you need do this. On execution, CLUSTER generates the files `XIII.ornt` and `XIII.prss`.

We will now calculate localized polarizabilities using the localization procedure of Le Sueur and Stone [1994] that is programmed in ORIENT. Let's look at the file `XIII.ornt`:

```
$ cat XIII.ornt
! ORIENT localization file for XIII

UNITS BOHR

Parameters
      Sites     13 polarizable     13
      S-functions 10000
      Alphas 15000
End

Types
      C    Z     6
      Br   Z    35
      Cl   Z    17
      F    Z     9
      H    Z     1
End

Molecule  XIII at  0.0 0.0 0.0
  ...molecule geometry...
End

Polarizabilities for XIII
   Read rank 4 sites +++
      C1  C2  C3  C4  C5  C6  BR1 BR2 CL  F    +++
      H1  H2
   #include XIII_NL4_<INDEX!000>.pol
   Limit rank 2 for +++
      C1  C2  C3  C4  C5  C6  BR1 BR2 CL  F    +++
      H1  H2
   Sum-rule test 1e-6
   Localize test 1e-6 sites +++
      C1  C2  C3  C4  C5  C6  BR1 BR2 CL  F    +++
      H1  H2
End
Edit XIII
   #include <LOCAL-AXES!local_axes>
   Bonds Auto
End

Polarizabilities for XIII
   Write limit 2 file XIII_L2_<INDEX!000>.pol sites +++
      C1  C2  C3  C4  C5  C6  BR1 BR2 CL  F    +++
      H1  H2
   Print limit 2 +++
      C1  C2  C3  C4  C5  C6  BR1 BR2 CL  F    +++
      H1  H2
End

Finish
```

There are two things we need to change:

- The local axes need to be set using the `LOCAL-AXES` tag. So replace `LOCAL-AXES` with the name of our local axes file. This can be done by editing the file, or using the `replace` command described next.

  If you are working in the global axis system, comment out or delete the line containing the `LOCAL-AXES` tag.

- `INDEX` is the frequency index. It is a three digit tag of the form `000, 001, 002` etc. We will have to replace `INDEX` with each of these indices, and run ORIENT for each one in turn.

Both this and the LOCAL-AXES replacements can be done from the command line using the replace command supplied with CamCASP:

```
$ ${CAMCASP}/bin/replace LOCAL-AXES=local_axes INDEX=001 < XIII.ornt > XIII_1.ornt
```

Actually the default value of LOCAL-AXES is local_axes so the first argument of the replace command could be omitted altogether.

NOTE: Do not try to overwrite the XIII.ornt file using

```
$ ${CAMCASP}/bin/replace INDEX=001 < XIII.ornt > XIII.ornt
```

This will result in you losing the file XIII.ornt. Don't worry if you do do this my mistake as it can be re-generated using the Cluster code in a single command.

After Orient has been run for each of the frequency indices, we will have the localized polarizabilities in binary format in the files XIII_L2_000.pol, XIII_L2_001.pol, etc.

Before going onto the next step, we need to do a couple of things:

- Concatenate these binary localized polarizability files into one file. Use the script cat_local.bash (located in $CAMCASP/scripts/dispersion/) for this:

  ```
  $ $CAMCASP/scripts/dispersion/cat_local.bash XIII
  ```

  This will create the file XIII_L2_0f10.pol. The '0f10' indicates the file contains the static polarizabilities and frequency-dependent polarizabilities at 10 frequencies.

- Split the point-to-point polarizabilities file into 11 parts, one for each frequency. The point-to-point polarizabilities will be used in the refinement procedure. At present, they are computed at 11 frequencies (default) by CamCASP and put in one file. But Pfit can handle only one frequency at a time and consequently expects to read in the point-to-point polarizabilities computed at a single frequency. So we split the file into 11 parts using Process. The input file for this is:

  ```
  read P2P pols for XIII
    Frequencies Static + 10
    ! Use this command to split the P-2-P pol file:
    P2P-Pols XIII_lim2.0_4.0_p2000_f11.p2p SPLIT
  End

  Finish
  ```

  The split point-to-point polarizabilities will be put in files XIII_000.p2p, XIII_001.p2p, etc.

We are now set to refine these localized polarizabilities. The refinement is done using the Pfit program. The input is somewhat complicated, but Process will write the input files for Pfit. We have already created the input file for Process. It is file XIII.prss. Here's what it looks like:

```
$ cat XIII.prss
TITLE  PROCESS file for XIII
! Usage: process < XIII.prss

CAMCASP /usr/local/camcasp

Read local pols for XIII
   Use binary file XIII_L2_0f10.pol
   Maximum rank  2
   Sites +++
      C1  C2  C3  C4  C5  C6  BR1 BR2 CL  F    +++
      H1  H2
```

```
   Limit rank to 2
   Limit rank to 1 for sites +++
      H1  H2
   Frequencies STATIC + 10
   ! Use this command to split the P-2-P pol file into 11 parts:
   ! P2P-Pols <point-2-point pol file> SPLIT
End

Write
   PFIT file for XIII +++
      with cutoff 0.0001 and freq <INDEX!0> +++
      with penalties and weight 4
End

Finish
```

Once again we see the tag `INDEX`. As before, this tag will need to be replaced by the three digit frequency index (`000`, `001`, etc.) using:

```
  $ ${CAMCASP}/bin/replace INDEX=000 < XIII.prss > XIII_0.prss
```

So do the replacement, and run PROCESS for each of them and send the output of PROCESS to a suitable file. The output is the input for the PFIT code. So, for the static case, replace `INDEX` by `000` and use the command:

```
$ process < XIII.prss > ref_000.pfit
```

So what do one of these PFIT input files look like? Here's a part of one file (some parts removed):

```
$ cat ref_000.pfit
 Allocate
    points   2000
    batches 1
    sites    12
    rank     2
    parameters  218
 End

 Data XIII_000.p2p

 #include <SITEFILE!local_axes_pfit>

 Polarizabilities
      C1   C1    10   10 =      C1_10_10_A
      C1   C1    10  21c =     C1_10_21c_A
      C1   C1    10  21s =     C1_10_21s_A
      C1   C1   11c  11c =    C1_11c_11c_A
         ...
         ...
         ...
      H2   H2   11c  11s =    H2_11c_11s_A
      H2   H2   11s  11s =    H2_11s_11s_A

 End

 Penalties
          C1_10_10_A +++
             5.74440962      0.00001000
         C1_10_21c_A +++
             1.94202436      0.00000000
         C1_10_21s_A +++
             8.35043368      0.00000000
         ...
```

```
        ...
        ...
       H2_11c_11s_A +++
           -1.02508681      0.00001000
       H2_11s_11s_A +++
            2.94522570      0.00001000
End

! Damping <factor>

Start

Export Polarizabilities to file XIII_ref_wt4_L2_f0.pol
Print Total Origin 0.0 0.0 0.0

Finish
```

The tag SITEFILE needs to be replaced with the name of the molecule definition and local axes file.

Now we are ready to refine the local polarizabilities at zero frequency (static):

```
$ pfit < ref_000.pfit > ref_000_pfit.out
```

The refined polarizabilities will be in file XIII_ref_wt4_L2_f0.pol. These are in ASCII format. Check them. Also check the errors reported in the PFIT output in file ref_000_pfit.out.

Do this for all frequencies. By the way, we do not need the static polarizabilities for calculating the dispersion coefficients; we just used them as an example. We will need the static polarizabilities for the induction energy calculation. But that is the subject of another section (as yet unwritten).

Finally, we now have all the data needed to calculate the dispersion coefficients using the CASIMIR program. The input file for this program is probably the most complicated of all. But once again, PROCESS will come to our aid and write out this file.

First concatenate all the refined local polarizabilities into one file using the script cat_refined.bash located in $CAMCASP/scripts/dispersion/:

```
$ $CAMCASP/scripts/dispersion/cat_refined.bash XIII_ref_wt4_L2
```

The argument given to this script is the bit of the name given to the refined polarizability files: {NAME}_f<n>.pol, where n is the polarizability index. Yes, this time n takes the values 0,1,2,3...,10 and not 000,001...,010 as before. We haven't yet got all the bits consistent, but you get the idea.

This creates the file XIII_ref_wt4_L2_0f10_ref.pol. We added the extra ref to avoid accidentally overwriting one of the earlier polarizability files. Either leave it there or move it to file XIII_ref_wt4_L2_0f10.pol.

Now copy file XIII.prss to XIII_casimir.prss and edit the latter to look like:

```
$ cat XIII_casimir.prss
TITLE  PROCESS file for XIII
! Usage: process < XIII.prss

Global-Data
  Camcasp /usr/local/camcasp
End

Read local pols for XIII
   Use ascii file XIII_ref_wt4_L2_0f10.pol
   Maximum rank  2
   Sites +++
      C1  C2  C3  C4  C5  C6  BR1 BR2 CL  F    +++
      H1  H2
   Limit rank to 2
   Limit rank to 1 for sites +++
```

```
      H1  H2
   Frequencies STATIC + 10
End

Write
   CASIMIR file for XIII and XIII +++
      with cutoff 0.0001
End

Finish
```

Run PROCESS:

```
$ process < XIII_casimir.prss > XIII_ref_wt4_L2.casimir
```

Now run the CASIMIR program:

```
$ casimir < XIII_ref_wt4_L2.casimir > XIII_ref_wt4_L2_casimir.out
```

And we've finially got our dispersion coefficients (here's part of the output):

```
$ cat XIII_ref_wt4_L2_casimir.out
Title XIII ... XIII
XIII
Frequencies   0.5    10
Skip  0
Print nonzero

Molecule  XIII
Site  C1
10 10
             5.72200000      5.70700000      5.61200000      5.27400000 +++
             4.44000000      3.07700000      1.62100000      0.58700000 +++
             0.11800000      0.00600000
   ...
   ...
   ...
Dispersion coefficients for XIII and XIII
  C1  C1               C6             C7              C8
    00   00   0    88.39684131     0.0         4820.75066614
    00   11c  1     0.0          809.88012198
    00   11s  1     0.0          477.13620553
    00   20   2   -24.31832249     0.0        -3056.70651103
   ...
   ...
   ...
Finish
```

C Older Scripts

These scripts are still present, but have been superseded by Python scripts. However they are still available in case there are issues with Python installations.

For an up-to-date usage of these script use `<script> --help`.

C.1 High-level scripts

Most CamCASP calculations can be set up and run using the high-level scripts in conjunction with one simple data file.

The following scripts are currently available. They can all be executed with the single argument `--help` to print brief usage details and exit.

- `runSAPT`
  Script to run a complete SAPT(DFT) calculation to calculate the components of the interaction energy between two molecules, or to calculate the properties (multipole moments and polarizabilities) of a single molecule. The command is just
  `runSAPT` *JOB* `-q` *queue* `-M` *memory*
  The 'queue' may be `bg` to run the calculation in the background, or `none` just to set up all the files and prepare a script for submitting the calculation. This allows you to modify any of the files if necessary. You need to provide one simple data file, *JOB*.clt, to define the system and specify the calculation. For details see §7 and the example in §8.1 below.

  Because of an inconsistency in the site order used by these scripts and the newer Python scripts, your Cluster input file needs to have the line `SITE-ORDER OLD` in the `RUN-TYPE` block. If this is not present, the CamCASP interaction energy calculation will still proceed, but the results may be meaningless.

- `search.pl`
  The output of a SAPT(DFT) calculation can be analysed using the script `search.pl`. It picks out the energy terms from the CamCASP output file and displays them in a compact form.

- `localize`
  Using the polarizability files from a properties calculation, this script will generate local polarizabilities (static and dynamic). See §8.2.1 on p. 32 for details. The dynamic polarizabilities are then used to construct dispersion coefficients, if required.

C.2 Low-level scripts

Some of the scripts provided in the CamCASP package are low-level ones that just perform one step in setting up the job, for example by generating further files that are needed for the calculations. If you need to do something unusual you may have to use the lower-level scripts directly, but for most purposes the high-level scripts will suffice.

In all the following, *JOB* is the job-name, which identifies all the files belonging to the job — in particular, the cluster input file, usually called *JOB*`.clt`. Most jobs involve a large number of files, however, so it is not a good idea to rely on the job-name to distinguish files for different jobs from each other. Recommended practice is to run every job in its own private directory (folder).

The starting-point for all tasks is the 'cluster' file, conventionally given the `.clt` suffix, which contains information about the molecules involved, the basis sets to be used, and other details. Details are in §7. Once you have prepared the input file for Cluster, called for example `setup.clt`, the `cluster` command is the first step in setting up the job:

```
cluster < setup.clt
```

(It will usually be called by a script rather than directly.) This will produce a file called *JOB*`.template`, where *JOB* is the file prefix specified (or implied) in the Cluster data file. It will also produce a number of other files — for example *JOB*`.cks` and *JOB*`_P.data` — depending on the information in the `.clt` file.

Next, `generate.pl` is a low-level script that generates the input files for the DALTON *ab initio* calculation. These have names like `A.mol`, and `MA.mol`. Again, the particular files that are produced depend on the nature of the calculation.

`generate.pl` *options JOB*

The options to this command are as follows:

`-monomer`	This is to be a properties calculation for a single molecule.
`-mc+, -mc`	This is a dimer interaction calculation, using monomer-centred basis sets, with or without midbond functions. `mc+` is the default calculation type.
`-dc+, -dc`	This is a dimer interaction calculation, using dimer-centred basis sets, with or without midbond functions.
`-h, -help`	Prints a help page to standard output.

DALTON also requires .dal files that describe the type of calculation to be done. Usually, these files do not depend on the particular molecule or molecules for which the calculation is being done (they are described in the `.mol` files) but in the case of DFT calculations using the asymptotically-corrected PBE0 functional it is necessary to specify the molecular ionization potential in order to set up the asymptotic correction properly. Accordingly the `dal.pl` command that sets up these files requires the ionization potentials:

`dal.pl` *options*

where the options are

`[-j]` *JOB*	Specifies the jobname to be used as the file prefix
`-a` *moleculeA*	Alternatively, the molecule names can be specified, in which case the jobname will be constructed from them.
`-b` *moleculeB*	For a dimer calculation, both molecules need to be specified.
`-ipa` or `-ip` *IP*	Vertical ionization potential for molecule *A*, or for a monomer (in Hartree).
`-ipb` *IP*	Vertical ionization potential for molecule *B*, if present.
`-monomer`	This is to be a properties calculation for a single molecule.
`-mc+, -mc`	This is a dimer interaction calculation, using monomer-centred basis sets, with or without midbond functions.
`-dc+, -dc`	This is a dimer interaction calculation, using dimer-centred basis sets, with or without midbond functions.

The file `CAMCASP/misc/IP.txt` contains a list of ionization potentials for some small molecules; many others are available in the NIST Chemistry Webbook at `webbook.nist.gov/chemistry/`. If the IP cannot be found there or elsewhere in the literature, it is necessary to calculate it.

References

O. Akin-Ojo, R. Bukowski, and K. Szalewicz. Ab initio studies of He-HCCCN interaction. *J. Chem. Phys.*, 119: 8379–8396, 2003.

R. Bukowski, R. Podeszwa, and K. Szalewicz. Efficient generation of the coupled Kohn–Sham dynamic susceptibility functions and dispersion energy with density fitting. *Chem. Phys. Lett.*, 414:111–116, 2005.

M. E. Casida. Time-dependent density-functional response theory for molecules. In D. P. Chong, editor, *Recent Advances in Density-Functional Theory*, page 155. World Scientific, 1995.

M. E. Casida and D. R. Salahub. Asymptotic correction approach to improving approximate exchange-correlation potentials: Time-dependent density-functional theory calculations of molecular excitation spectra. *J. Chem. Phys.*, 113:8918–8935, 2000.

S. M. Colwell, N. C. Handy, and A. M. Lee. Determination of frequency-dependent polarizabilities using current density-functional theory. *Phys. Rev. A*, 53:1316–1322, 1995.

S. Grimme. Semiempirical GGA-type density functional constructed with a long-range dispersion correction. *J. Comp. Chem*, 27:1787 – 1799, 2006.

M. Gruning, O. V. Gritsenko, S. J. A. van Gisbergen, and E. J. Baerends. Shape corrections to exchange-correlation potentials by gradient-regulated seamless connection of model potentials for inner and outer region. *J. Chem. Phys.*, 114:652–660, 2001.

A. Hesselmann, G. Jansen, and M. Schütz. Density-functional theory-symmetry-adapted intermolecular perturbation theory with density fitting: A new efficient method to study intermolecular interaction energies. *J. Chem. Phys.*, 122:014103, 2005.

M. P. Hodges and R. J. Wheatley. Flexible multipole models for hydrogen fluoride. *Phys. Chem. Chem. Phys*, 2: 1631–1638, 2000.

B. Jeziorski, R. Moszynski, A. Ratkiewicz, S. Rybak, K. Szalewicz, and H.L. Williams. *SAPT: A program for many-body symmetry-adapted perturbation theory calculations of intermolecular interaction energies*, volume B, page 79. STEF, Cagliari, 1993.

B. Jeziorski, R. Moszynski, and K. Szalewicz. Perturbation theory approach to intermolecular potential energy surfaces of van der waals complexes. *Chem. Rev.*, 94:1887–1930, 1994.

C. R. Le Sueur and A. J. Stone. Localization methods for distributed polarizabilities. *Molec. Phys.*, 83:293–308, 1994.

T. C. Lillestolen and R. Wheatley. Atomic charge densities generated using an iterative stockholder approach. *J. Chem. Phys.*, 131:144101, 2009.

T. C. Lillestolen and R. J. Wheatley. First-principles calculation of local atomic polarizabilities. *J. Phys. Chem. A*, 111:11141–11146, 2007. doi: 10.1021/jp073151y.

A. J. Misquitta. Charge-transfer from regularized symmetry-adapted perturbation theory. *J. Chem. Theory Comput.*, 9:5313–5326, 2013. doi: 10.1021/ct400704a.

A. J. Misquitta and A. J. Stone. Distributed polarizabilities obtained using a constrained density-fitting algorithm. *J. Chem. Phys.*, 124:024111, 2006.

A. J. Misquitta and A. J. Stone. Accurate induction energies for small organic molecules: I. Theory. *J. Chem. Theory Comput.*, 4:7–18, 2008a.

A. J. Misquitta and A. J. Stone. Dispersion energies for small organic molecules: first row atoms. *Molec. Phys.*, 106:1631 – 1643, 2008b.

A. J. Misquitta and K. Szalewicz. Intermolecular forces from asymptotically corrected density functional description of monomers. *Chem. Phys. Lett.*, 357:301–306, 2002.

A. J. Misquitta and K. Szalewicz. Symmetry-adapted perturbation-theory calculations of intermolecular forces employing density-functional description of monomers. *J. Chem. Phys.*, 122:214109, 2005.

A. J. Misquitta, B. Jeziorski, and K. Szalewicz. Dispersion energy from density-functional theory description of monomers. *Phys. Rev. Lett.*, 91:33201, 2003.

A. J. Misquitta, R. Podeszwa, B. Jeziorski, and K. Szalewicz. Intermolecular potentials based on symmetry-adapted perturbation theory with dispersion energies from time-dependent density-functional theory. *J. Chem. Phys.*, 123:214103, 2005.

A. J. Misquitta, A. J. Stone, and S. L. Price. Accurate induction energies for small organic molecules. 2. Development and testing of distributed polarizability models against SAPT(DFT) energies. *J. Chem. Theory Comput.*, 4:19–32, 2008a. doi: 10.1021/ct700105f.

A. J. Misquitta, G. W. A. Welch, A. J. Stone, and S. L. Price. A first principles prediction of the crystal structure of $C_6Br_2ClFH_2$. *Chem. Phys. Lett.*, 456:105–109, 2008b.

Alston J. Misquitta, Anthony J. Stone, and Farhang Fazeli. Distributed multipoles from a robust basis-space implementation of the iterated stockholder atoms procedure. *J. Chem. Theory Comput.*, 2014. doi: 10.1021/ct5008444.

K. Patkowski, B. Jeziorski, and K. Szalewicz. Symmetry-adapted perturbation theory with regularized Coulomb potential. *J. Mol. Struct. (TheoChem)*, 547:293–307, 2001a.

K. Patkowski, B. Jeziorski, and K. Szalewicz. Symmetry-adapted perturbation theory with regularized coulomb potential. *J. Mol. Struct. (Theochem)*, 547:293–307, 2001b.

K. Patkowski, B. Jeziorski, and K. Szalewicz. Unified treatment of chemical and van der waals forces via symmetry-adapted perturbation expansion. *J. Chem. Phys.*, 120:6849–6862, 2004.

Konrad Patkowski, Krzysztof Szalewicz, and Bogumil Jeziorski. Induction and exchange-induction energy with a regularized coulomb potential. private communication, 2012.

R. Podeszwa, R. Bukowski, and K. Szalewicz. Density-fitting method in symmetry-adapted perturbation theory based on kohn-sham description of monomers. *J. Chem. Theory Comput.*, 2:400–412, 2006.

Fazle Rob and Krzysztof Szalewicz. Asymptotic dispersion energies from distributed polarizabilities. *Chem. Phys. Lett.*, 572:146–149, 2013.

A. J. Stone. Distributed multipole analysis: Stability for large basis sets. *J. Chem. Theory Comput.*, 1:1128–1132, 2005.

A. J. Stone and A. J. Misquitta. Atom–atom potentials from *ab initio* calculations. *Int. Rev. Phys. Chem.*, 26: 193–222, 2007.

K. T. Tang and J. Peter Toennies. The damping function of the van der Waals attraction in the potential between rare gas atoms and metal surfaces. *Surf. Sci. Lett.*, 279:203–206, 1992.

David J. Tozer and Nicholas C. Handy. Improving virtual Kohn–Sham orbitals and eigenvalues: Application to excitation energies and static polarizabilities. *J. Chem. Phys.*, 109:10180–10189, 1998.

T. Verstraelen, P.W. Ayers, V. Van Speybroeck, and M. Waroquier. The conformational sensitivity of iterative stockholder partitioning schemes. *Chem. Phys. Lett.*, 545:138–143, 2012.

H. L. Williams, E. M. Mas, K. Szalewicz, and B. Jeziorski. On the effectiveness of monomer-, dimer-, and bond-centered basis functions in calculations of intermolecular interaction energies. *J. Chem. Phys.*, 103:7374–7391, 1995.