AIMAll AB Testing

AA'

The default method of running AIMAll in ICHOR is to calculate all atomic properties in the AA' format. The following command is used to run AIMAll on all atoms using AA':

```
aimall -atoms=all -encomp=3 ... METHANOL001.wfn
```

Running this command on an example methanol wavefunction file `METHANOL0001.wfn`, results in a `METHANOL0001_atomicfiles` directory containing the following files:

```
>> ls -l METHANOL0001_atomicfiles
.rw-r--r-- 468 mfbx4mb9 20 Jul 14:10 c1.inp
.rw-r--r-- 35k mfbx4mb9 20 Jul 14:10 c1.int
.rw-r--r-- 240 mfbx4mb9 20 Jul 14:10 h2.inp
.rw-r--r-- 31k mfbx4mb9 20 Jul 14:10 h2.int
.rw-r--r-- 240 mfbx4mb9 20 Jul 14:10 h4.inp
.rw-r--r-- 31k mfbx4mb9 20 Jul 14:11 h4.int
.rw-r--r-- 240 mfbx4mb9 20 Jul 14:10 h5.inp
.rw-r--r-- 31k mfbx4mb9 20 Jul 14:12 h5.int
.rw-r--r-- 240 mfbx4mb9 20 Jul 14:10 h6.inp
.rw-r--r-- 31k mfbx4mb9 20 Jul 14:12 h6.int
.rw-r--r-- 316 mfbx4mb9 20 Jul 14:10 o3.inp
.rw-r--r-- 33k mfbx4mb9 20 Jul 14:11 o3.int
```

Focusing on the IQA Energy components only, the outputted `c1.int` file contains the following (some values cutout for clarity):

```
IQA Energy Components (see "2EDM Note"):
----------------------------------------
T(A)                                  =  3.7361664519E+01
...
E_IQA0(A)                             = -3.7837777138E+01
E_IQA(A)                              = -3.8055484376E+01
E_IQA_Intra0(A)                       = -3.7297313951E+01
E_IQA_Intra(A)                        = -3.7515021190E+01
E_IQA_Inter0(A)                       = -5.4046318648E-01
E_IQA_Inter(A)                        = -5.4046318648E-01
```

As you can see, the IQA energy (`E_IQA(A)`) is roughly `-38 Ha`. The IQA energy is made up of two parts: the intra-atomic energy (`E_IQA_Intra(A)`) and a sum over the interatomic interaction energies (`E_IQA_Inter(A)`). In the example above, the intra-atomic energy is roughly `-37.5 Ha` and the interatomic energy is roughly `-0.5 Ha`, the sum of which results in the correct total IQA energy of `-38 Ha`. In the AA' format (specified by the `-encomp=3` flag), the interatomic interaction energy is the sum over the interaction between every other atom in the system.

AB

It is possible to calculate the interaction between individual atom pairs known as AB interactions. For model knitting, the whole system of the source molecule will not be available during the simulation and therefore one may wish to use a subsystem (target) molecule where the energies are computed with only the atoms that will be available during the simulation (think of the central amino acid in a peptide capped amino acid). For clarity, the following atom labels will be used when referring to the example methanol:

```
H6
  \
   O3
   |
   C1
/  |  \
H2 H3 H4
```

To run AIMAll to calculate AB interactions, one must use the `-encomp=4` flag:

```
aimall -atoms=all -encomp=4 ... METHANOL001.wfn
```

Running the above again results in a `METHANOL0001_atomicfiles` directory containing the following:

```
>> ls -l METHANOL0001_atomicfiles
.rw-r--r--  468 mfbx4mb9 20 Jul 14:05 c1.inp
.rw-r--r--  35k mfbx4mb9 20 Jul 14:05 c1.int
.rw-r--r--  467 mfbx4mb9 20 Jul 14:05 c1_h2.inp
.rw-r--r-- 5.2k mfbx4mb9 20 Jul 14:07 c1_h2.int
.rw-r--r--  467 mfbx4mb9 20 Jul 14:05 c1_h4.inp
.rw-r--r-- 5.2k mfbx4mb9 20 Jul 14:08 c1_h4.int
.rw-r--r--  467 mfbx4mb9 20 Jul 14:05 c1_h5.inp
.rw-r--r-- 5.2k mfbx4mb9 20 Jul 14:08 c1_h5.int
.rw-r--r--  467 mfbx4mb9 20 Jul 14:05 c1_h6.inp
.rw-r--r-- 5.2k mfbx4mb9 20 Jul 14:08 c1_h6.int
.rw-r--r--  467 mfbx4mb9 20 Jul 14:05 c1_o3.inp
.rw-r--r-- 5.2k mfbx4mb9 20 Jul 14:08 c1_o3.int
.rw-r--r--  240 mfbx4mb9 20 Jul 14:05 h2.inp
.rw-r--r--  31k mfbx4mb9 20 Jul 14:05 h2.int
.rw-r--r--  239 mfbx4mb9 20 Jul 14:05 h2_h4.inp
.rw-r--r-- 5.0k mfbx4mb9 20 Jul 14:08 h2_h4.int
.rw-r--r--  239 mfbx4mb9 20 Jul 14:05 h2_h5.inp
.rw-r--r-- 5.0k mfbx4mb9 20 Jul 14:08 h2_h5.int
.rw-r--r--  239 mfbx4mb9 20 Jul 14:05 h2_h6.inp
.rw-r--r-- 5.0k mfbx4mb9 20 Jul 14:08 h2_h6.int
.rw-r--r--  239 mfbx4mb9 20 Jul 14:05 h2_o3.inp
.rw-r--r-- 5.0k mfbx4mb9 20 Jul 14:08 h2_o3.int
.rw-r--r--  240 mfbx4mb9 20 Jul 14:05 h4.inp
.rw-r--r--  31k mfbx4mb9 20 Jul 14:06 h4.int
.rw-r--r--  239 mfbx4mb9 20 Jul 14:05 h4_h5.inp
.rw-r--r-- 5.0k mfbx4mb9 20 Jul 14:09 h4_h5.int
.rw-r--r--  239 mfbx4mb9 20 Jul 14:05 h4_h6.inp
.rw-r--r-- 5.0k mfbx4mb9 20 Jul 14:09 h4_h6.int
```

```
.rw-r--r--  240 mfbx4mb9 20 Jul 14:05 h5.inp
.rw-r--r--  31k mfbx4mb9 20 Jul 14:07 h5.int
.rw-r--r--  239 mfbx4mb9 20 Jul 14:05 h5_h6.inp
.rw-r--r-- 5.0k mfbx4mb9 20 Jul 14:09 h5_h6.int
.rw-r--r--  240 mfbx4mb9 20 Jul 14:05 h6.inp
.rw-r--r--  31k mfbx4mb9 20 Jul 14:07 h6.int
.rw-r--r--  316 mfbx4mb9 20 Jul 14:05 o3.inp
.rw-r--r--  33k mfbx4mb9 20 Jul 14:06 o3.int
.rw-r--r--  315 mfbx4mb9 20 Jul 14:05 o3_h4.inp
.rw-r--r-- 5.1k mfbx4mb9 20 Jul 14:09 o3_h4.int
.rw-r--r--  315 mfbx4mb9 20 Jul 14:05 o3_h5.inp
.rw-r--r-- 5.1k mfbx4mb9 20 Jul 14:09 o3_h5.int
.rw-r--r--  315 mfbx4mb9 20 Jul 14:05 o3_h6.inp
.rw-r--r-- 5.1k mfbx4mb9 20 Jul 14:09 o3_h6.int
```

As you can see there are a lot more files, not only are there individual atom `.int` files but there are also `A_B.int` files. Looking again at the `c1.int` file at the IQA energy component section reveals the following:

```
IQA Energy Components (see "2EDM Note"):
----------------------------------------
T(A)                                  =  3.7361664519E+01
...
E_IQA0(A)                             = -3.7837777138E+01
E_IQA(A)                              = -3.8055484376E+01
E_IQA_Intra0(A)                       = -3.7297313951E+01
E_IQA_Intra(A)                        = -3.7515021190E+01
E_IQA_Inter0(A)                       = -5.4046318648E-01
E_IQA_Inter(A)                        = -5.4046318648E-01
```

As you can see, this is the exact same output as before using AA' (`-encomp=3`). This is to be expected, if we wanted the interaction energy between `C1` and `H2` only, this would be found in the `c1_h2.int` file. Again only looking at the IQA energy component section of `c1_h2.int` file reveals the following (again with some values removed for clarity):

```
 IQA Atomic and Diatomic Contributions to Diatomic ("Interaction") Energy
Components (See "2EDM Note"):
 --------------------------------------------------------------------------------
---------------------
 Atom A = C1
 Atom B = H2
 Vne(A,B)/2          = -1.318843+00    Vne(A,B)          = -2.6376872+00
 ...
 VC_IQA(A,B)/2       =  2.223259E-02   VC_IQA(A,B)       =  4.446518E-02
 VX_IQA(A,B)/2       = -1.416020E-01   VX_IQA(A,B)       = -2.832041E-01
 E_IQA_Inter(A,B)/2  = -1.193694E-01   E_IQA_Inter(A,B)  = -2.387389E-01
 Vne(B,A)/2          = -1.333995E+00   Vne(B,A)          = -2.667991E+00
 ...
 V_IQA(B,A)/2        = -1.193694E-01   V_IQA(B,A)        = -2.387389E-01
 VC_IQA(B,A)/2       =  2.223259E-02   VC_IQA(B,A)       =  4.446518E-02
 VX_IQA(B,A)/2       = -1.416020E-01   VX_IQA(B,A)       = -2.832041E-01
 E_IQA_Inter(B,A)/2  = -1.193694E-01   E_IQA_Inter(B,A)  = -2.387389E-01
```

This file is different to the single atom `.int` file as it only contains information about the interaction between atoms A and B. As mentioned before, we're interested in the interaction between atoms A and B (`E_IQA_Inter(A,B)`), more specifically we need `E_IQA_Inter(A,B)/2` as `E_IQA_Inter(A,B)` will include double counting. As you may have noticed, there exists `E_IQA_Inter(A,B)/2` and `E_IQA_Inter(B,A)/2`, this is because numerical errors may result in the integration of the reverse interaction having a different value to the initial integration (AB and BA may differ slightly). To correct for this, both values are provided and the mean of the two should be used.

Now to recreate the `E_IQA_Inter(A)` from the `c1.int` file, all `E_IQA_Inter(A,B)` (in reality `(E_IQA_Inter(A,B)/2 + E_IQA_Inter(B,A)/2)/2`) from each `c1_XX.int` file must be summed up (note that AIMAll only writes the `c1_h2.int` and not `h2_c1.int`, to get the `H2` interaction with `C1`, simply look in the `c1_h2.int` file).

As mentioned earlier, the reason to use `-encomp=3` is to only compute the IQA energies for an atom within a subsystem. To perform such a calculation in AIMAll, one can use the `-atoms` flag. For example if the IQA energy of `C1` is required for the `C1, H2` subsystem then one would use the following:

```
aimall -atoms=all_1,2 -encomp=4 ... METHANOL001.wfn
```

The above command computes the connectivity information for the entire system but only computes atomic properties for atoms 1 and 2 (`C1` and `H2`). Running the above command again produces a `METHANOL0001_atomicfiles` directory containing the following:

```
>> ls -l METHANOL0001_atomicfiles
.rw-r--r--  468 mfbx4mb9 20 Jul 14:01 c1.inp
.rw-r--r--  35k mfbx4mb9 20 Jul 14:02 c1.int
.rw-r--r--  467 mfbx4mb9 20 Jul 14:01 c1_h2.inp
.rw-r--r-- 5.2k mfbx4mb9 20 Jul 14:02 c1_h2.int
.rw-r--r--  240 mfbx4mb9 20 Jul 14:01 h2.inp
.rw-r--r--  31k mfbx4mb9 20 Jul 14:02 h2.int
```

As you can see there is a atomic `.int` file for both `C1` and `H2` and the interaction `.int` file for the single `C1` `H2` interaction. Looking in the `c1.int` file reveals the following (again only IQA energy components and trimmed down):

```
IQA Energy Components (see "2EDM Note"):
----------------------------------------
T(A)                               =  3.7361664519E+01
...
E_IQA0(A)                          = -3.7837777138E+01
E_IQA(A)                           = -3.8055484376E+01
E_IQA_Intra0(A)                    = -3.7297313951E+01
E_IQA_Intra(A)                     = -3.7515021190E+01
E_IQA_Inter0(A)                    = -5.4046318648E-01
E_IQA_Inter(A)                     = -5.4046318648E-01
```

As you can see the `E_IQA(A)` is still the IQA energy for the entire system (essentially the AA' result), therefore to get the `C1` IQA energy of the `C1`, `H2` subsystem only, we need to take the `E_IQA_Intra(A)` energy from the `c1.int` file (`-37.515021190`) and the `E_IQA_Inter(A,B)/2` (remembering to average both integration results) from the `c1_h2.int` (`-0.11936946953`) and add them together to get the `E_IQA(C1, [C1,H2])` value of `-37.63439066 Ha`.

If we expand our subsystem to say `C1`, `H2`, `O3`, `H4`, `H5` (all but `H6`) and want to get the IQA energy of `C1` then we must run the command:

```
aimall -atoms=all_1,2,3,4,5 -encomp=4 ... METHANOL001.wfn
```

Which will result in a `METHANOL0001_atomicfiles` directory containing the following:

```
>> ls -l METHANOL0001_atomicfiles
.rw-r--r--  468 mfbx4mb9 20 Jul 13:56 c1.inp
.rw-r--r--  35k mfbx4mb9 20 Jul 13:57 c1.int
.rw-r--r--  467 mfbx4mb9 20 Jul 13:56 c1_h2.inp
.rw-r--r-- 5.2k mfbx4mb9 20 Jul 13:59 c1_h2.int
.rw-r--r--  467 mfbx4mb9 20 Jul 13:56 c1_h4.inp
.rw-r--r-- 5.2k mfbx4mb9 20 Jul 13:59 c1_h4.int
.rw-r--r--  467 mfbx4mb9 20 Jul 13:56 c1_h5.inp
.rw-r--r-- 5.2k mfbx4mb9 20 Jul 14:00 c1_h5.int
.rw-r--r--  467 mfbx4mb9 20 Jul 13:56 c1_o3.inp
.rw-r--r-- 5.2k mfbx4mb9 20 Jul 13:59 c1_o3.int
.rw-r--r--  240 mfbx4mb9 20 Jul 13:56 h2.inp
.rw-r--r--  31k mfbx4mb9 20 Jul 13:57 h2.int
.rw-r--r--  239 mfbx4mb9 20 Jul 13:56 h2_h4.inp
.rw-r--r-- 5.0k mfbx4mb9 20 Jul 14:00 h2_h4.int
.rw-r--r--  239 mfbx4mb9 20 Jul 13:56 h2_h5.inp
.rw-r--r-- 5.0k mfbx4mb9 20 Jul 14:00 h2_h5.int
.rw-r--r--  239 mfbx4mb9 20 Jul 13:56 h2_o3.inp
.rw-r--r-- 5.0k mfbx4mb9 20 Jul 14:00 h2_o3.int
.rw-r--r--  240 mfbx4mb9 20 Jul 13:56 h4.inp
.rw-r--r--  31k mfbx4mb9 20 Jul 13:58 h4.int
.rw-r--r--  239 mfbx4mb9 20 Jul 13:56 h4_h5.inp
.rw-r--r-- 5.0k mfbx4mb9 20 Jul 14:00 h4_h5.int
.rw-r--r--  240 mfbx4mb9 20 Jul 13:56 h5.inp
.rw-r--r--  31k mfbx4mb9 20 Jul 13:58 h5.int
.rw-r--r--  316 mfbx4mb9 20 Jul 13:56 o3.inp
.rw-r--r--  33k mfbx4mb9 20 Jul 13:58 o3.int
.rw-r--r--  315 mfbx4mb9 20 Jul 13:56 o3_h4.inp
.rw-r--r-- 5.1k mfbx4mb9 20 Jul 14:00 o3_h4.int
.rw-r--r--  315 mfbx4mb9 20 Jul 13:56 o3_h5.inp
.rw-r--r-- 5.1k mfbx4mb9 20 Jul 14:00 o3_h5.int
```

Then to get the IQA energy for `C1` only, we need to get the `E_IQA_Intra(A)` value from the `c1.int` file and the `E_IQA_Inter` values from each of the interaction files that contain `C1`. Again the `E_IQA_Intra(A)` value is `-37.515021190` and the interaction energies are as follows:

Interaction File	E_IQA_Inter(A,B)/2 / Ha
`c1_h2.int`	`-0.119369470`
`c1_o3.int`	`-0.231242113`
`c1_h4.int`	`-0.114870472`
`c1_h5.int`	`-0.118706820`

The sum of which results in a value of `-0.584188875`, adding this to the `E_IQA_Intra(A)` value above results in a total IQA energy (`E_IQA(C1, [C1, H2, O3, H4, H5])`) of `-38.099210065 Ha`.

For completeness, the `c1_h6.int` file from the `-atoms=all -encomp=4` run above contains a (averaged) `E_IQA_Inter(A,B)/2` value of `0.043633966`, adding this to the other `E_IQA_Inter` values, a total `E_IQA_Inter` value of `-0.540554910` is achieved, compare this to the `E_IQA_Inter(A)` value from the `c1.int` value and it is very close to `-0.540463186` (difference is `0.24 kJ/mol`). This highlights that `-encomp=4` performs more integrations and therefore more opportunity for integration errors hence the difference.

Implementing AB in ICHOR

Now for a quick note on how this may be implemented in ICHOR. At the time of writing, in the ICHOR repository, there is a branch called v3.1-ab, in this branch is a simple implementation of how to read and represent AB `.int` files. In `ichor.core.files.int` a `ABInt` class has been added which is responsible for reading the `A_B.int` files and storing the contents in the dictionary `ABInt.iqa_diatomic_contributions`. The `ABInt` class performs the averaging of each property whilst reading in the file and so the stored values are ready for use. If the `E_IQA_Inter(A,B)/2` value is needed, this is available (for an example `ABInt` instance `i`) with the following:

```
>>> i = ABInt("c1_h2.int")
>>> i.iqa_diatomic_contributions["E_IQA_Inter"]
-0.11936947
```

If you have used ICHOR in the past, you may be aware that is is rare to use the `INT` class directly but it is more common to interact with `.int` files through the `INTs` class. The v3.1-ab branch adds the `interaction_ints` to the `INTs` class which is a dictionary used to store the interaction files. Alongside the `interaction_ints` in the `INTs` class, the `interactions` attribute has also been added to the `INT` class which holds a dictionary of AB interaction files with the atom being interacted with. Take for example the `-atoms=all -encomp=4` run from above:

```
>>> i = INTs("METHANOL0001_atomicfiles")
>>> i["O3"].interactions
{
    'C1': ABInt('c1_o3.int'),
    'H2': ABInt('h2_o3.int'),
    'H4': ABInt('o3_h4.int'),
    'H5': ABInt('o3_h5.int'),
    'H6': ABInt('o3_h6.int'),
}
```

Then the IQA energy is implemented as a `property` in the `INT` class which computes the sum of the `intra` and `inter` atomic energies:

```
@property
def iqa(self) -> float:
    return self.e_intra + self.e_inter
```

Where `e_intra` and `e_inter` are given by the following:

```
@property
def e_intra(self) -> float:
    return self.iqa_energy_components["E_IQA_Intra(A)"]

@property
def e_inter(self) -> float:
    if len(self.interactions) > 0:
        return sum(i.e_inter for i in self.interactions.values())
    else:
        return self.iqa_energy_components["E_IQA_Inter(A)"]
```

The current implementation leaves much to be desired, the problem arises from the fact that the `e_inter` value from the `INT` class is reliant on which AB interaction files are present. This is undesirable as imagine the scenario where I run AIMAll using `-atoms=all -encomp=4` but only want the `C1`, `H2`, `H4`, `H5` subsystem. With the current implementation I would need to move or delete the files that are not part of this subsystem to avoid them being read. The fix to this issue is slightly complicated due to ICHOR's reliance on the `property` decorator. A better implementation may look like so:

```
def iqa(self, subsystem: Optional[List[str]] = None) -> float:
    return self.e_intra + self.e_inter(subsystem)

@property
def e_intra(self) -> float:
    return self.iqa_energy_components["E_IQA_Intra(A)"]

def e_inter(self, subsystem: Optional[List[str]] = None) -> float:
    if subsystem is None:
        return self.iqa_energy_components["E_IQA_Inter(A)"]
    else:
         return sum(self.interactions[atom].e_inter for atom in subsystem)
```

Where a `subsystem` (list of atom names to use as a subsystem) can be passed into `iqa` and `e_inter` to alter the output dynamically and not relying on the underlying files. The reason why the implementation does not currently look like the above is due to the `HasProperties` abstract base class which underpins a lot of ICHOR's file reading, from `ichor.core.files.file_data.HasProperties`:

```
@property
@abstractmethod
def properties(self) -> Dict[str, Any]:
    raise NotImplementedError(
        f"'properties' not defined for '{self.__class__.__name__}'"
    )
```

Then the `INT` implementation of `properties`:

```
@property
def properties(self) -> Dict[str, float]:
    return {
        **{"integration_error": self.integration_error, "iqa": self.iqa},
        **self.local_spherical_multipoles(),
    }
```

As you can see due to the use of `property` there is no way to pass in `subsystem` to `iqa` hence the naive basic implementation. There is also an issue with calling `local_spherical_multipoles` without explicitly passing in arguments and so a fix to the above issue is already underway (at the time of writing Yulian is taking lead on the fix) and the same solution can hopefully be used to solve the `iqa` issue.

Now onto how do we run AIMAll, in `ichor.hpc.submission_script.aimall_command` there exists the `AIMAllCommand` class which is responsible for informing the submission script how to run AIMAll. The constrictor is as follows:

```
class AIMAllCommand(CommandLine):
    """
    A class which is used to add AIMALL-related commands to a submission script.
It is used to write the submission script line where
    AIMALL modules are loaded. It is also used to write out the submission script
line where AIMALL is ran on a specified array of files (usually
    AIMALL is ran as an array job because we want to run hundreds of AIMALL tasks
in parallel). Finally, depending on the `check` and `scrub` arguments,
    additional lines are written to the submission script file which rerun failed
tasks as well as remove any points that did not terminate normally (even
    after being reran).
    :param wfn_file: Path to a .wfn file. This is not needed when running auto-
run for a whole directory.
    :param aimall_output: Optional path to the AIMALL job output. Default is None
as it is set to the `gjf_file_name`.aim if not specified.
    """

    def __init__(
        self,
        wfn_file: Path,
        atoms: Optional[Union[str, List[str]]] = None,
        aimall_output: Optional[Path] = None,
    ):
        ...
```

Where `atoms` in this case needs to be the `subsystem` that is currently being ran. AIMAll has a lot of arguments that are controlled by `GLOBALS`:

```python
@property
def arguments(self) -> List[str]:
    from ichor.hpc import GLOBALS

    logger.debug(f"atoms: {self.atoms}")

    atoms = (
        self.atoms
        if isinstance(self.atoms, str)
        else "all_"
        + ",".join(map(str, [get_digits(a) for a in self.atoms]))
    )

    return [
        "-nogui",
        "-usetwoe=0",
        f"-atoms={atoms}",
        f"-encomp={Encomp(GLOBALS.AIMALL_ENCOMP).value}",
        f"-boaq={Boaq(GLOBALS.AIMALL_BOAQ).value}",
        f"-iasmesh={IASMesh(GLOBALS.AIMALL_IASMESH).value}",
        f"-nproc={self.ncores}",
        f"-naat={self.ncores if self.atoms == 'all' else min(len(self.atoms),
self.ncores)}",
        f"-bim={BasinIntegrationMethod(GLOBALS.AIMALL_BIM).value}",
        f"-capture={Capture(GLOBALS.AIMALL_CAPTURE).value}",
        f"-ehren={Ehren(GLOBALS.AIMALL_EHREN).value}",
        f"-feynman={str(GLOBALS.AIMALL_FEYNMAN).lower()}",
        f"-iasprops={str(GLOBALS.AIMALL_IASPROPS).lower()}",
        f"-magprops={MagProps(GLOBALS.AIMALL_MAGPROPS).value}",
        f"-source={str(GLOBALS.AIMALL_SOURCE).lower()}",
        f"-iaswrite={str(GLOBALS.AIMALL_IASWRITE).lower()}",
        f"-atidsprops={ATIDSProps(GLOBALS.AIMALL_ATIDSPROPS).value}",
        f"-warn={str(GLOBALS.AIMALL_WARN).lower()}",
        f"-scp={SCP(GLOBALS.AIMALL_SCP.lower()).value}",
        f"-delmog={str(GLOBALS.AIMALL_DELMOG).lower()}",
        f"-skipint={str(GLOBALS.AIMALL_SKIPINT).lower()}",
        f"-f2w={F2W(GLOBALS.AIMALL_F2W).value}",
        f"-f2wonly={str(GLOBALS.AIMALL_F2WONLY).lower()}",
        f"-mir={MIR.Auto.value if GLOBALS.AIMALL_MIR < 0 else
GLOBALS.AIMALL_MIR}",
        f"-cpconn={CPConn(GLOBALS.AIMALL_CPCONN).value}",
        f"-intveeaa={IntVeeAA(GLOBALS.AIMALL_INTVEEAA).value}",
        f"-atlaprhocps={str(GLOBALS.AIMALL_ATLAPRHOCPS).lower()}",
        f"-wsp={str(GLOBALS.AIMALL_WSP).lower()}",
        f"-shm_lmax={SHMMax(GLOBALS.AIMALL_SHM).value}",
        f"-maxmem={GLOBALS.AIMALL_MAXMEM}",
        f"-verifyw={VerifyW(GLOBALS.AIMALL_VERIFYW).value}",
        f"-saw={str(GLOBALS.AIMALL_SAW).lower()}",
        f"-autonnacps={str(GLOBALS.AIMALL_AUTONNACPS).lower()}",
    ]
```

The third argument in is the `atoms` argument controlling which atoms to calculate atomic properties for. As you can see this is controlled by the `atoms` attribute passed into the `__init__` method whilst the `encomp` flag is controlled by the `GLOBALS.ENCOMP` variable (which is defaulted to `3`). Starting with the `encomp` flag, in order to perform AB calculations, this needs to be set to `4`. Now this can be done by the user in the config file but it would be better if it could be set automatically, for this reason (and to remove the dependence on `GLOBALS`) it would be best to move the `encomp` to the `__init__` method. As both variables are now controlled by the `__init__` method it would be reasonable to ask where is the `AIMAllCommand` class called from.

To answer that, the `AIMAllCommand` is called from three places: `ichor.hpc.main.aimall.submit_wfns`, `ichor.hpc.auto_run.auto_run_aimall.submit_aimall_job_to_auto_run` and `ichor.hpc.submission_script.morfi_command`. All three would need to be modified to accept the new changes. A proposed implementation would be to create a `GLOBALS.SUBSYSTEM` variable which would be a `Optional[List[str]]` (i.e. an optional list of atom names). If `GLOBALS.SUBSYSTEM` isn't set (i.e. is set to `None`) then normal running behaviour is followed, otherwise `GLOBALS.SUBSYSTEM` is used to initialise `atoms` and `encomp` is set to `4`. This may be a simple implementation which could be improved but should work.

> Note: `ichor.hpc.auto_run.auto_run_aimall.submit_aimall_job_to_auto_run` is called from `ichor.hpc.auto_run.standard_auto_run` using `IterArgs` and therefore would need to be set in `ichor.hpc.auto_run.standard_auto_run.setup_iter_args`