**Improvement to convergence test**

This is a simple, proof-of-concept implementation of the convergence tester. Below, I outline how the software operates, give notes on potential pitfalls, and suggest improvements.

**Step 1:** Given a pw.in file, a new material is generated.

* Running the software on the same structure will lead to duplicate materials in Mat3ra. An improved version would first search for a matching material in Mat3ra and only generate a new material if it doesn’t already exist.
* My structure parsing code extracting the structure from the pw.in file is extremely crude and can be improved.

**Step 2:** A search is carried out, calculating the energy for various k-point grids.

**Step 2a:** A new k-point grid is selected:

* In my implementation, I only evaluate k-point grids that have directional sampling ratios identical to some initial starting guess (e.g. 1x2x3, 2x4x6, etc.). This works perfectly for unit cells that have reciprocal lattice vectors with identical lengths. However, this is not the case in many unit cells. For these unit cells, a totally even sampling of reciprocal space may not be possible; therefore, an improved version would choose candidate k-point grids more intelligently, for example by finding grids that sample reciprocal space within some “evenness threshold”.

**Step 2b**: A new workflow is created for that k-point grid and a new job is run.

* Creation of a new workflow for each pw.in file is reasonable. However, having a workflow that takes the k-point grid as input would be much more efficient, allowing us to not create so many nearly identical workflows.
* Better yet, rather than repeatedly submitting jobs from the client, a larger workflow could be built that did the k-point convergence test entirely on Mat3ra. The results from that workflow could then be fetched by the client and given to the user. This approach does have some downsides, including difficulty to debug and perhaps less flexibility when restarting a job.