Command Files

Parameter Files (Prev)

The command file contains a sequence of commands that are read and executed in serial. The commands are organized into a JSON file. Below is an example of a command file for single-phase calculation:

```
[
    { "CaseId": "1" },
    {
        "FieldIO": {
            "IO" : "read",
            "Type": "omega",
            "Format": "basis",
            "Directory": "in/"
        }
    },
    {
        "SinglePhaseSCF": {
            "OutputDirectory": "out/"
        }
    },
    {
        "FieldIO": {
            "IO" : "write",
            "Type": "omega",
            "Format": "basis",
            "Directory": "out/omega/"
        }
    },
    {
        "FieldIO": {
            "IO" : "write",
            "Type": "phi",
            "Format": "real",
            "Directory": "out/phi/"
        }
    }
]
```

All commands are put in a pair of square brackets, and they are divided into different blocks. (Here gives an introduction to JSON)

The following explain the usage of each command block.

- The first block must be the `"CaseId"` block.

  ```
  { "CaseId": "your_case_id" }
  ```

  This command specifies the case ID of the calculation, `your_case_id`, which is part of the names of output files. The case ID can be anything, even an empty string.
- To read or write a field (*e.g.*, volume-fraction or conjugate field) file in a specified format, use `"FieldIO"` block.

  ```
  {
      "FieldIO": {
          "IO" : "read",
          "Type": "omega",
          "Format": "basis",
          "Directory": "in/"
      }
  }
  ```

"`IO`" is either "`read`" or "`write`" for reading from or writing to a file, respectively. "`Type`" specifies the field, which can be either "`omega`" for conjugate field or "`phi`" for volume-fraction field. "`Format`" specifies the format of the field, which can be either "`basis`" for the basis format, "`real`" for the real-space-grid format, or "`reciprocal`" for the reciprocal-space-grid format; see PSCF documentation for the explaination of these formats. "`Directory`" specifies the directory of the file. Finally, the file name is specified by the case ID, field type and format as `your_case_id_type.format`; for example, to read a conjugate field in the basis format as input of your SCF calculation with the case ID 1234, the file name must be `1234_omega.basis`.

- To perform SCF calculation of a single phase with given initial guess (the conjugate field of which should be read before), use the "`SinglePhaseSCF`" block.

  ```
  {
      "SinglePhaseSCF": {
          "OutputDirectory": "out/"
      }
  }
  ```

  "`OutputDirectory`" specifies the directory of the output file for the system free energy and its components. This output file name is `your_case_id_out.json`. For example, with the case ID 1234, the name of the output file is `1234_out.json`.

To perform automated calculation along a path (ACAP; see ACAP.pdf for details.), use the block "`ACAP`".

```
[
    { "CaseId": "1" },
    {
        "FieldIO": {
            "IO" : "read",
            "Type": "omega",
            "Format": "basis",
            "Directory": "in/"
        }
    },
    {
        "ACAP":{
            "Variable": ["chi", 0, 1],
            "InitialValue": 16,
            "FinalValue": 15.5,
            "InitialStep": 0.1,
            "SmallestStep": 0.001,
            "LargestStep": 0.5,
            "StepScale": 1.1,
            "OutputDirectory": "out/",
            "IntermediateOuput":
            [
                {
                    "OutputPoints": [15.4, 15.6, 15.8]
                },
                {
                    "Field" : "omega",
                    "Format": "basis",
                    "OutputDirectory": "out/omega/"
                },
                {
                    "Field" : "phi",
                    "Format": "real",
                    "OutputDirectory": "out/phi/"
                }
            ]
        }
    },
    {
        "FieldIO": {
            "IO" : "write",
            "Type": "omega",
            "Format": "basis",
```

```
            "Directory": "out/omega/"
        }
    },
    {
        "FieldIO": {
            "IO" : "write",
            "Type": "phi",
            "Format": "real",
            "Directory": "out/phi/"
        }
    }
]
```

"Variable" specifies the paramter whose value is varied along the path; this is so far either "chi", the Flory-Huggins parameter bewteen two segments of different types, or "b", the statistical segment length of a segment type. If the varing parameter is "chi", user needs to specify the two segment types as shown in the above example. If the varing parameter is "b", user needs to specify the corresponding segment type (*e.g.*, ["b", 0]). "InitialValue" and "FinalValue" give the starting and ending parameter values of the path, respectively. "InitialStep", "SmallestStep", and "LargestStep" specifies the initial, smallest and largest absolute values of the stepsize, respectively, used for varing the parameter along the path. "StepScale" specifies the scaling factor used to vary the stepsize. "OutputDirectory" specifies the directory of the output file for the system free energy and its components along the path. "IntermediateOuput" is needed when user wants to output field files during ACAP. The first block in "IntermediateOuput" specifies the parameter values at which the fields are output along the path (the order of these values does not matter, which means [1.1, 1.2, 1.3] and [1.2, 1.3, 1.1] result in the same intermediate output files). Each of the following blocks specifies the type of the field, its format, and the directory of the output files via "Field", "Format", and "IntermediateDirectory", respectively.

To find a boundary point between two specified phases, where they have the same Helmholtz free-energy density, use the "PhaseBoundaryPoints" block as in the following example:

```
[
    {   "CaseId": "1"   },
    {
        "FieldIO": {
            "PhaseId": 1,
            "IO" : "read",
            "Type": "omega",
            "Format": "basis",
            "Directory": "in/1/"
        }
    },
    {
        "FieldIO": {
            "PhaseId": 2,
            "IO" : "read",
            "Type": "omega",
            "Format": "basis",
            "Directory": "in/2/"
        }
    },
    {
        "PhaseBoundaryPoints": {
            "epsilon": 1e-5,
            "b": [1, 1.0],
            "InitialGuess(chi)": [0, 1, 19.1, 19.3]
        }
    }
    {
        "FieldIO": {
            "PhaseId": 1,
            "IO" : "write",
            "Type": "omega",
            "Format": "basis",
            "Directory": "out/1/omega/"
        }
    },
    {
        "FieldIO": {
            "PhaseId": 2,
            "IO" : "write",
            "Type": "phi",
            "Format": "real",
            "Directory": "out/2/phi/"
        }
    }
]
```

Here, the initial guess of each phase is read first by the two `"FieldIO"` blocks; different from the above single-phase calculation, `"PhaseId"` is needed in each `"FieldIO"` block, which takes the value of `1` or `2` in accordance to the command-line arguments of `-d`, `D1` and `D2`, respectively (see **Invoking an Executable**). In the `"PhaseBoundaryPoints"` block, `"epsilon"` specifies the criterion of convergence, which is the absolute difference in the Helmholtz free-energy density between the two phases; the next line specifies that the calculation is performed at the constant value for the statistical segment length (*i.e.*, `"b"`) of segment type 1, which is 1.0; in this case, the calculation solves for the χ value between segment types `0` and `1`, which falls in the interval of [19.1, 19.3] as shown in third line. Note that this interval is required by the Ridders' method used for the phase-boundary calculation.

Parameter Files (Prev)   **User Guide** (Up)