The listed locations include a local *$HOME/.OpenFOAM* directory and follow a descending order of precedence, *i.e.* the last location listed (*etc*) is lowest precedence.

If a user therefore wished to work permanently in USCS units, they could maintain a *controlDict* file in their *$HOME/.OpenFOAM* directory that includes the following entry.

```
DimensionedConstants
{
    unitSet  USCS;
}
```

OpenFOAM would read the `unitSet` entry from this file, but read all other *controlDict* keyword entries from the global *controlDict* file.

Alternatively, if a user wished to work on a *single case* in USCS units, they could add the same entry into the *controlDict* file in the *system* directory for their *case*. This file is discussed in the next section.

4.4 Time and data input/output control

The OpenFOAM solvers begin all runs by setting up a database. The database controls I/O and, since output of data is usually requested at intervals of time during the run, time is an inextricable part of the database. The *controlDict* dictionary sets input parameters *essential* for the creation of the database. The keyword entries in *controlDict* are listed in the following sections. Only the time control and `writeInterval` entries are mandatory, with the database using default values for any of the optional entries that are omitted. Example entries from a *controlDict* dictionary are given below:

```

application     icoFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         0.5;

deltaT          0.005;

writeControl    timeStep;

writeInterval   20;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;


// ************************************************************************* //
```

4.4.1 Time control

`startFrom` Controls the start time of the simulation.

- `firstTime`: Earliest time step from the set of time directories.
- `startTime`: Time specified by the `startTime` keyword entry.
- `latestTime`: Most recent time step from the set of time directories.

`startTime` Start time for the simulation with `startFrom startTime;`

`stopAt` Controls the end time of the simulation.

- `endTime`: Time specified by the `endTime` keyword entry.
- `writeNow`: Stops simulation on completion of current time step and writes data.
- `noWriteNow`: Stops simulation on completion of current time step and does not write out data.
- `nextWrite`: Stops simulation on completion of next scheduled write time, specified by `writeControl`.

`endTime` End time for the simulation when `stopAt endTime;` is specified.

`deltaT` Time step of the simulation.

4.4.2 Data writing

`writeControl` Controls the timing of write output to file.

- `timeStep`: Writes data every `writeInterval` time steps.
- `runTime`: Writes data every `writeInterval` seconds of simulated time.
- `adjustableRunTime`: Writes data every `writeInterval` seconds of simulated time, adjusting the time steps to coincide with the `writeInterval` if necessary — used in cases with automatic time step adjustment.
- `cpuTime`: Writes data every `writeInterval` seconds of CPU time.
- `clockTime`: Writes data out every `writeInterval` seconds of real time.

`writeInterval` Scalar used in conjunction with `writeControl` described above.

`purgeWrite` Integer representing a limit on the number of time directories that are stored by overwriting time directories on a cyclic basis. For example, if the simulations starts at $t = 5$s and $\Delta t = 1$s, then with `purgeWrite 2;`, data is first written into 2 directories, *6* and *7*, then when *8* is written, *6* is deleted, and so on so that only 2 new results directories exists at any time. *To disable the purging, specify* `purgeWrite 0;` (default).

`writeFormat` Specifies the format of the data files.

- `ascii` (default): ASCII format, written to `writePrecision` significant figures.
- `binary`: binary format.

`writePrecision` Integer used in conjunction with `writeFormat` described above, 6 by default.

`writeCompression` Switch to specify whether files are compressed with gzip when written: `on`/`off` (`yes`/`no`, `true`/`false`)

`timeFormat` Choice of format of the naming of the time directories.

- `fixed`: $\pm m.dddddd$ where the number of ds is set by `timePrecision`.
- `scientific`: $\pm m.dddddde\pm xx$ where the number of ds is set by `timePrecision`.
- `general` (default): Specifies `scientific` format if the exponent is less than -4 or greater than or equal to that specified by `timePrecision`.

`timePrecision` Integer used in conjunction with `timeFormat` described above, 6 by default.

`graphFormat` Format for graph data written by an application.

- `raw` (default): Raw ASCII format in columns.
- `gnuplot`: Data in gnuplot format.
- `xmgr`: Data in Grace/xmgr format.
- `jplot`: Data in jPlot format.

4.4.3 Other settings

`adjustTimeStep` Switch used by some solvers to adjust the time step during the simulation, usually according to `maxCo`.

`maxCo` Maximum Courant number, *e.g.* `0.5`

`runTimeModifiable` Switch for whether dictionaries, *e.g.controlDict*, are re-read during a simulation at the beginning of each time step, allowing the user to modify parameters during a simulation.

`libs` List of additional libraries (on $LD_LIBRARY_PATH) to be loaded at run-time, *e.g.*`("libNew1.so" "libNew2.so")`

`functions` Dictionary of functions, *e.g.* `probes` to be loaded at run-time; see examples in *$FOAM_TUTORIALS*

4.5 Numerical schemes

The *fvSchemes* dictionary in the *system* directory sets the numerical schemes for terms, such as derivatives in equations, that are calculated during a simulation. This section describes how to specify the schemes in the *fvSchemes* dictionary.

The terms that must typically be assigned a numerical scheme in *fvSchemes* range from derivatives, *e.g.* gradient ∇, to interpolations of values from one set of points to another. The aim in OpenFOAM is to offer an unrestricted choice to the user, starting with the choice of discretisation practice which is generally standard Gaussian finite volume integration. Gaussian integration is based on summing values on cell faces, which must be interpolated from cell centres. The user has a wide range of options for interpolation scheme, with certain schemes being specifically designed for particular derivative terms, especially the advection divergence $\nabla\bullet$ terms.