# Instructions

February 15, 2018

- The following five files will be output from the interface (under `inputs/` directory):
  - `eweld.in`
  - `eweld_weld_parameters.in`
  - `eweld_boundary_condition.in`
  - `eweld_preheat_interpass_temperature.in`
  - `eweld_temperature_monitor.in`
  - `eweld_mesh_key.txt` (Not need to do now. This option will allow users to input their own meshes. )
- For automatic mesh, the following steps will be run:
  1. Copy the `.in` files from one of these two directories to =inputs/
     - Under `inputs/test_run`: Input files for a fast/test run
     - Under `inputs/long_run`: Input files for an actual V-Groove simulation which take much longer to run

     For a test/quick case run:

     ```
     cp inputs/test_run/*.in inputs/
     ```

  2. Check if `pass_coordinates.out` exists in `input` directory, if not, run `utils/determine_passes_arc_v4.exe` to create `inputs/pass_coordinates.out`[1]:

     ```
     mkdir -p results/case_1/
     ./utils/determine_passes_arc_v4.out inputs/
     ```

     `eweld.in` will be input.

[1] On Linux, compile `determine_passes_arc_v4.out`, to get `determine_passes_arc_v4.out` via `gfortran determine_passes_arc_v4.for -o determine_passes_arc_v4.out`

3. Run `utils/Automesh_v14.py` from SALOME's Python Console to create `Mesh_3D.unv`, or run without Salome GUI:

   ```
   $SALOMEPATH/salome start -t -w 1 utils/Automesh_v14.py \
   --write_separate_step_files
   ```

   (a) The input files are:
   - `./inputs/eweld.in`
   - `./inputs/eweld_weld_parameters.in`
   - `./setting/Setting_arc_efficiency_dfault.in`
   - `./inputs/pass_coordinates.out`

   (b) The output files will be:
   - `Mesh_3D.unv`
   - `model_dflux.for`
   - `model_step1.inp`
   - `model_step2.inp`
   - `model_step3.inp`
   - `model_step4.inp`
   - `model_step5.inp`
   - `model_step6.inp`
   - `model_step7.inp`

   **Note:** The `model_step.inp` file has been broken into several files (`model_stepX.inp`) to resolve and issue with post-processing of cases where elements get added during the simulation.

4. Run `./utils/runCGX.sh` to generate the `nodesElems.inp` and `model_film.in` files (using cgx and unical):

   ```
   ./utils/runCGX.sh Mesh_3D.unv ./utils/write_film.fbd
   ```

5. Run

   ```
   python2 utils/Analysis_file_create.py
   ```

   - The input files are:
     * `./inputs/eweld.in`
     * `./inputs/eweld_boundary_condition.in`
     * `./inputs/eweld_preheat_interpass_temperature.in`
     * `./nodesElems.inp`
   - The output files will be:

* `model_bc.in`
* `model_ini_temperature.in`
* `model_material.in`

6. Move `model_dflux.for` to the CalculiX directory (to `CalculiX-PW/src/`) and rename to `dflux.f`, and compile CalculiX (see the notes in `CalculiX-PW/README.md` for compilation/installation of CalculiX). Or decompress `tools/CalculiX-PW.tar` (works under Ubuntu 14.04, if all the required packages for CalculiX are installed):

   ```
   tar -xf tools/CalculiX-PW.tar
   ```

   **The script `./utils/compileCcx.sh` performs the above automatically on Ubuntu 14.04 if all required packages for compiling CalculiX are installed:**

   ```
   ./utils/compileCcx.sh model_dflux.for tools/CalculiX-PW.tar ccx_2.12_MT
   ```

7. Run `analysis.inp` with CalculiX: To get around an issue with post-processing of results in cases with element addition/removal during simulation, we have divided the simulation into several steps. In each step, the number of elements remain constant. To run the whole simulation, restart files written at the end of one step are used as a start point for the next step.
   The bash script `./tests/runCCX_manual.sh` runs the multiple simulation steps consecutively. When running `./utils/runCCX_manual.sh`, the number of processors can be specified (the default number of processors is 1):

   ```
   ./tests/runCCX_manual.sh  4
   ```

   The output exo, sta and cvg files with be compressed in `ccx-results.tar` after the simulations are complete.

8. Post processing: To post process the simulation results using Paraview, the Metrics Extraction Python library can be used. The properties of image and metrics can be specified via a json file. An example json file is in `file:///setting/mex/welding_anim.json`. For details of the json syntax please see
   `https://github.com/parallelworks/MetricExtraction`.
   To generate images and animations and extract statistics, follow these steps:
   (a) Set the environment variable `PARAVIEWPATH` to the path of Paraview on your system, i.e, to the directory of `pvpython`. For example if `pvpython` is in directory `/opt/paraview530/bin`, run:
   ```
   export PARAVIEWPATH=/opt/paraview530/bin
   ```
   (b) Archive the `model_stepX.inp` files:
   ```
   tar -cf model_step.tar  model_step?*.inp
   ```
   (c) Run `utils/mexdex/extract.sh`:

```
./utils/mexdex/extract.sh model_step.tar ccx-results.tar \
setting/mex/welding_anim.json results/ results/metrics.csv \
pass_coordinates.out
```

The images, animations and `metrics.csv` file (with extracted statistics) will be written under the directory `results`