Altair Flux™ 2020

How to Launch Flux via Command Line?

How to Launch Flux via Command Line?

What's New

Here is a quick look at the latest features available in Flux 2019.

The three command line options to run python file are renamed:

- `-runPy` replaces `-executeSpy`
- `-runPyInSilentMode` replaces `-executeBatchSpy`
- `-runPyInSilentModeAndExit` replaces `-executeBatchSpyAndExit`

Command-line interface

The following commands are valid for Flux 2019. The Flux executable can be found in the installation directory `FLUX_HOME`. By default, `FLUX_HOME` is:

- On Windows: `FLUX_HOME= C:\Program Files\Altair\2019\flux\Flux`
- On Linux: `FLUX_HOME=/root/2019/altair/flux/Flux`

The Flux executable is located at `FLUX_EXE`:

- On Windows: `FLUX_EXE=%FLUX_HOME%\Bin\prg\win64\flux.exe`
- On Linux: `FLUX_EXE=$FLUX_HOME/Bin/prg/lin64/flux`

You can find below the command line to launch Flux depending on the application 2D, Skew, 3D or PEEC:

- On Windows: `%FLUX_EXE% -application <app> [options]`
- On Linux: `$FLUX_EXE -application <app> [options]`

with `<app> = Flux2D`, `FluxSkewed`, `Flux3D` or `FluxPEEC`

The `options` are listed below:

[options]	Function
`-batch`	to launch Flux in batch mode. The batch mode enables Flux to be launched without user interface.
`-newproject`	to create a new project with the `<app>` application.
`-file <fluxProject>`	to open an existing project located at `<fluxProject>`. `fluxProject` is the path to

[options]	Function
	the Flux project (`C:\work\project1.FLU`, for example).
`-runPy <pyFile>`	to launch Flux and run a python script located at `<pyFile>`.
`-runPyInSilentMode <pyFile>`	to run a python script located at `<pyFile>` with the user interface frozen and being refreshed only at the end of the execution of the python script.
`-runPyInSilentModeAndExit <pyFile>`	to run a python script located at `<pyFile>` with no graphical refresh, and then exit. Using the command `runPyInSilentMode` may not terminate the Flux process when encountering an error.
`-help`	to display the help.

Note:

- `<pyFile>` is the path to the python script to run (Example: `C:\python\project1.py`).
- the application `<app>` must be the same as the python script to run or the Flux project to open or create.

To be able to launch Flux, the next environment variables must be defined.

Environment variables

To set an environment variable:

- On Windows, use the `set` command, as below:

  ```
  set MY_VAR=2
  ```

- On Linux, in the case of common interpreter **sh**, **ksh** and **bash**, use the `export` command (see example below):

  ```
  export MY_VAR=2
  ```

1. Mandatory environment variables

The mandatory environment variables are listed below:

- Number of cores to use: *FLUX_NCORES*.

  Use *FLUX_NCORES=<NCORES>* to use *NCORES* cores.

  If *NCORES=0*, Flux will use all the cores available.

- Numerical memory (in Bytes): *MEMSIZN3*.

  The numerical memory is the memory used for the different modeling actions (import, mesh, solving process, ...).

> **Warning:** For PEEC, this is not this memory which is used for the solving process, but *JVM_MEMORY* (see below "GUI memory + PEEC Solver").

Default value for

- 2D and PEEC: *MEMSIZN3=2147483648*
- Skew and 3D: *MEMSIZN3=4294967296*

- Character memory (in Bytes): *MEMSIZC3*.

  The character memory is the memory used for the storage of entity names (parameters, transformations, regions, conductors, …) and the project names in the working directory.

  Default value for 2D, Skew, 3D and PEEC: *MEMSIZC3=134217728*

- GUI memory + PEEC Solver (in MiB): *JVM_MEMORY*.

  This memory is the memory used for everything related to the Graphical User Interface (graphical display, …).

  > **Warning:** For PEEC, this memory is also used for the solving process.

  Default value for:

  - 2D: *JVM_MEMORY=1024*
  - Skew: *JVM_MEMORY=1536*
  - 3D: *JVM_MEMORY=2048*
  - PEEC: *JVM_MEMORY=4096*

2. Additional environment variables

You can find below a list of optional variables:

- Allow multiple instances of Flux on the same machine:

  Set the variable *SYNC_SOCKET=0*.

- Force linear solver for pre and main resolution:

  FORCE_FLUX_SOLVER and *FORCE_FLUX_PRESOLVER*, respectively.

  These 2 variables allow to force a specific solver (Mumps, ICCG, SuperLu…). When solving the project, the solver used is the one defined in the environment variable.

  For example:

  FORCE_FLUX_SOLVER = LinearSystemSolverMumps

  FORCE_FLUX_PRESOLVER = LinearSystemSolverIccg

  will set ICCG for pre-solving and Mumps sequential for main solving.

  The possible choices are:

  - Direct solvers:
    - *LinearSystemSolverMumps*: Mumps

        - *LinearSystemSolverMumpsDistributed*: Mumps distributed
        - *LinearSystemSolverSuperlu*: Super LU
    - Iterative solvers:
        - *LinearSystemSolverIccg*: Incomplete Cholesky Conjugate Gradient
        - *LinearSystemSolverBiCGStab*: BiConjugate Gradient Stabilized
        - *LinearSystemSolverGmres*: Generalized Minimum RESidual
    - Others:
        - *LinearSystemSolverByUserSubroutine*
        - *LinearSystemSolverAutomatic*

Short example

Assuming all the environment variables set, the command line to open a Flux 3D project is:

- On Windows:

  If Flux is installed in `C:\Program Files\Altair\2019\flux\Flux`

  And if the Flux 3D project named Project_3D i.e. `Project_3D.FLU` is located in the directory `C:\work`

  ```
  "C:\Program Files\Altair\2019\flux\Flux\Bin\prg\win64\flux.exe" -application Flux3D -file C:\work\Project_3D.FLU
  ```

- On Linux:

  If Flux is installed in `/root/2019/altair/flux/Flux`

  And if the Flux 3D project named Project_3D i.e. `Project_3D.FLU` is located in the directory `/home`

  ```
  /root/2019/altair/flux/Flux/Bin/prg/lin64/flux -application Flux3D -file /home/Project_3D.FLU
  ```