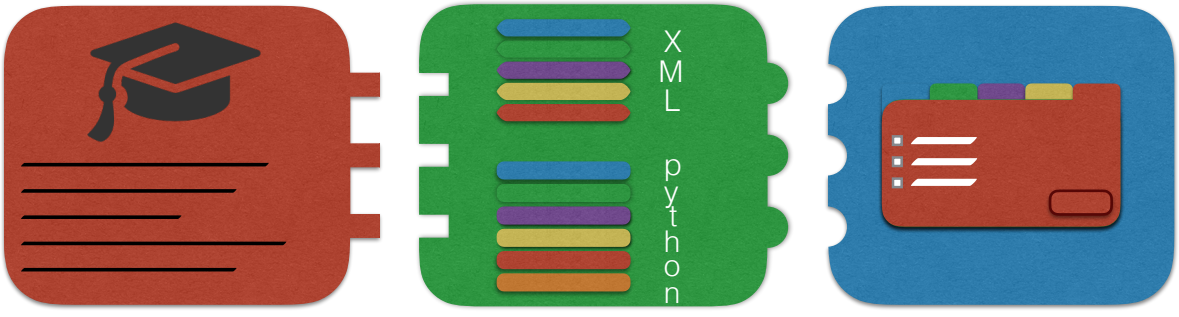
Prototype d'une application OpenTEA - Exercice pratique

Démonstration du prototypage d'une application avec le moteur OpenTEA

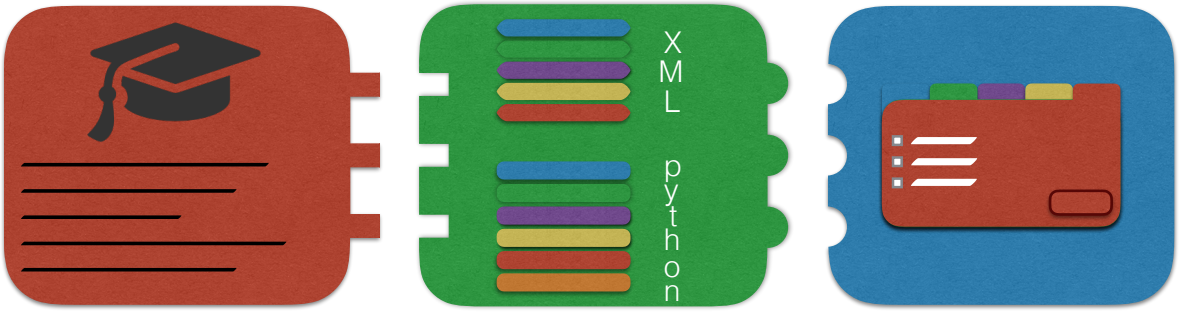
Une des forces du moteur OpenTEA est la possibilité de réaliser un prototype concret d'application en moins d'une demi-heure.



OpenTEA

Démonstration du prototypage d'une application avec le moteur OpenTEA

- Définition de la maquette
- Construction de l'interface graphique
- Specification des actions associées aux onglets



Réalisation d'une interface de code CFD de type recherche.

Exemple autour d'un solveur très simple codé en python basé sur la LBM en 2D.

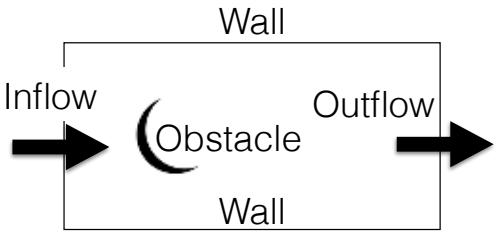
Inputfile

```
{
    "mesh": "mesh.dat",
    "characteristic_dimension": 0.0022,
    "scale": 2,
    "lu_x": 0.0001,
    "reynolds": 105,
    "viscosity": 1e-06,
    "bnd_bottom": "wall_slip",
    "bnd_right": "outlet",
    "bnd_up": "wall_slip",
    "bnd_left": "inlet_neq"
    "tau": 0.6,
    "max_iter": 10000,
    "postproc_dump_niter": 50,
    "postproc_info_niter": 10,
    "postproc_vel_ux": true,
    "postproc_density": true,
    "postproc_vel_mag": true,
}
```

Architecture similaire aux codes de recherche AVBP / YALES2 avec un fichier de maillage et un fichier de configuration en texte formaté.

Nombre de paramètres:
- Exemple présenté ~10
- AVBP / YALES2 ~1000

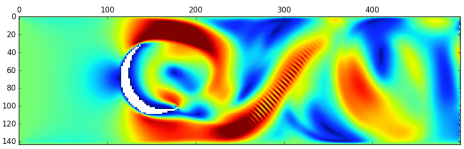
1. Définition de la maquette



Reynolds = 100

Inputfile + Maillage

```
~/lbm> python lbm_solver.py
```



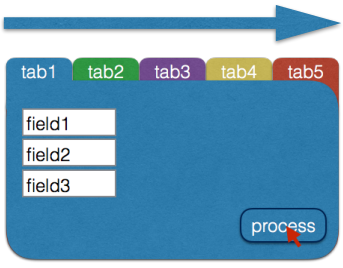
Exemple autour de la simulation d'un canal 2D avec un obstacle.

Les informations de la simulation sont spécifiées dans un fichier de maillage et un fichier de configuration.

Execution du calcul via le script python.

Visualisation du résultat.

1. Définition de la maquette



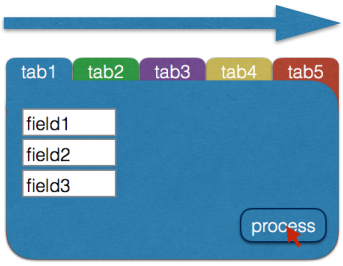
Exercice: Comment découper et ordonner la mise en donnée d'une simulation CFD ?

Contraintes de dépendances.
Ordre du processus métier.

?

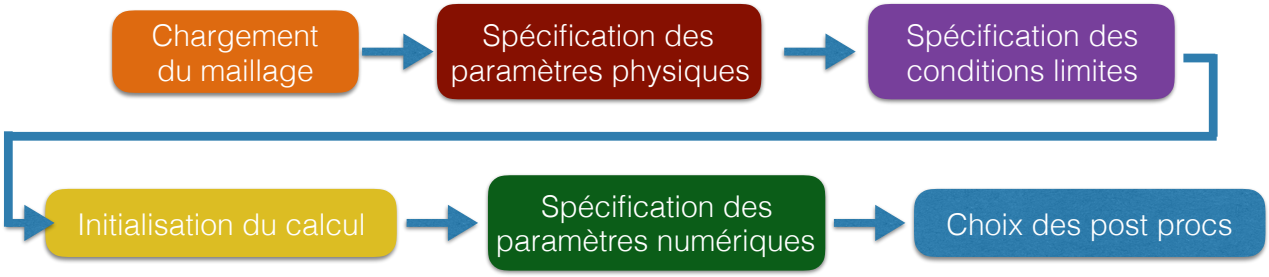
- Spécification des paramètres numériques
- Choix des post procs
- Spécification des conditions limites
- Chargement du maillage
- Initialisation du calcul
- Spécification des paramètres physiques

1. Définition de la maquette

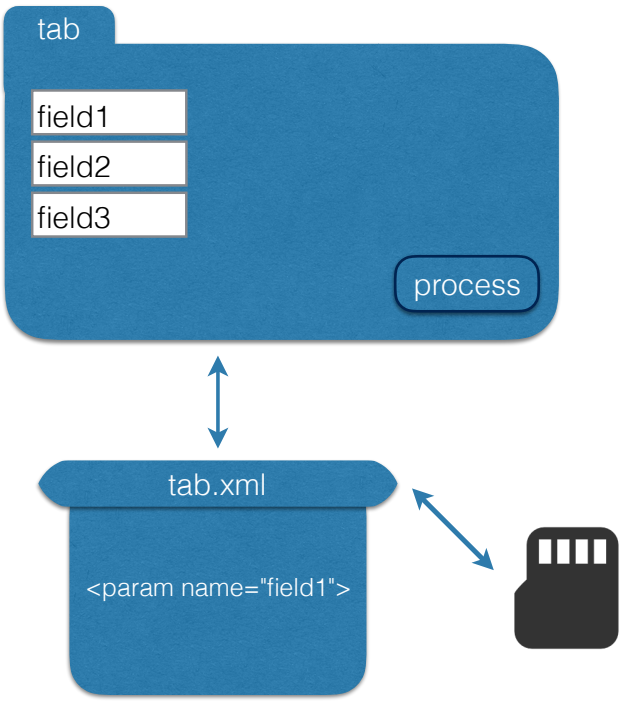


Exercice: Comment découper et ordonner la mise en donnée d'une simulation CFD ?

Exemple de réponse:



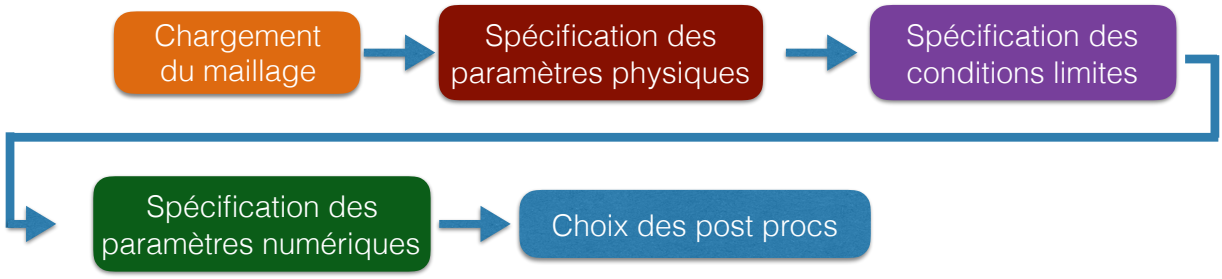
2. Construction de l'interface graphique



L'interface graphique est construite via des fichiers XML interprétables par le moteur OpenTEA.

Un fichier XML par onglet.

2. Construction de l'interface graphique




```
{
    "mesh": "mesh.dat",
    "characteristic_dimension": 0.0022,
    "scale": 2,
    "lu_x": 0.0001,
    "reynolds": 105,
    "viscosity": 1e-06,
    "bnd_bottom": "wall_slip",
    "bnd_right": "outlet",
    "bnd_up": "wall_slip",
    "bnd_left": "inlet_neq"
    "tau": 0.6,
    "max_iter": 10000,
    "postproc_dump_niter": 50,
    "postproc_info_niter": 10,
    "postproc_vel_ux": true,
    "postproc_density": true,
    "postproc_vel_mag": true,
}
```

Maillage / Géométrie

Paramètres physiques

Conditions limites

Paramètres numériques

Gestion des sorties

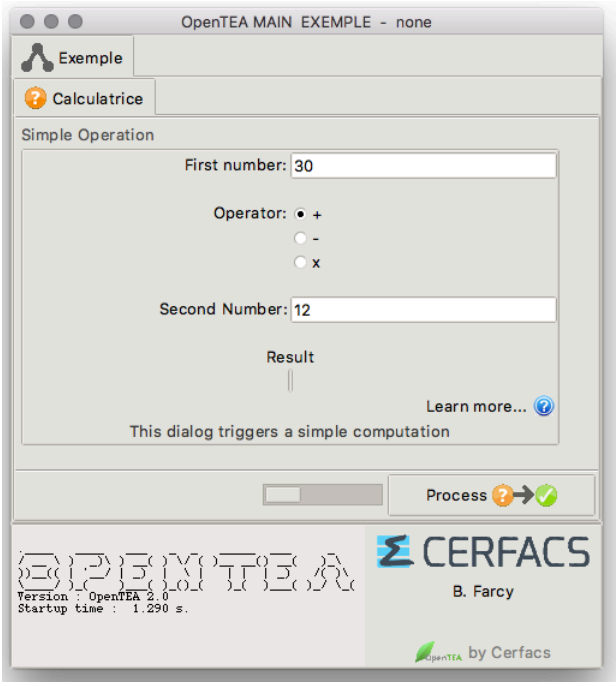
Exercice: Ecriture des fichiers XML pour un découpage en cinq onglets permettant de specifier les paramètres du fichier d'entrée du code CFD.

2. Construction de l'interface graphique

```
farcy@laptop:~> openTea -code exemple
farcy@laptop:~> cd lbm
farcy@laptop:~/lbm> ll
-rw-r--r--  1 farcy  3000   8112 19 jan 16:29 mesh.dat
-rwxr-xr-x  1 farcy  3000  26670 19 jan 16:29 lbm_solver.py
-rw-r--r--  1 farcy  3000    451 19 jan 16:29 inputfile
-rw-r--r--  1 farcy  3000   7550 19 jan 16:29 cylinder_wall.dat
-rw-r--r--  1 farcy  3000   7248 19 jan 16:29 cylinder_nowall.dat
-rw-------  1 farcy  3000   4401 19 jan 16:29 velocity_mag.html
-rw-r--r--  1 farcy  3000   7550 19 jan 16:29 triangle.dat
drwxr-xr-x  2 farcy  3000     68 19 jan 16:30 output
farcy@laptop:~/lbm> mkdir openTea_app
farcy@laptop:~/lbm/openTea_app> cp -r opentea/library/exemple ./
farcy@laptop:~/lbm/openTea_app> mv exemple lbm_2D
farcy@laptop:~/lbm/openTea_app> cd lbm_2D/
farcy@laptop:~/lbm/openTea_app/lbm_2D > ll
-rw-r--r--  1 farcy  3000   43 19 jan 12:12 folderParameters.xml
drwxr-xr-x  4 farcy  3000  136 19 jan 12:12 calculatrice
-rw-r--r--  1 farcy  3000  775 19 jan 12:12 N3S100.gif
```

Une interface = un dossier
Si possible inclus dans le code source du solveur.

Construction à partir de la copie de l'exemple le plus simple: une calculette



2. Construction de l'interface graphique

```
LBM_2D/XML/lbm_2D> mv calculatrice mesh
LBM_2D/XML/lbm_2D> cd mesh/
LBM_2D/XML/lbm_2D/mesh> ls
calc.xml              folderParameters.xml
LBM_2D/XML/lbm_2D/mesh> mv calc.xml mesh.xml
LBM_2D/XML/lbm_2D/mesh> vi folderParameters.xml
LBM_2D/XML/lbm_2D/mesh cat folderParameters.xml
<tab title="Mesh" order="1"/>>
```

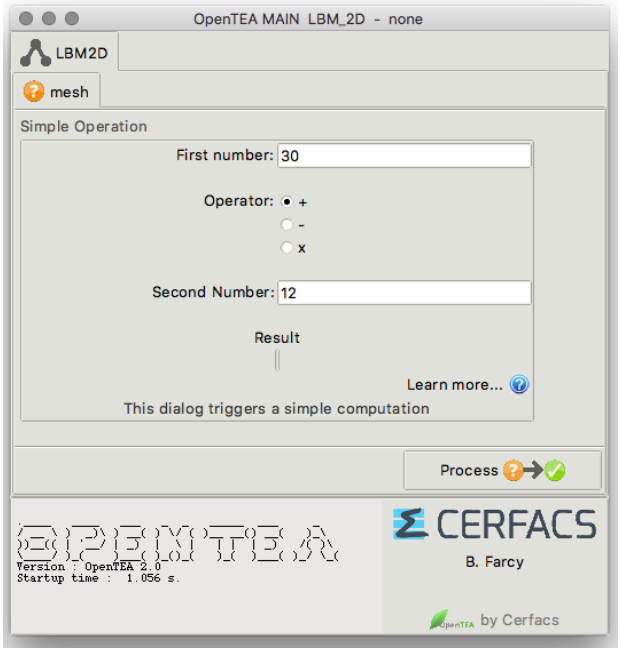
1. Changement des noms de dossier / fichiers
2. Mise à jour du fichier folderParameter.xml spécifiant le titre et l'ordre des onglets

OU utilisation d'un script utilitaire.

```
farcy@laptop:~/opentea/library> sh script.sh
Create a new app
Enter the name of the app and press ENTER
test
```

Lien de l'application créée dans le dossier library du moteur OpenTea

```
farcy@laptop:~/opentea/library> ln -s ~/lbm/openTea_app/LBM_2D ./
farcy@laptop:~> openTea -code LBM_2D
```



2. Construction de l'interface graphique

Répétition pour tous les onglets

```
{
    "mesh": "mesh.dat",
    "characteristic_dimension": 0.0022,
    "scale": 2,
    "lu_x": 0.0001,
    "reynolds": 105,
    "viscosity": 1e-06,
    "bnd_bottom": "wall_slip",
    "bnd_right": "outlet",
    "bnd_up": "wall_slip",
    "bnd_left": "inlet_neq"
    "tau": 0.6,
    "max_iter": 10000,
    "postproc_dump_niter": 50,
    "postproc_info_niter": 10,
    "postproc_vel_ux": true,
    "postproc_density": true,
    "postproc_vel_mag": true,
}
```

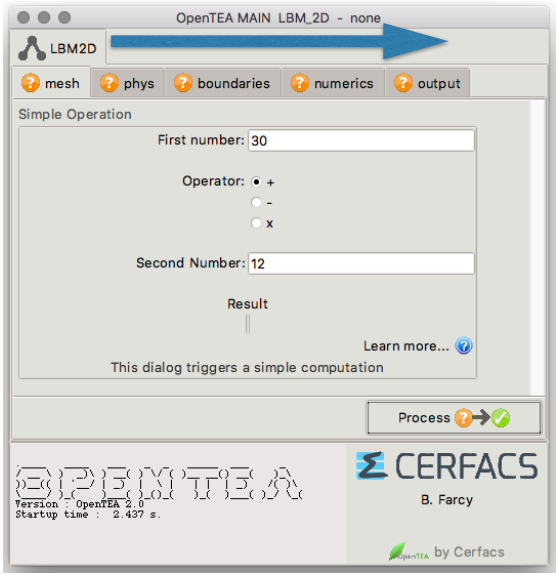
Maillage / Géométrie

Paramètres physiques

Conditions limites

Paramètres numériques

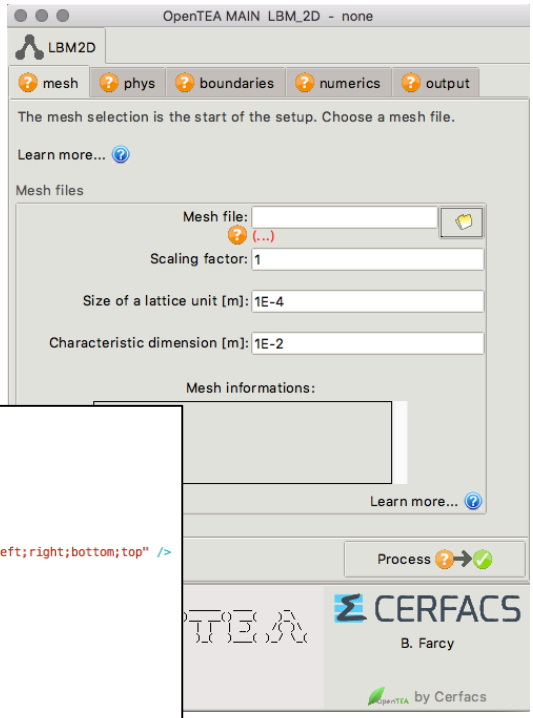
Gestion des sorties



2. Construction de l'interface graphique

Onglet mesh

1. Entrée d'informations
   1. Choix du fichier de maillage (string)
   2. Paramètre de scaling (int)
   3. Taille physique de lattice
2. Liste pour stocker les conditions limites.
3. Retour d'informations prévu via une zone *comment* grisée.

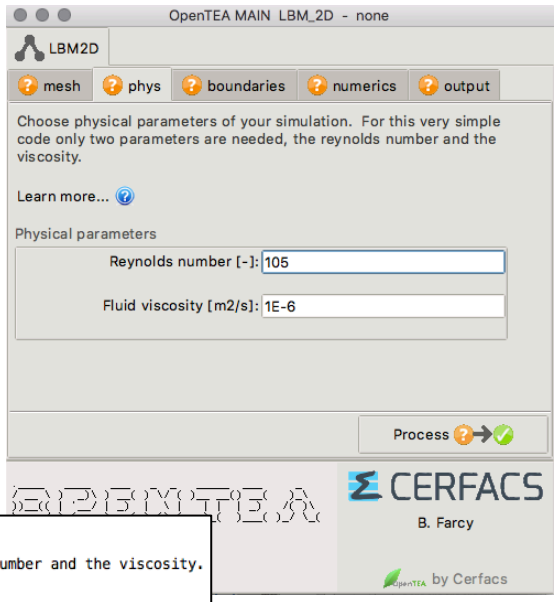



```
<desc>
        The mesh selection is the start of the setup.
        Choose a mesh file.
</desc>
<docu>
        The mesh is a text file filled with 0 and 1.
        Each number is a lattice unit, 0 for fluid and 1 for solid.
</docu>

<info name="listofpatches" title="List of patches" type="liststring" visibility="hidden" default="left;right;bottom;top" />

<model name="mesh_params" title="Mesh files" >
    <param name="meshfile" title="Mesh file" type="file" default="" />
    <param name="scaling" title="Scaling factor" type="integer_ge1" default="1" />
    <param name="lu_x" title="Size of a lattice unit [m]" type="double_g0" default="1E-4" />
    <param name="char" title="Characteristic dimension [m]" type="double_g0" default="2E-3" />

    <comment name="info_mesh" title="Mesh informations" default="" state="disabled" />
    <docu>
        The mesh can be scaled to increase the number of points.
        Warning, the size of the lattice unit is used after the scaling.
        Give a characteristic dimension of the mesh that will be used after to compute the Reynolds number.
    </docu>
</model>
```


2. Construction de l'interface graphique

Onglet phys

1. Entrée d'informations
   1. Entrée du nombre de Reynolds
   2. Entrée de la viscosité du fluide




```
<desc>
      Choose physical parameters of your simulation.
      For this very simple code only two parameters are needed, the reynolds number and the viscosity.
</desc>
<docu>
      Reynolds number
</docu>

<model name="phys_params" title="Physical parameters" >
    <param name="reynolds" title="Reynolds number [-]" type="double_g0" default="105" />
    <param name="viscosity" title="Fluid viscosity [m2/s]" type="double_g0" default="1E-6" />
</model>
```

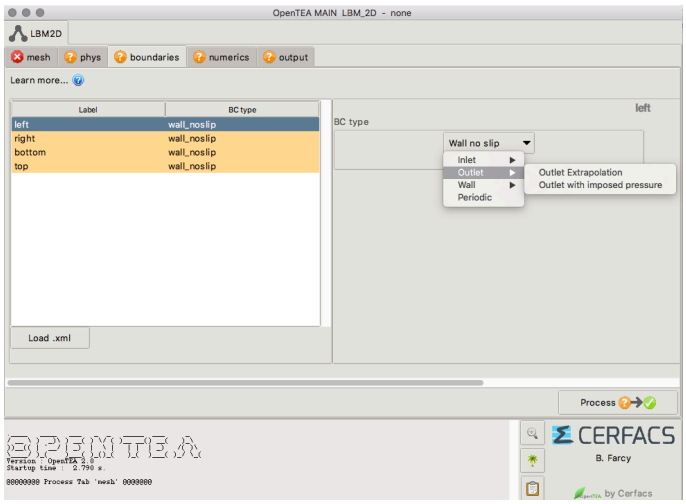
2. Construction de l'interface graphique

Onglet boundaries

1. Pour ce code, il n'existe que 4 conditions limites associés: Left/Up/Right/Bottom. Elles ont été rentrées en dur dans le mesh.xml. Pour un code classique, cette liste est lue à partir du fichier de maillage.
2. Un *multiple* est utilisé pour lister les conditions limites et un *xor* pour choisir le type de CL.

```
<info name="listofpatches" title="List of patches" type="liststring" visibility="hidden" default="left;right;bottom;top" />
```

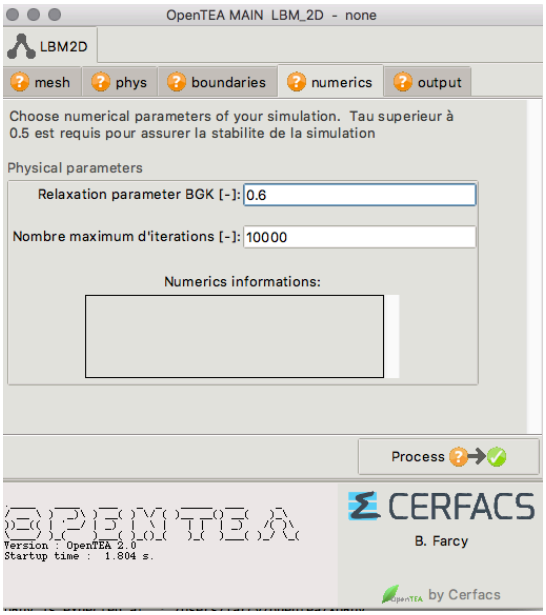
```
<docu>
    Conditions limites.
    Les conditions d'entrées ne peuvent être imposées que sur la face de gauche.
    Les conditions de sortie ne peuvent être imposées que sur la face de droite.
</docu>

<multiple name="bc" title="Conditions limites" require="listofpatches"  size="1.0;1.5" subcolumns="2">
    <xor name="bnd_type" title="BC type" default="wall_noslip" groups="on">
        <model name="inlet_bounceback" title="Inlet Bounce Back">
            <docu>
                Non equilibrium Bounce back method.
            </docu>
        </model>
        <model name="inlet_equilibrium" title="Inlet Equilibrium">
            <docu>
                Equilibrium mehod.
            </docu>
        </model>
        <model name="inlet_nonequilibrium" title="Inlet Non Equilibrium Bounce Back">
            <docu>
                Non equilibrium extrapolation method.
            </docu>
        </model>
        <model name="outlet_extrapolation" title="Outlet Extrapolation">
            <docu>
                The unknown quantities at the outlet are extrapolated from the inside nodes.
            </docu>
        </model>
        <model name="outlet_pressure" title="Outlet with imposed pressure">
            <docu>
                The unknown quantities at the outlet are computed to impose a pressure.
            </docu>
        </model>
        <model name="periodic" title="Periodic">
        </model>
        <model name="wall_noslip" title="Wall no slip">
        </model>
        <model name="wall_slip" title="Wall slip">
        </model>
    </xor>
</multiple>
```



2. Construction de l'interface graphique

Onglet numerics

1. Entrée d'informations
    1. Paramètre numérique de relaxation
    2. Nombre d'itérations max
2. Retour d'informations via une zone grisée

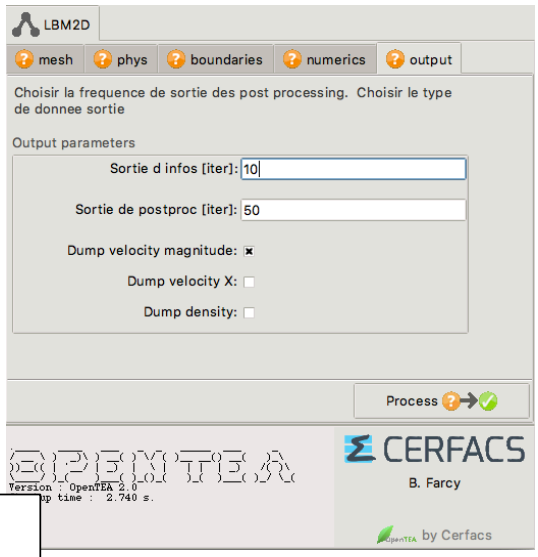



```
<desc>
      Choose numerical parameters of your simulation.
      Tau superieur à 0.5 est requis pour assurer la stabilite de la simulation
</desc>

<model name="num_params" title="Physical parameters" >
    <param name="tau" title="Relaxation parameter BGK [-]" type="double_g0" default="0.6" />
    <param name="max_iter" title="Nombre maximum d'iterations [-]" type="integer_g0" default="10000" />

    <comment name="info_num" title="Numerics informations" default="" state="disabled" />
</model>
```

2. Construction de l'interface graphique

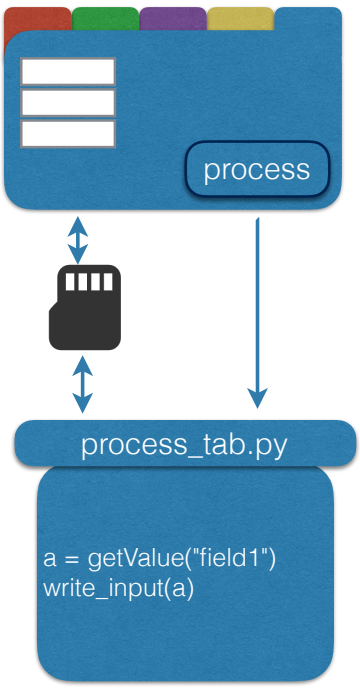
Onglet output

1. Entrée d'informations
    1. Fréquence de sortie d'infos
    2. Fréquence de sorite d'images
    3. Choix de type de sortie




```
<desc>
    Choisir la frequence de sortie des post processing.
    Choisir le type de donnee sortie
</desc>

<model name="output_params" title="Output parameters" >
    <param name="info_iter" title="Sortie d infos [iter]" type="integer_g0" default="10" />
    <param name="dump_iter" title="Sortie de postproc [iter]" type="integer_g0" default="50" />

    <param name="dump_u" title="Dump velocity magnitude" type="onoff" default="1"/>
    <param name="dump_ux" title="Dump velocity X" type="onoff" default="0"/>
    <param name="dump_rho" title="Dump density" type="onoff" default="0"/>
</model>
```


3. Specification des actions



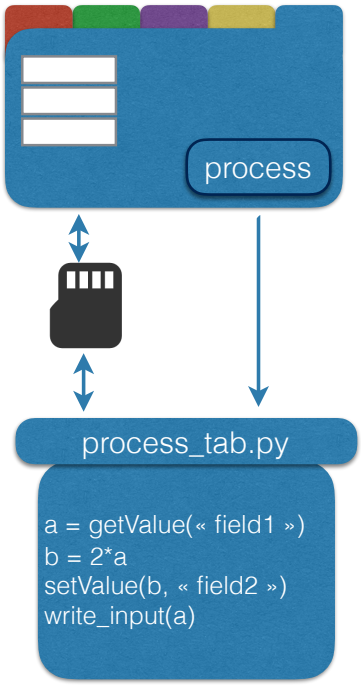
L'interface graphique est terminée. Toutes les données remplies par l'utilisateur lors de la mise en donné sont stockées dans un fichier XML

Il est possible d'associer un script python à chaque bouton process de chaque onglet.

Ce script python peut accéder à la mémoire et la modifier.

Le script python peut également executer des outils, copier des fichiers, écrire des fichiers, lancer le code sur un calculateur..

3. Specification des actions



Pour l'exemple, les actions suivantes vont êtres ajoutée à l'interface.

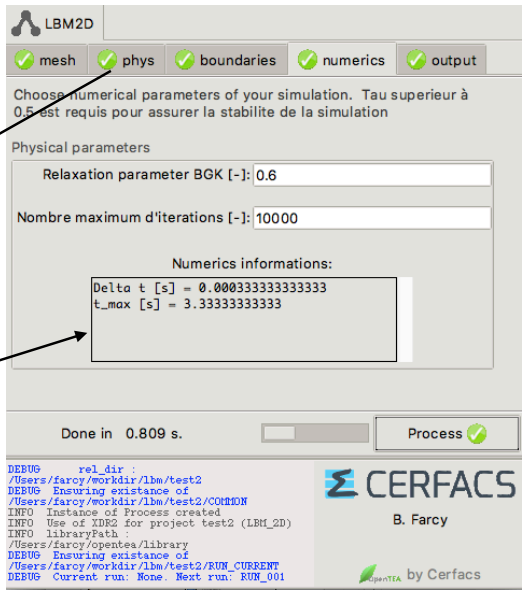
- Onglet maillage:
  - Copie du fichier de maillage
  - Retour sur la taille du maillage
- Onglet numerics
  - Retour d'informations sur les paramètres choisis
- Onglet output:
  - Ecriture du fichier d'entrée du code CFD

3. Specification des actions

Onglet numerics

Acquisition d'informations de la mémoire de différents onglets
Calcul interne
Retour visuel d'informations

```
""" Module to process the numerical parameters"""

from XDR2 import CodeProcess

def process_num(process):
    """ Process the tab """

    dataset = process.ds
    process.log.info("0% - Debut du traitement ")

    niter = float(dataset.getValue("max_iter"))
    tau = float(dataset.getValue("tau"))
    lu_x = float(dataset.getValue("lu_x"))
    viscosity = float(dataset.getValue("viscosity"))

    delta_t = 1. / 3. * (tau - 1. / 2.) * lu_x * lu_x / viscosity
    t_max = niter * delta_t

    info_num = "Delta t [s] = " + str(delta_t) + "\n"
    info_num += "t_max [s] = " + str(t_max) + "\n"

    dataset.setValue(info_num, "info_num")

    process.log.info("100% - Fin du traitement ")

if __name__ == '__main__':
    process = CodeProcess("dataset.xml")
    process_num(process)
    process.finish()
```



3. Specification des actions

Onglet mesh

```
""" Module to process the mesh"""

from XDR2 import CodeProcess

def process_mesh(process):
    """ Process the tab """

    dataset = process.ds
    process.log.info("0% - Debut du traitement du maillage ")


    process.log.info("100% - Fin du traitement du maillage ")

if __name__ == '__main__':
    process = CodeProcess("dataset.xml")
    process_mesh(process)
    process.finish()
```

Le script basique se constitue

- D'un programme principal où l'objet process de type CodeProcess est créé à partir de l'arbre de données XML.
- D'une fonction où des données sont traitées

- Onglet maillage:
  - Copie du fichier de maillage
  - Retour sur la taille du maillage

```
""" Module to process the mesh"""

import numpy
from XDR2 import CodeProcess

def process_mesh(process):
    """ Process the tab """

    dataset = process.ds
    process.log.info("0% - Debut du traitement du maillage ")

    mesh_filename = dataset.getValue("meshfile")
    scale = float(dataset.getValue("scaling"))
    lu_x = float(dataset.getValue("lu_x"))

    process.copy_file(mesh_filename, process.COMMON)

    mesh_file = open(mesh_filename, 'r')
    mesh_lines = mesh_file.readlines()
    n_line = 0
    for line in mesh_lines:
        n_line = n_line + 1
    n_char = 0
    for point in mesh_lines[0]:
        n_char = n_char + 1
    n_char = n_char - 1 # Remove newline
    mesh_file.close()

    np_x = n_char * scale
    np_y = n_line * scale
    np = np_x*np_y
    l_x = np_x*lu_x
    l_y = np_y*lu_x

    info_mesh = "Np_x = " + str(int(np_x)) + "\n"
    info_mesh += "Np_y = " + str(int(np_y)) + "\n"
    info_mesh += "Np = " + str(int(np)) + "\n"
    info_mesh += "L_x = " + str(l_x) + "\n"
    info_mesh += "L_y = " + str(l_y)

    dataset.setValue(info_mesh, "info_mesh")

    process.log.info("100% - Fin du traitement du maillage ")

if __name__ == '__main__':
    process = CodeProcess("dataset.xml")
    process_mesh(process)
    process.finish()
```

Acquisition des données via les noms des paramètres XML

Copie du fichier de maillage

Retour d'information vers l'interface

3. Specification des actions

Onglet output

Ecriture du fichier de configuration au format du solveur CFD.

```
""" Module to write the inputfile"""

import json
from XDR2 import CodeProcess

def process_output(process):
    """ Process the tab write the inputfile """

    dataset = process.ds
    process.log.info("0% - Debut du traitement ")

    inputfile = open("inputfile", 'w')
    dico = dict(max_iter = int(dataset.getValue("max_iter")),
                mesh = dataset.getValue("meshfile"),
                reynolds = float(dataset.getValue("reynolds")),
                lu_x = float(dataset.getValue("lu_x")),
                scale = int(dataset.getValue("scaling")),
                characteristic_dimension = float(dataset.getValue("char")),
                viscosity = float(dataset.getValue("viscosity")),
                bnd_bottom = "wall_slip",
                bnd_up = "wall_slip",
                bnd_left = "inlet_neq",
                bnd_right = "outlet",
                tau = float(dataset.getValue("tau")),
                postproc_dump_niter = int(dataset.getValue("dump_iter")),
                postproc_info_niter = int(dataset.getValue("info_iter")),
                postproc_vel_mag = bool(dataset.getValue("dump_u")),
                postproc_density = bool(dataset.getValue("dump_rho")),
                postproc_vel_ux = bool(dataset.getValue("dump_u")))
    json.dump(dico, inputfile)
    inputfile.close()

    process.log.info("100% - Fin du traitement ")

if __name__ == '__main__':
    process = CodeProcess("dataset.xml")
    process_output(process)
    process.finish()
```