POLYTECHNIQUE
MONTRÉAL

Enrichissement des fichiers APEX avec Version5

A. Hébert

2022/07/04

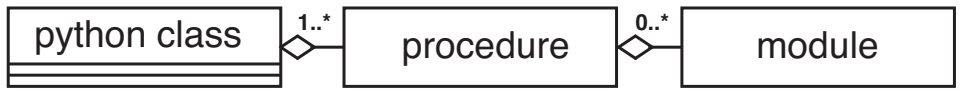
POLYTECHNIQUE MONTRÉAL

Table des matières

Introduction
Le code de réseau Dragon5
Le code de simulation coeur-entier Donjon5
Équivalence SPH
Utilisation de Version5
Le module d'extension lcm
Le module d'extension lifo
Le module d'extension cle2000
Enrichissement des APEX
Conclusions
Ressources

Introduction

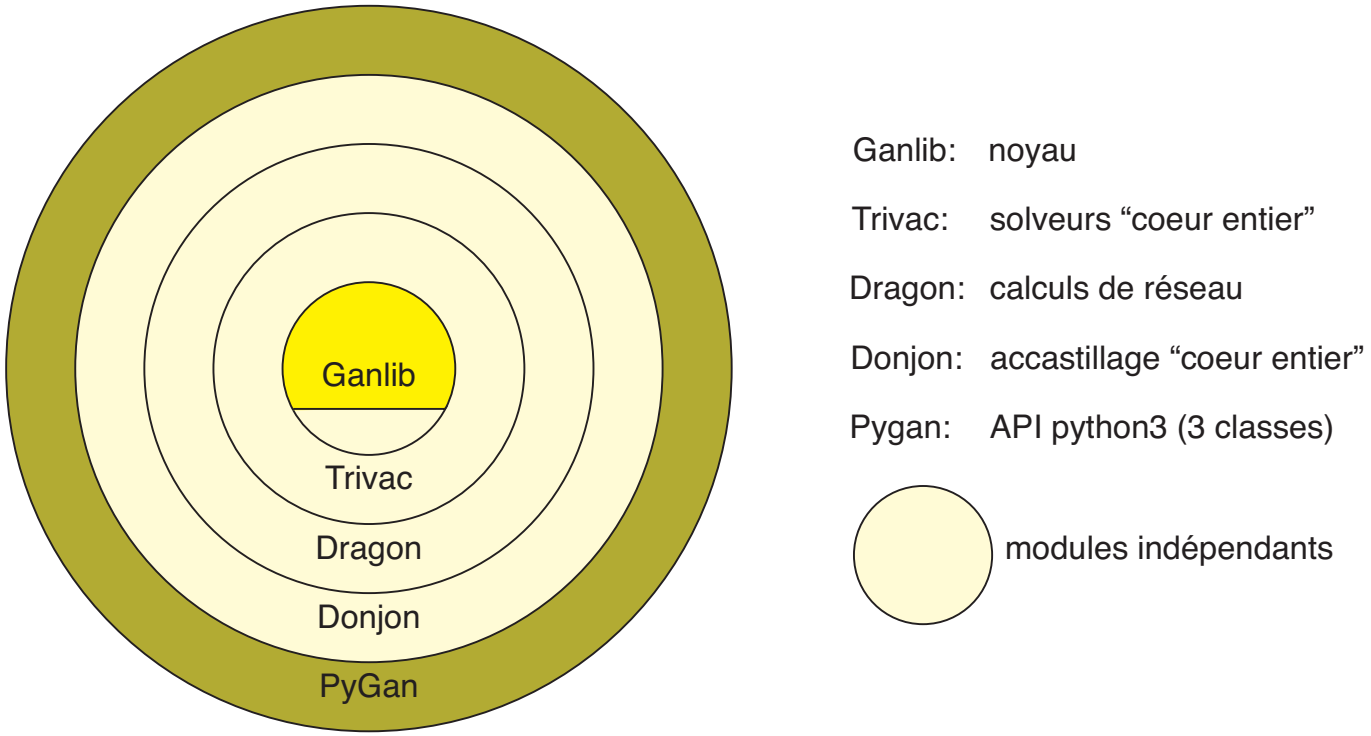
- Version5 est un système de codes, développés par Polytechnique Montréal, en physique des réacteurs nucléaires:
  - Dragon5 est un code de réseau construit autour de solveurs de l'équation de Boltzmann
  - Donjon5 est un code de simulation du réacteur entier.
- Version5 est un projet Open-Source distribué sous license LGPL.
- Version5 permet le prototypage rapide de méthodes et de schémas de calcul:
  - test et développement de nouvelles méthodes avant inclusion dans les codes de production
  - outil de recherche R&D
- Version5 est un code modulaire construit selon un modèle fonctionnel:
  - Version5 contient ≃ 90 modules de calcul indépendants (sans effet de bord)
  - Le superviseur CLE-2000 permet de programmer le chainage entre les modules pour produire des procédures
  - Un API python3 permet d'encapsuler une ou plusieurs procédures CLE-2000 dans une classe python.



langage:	python3	CLE-2000	Fortran 2003

Introduction

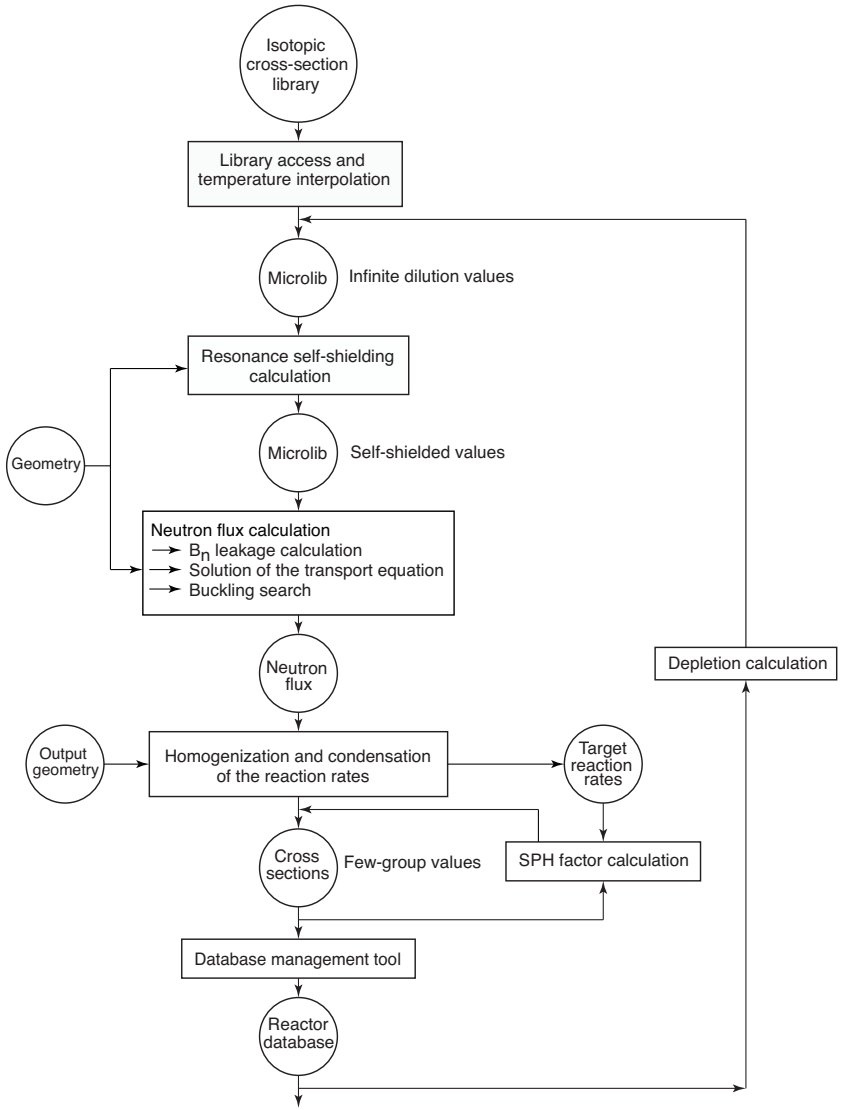
- Le noyau Ganlib contient le superviseur CLE-2000, l'implémentation des objets LCM (semblables aux segments Ésope), les API HDF5 et certains modules utilitaires.
- Trivac, Dragon et Donjon contiennent des modules indépendants liés à la physique des réacteurs.
- PyGan contient les API python3 permettant la pythonnisation des procédures Version5.



Le code de réseau Dragon5

Dragon5 est un code de réseau de seconde génération avec les fonctionnalitées suivantes:

1. Accès aux bibliothèques de sections efficaces et interpolation en température.
2. Calcul d'autoptotection des résonances.
3. Calcul de flux principal
4. Homogénéisation et condensation des taux de réactions.
5. Calcul des facteurs d'équivalence SPH.
6. Calcul d'évolution isotopique (solution des équations de Bateman).
7. Création d'une base de données réacteur multi-paramètres en format Saphyb ou Multicompo



Le code de réseau Dragon5

Modules importants de Trivac5 et Dragon5

- Les modules Dragon5 sont en bleu et les modules Trivac5 sont en vert.
- Les modules Trivac5 peuvent être utilisés par Dragon5.
- Les modules Dragon5 peuvent être utilisés par Donjon5.
- Les codes Trivac5, Dragon5 et Donjon5 sont une collection de modules indépendants, sans connexions entre eux et sans effets de bord.

GEO: Entrée de la géométrie

SYBILT:, SNT:, NXT:, SALT:, etc. Tracking de la géométrie (numérotation, calcul des connexions et des trajectoires pour la méthode Multicell, SN, PIJ ou MOC

BIVACT:, TRIVAT: Numérotation de la géométrie pour une méthode d'éléments finis dans Trivac5

LIB: Fabrication de la microlib (sections efficaces microscopiques par isotope)

SHI:, USS:, TONE: Calcul d'autoprotection des résonances

ASM: Calcul des matrices de PIJ ou des quantités multigroupes non-itératives

BIVACA:, TRIVAA: Calcul des matrices d'éléments finis dans Trivac5

FLUD:, FLU: Calcul du flux

OUT:, EDI: Homogénéisation et condensation

SPH: Équivalence SPH et/ou enrichissement des base de données réacteur multi-paramètres

EVO: Solution des équations de Bateman (calcul de burnup)

COMPO: Création d'une multicompo

SAP: Création d'une Saphyb

Le code de simulation coeur-entier Donjon5

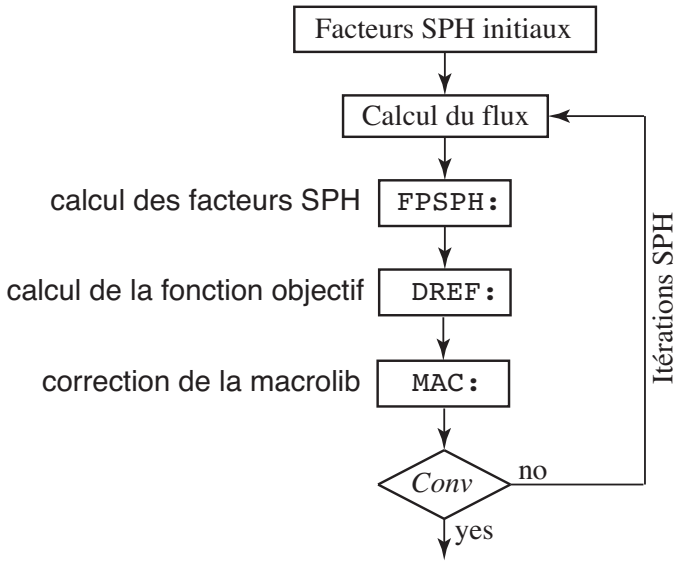
Modules importants de Donjon5

- L'équivalence SPH peut être effectuée dans l'environnement Donjon5 pour obtenir plus de flexibilité

  FPSPH: Itération SPH point fixe unique (aka. Picard) ou itération Newton unique.

  DREF: calcul de la fonction objectif du processus d'optimisation et définition (optionnelle) des sources fixes qui peuvent être utilisées dans le terme de droite d'un problème à source singulier adjoint (*adjoint fixed source eigenvalue problem*) pour la méthode de perturbation généralisée (GPT).

- L'itération SPH de point fixe est favorisée dans les cas de mode fondamental.

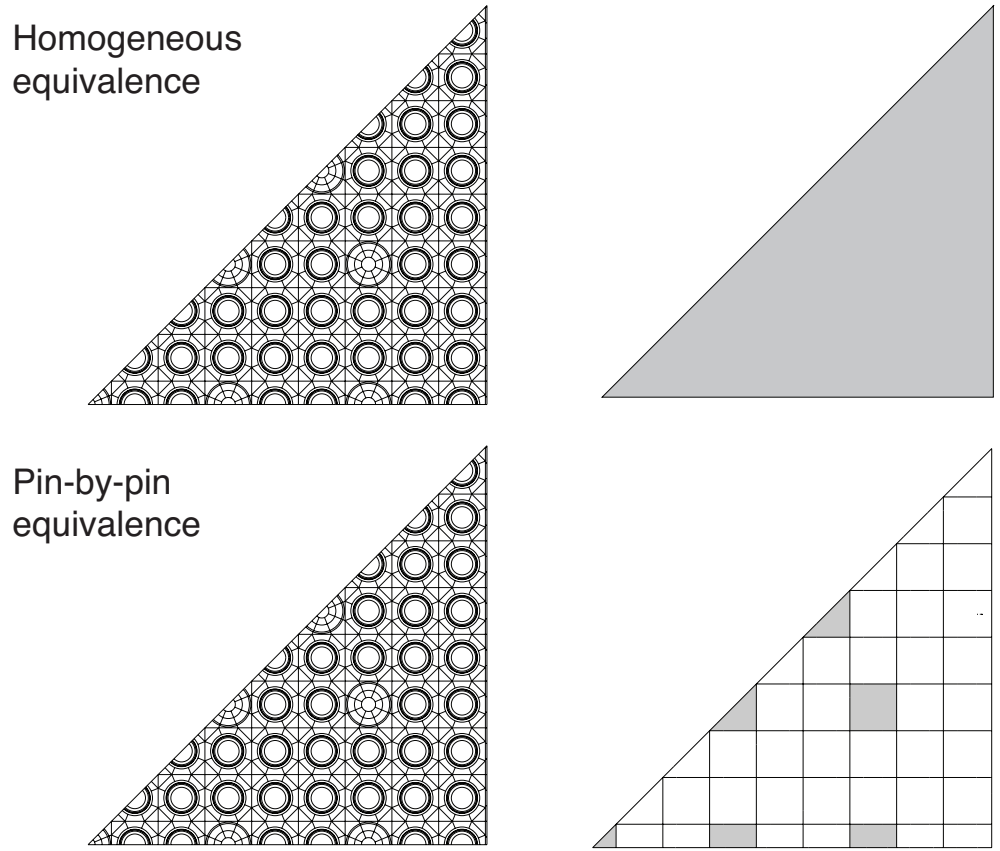


Équivalence SPH

- Les procédures d'équivalence sont des procédures indispensables en neutronique
  - Utilisées dans les schémas de calcul déterministes
  - Permettent de remplacer une équation de transport par une équation plus simple à résoudre
  - Permettent de simplifier la discrétisation en énergie, espace et direction (angle solide)
  - Il existe deux familles (doctrines) de procédures d'équivalence:
    - les facteurs de discontinuité (DF et ADF) – on ne modifie pas les sections efficaces
    - les facteurs de superhomogénéisation (SPH) – on ne modifie pas la programmation des opérateurs de calcul
  - Des procédures d'équivalence existent pour les deux situations suivantes:
    - en mode fondamental, lorsque le motif géométrique est une maille de réseau se répétant à l'infini
    - en mode non-fondamental, lorsque le motif géométrique est entouré de vide.
- Les facteurs DF ou ADF et les facteurs SPH sont intégrés dans la base de données réacteur multi-paramètres produite par le code de réseau
  - les sections efficaces de la base de données réacteur multi-paramètres ne sont pas modifiées.

Équivalence SPH

Équivalence d'un huitième d'assemblage 17x17 en mode fondamental



- à gauche:, assemblage décrit par une géométrie surfacique TDT native
- à droite, assemblage décrit par un maillage cartésien.

- Les applications de la méthode SPH en mode fondamental sont nombreuses.

Équivalence SPH

Définitions

géométrie de référence: Géométrie détaillée utilisée par le calcul de référence

calcul de référence: Calcul détaillé réalisé sur la géométrie de référence en utilisant des sections efficaces non-corrigées et une équation de transport précise.

macro-géométrie: Géométrie simplifiée utilisée par le macro-calcul

macro-calcul: Calcul simplifié réalisé sur la macro-géométrie en utilisant des sections efficaces corrigées SPH, des albédos corrigés et/ou des facteurs de discontinuité

calcul de vérification: Macro-calcul réalisé avec les facteurs d'équivalence convergés qui permet de s'assurer que la procédure d'équivalence a bien fonctionnée.

Utilisation de Version5

L'installation de la version python de Version5 nécessite :

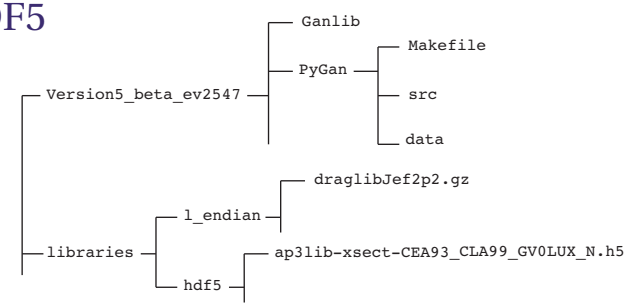
1. Ajout des lignes suivantes dans le fichier `.bashrc`:

```
# Support for HDF5
export HDF5_INC="/usr/include" # HDF5 include directory
export HDF5_API="$HDF5_INC/../lib64" # HDF5 C API
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:$HDF5_API"
# Support for Python3 API
export FORTRANPATH=/usr/lib/gcc/x86_64-redhat-linux/8/ # contains libgfortran.so
```

2. Installation de l'exécutable Version5:

```
tar xvfzp Version5.0.8_ev2547.tgz
cd Version5_beta_ev2547/PyGan
make hdf5=1
```

3. Installation des bibliothèques de sections efficaces (optionnel):
   - Téléchargez des Draglibs à partir de website ou récupérez une Apolib XSM ou HDF5
   - Copier les bibliothèques dans le répertoire `libraries/`



4. Exécution des tests de non régression:

```
cd Version5_beta_ev2547/PyGan
make tests hdf5=1
```

5. Exécution d'un seul jeu de données:

```
cd Version5_beta_ev2547/PyGan
./rpython Equivalence_SPH_APEX.py
```

Utilisation de Version5

- Version5 nécessite les prérequis suivant:
  - UNIX ou Linux OS
  - compilateur Fortran 2003
  - python3 et numpy
  - API C HDF5
  - LaTeX (optionnel)
- L'enchaînement des modules et la génération de schémas de calcul sont réalisés à l'aide du langage de supervision CLE-2000.
- L'encapsulation des composants métiers dans les classes python3 est possible à l'aide de l'API PyGan (basée sur l'utilitaire distutils).
  - PyBind, Swig ou Boost ne sont pas utilisés.
- Polytechnique Montréal ne fournit pas de schémas de calcul validés ni de composants métiers pour une utilisation en production.
- Les schémas de calcul sont entièrement écrits avec la syntaxe de script CLE-2000.
- Les schémas de calcul sont spécifiques à chaque type de réacteur ou d'application et contiennent la propriété intellectuelle (IP) des utilisateurs.
- Les schémas de calcul ne sont pas soumis à la licence LGPL de DRAGON.
  - Cette distinction est possible car Dragon5 et Donjon5 sont publiés sous la forme lesser de la licence et non sous la version GPL plus restrictive de celle-ci.
  - L'inconvénient de cette approche est que les utilisateurs doivent apprendre à construire leurs propres schémas de calcul pour utiliser le code.
  - Nécessite plus de savoir-faire que pour des codes concurrents tels que CASMO5.

Le module d'extension lcm

Le module d'extension `lcm` permet un accès in-out aux objets LCM de DRAGON5/DONJON5 et à leur contenu au moyen d'un mapping de l'opérateur " `[]` " en Python. Voici un tutoriel interactif:

```
cd Version5_beta_ev2547/PyGan
export PYTHONPATH=lib/Linux_x86_64/python/
export PYTHONMALLOC=debug
python3
>>> import lcm
>>> from assertS import *
>>> import numpy as np
>>> my_lcm=lcm.new('LCM_INP','nonfuel')
>>> my_lcm._impx=3
>>> my_lcm.lib()
>>> my_lcm.keys()
>>> sign=my_lcm['SIGNATURE']
>>> print('object signature=', sign)
>>> daughter=my_lcm['REFL']
>>> daughter.lib()
>>> o2=lcm.new('LCM_INP','new_dictionary',pyobj=daughter)
>>> state=o2['STATE-VECTOR']
>>> print('state vector=', state)
>>> o3=daughter['MIXTURES']
>>> ia=np.array([8, 7, 8, 4, 9, 1, 0, 4], dtype='i')
>>> ra=np.array([8.0,6.0,5.0,2.0,1.0], dtype='f')
>>> da=np.array([8.0,6.0,5.0,2.0,1.0], dtype='d')
>>> o2['key1']='new comments for this record'
>>> o2['key2']=ia
>>> o2['key3']=ra
>>> o2['key4']=da
```

`olcm=lcm.new(type[,name][,iact][,pyobj][,lrda][,impx])`

`olcm._impx`
`olcm._access`
`olcm._type`
`olcm._name`

`olcm.lib()`
`olcm.val()`
`tuple=olcm.keys()`
`length=olcm.len()`
`o2lcm=olcm.rep({hkey|ikey})`
`o2lcm=olcm.lis({hkey|ikey},length)`
mapping: `olcm[hkey]`
mapping: `olcm[ikey]`

Le module d'extension lifo

Le module d'extension `lifo` permet un accès in-out aux objets lifo (stack "last in first out") utilisés par CLE-2000.

```
cd Version5_beta_ev2547/PyGan
export PYTHONPATH=lib/Linux_x86_64/python/
export PYTHONMALLOC=debug
python3
>>> import lifo, lcm
>>> import numpy as np
>>> x=lifo.new()
>>> x._impx=1
>>> x.push('tata')
>>> x.push(int)
>>> x.push(12345678)
>>> x.push(2.5)
>>> x.push(3.5)
>>> x.push(float)
>>> x.pushEmpty('my_new_LCM_object','LCM')
>>> x.push('this is very looong text')
>>> print('val(3)=',x.node(3)
>>> val=x.pop()
>>> print('val(pop)=',val)
>>> ilen=x.getMax()
>>> print('stack length=',ilen)
>>> my_lcm=lcm.new('LCM_INP','nonfuel')
>>> x.push(my_lcm)
>>> x.push('more text')
>>> new_lcm=x.node('nonfuel')
>>> x.pop()
>>> x.pop()
>>> x.lib()
```

```
olifo=lifo.new([impx])
olifo._impx
olifo.lib()
olifo.push(obj)
olifo.pushEmpty(name[,type])
obj=olifo.pop()
obj=olifo.node({ipos|name})
length=olifo.getMax()
name=olifo.OSName(ipos)
```

Le module d'extension cle2000

Le module d'extension `cle2000` permet d'encapsuler Ganlib5, Trivac5, Dragon5 ou Donjon5 et d'exécuter une procédure CLE-2000 appelant des modules de ces codes ou des sous-procédures CLE-2000.

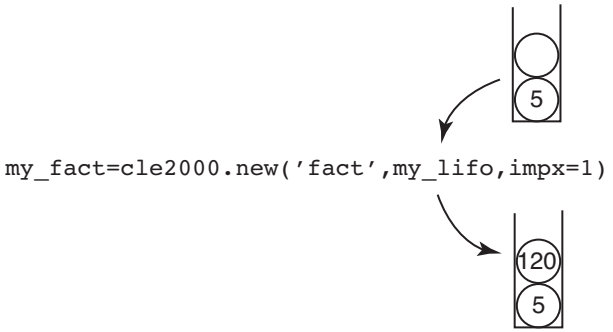
`ocle2000=cle2000.new(procname,olifo[,impx])`
`ocle2000._impx`

`ocle2000.exec()`
`olifo=ocle2000.getLifo()`
`ocle2000.putLifo(olifo)`

```
cd Version5_beta_ev2547/PyGan
export PYTHONPATH=lib/Linux_x86_64/python/
export PYTHONMALLOC=debug
python3
>>> import lifo,cle2000
>>> my_lifo=lifo.new()
>>> my_lifo.push(5)
>>> my_lifo.push(int)
>>> my_fact=cle2000.new('fact',my_lifo,1)
>>> my_fact.exec()
>>> print('factorial(5)=', my_lifo.node(1))
```

On rappelle la procédure CLE-2000 `fact.c2m`:

```
INTEGER   n n_fact prev_fact ;
 ::  >>n<< ;
IF n 1 = THEN
  EVALUATE n_fact := 1 ;
ELSE
  EVALUATE n := n 1 - ;
  ! Here, "fact" calls itself
  PROCEDURE fact ;
  fact ::   <<n>>  >>prev_fact<< ;
  EVALUATE n_fact := n 1 + prev_fact * ;
ENDIF ;
 :: <<n_fact>>  ;
QUIT " Recursive procedure *fact* XREF " .
```



Le module d'extension cle2000

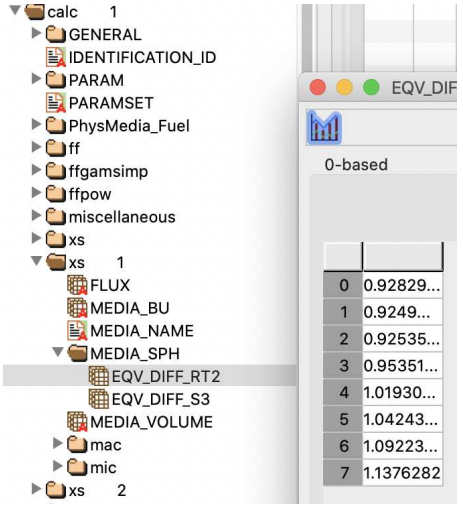
Remarque: L'équivalent python3 de la procédure `fact.c2m` est:

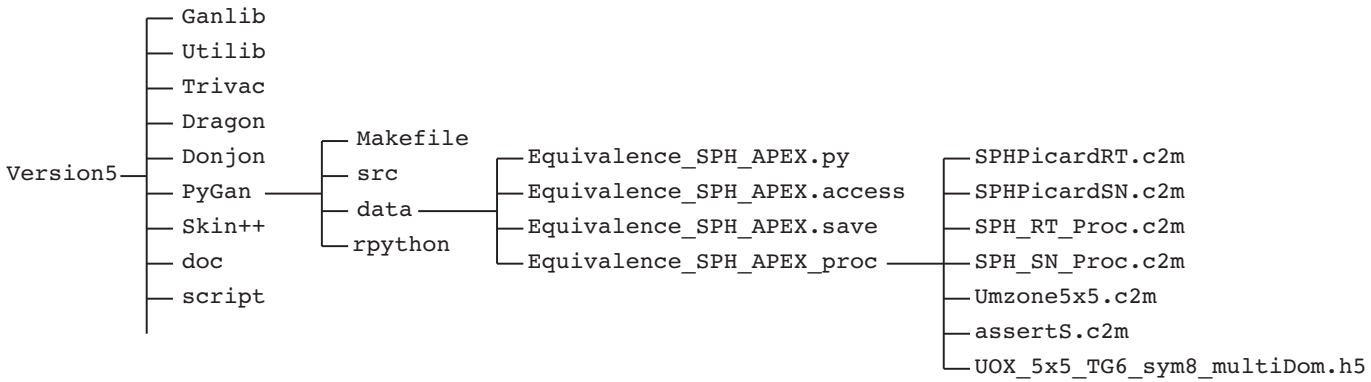
```
def factorial(n):
    if n == 1:
        return 1
    else:
        return (n * factorial(n-1))
```

Enrichissement des APEX

- Un groupe nommé `MEDIA_SPH` est ajouté à un fichier APEX de l'étude UMZONE.
- Cet enrichissement APEX est implémenté dans `Equivalence_SPH_APEX.py` :
  - en instanciant les 3 objets python3 `GeomProc`, `SPH_RT_Proc` et `SPH_SN_Proc`
  - en utilisant six procédures CLE-2000.
- Les modules d'extension `lifo` et `cle2000` sont utilisés pour enrichir le fichier APEX.

- Le fichier APEX `UOX_5x5_TG6_sym8_multiDom.h5` est modifié pour devenir `UOX_5x5_TG6_sym8_multiDom-SPH.h5` .





Enrichissement des APEX

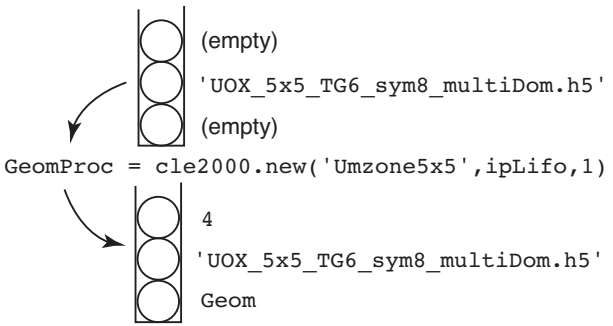
fichier Equivalence_SPH_APEX.py

```
#
# Equivalence_SPH_APEX: incorporation of SPH factors in an APEX file
#
import lifo
import lcm
import cle2000
from assertS import *
import numpy as np

# recover the macro-geometry
ipLifo=lifo.new()
ipLifo.pushEmpty("Geom", "LCM")
ipLifo.push('UOX_5x5_TG6_sym8_multiDom.h5')
ipLifo.push(int) # ncals
GeomProc = cle2000.new('Umzone5x5',ipLifo,1)
GeomProc.exec()
ipGeom = ipLifo.node("Geom")
ncals = ipLifo.node(2)
print("test Equivalence_SPH_APEX: number of calculations=",ncals)
ipGeom.lib()

# perform transport-diffusion SPH equivalence
SPH_RT_Proc = cle2000.new('SPH_RT_Proc',ipLifo,1)
SPH_RT_Proc.exec()

# perform transport-SN SPH equivalence
SPH_RT_Proc = cle2000.new('SPH_SN_Proc',ipLifo,1)
SPH_RT_Proc.exec()
print("test Equivalence_SPH_APEX completed")
```



Enrichissement des APEX

fichier Equivalence_SPH_APEX.access

```
#!/bin/sh
if [ $# = 0 ]
   then
   echo "usage: Equivalence_SPH_APEX.access directory" 1>&2
   exit 1
fi
cp $1/../Dragon/data/UOX_5x5_TG6_sym8_multiDom_proc/UOX_5x5_TG6_sym8_multiDom.h5 .
chmod 644 *.h5
ls -l
echo "Equivalence\_SPH\_APEX terminated"
```

fichier Equivalence_SPH_APEX.save

```
#!/bin/sh
#
if [ $# = 0 ]
   then
   echo "usage: Equivalence_SPH_APEX.save directory" 1>&2
   exit 1
fi
echo "access Equivalence_SPH_APEX.save"
MACH=‘uname -s‘
Sysx="‘echo $MACH | cut -b -6‘"
if [ $Sysx = "CYGWIN" ]; then
   MACH=‘uname -o‘
elif [ $Sysx = "AIX" ]; then
   MACH=‘uname -s‘
else
   MACH=‘uname -sm | sed ’s/[ ]/_/’‘
fi
ls -l
mv UOX_5x5_TG6_sym8_multiDom.h5 $1/"$MACH"/UOX_5x5_TG6_sym8_multiDom-SPH.h5
echo "Equivalence_SPH_APEX.save completed"
```

Enrichissement des APEX

fichier Umzone5x5.c2m

```
************************************************************
*                                                          *
* Procedure : Umzone5x5.c2m                                *
* Purpose   : Define the macro-geometry                    *
* Author    : A. Hebert                                    *
*                                                          *
* CALL      :                                              *
*  Geom := Umzone5x5 ;                                     *
*                                                          *
*  Input object:                                           *
*    ApexName  : Apex file name                            *
*                                                          *
*  Output objects:                                         *
*    Geom      : macro-geometry                            *
*    ncals     : number of calculations in the Apex file   *
*                                                          *
************************************************************
PARAMETER  Geom ::
  ::: LINKED_LIST Geom ; ;
MODULE GEO: HUTL: END: ;
STRING ApexName ;
  ::  >>ApexName<< ;
HDF5_FILE UOX_5x5 :: FILE <<ApexName>> ;
INTEGER ncals ;
*
REAL side1 := 1.26 ;
REAL side2 := side1 1.26 + ;
REAL side3 := side2 1.26 + ;
REAL side4 := side3 1.26 + ;
REAL side5 := side4 1.26 + ;
Geom := GEO: :: CAR2D 5 5 (*ASSEMBLY 5 X 5*)
          X- DIAG X+ REFL
          Y- REFL Y+ DIAG
          MESHX 0.0 <<side1>> <<side2>> <<side3>> <<side4>> <<side5>>
          MIX 6 5 4 5 6
                3 2 3 5
                  1 2 4
                    3 5
                      6 ;
HUTL: UOX_5x5 :: GREP 'NCALS' 1 >>ncals<< ;
  :: <<ncals>>  ;
END: ;
```

Enrichissement des APEX

fichier Equivalence_SPH_RT_Proc.c2m

```
************************************************************
*                                                          *
* Procedure : SPH_RT_Proc.c2m                              *
* Purpose   : Perform a transport-diffusion SPH equivalence*
* Author    : A. Hebert                                    *
*                                                          *
* CALL      :                                              *
*  SPH_RT_Proc Geom ;                                      *
*                                                          *
*  Input objects:                                          *
*    Geom      : macro-geometry LCM object                 *
*    ApexName  : Apex file name                            *
*    ncals     : number of calculations in the Apex file   *
*                                                          *
************************************************************
PARAMETER  Geom ::
  ::: LINKED_LIST Geom ; ;
STRING ApexName ;
  ::  >>ApexName<< ;
INTEGER ncals ;
  ::  >>ncals<< ;
MODULE SPH: TRIVAT: TRIVAA: FLUD: UTL: GREP: DELETE: END: ;
LINKED_LIST TRACK MACRO2 MACRO OPTIM SYSTEM FLUX ;
HDF5_FILE APEX_FILE :: FILE <<ApexName>> ;
INTEGER IterEmax := 1000 ;
REAL K_EFF_REF ;
PROCEDURE SPHPicardRT assertS ;
*
ECHO "SPH_SN_Proc: ApexName=" ApexName "ncals=" ncals ;
TRACK := TRIVAT: Geom ::
     TITLE 'SIMPLE 5x5 FUEL ASSEMBLY'
     EDIT 0 MAXR 3000 DUAL 3 1 ;
```

Enrichissement des APEX

fichier Equivalence_SPH_RT_Proc.c2m (cont'n)

```
*----
* Perform fixed point SPH equivalence for each branch calculation
*----
INTEGER ical := 0 ;
REPEAT

  EVALUATE ical := ical 1 + ;
  ECHO "process branch=" ical "/" ncals ;
  MACRO2 := SPH: APEX_FILE :: EDIT 1 STEP AT <<ical>> MACRO OFF LEAK ;
  GREP: MACRO2 :: GETVAL 'K-EFFECTIVE' 1 >>K_EFF_REF<< ;
  ECHO "ical=" ical "reference k-effective=" K_EFF_REF ;

  MACRO OPTIM := SPHPicardRT MACRO2 Geom TRACK
   :: 0.0 10.0 1.0E-6 <<IterEmax>> ;
  ECHO "control variables at convergence for ical=" ical ;
  UTL: OPTIM :: IMPR 'VAR-VALUE' * ;
  MACRO2 := DELETE: MACRO2 ;

*----
* Write SPH factors on APEX file
*----
  APEX_FILE := SPH: APEX_FILE OPTIM :: EDIT 1 STEP AT <<ical>>
     IDEM SPOP EQUI 'EQV_DIFF_RT2' ;
  OPTIM := DELETE: OPTIM ;

*----
* Verification calculation
*----
  SYSTEM := TRIVAA: MACRO TRACK :: EDIT 0 ;
  FLUX := FLUD: SYSTEM TRACK :: EDIT 2 ADI 3 EXTE 1.E-07 200 ;
  assertS FLUX :: 'K-EFFECTIVE' 1 <<K_EFF_REF>> ;

  MACRO SYSTEM FLUX := DELETE: MACRO SYSTEM FLUX ;
UNTIL ical ncals = ;

ECHO "Equivalence_SPH_RT_Proc completed" ;
END: ;
```

Conclusions

- Version5 offre un environnement pratique de prototypage rapide
  - utile pour les projets ODYSSEE et CamiVVER
  - permet d'enrichir un fichier APEX ou MPO avec des facteurs ADF ou SPH en mode fondamental
  - permet d'enrichir un fichier APEX ou MPO avec des facteurs DF ou SPH de réflecteur
    - réflecteur Baff-Refl de Framatome
    - réflecteur Lefebvre-Lebigot de EDF
    - réflecteur Richebois du CEA.
- Une thèse pourrait être lancée à Framatome sur le thème de l'équivalence nodale.
  - enchainement APOLLO3–Brisingr

Ressources

- Archive `Version5.0.8_ev2547.tgz`
- `Documentation`
- Textbook:
  Alain Hébert, Applied Reactor Physics, Presses Internationales Polytechnique, Third edition, Montréal, 2020.
  (to order)