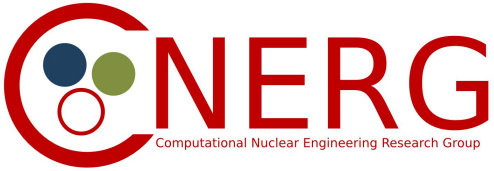
Empowering Fuel Cycle simulation with HTC Condor

B. Mouginot

HTC Condor Week - 05/2019

DEPARTMENT OF
Engineering Physics
UNIVERSITY OF WISCONSIN–MADISON

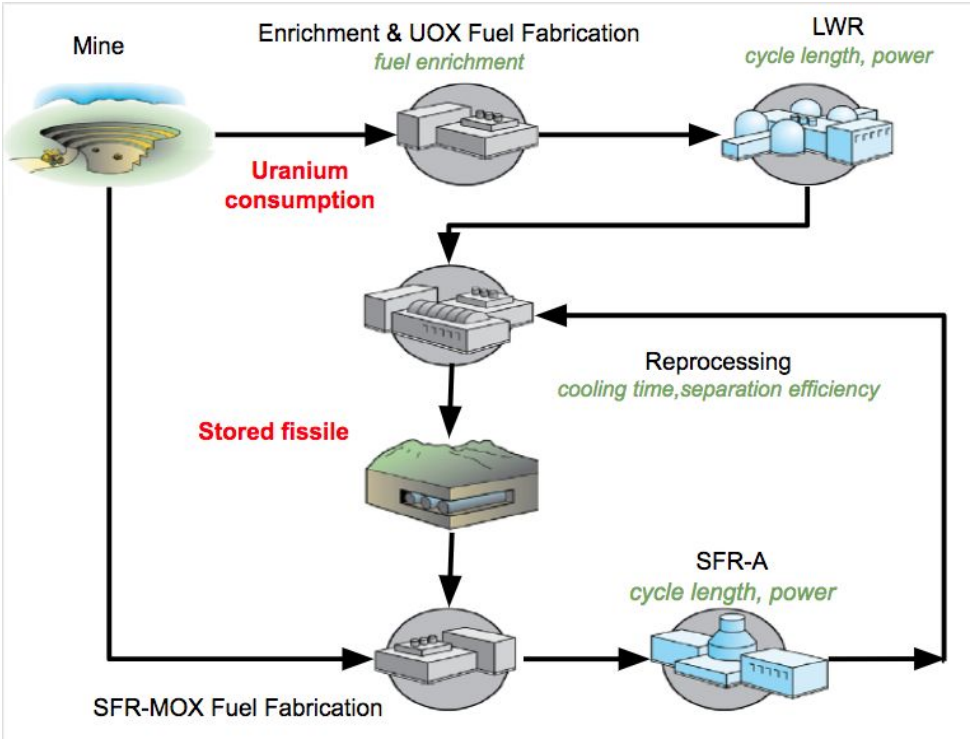


Nuclear Fuel Cycle Simulation - **What ?**

Simulation of the life of the nuclear fuel:

- From the mine to the ultimate waste
- Following all **fuel transformations**:
  - Fuel fabrication
  - Transmutation
  - Decay
  - Separation
  - …

- Scales:
  - Number of unit/facility:
    $10 - 10^3$ (country size)
  - Time scale
    10 days - 10^2 years

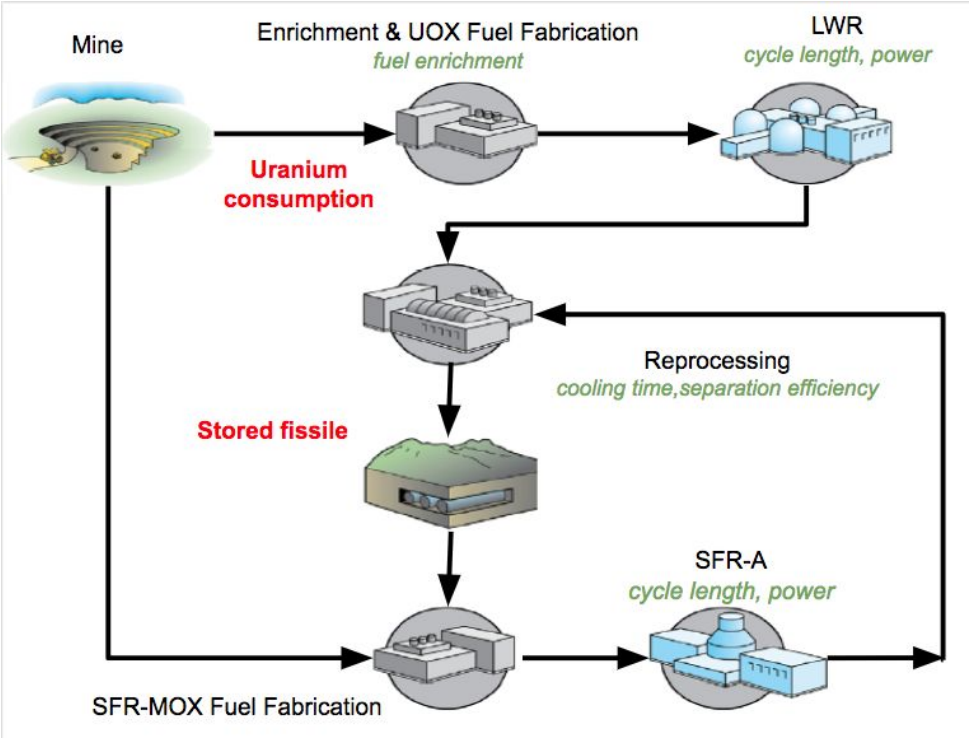


Nuclear Fuel Cycle Simulation - **Why ?**

Simulation of the life of the nuclear fuel:

- Assessing possible future:
  - Cost
  - Waste production
  - Proliferation risk
  - Deployment optimisation
  - Phase-out

- Retrieve past production:
  - Benchmark
  - Non-proliferation:
    - Verification
    - Nuclear archeology



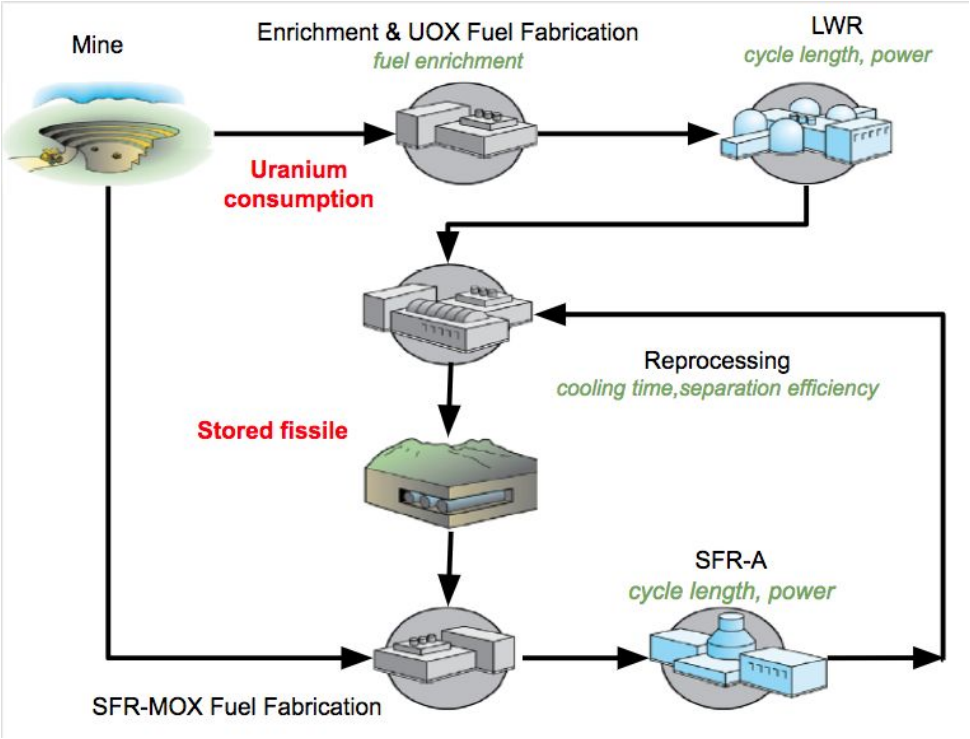
Nuclear Fuel Cycle Simulation - **How ?**

Simulation of the life of the nuclear fuel:

- NFC Simulators:
  - ORION[1], DYMOND[2], VISION[3],
  - CLASS[4], COSI[5], CYCLUS[6]

- CYCLUS: the (real) Agent based simulator

- Each facility:
  - individual decision making algorithm
  - Interacts through the Dynamical Resource exchange
  - could be as detailed as possible



First use of HTC

Neural network (NN) Prediction :

- 1 fuel/reactor type
- Neutron multiplication factor evolution
- ~700 Macroscopic reaction cross sections

Neural network (NN) training :

- trained on 1.10^3 to 10.10^3 nuclear fuel evolutions
- Each evolution requires:
  - ~10 CPU.h
  - 180 Mo raw data produced
  - 1.9 Mo of useful data
- Up to:
  - ~100 000 CPU.h per model
  - ~1.8 To of raw data
  - ~1.8 Go of useful data

First use of HTC

Neural network (NN) Prediction :

- 1 fuel/reactor type
- Neutron multiplication factor evolution
- ~700 Macroscopic reaction cross sections

Neural network (NN) training :

- trained on 1.10^3 to 10.10^3 nuclear fuel evolutions
- Each evolution requires:
  - ~10 CPU.h
  - 180 Mo raw data produced
  - 1.9 Mo of useful data
- Up to:
  - ~100 000 CPU.h per model
  - ~1.8 To of raw data
  - ~1.8 Go of useful data

Solution:

- tar.gz raw datas (from 180 to 50 Mo)
- Increased disk quotat
- batch of 5000 jobs

- Directly upload raw data on Box

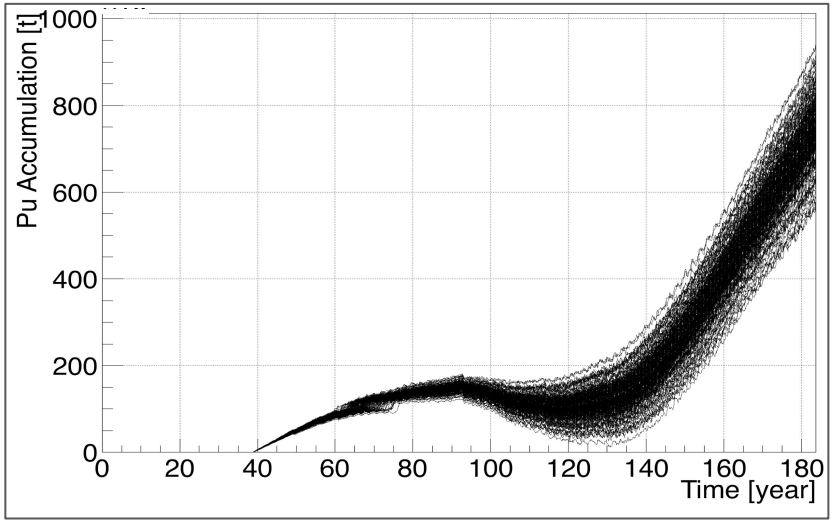
```
> cp $home/.netrc ~/

> curl -1 -v --disable-epsv --ftp-skip-pasv-ip
--ftp-ssl --upload-file ${1}_raw.tar.gz  -netrc
ftp://ftp.box.com/.../${1}_raw.tar.gz

 > rm -rf ~/.netrc


.netrc:
"machine ftp.box.com login YYYYY password XXXXXXX"
```

Stochastic Uncertainty Propagation

Stochastic uncertainty propagation:

- 1/10 of the US nuclear fleet over 180 years
- Main uncertainty propagation: 400 simulations
- Parameters normally distributed

- Output metrics uncertainty: std deviation
- Individual contribution One At the Time:
  200 simulations each

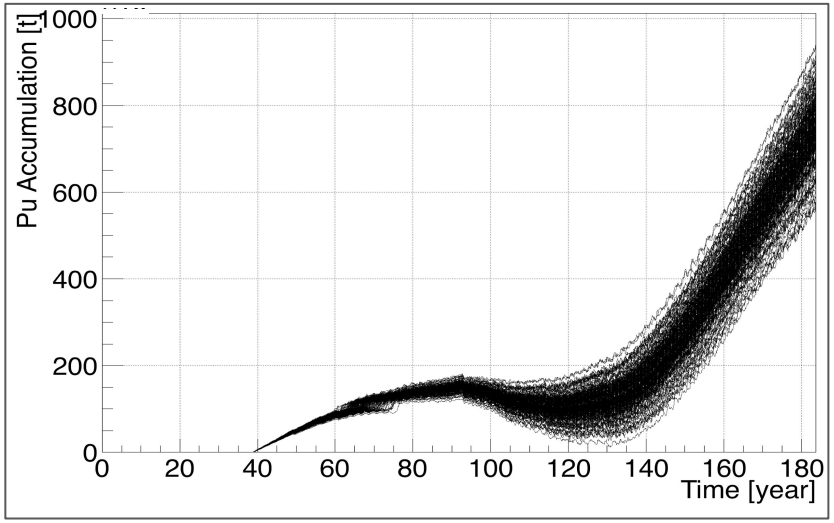


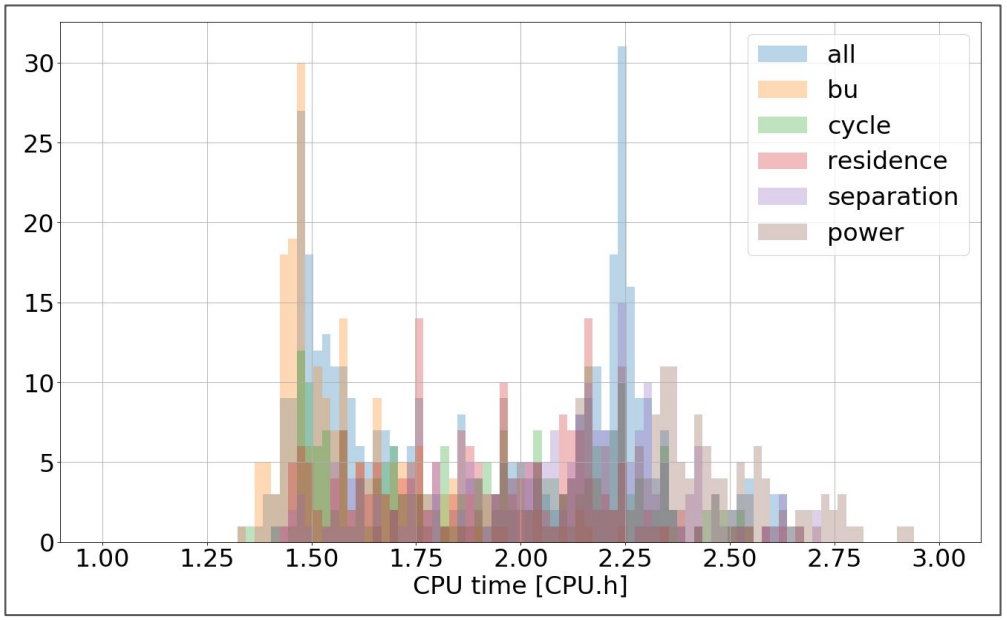
Stochastic Uncertainty Propagation

Stochastic uncertainty propagation:

- 1/10 of the US nuclear fleet over 180 years
- Main uncertainty propagation: 400 simulations
- Parameters normally distributed

- Output metrics uncertainty: std deviation
- Individual contribution One At the Time:
  200 simulations each





Computational needs:

- Per jobs:
  - ~ 2.2CPU.h
  - 20 Go memory
  - 22 Mo of generated
- 1400 jobs
- Total of ~2700 CPU.h
- x 2 (bug fixing, errors, crashes, data loss, …)

Issues: setup

Worker setup:

- Build a tar.gz package:
  - simulation software
  - Dependencies
  - Need to update the tar.gz frequently

    once/twice a year

Issues: setup

Worker setup:

- Build a tar.gz package:
  - simulation software
  - Dependencies
  - Need to update the tar.gz frequently

    once/twice a year

Typical dependency list:

- c/cpp compiler
- python
- CMake
- Boost
- libxml2
- libxml++
- sqlite3
- HDF5
- Coin-Cbc
- py_Jinja2
- py_NumPy
- py_Pandas
- py_Cython
- ROOT + pyROOT
- *CLASS*
- *cyCLASS*
- *cycamore*
- ...

Issues: setup

Worker setup:

- Build a tar.gz package:
  - simulation software
  - Dependencies
  - Need to update the tar.gz frequently
    once/twice a year

- Script:
  - No direct intervention:
    hard to debug and fix problem
  - Need to wait for the job to come back

- Interactive:
  - Wait to get an interactive worker
  - Connection issues:
    - connection/wifi loss
    - Not done when need to move
      (meeting, back home, …)

Typical dependency list:

- c/cpp compiler
- python
- CMake
- Boost
- libxml2
- libxml++
- sqlite3
- HDF5
- Coin-Cbc
- py_Jinja2
- py_NumPy
- py_Pandas
- py_Cython
- ROOT + pyROOT
- *CLASS*
- *cyCLASS*
- *cycamore*
- ...

Issu

Solutions:

- install scripts
- tmux/screen (from desktop computer) to keep connection alive

Worl

-

- Dependencies
- Need to update the tar.gz frequently
  once/twice a year

- Script:
  - No direct intervention:
    hard to debug and fix problem
  - Need to wait for the job to come back

- Interactive:
  - Wait to get a interactive worker
  - Connection issues:
    - connection/wifi loss
    - Not done when need to move
      (meeting, back home, …)

st:

- libxml++
- sqlite3
- HDF5
- Coin-Cbc
- py_Jinja2
- py_NumPy
- py_Pandas
- py_Cython
- ROOT + pyROOT
- *CLASS*
- *cyCLASS*
- *cycamore*
- ...

Issu

Wor

- Dependencies
- Need to update the tar.gz frequently
  once/twice a year

- Script:
  -
  -
- Inte
  -
  -

st:

- libxml++
- sqlite3
- HDF5
- Coin-Cbc
- py Jinja2

OT

Solutions:

- install scripts
- tmux/screen (from desktop computer) to keep connection alive

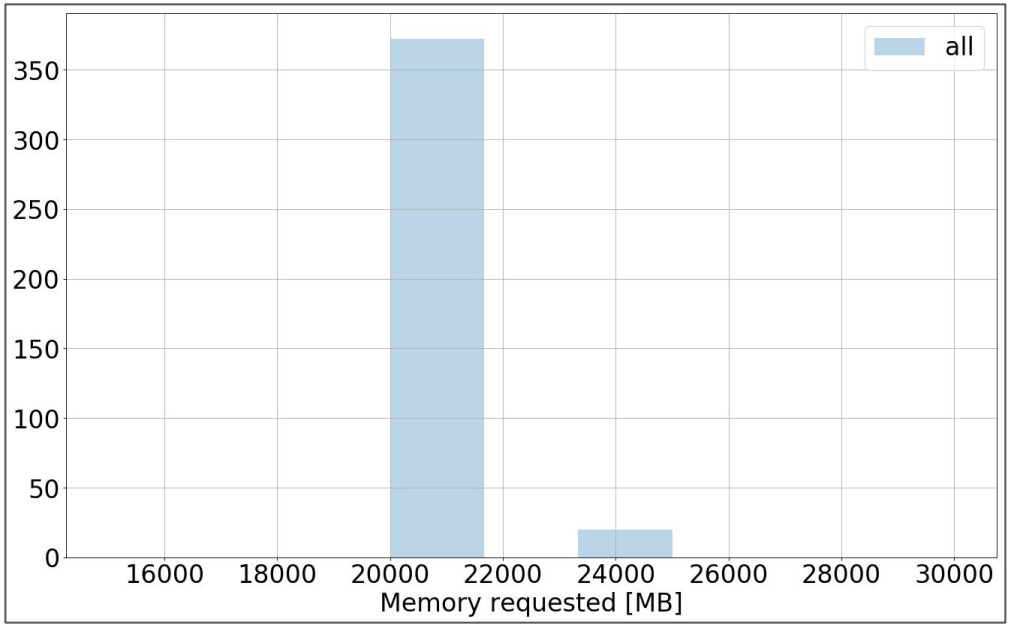
Future workflow:

- Docker:
  - pro:
    - Complete control of the environment
    - Easy to test and install softwares
  - Cons:
    - Image size
    - Speed ?

Issues: exceeding memory allocation

5-10% of the job on hold:

- Some jobs used too much memory



- Solution: restart them requesting more memory (x1.2) up to 4 times

```
cvt.sub:
# Conditionnal memory request
request_memory = ifthenelse(MemoryUsage =?= undefined, 20000, (MemoryUsage * 1.2))

# Periodidic release
periodic_release = (JobStatus == 5)
                        && ((CurrentTime - EnteredCurrentStatus) > 180)
                        && (JobRunCount < 4)
                        && (HoldReasonCode =?= 34)
```

Workflow Improvement

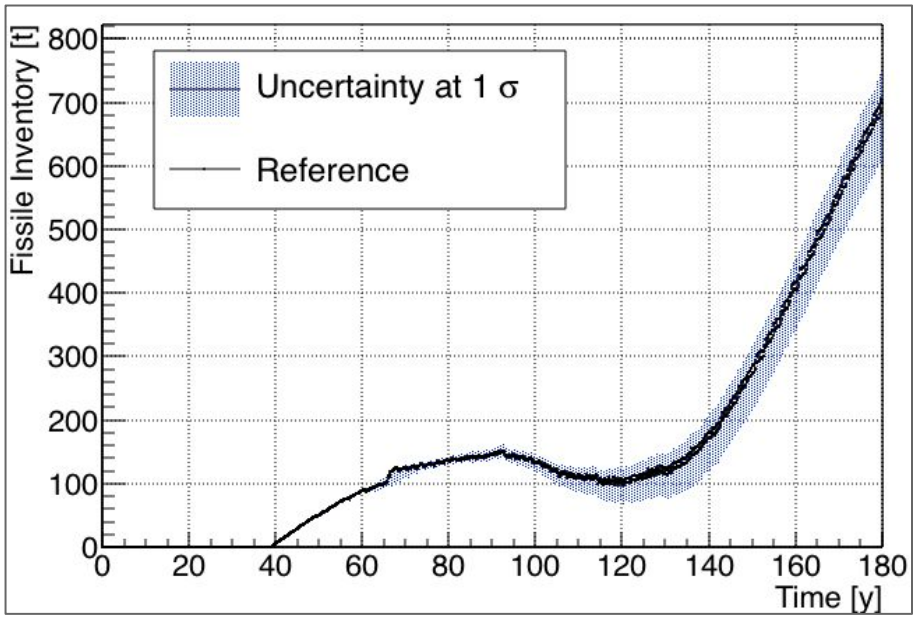
Post-processing:

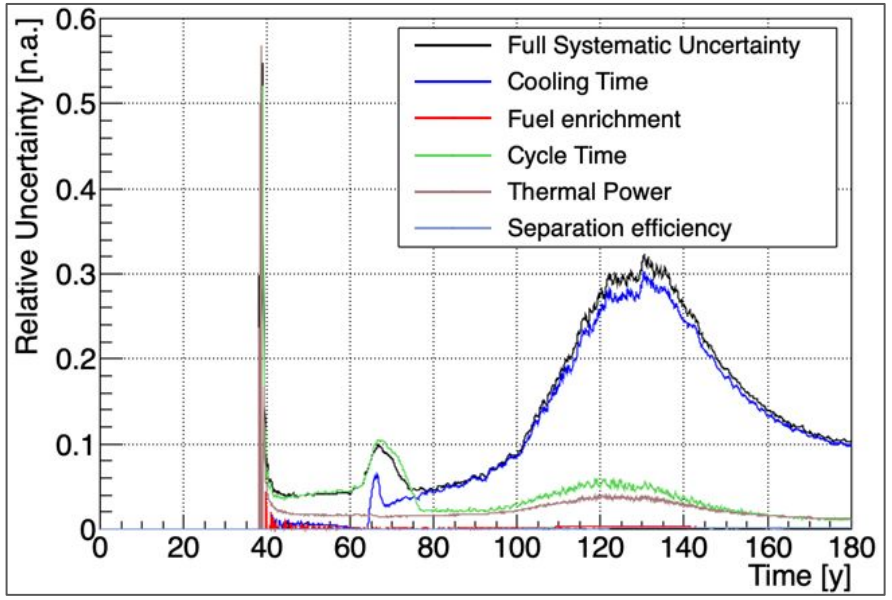
- Combining different simulations
- Computing the uncertainty
- ~2h per set / ~12h total
- Was run locally :(

Use flow manager (*makeflow* / *dagman* / *?*):

- combine simulation
- compute the main interesting metrics
- Produce plots

Final Results





References

1. R. GREGG and K. HESKETH, “The benefits of a fast reactor closed fuel cycle in the UK,” American Nuclear Society - ANS; La Grange Park (UnitedStates) (2013).
2. Moisseytsev, A., “DYMOND – A Dynamic Model of Nuclear Development,” Internal Report, Argonne National Laboratory, August (2001).
3. J.J. JACOBSON et al., “Verifiable Fuel Cycle Simulation Model (VISION): A Tool for Analyzing Nuclear Fuel Cycle Futures,” Nuclear Technology, 172, 157-178 (2010).
4. B. Mouginot, B. Leniau, N. Thiolliere, M. Ernoult, S. David, X. Doligez, A. Bidaud, O. Meplan, R. Montesanto, G. Bellot, J.B. Clavel, I. Duhamel, E. Letang, J. Miss., "Core library for advanced scenario simulation, {C.L.A.S.S.}: principle & application.", *PHYSOR, (*2014)
5. C. Coquelet-Pascal, M. Tiphine, G. Krivtchik, D. Freynet, C. Cany, R. Eschbach & C. Chabert. “COSI6: A Tool for Nuclear Transition Scenario Studies and Application to SFR Deployment Scenarios with Minor Actinide Transmutation”, NUclear Technology, Vol 192, 2 (2015)
6. K. D. Huff, M. J. Gidden, R. W. Carlsen, R. R. Flanagan, M. B. McGarry, A. C. Opotowsky, E. A. Schneider, A. M. Scopatz, P. P.H. Wilson, “Fundamental concepts in the Cyclus nuclear fuel cycle simulation framework“, *Advances in Engineering Software,* 94, (April 2016)

DEPARTMENT OF Engineering Physics
UNIVERSITY OF WISCONSIN–MADISON

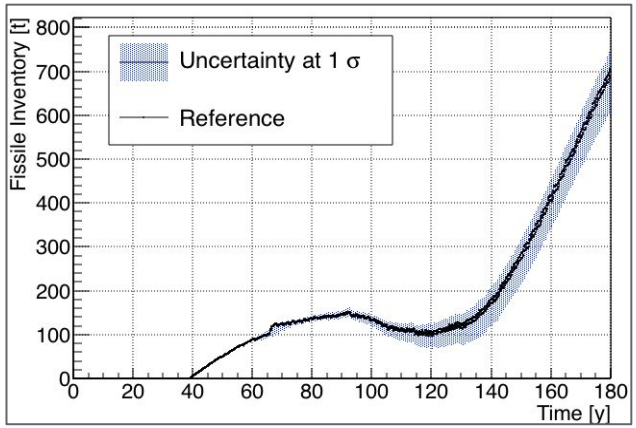
CONSORTIUM FOR VERIFICATION TECHNOLOGY

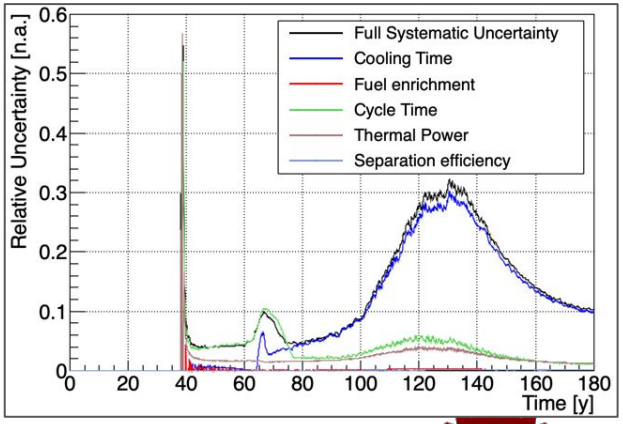
CNERG
Computational Nuclear Engineering Research Group

Thank you !

Questions/Comments ?

Thanks to the Research Computing Facilitators for all the help and advices.





CENTER FOR HIGH THROUGHPUT COMPUTING

CYCLUS

Acknowledgements:

This work was funded by the Consortium for Verification Technology under Department of Energy National Nuclear Security Administration award number DE-NA0002534.

This research was performed using the compute resources and assistance of the UW-Madison Center For High Throughput Computing (CHTC) in the Department of Computer Sciences. The CHTC is supported by UW-Madison, the Advanced Computing Initiative, the Wisconsin Alumni Research Foundation, the Wisconsin Institutes for Discovery, and the National Science Foundation, and is an active member of the Open Science Grid, which is supported by the National Science Foundation and the U.S. Department of Energy's Office of Science.