Integration of Thermochimica with ORIGEN using the ORIGEN-API

and its application to molten salt reactors

B.W.N. Fitzpatrick and M.H.A. Piro
University of Ontario Institute of Technology
Faculty of Energy Systems and Nuclear Science

Agenda

- What’s Thermochimica?
- How can Thermochimica and ORIGEN together solve nuclear fuel problems?
- Integrating (FORTRAN) Thermochimica into C++ code
- Very basics of the ORIGEN-API, and an example
- ORIGEN-API for molten salt reactors
- Integrating Thermochimica into the ORIGEN-API
- Results for a simple molten salt reactor problem

What’s Thermochimica?

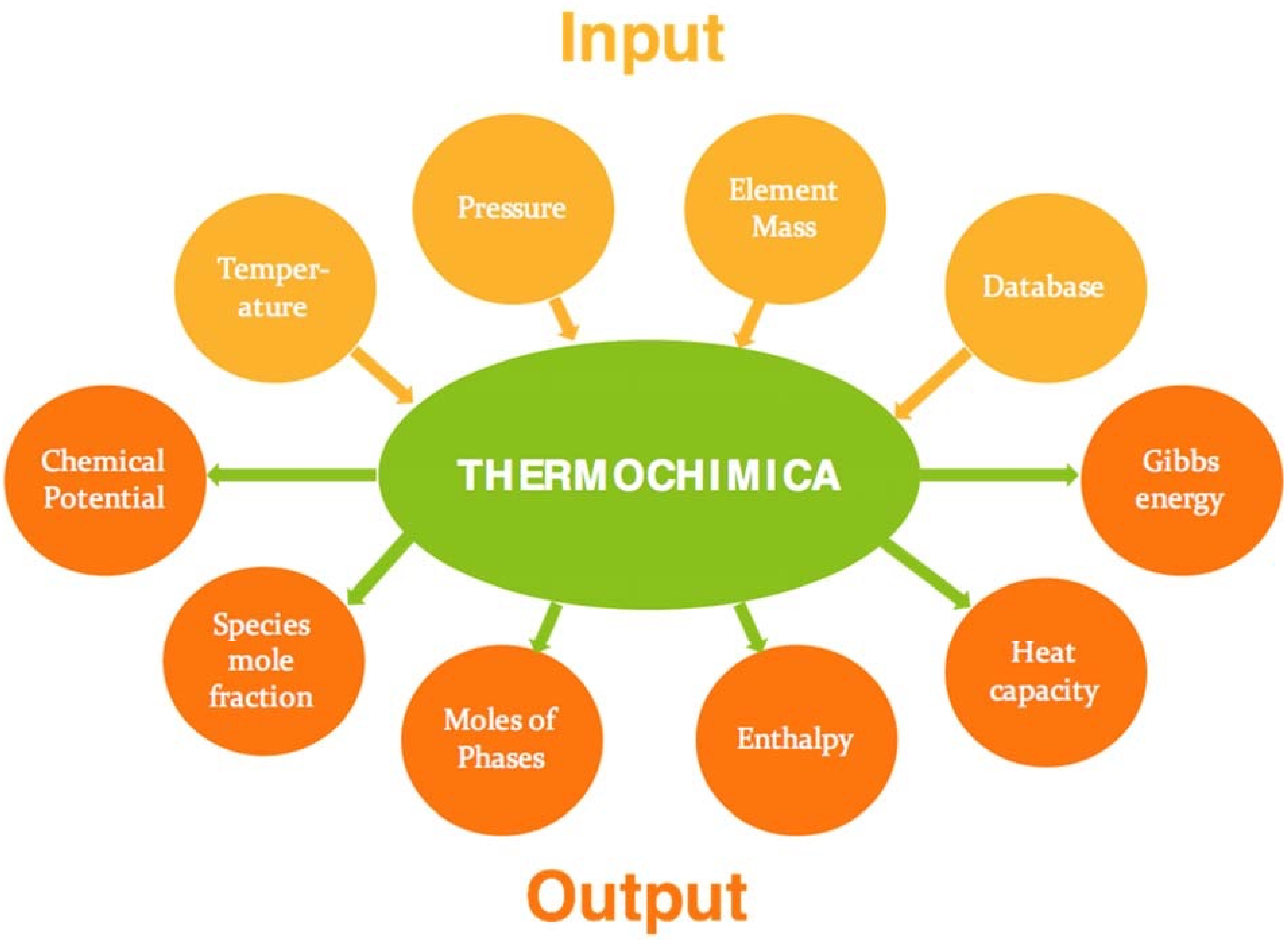
and why would I want to use it?

Thermochimica is…

- equilibrium thermodynamics solving system
- particularly good for large, multicomponent, multiphase systems
- designed to work with high-performance multiphysics computing codes
- open source

Is ‘burning’ nuclear fuel at equilibrium?

- Close enough
- Chemical equilibrium is attained at relatively short time periods due to the high temperatures of nuclear fuel
- Different elements in nuclear fuel are randomly mixed
- Time scales in nuclear performance simulations are very long *(M.H.A. Piro, J. Banfield, K.T. Clarno, S. Simunovic, T.M. Besmann, B.J. Lewis, et al., Journal of Nuclear Materials. 441 (2013) 240–251).*
- In molten salts, diffusion through the fluid adds to mixing


Input
Temper-ature
Pressure
Element Mass
Database
THERMOCHIMICA
Chemical Potential
Gibbs energy
Species mole fraction
Moles of Phases
Enthalpy
Heat capacity
Output

FactSage / Thermochimica Database File

```
System U-F-Li
    3    2    2    3   12
U F Li
238.02891000 6123456 6123456
 gas_ideal
 IDMX
 LiF
   1  1    0.0    1.0    1.0
  6000.0000     -351581.57
 0.27571767E-07 0.00000000
UF4
   1  1    1.0    4.0    0.0
  6000.0000     -1639992.8
 0.24183333E-06  510660.00
LIQUsoln
 SUBG
2.40000 23
```

Thermochimica and Multiphysics

Thermochimica Output		**Physical phenomena**
Heat capacity and enthalpy	$\xrightarrow{\text{useful for}}$	heat transfer calculations
Speciation of a salt	$\xrightarrow{\text{useful for}}$	viscosity for momentum equations
Phase quantities	$\xrightarrow{\text{useful for}}$	multi-phase flow calculations

How Thermochimica works…

- Uses the fundamentals of equilibrium thermodynamics to simplify the numerical approach (1st & 2nd laws, minimization of Gibbs free energy)

- Several numerical advantages are obtained, which speed up convergence while increasing numerical stability.

- Read more: *M.H.A. Piro, S. Simunovic, T.M. Besmann, B.J. Lewis, W.T. Thompson, Computational Materials Science, Computational Materials Science. 67 (2013) 266–272.*

How Thermochimica Works

```
program thermo

    USE ModuleThermoIO
    USE ModuleThermo
    USE ModuleGEMSolver

    implicit none

    ! Specify units:
    cInputUnitTemperature = 'K'
    cInputUnitPressure    = 'atm'
    cInputUnitMass        = 'moles'
    cThermoFileName       = '../data/deleteme2.dat'

    ! Specify values:
    dTemperature          = 553.15D0
    dPressure             = 1D0
    dElementMass          = 0D0
    dElementMass(40)      = 89.24D0    ! Zr
    dElementMass(50)      = 0.7D0      ! Sn
    dElementMass(8)       = 0.73D0     ! O
    dElementMass(1)       = 8.27D0     ! H
    dElementMass(41)      = 0.9D0      ! Nb
    dElementMass(23)      = 0.16d0     ! V
    ...
```

How Thermochimica Works

```
      …

    ! Specify output and debug modes:
    iPrintResultsMode = 2
    lDebugMode       = .FALSE.
    !lDebugMode       = .TRUE.

    ! Parse the ChemSage data-file:
    call ParseCSDataFile(cThermoFileName)

    ! Call Thermochimica:
    if (INFOThermo == 0) call Thermochimica

    ! Perform post-processing of results:
    if (iPrintResultsMode > 0) call PrintResults

    ! Destruct everything:
    if (INFOThermo == 0) call ResetThermoAll

    ! Call the debugger:
    call ThermoDebug

end program thermo
```

Thermochimica Output…

```
|              THERMOCHIMICA RESULTS              |
=================================================

     91.000 mol BCC_A2
          { 2.9960E-04 NB:H
          + 2.6446E-05 NB:O
          + 9.5641E-03 NB:VA
          + 2.3302E-04 SN:H
          + 2.0569E-05 SN:O
          + 7.4387E-03 SN:VA
          + 5.3262E-05 V:H
          + 4.7015E-06 V:O
          + 1.7003E-03 V:VA
          + 2.9707E-02 ZR:H
          + 2.6223E-03 ZR:O
          + 0.94833    ZR:VA }

     ----------------------------------------------
     Sublattice 1; stoichiometric coefficient: 1.0000
          { Nb      9.8901E-03
          + Sn      7.6923E-03
          + V       1.7582E-03
          + Zr      0.98066    }

     Sublattice 2; stoichiometric coefficient: 3.0000
          { H       3.0293E-02
          + O       2.6740E-03
          + VA      0.96703    }
```

```
============================================================
|              System properties              |
============================================================

 Temperature =     553.15 [K]
 Pressure    =     1.0000 [atm]

 System Component Mass [mol]  Chemical potential [J/mol]
 ---------------- ----------  --------------------------
 Sn              7.0000E-01  -2.601398E+05
 Nb              9.0000E-01  -2.424836E+04
 Zr              8.9240E+01  -2.166661E+04
 V               1.6000E-01  -2.195394E+04
 O               7.3000E-01  -5.873069E+05
 H               8.2700E+00  -8.083377E+04

 Integral Gibbs energy = -3.23819E+06 [J]
 Functional norm       =  1.25761E-12 [unitless]

 # of stable pure condensed phases =   0
 # of stable solution phases       =   1

============================================================
```

Thermochimica Output ct’d

Recent Developments

Modified Quasi-chemical Model (MQM)

- Of specific interest to molten salts
- Does not focus on chemical species on a lattice, but rather mixing of species in pairs.
- This captures short-range order in liquid or solid solutions

Relevant to molten salt calculations!

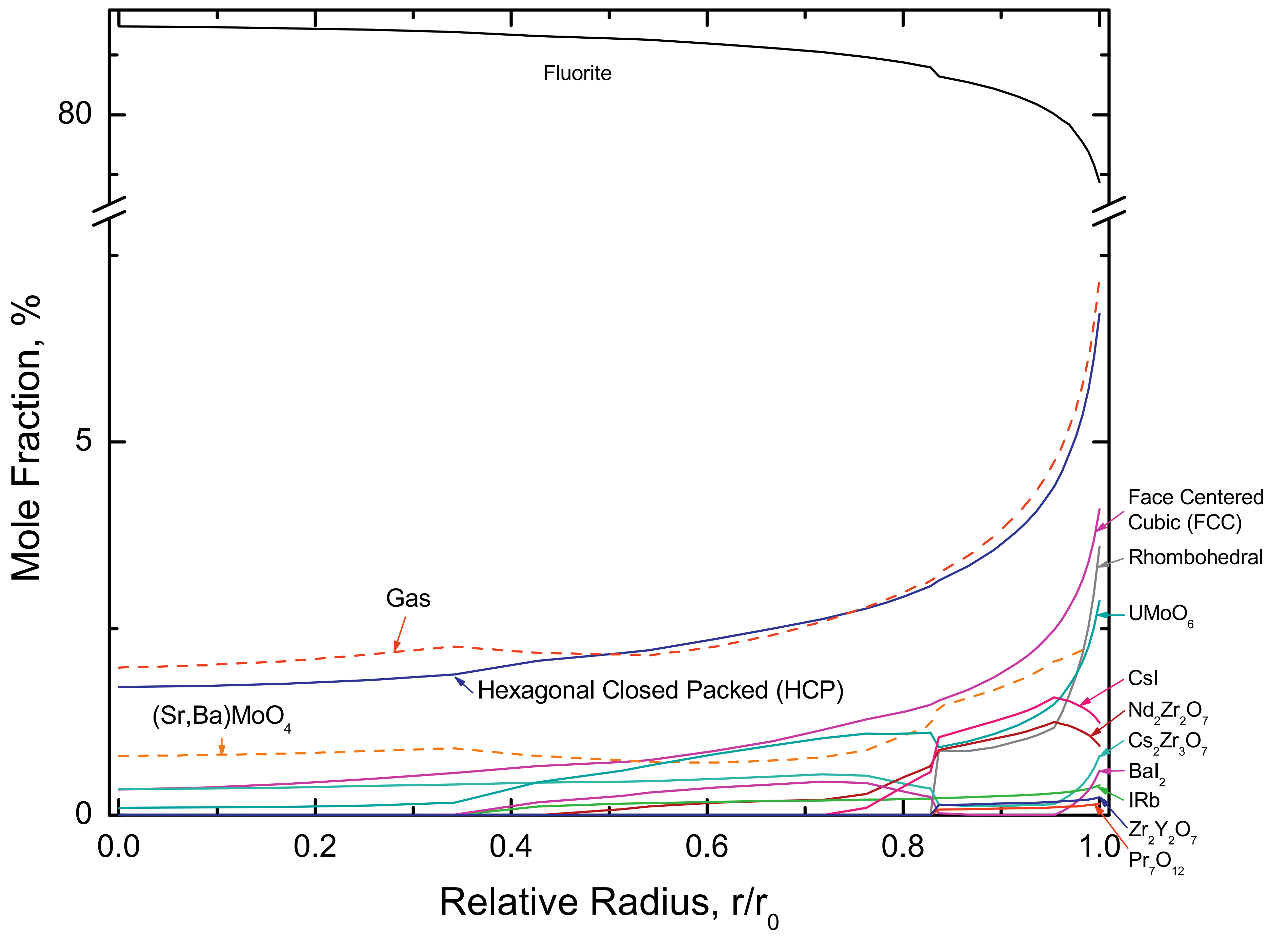
Previous Work

- Work by *M.H.A. Piro* et. al. simulated the irradiation of a nuclear fuel pellet in an LWR
- Combined Thermochimica, ORIGEN and AMP

Previous Work

Origen simulation of fission product formation in solid fuel pellet in an LWR

M.H.A. Piro, J. Banfield, K.T. Clarno, S. Simunovic, T.M. Besmann, B.J. Lewis, et al., Journal of Nuclear Materials, 441 (2013) 240–251.

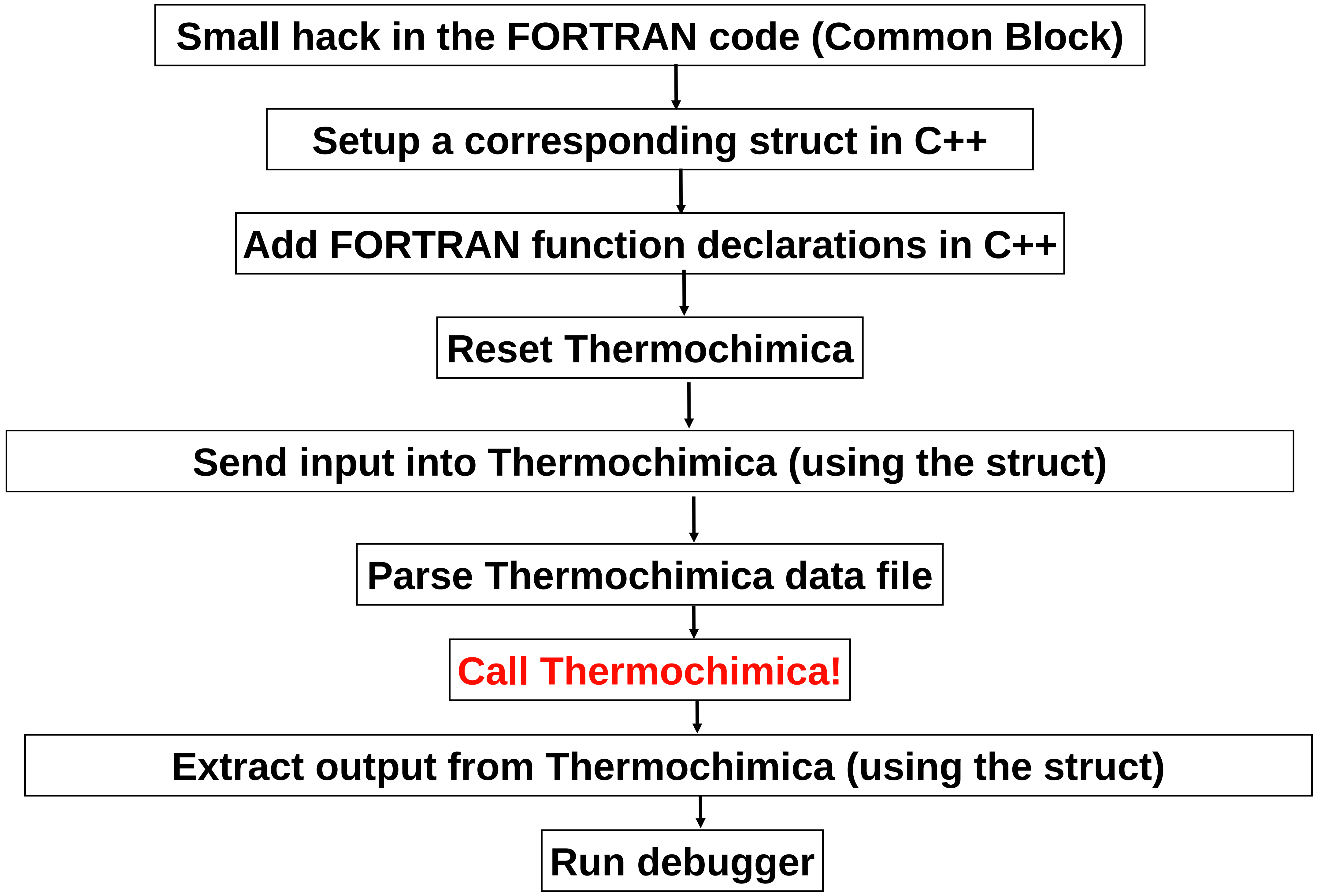


Previous Work

Simulated distribution of phases in solid fuel pellet in an LWR

M.H.A. Piro, J. Banfield, K.T. Clarno, S. Simunovic, T.M. Besmann, B.J. Lewis, et al., Journal of Nuclear Materials, 441 (2013) 240–251.

Integrating Thermochimica into C++ Code


Small hack in the FORTRAN code (Common Block)
Setup a corresponding struct in C++
Add FORTRAN function declarations in C++
Reset Thermochimica
Send input into Thermochimica (using the struct)
Parse Thermochimica data file
Call Thermochimica!
Extract output from Thermochimica (using the struct)
Run debugger

Integrating Thermochimica into C++ Code

small hack for input/output…

in `/thermochimica/src/shared/ModuleThermoIO.f90,` we have

```
module ModuleThermoIO

  ! INPUT VARIABLES:
  integer                              :: iCounter, iPrintResultsMode
  real(8)                              :: dTemperature, dPressure
  real(8),       dimension(0:118)       :: dElementMass
  character(15)                        :: cInputUnitTemperature, cInputUnitPressure, cInputUnitMass
  character(120)                       :: cThermoFileName
  ! OUTPUT VARIABLES
  integer                              :: nSolnPhasesOut, nPureConPhaseOut, nSpeciesOut, INFOThermo
  real(8)                              :: dGibbsEnergySys
  real(8),dimension(:),allocatable      :: dSolnPhaseMolesOut, dPureConPhaseMolesOut, dSpeciesMoleFractionOut
  character(25),dimension(:),allocatable :: cSolnPhaseNameOut, cPureConPhaseNameOut, cSpeciesNameOut, cSpeciesPhaseOut
  logical, dimension(:),allocatable     :: lSpeciesStable
```

just add…

```
COMMON/THERM/ dTemperature, dPressure, dGibbsEnergySys, dElementMass, iCounter, &
iPrintResultsMode, INFOThermo, nSolnPhasesOut, nPureConPhaseOut, nSpeciesOut, &
cInputUnitTemperature, cInputUnitPressure, cInputUnitMass, cThermoFileName
```

Integrating Thermochimica into C++ Code

```
extern "C" {


   struct{
      double dTemperature;
      double dPressure;
      double dGibbsEnergySys;
      double dElementMass[119];
      int iCounter;
      int iPrintResultsMode;
      int INFOThermo;
      int nSolnPhasesOut;
      int nPureConPhaseOut;
      int nSpeciesOut;
      char cInputUnitTemperature[15];
      char cInputUnitPressure[15];
      char cInputUnitMass[15];
      char cThermoFileName[120];

   } therm_;
```

corresponding struct in C++

Integrating Thermochimica into C++ Code

```
void fortfunc_(int *ii, float *ff);
void helloworld_(int *ii);
void parsecsfile0_(char cc[120]);
void goodbyeworld_(int *ii);
void parsecsdatafile_(char cc[120]);
void stringconverter_(char cc[15]);
void thermochimica_();
void thermodebug_(); //was int *ii
void resetthermoall_();
void printresults_();
void variablepointers_();
void resultstofile_();
```

Add FORTRAN function declarations to C++

Integrating Thermochimica into C++ Code

Reset Thermochimica

```
resetthermoall_();
```

Integrating Thermochimica into C++ Code

Send input to Thermochimica (using the struct)

```
therm_.dElementMass[1] = 0.5;
therm_.dElementMass[8] = 1;

therm_.dTemperature = 553.15;
therm_.dPressure = 0.5;
```

Integrating Thermochimica into C++ Code

Access output

```
printf("Species out: %i   nSolnPhasesOut %i, nPureConPhaseOut: %i, Gibbs
energy:%e   n",therm_.nSpeciesOut,therm_.nSolnPhasesOut,
therm_.nPureConPhaseOut, therm_.dGibbsEnergySys);
```

Integrating Thermochimica into C++ Code

Run the debugger

```
thermodebug_();
```

ORIGEN API

What’s the ORIGEN API?

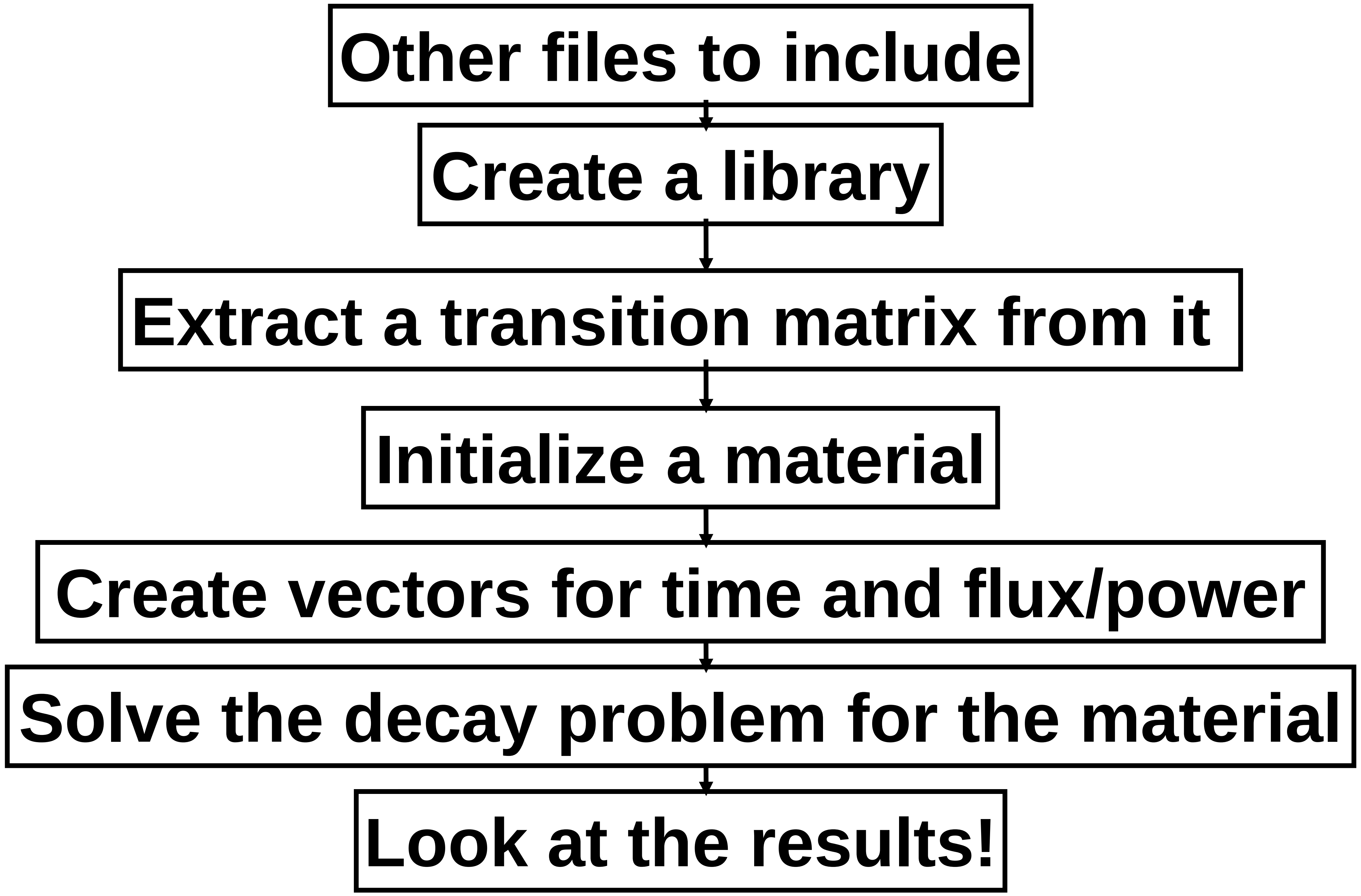
- Application Programming Interface is software that allows 2 applications to communicate with each other
- The ORIGEN-API is a set of classes for Oak Ridge Isotope GENeration (ORIGEN) for performing depletion / decay calculations in your own software.

(W. A. Wieselquist et al., ORIGEN API v0.5.2 Github page)

An Example

exNEAMS.cpp is an example that ships with the ORIGEN API.

exNEAMS.cpp - Test2
an Introductory Example


Other files to include
Create a library
Extract a transition matrix from it
Initialize a material
Create vectors for time and flux/power
Solve the decay problem for the material
Look at the results!

exNEAMS.cpp
the code

```
#include <algorithm>
#include <iostream>
#include <string>
#include <vector>

#include "Nemesis/gtest/nemesis_gtest.hh"
#include "Nemesis/harness/DBC.hh"
#include "Origen/Core/dc/FakeFactory.h"
#include "Origen/Core/dc/TransitionMatrixP.h"
#include "Origen/Core/io/LibraryIO.h"
#include "Origen/Core/xf/MultiZoneDepleter.h"
#include "Origen/Core/xf/Solver_Fake.h"
#include "Origen/Manager/libld/TransitionMatrixUpdater.h"
#include "Origen/Solver/matrex/Solver_matrex.h"
#include "ScaleUtils/IO/DB.h"
#include "Standard/Interface/AbstractWriter.h"
#include "Standard/Interface/BasicIOWriter.h"
#include "Standard/Interface/Communicator.h"
#include "Standard/Interface/jdebug.h"

using ScaleUtils::IO::nprintf;
using namespace Origen;

typedef TransitionMatrixP TransitionMatrix;
typedef SP_TransitionMatrixP SP_TransitionMatrix;
typedef std::vector<double> Vec_Dbl;
typedef std::shared_ptr<Vec_Dbl> SP_Vec_Dbl;

// GLOBAL COMMUNICATOR
Standard::Communicator world;
```

other files to include

exNEAMS.cpp
the code

Creating a library

```
// Get a general 2237-nuclide ORIGEN library.
  SP_Library lib( new Library() );
  FakeFactory::Library_scale_pwr( *lib );
```

exNEAMS.cpp
the code

Extracting a transition matrix

```
// Extract a transition matrix from it.  We will use this PWR transition
// matrix for all materials in this test.
SP_TransitionMatrix trx( lib->newsp_transition_matrix_at( 0 ) );
```

exNEAMS.cpp
the code

Initializing a material

```
// Create a single 3.6% enriched UO2 material.
SP_Material mat;
{
   std::vector<int> ids;            // nuclide ids in IZZZAAA format
   std::vector<double> numden;         // atoms/barn-cm
   double volume = 5.6;              // cm^3
   int id = 1234;                    // material id
   std::string name = "material_1234";  // material name

   // Use the FakeFactory to get a reasonable initial composition.
   FakeFactory::vera_uox_e360( ids, numden );

   // Initialize the material.
   mat = SP_Material( new Material( lib, name, id, volume ) );

   // Set the beginning of step number densities.
   mat->set_numden_bos( numden, ids );
}
```

exNEAMS.cpp
the code

Create vectors for time and flux/power

```
// Create depletion times/fluxes.
//  0 -->   3 days at 1e14 n/cm^2s
//  3 --> 250 days at 1e14 n/cm^2s
// 250 --> 500 days at 1e14 n/cm^2s
// 500 --> 515 days at 0 (decay)
// 515 --> 530 days at 0 (decay)
std::vector<double> time{0, 3, 250, 500, 515, 530};
std::vector<double> flux{1, 1, 1, 0, 0};
for( size_t j = 0; j < time.size(); ++j )
   time[j] *= 86400.0;  // scale to seconds
for( size_t j = 0; j < flux.size(); ++j )
   flux[j] *= 1e14;  // scale to n/cm^2s
size_t nsteps = flux.size();
```

exNEAMS.cpp
the code

```
// Create a Bell depletion/decay solver.
Solver_matrex slv;

// Enter step loop.
for( size_t j = 0; j < nsteps; ++j )
{
   // Add a new time step to this material.
   double dt = time[j + 1] - time[j];
   mat->add_step( dt );
   mat->set_flux( flux[j] );
   mat->set_transition_matrix( trx );

   // Solve the step using only data on the material, solve takes
   // the bos vector, time, flux, and pointer to eos.
   SP_Vec_Dbl n0 = mat->amount_bos();
   SP_Vec_Dbl n1 = mat->amount_eos();
   slv.set_transition_matrix( &*mat->transition_matrix() );
   slv.solve( *n0, mat->flux(), mat->dt(), &*n1 );
   slv.clear();
}
```

Solve the decay problem

exNEAMS.cpp
the code

Look at the results!

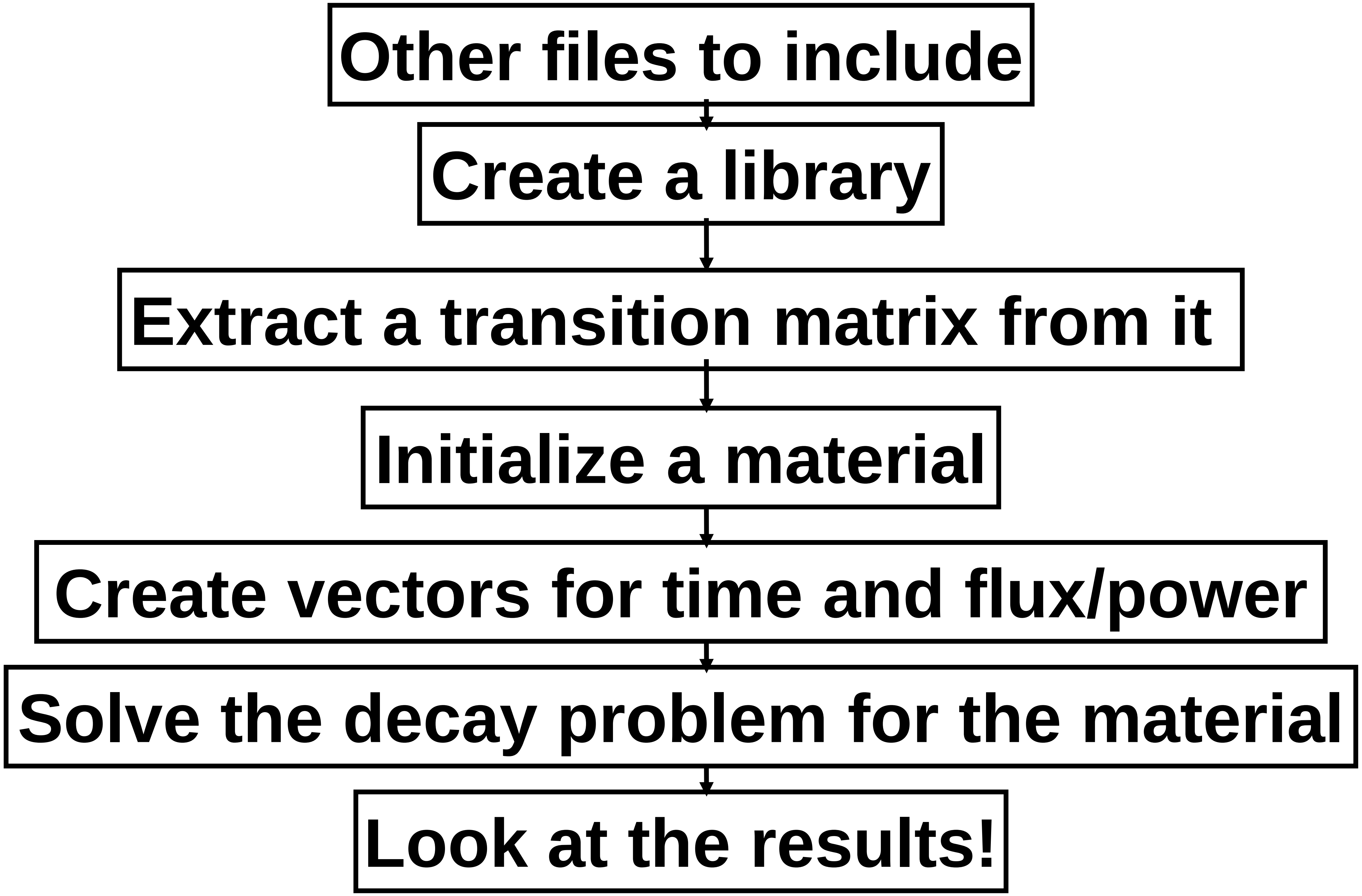
```
// Inspect results stored in material.
if( false ) std::cout << mat->to_string() << std::endl;
```

Let’s make this more applicable to molten salt reactors

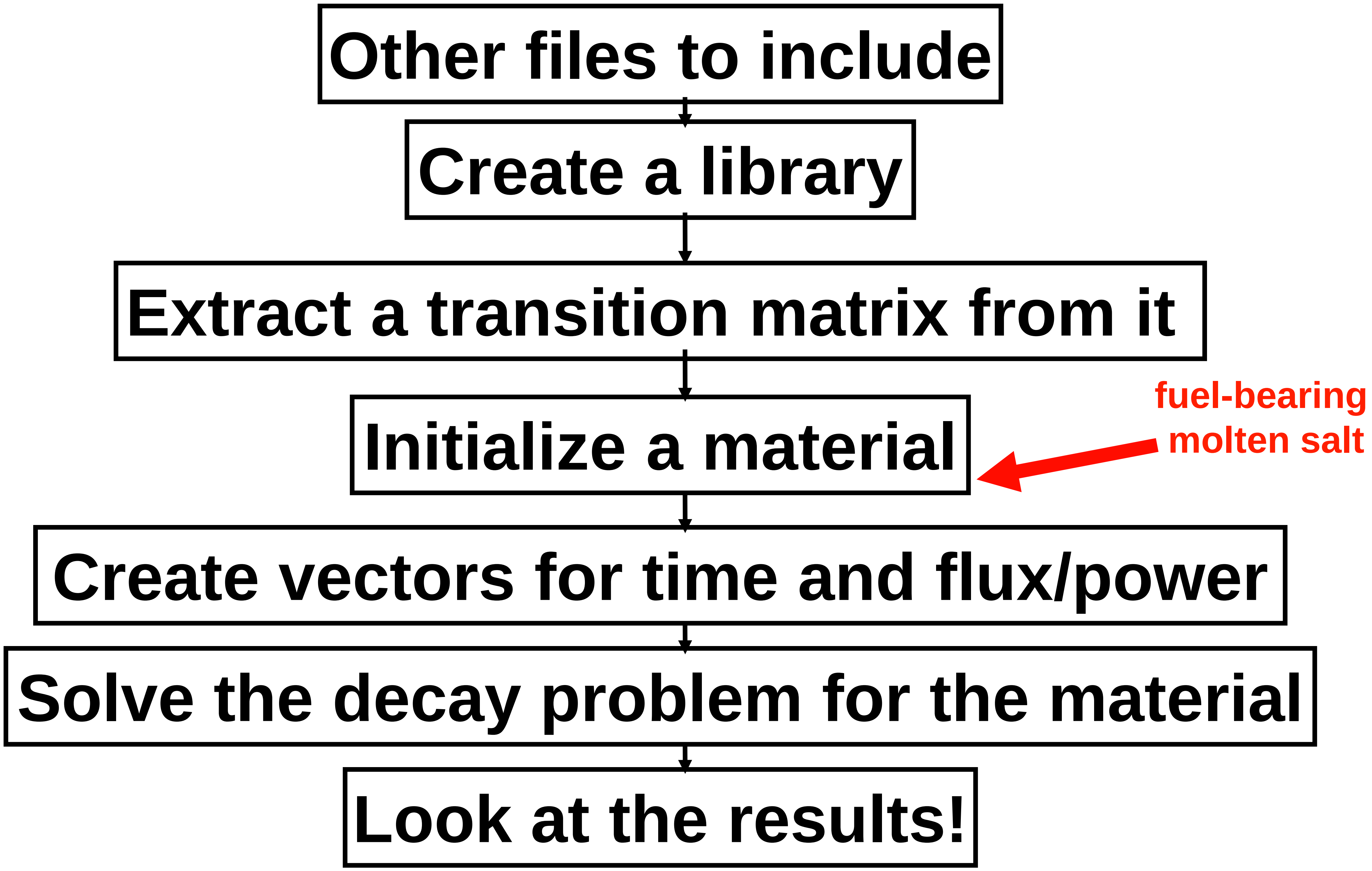
Let’s do a fictive simulation of the MSRE,

by modifying exNEAMS.cpp.

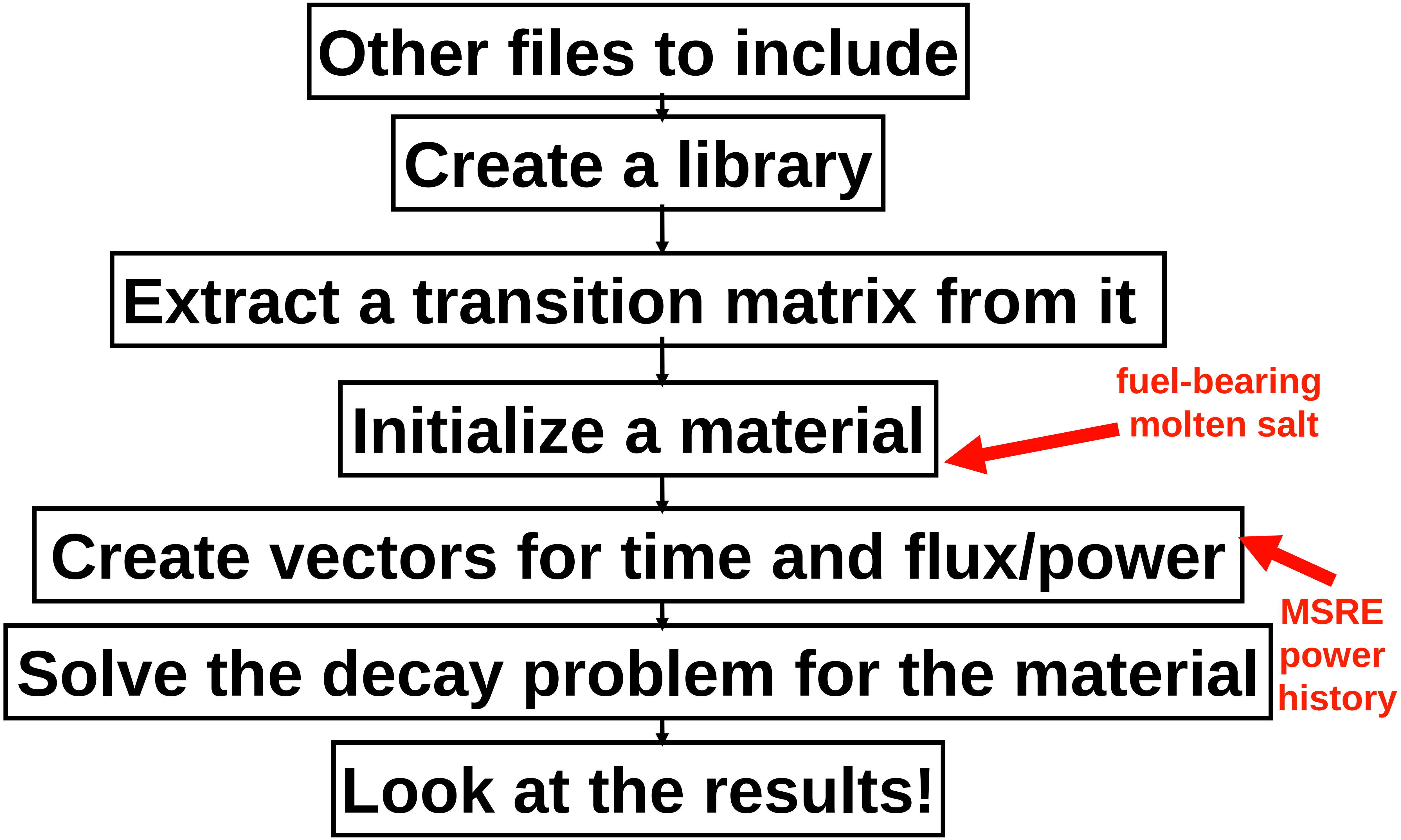
exNEAMS.cpp - Modifying for Molten Salt Simulation


Other files to include
Create a library
Extract a transition matrix from it
Initialize a material
Create vectors for time and flux/power
Solve the decay problem for the material
Look at the results!

exNEAMS.cpp - Modifying for Molten Salt Simulation


Other files to include
Create a library
Extract a transition matrix from it
Initialize a material
fuel-bearing molten salt
Create vectors for time and flux/power
Solve the decay problem for the material
Look at the results!

exNEAMS.cpp - Modifying for Molten Salt Simulation



exNEAMS.cpp Modifying for Molten Salt Simulation

Fuel-bearing molten salt

```
//FakeFactory::vera_uox_e360( ids, numden );

    std::map<int, double> map_numden;
    map_numden[3007] = 4.13510E-02; // Li-7
    map_numden[4009] = 1.23417E-02; // Be-9
    map_numden[9019] = 7.20399E-02; // F-19
    map_numden[40090] = 6.54746E-04; // Zr-90
    map_numden[40091] = 1.42502E-04; // Zr-91
    map_numden[40092] = 2.18206E-04; // Zr-92
    map_numden[40094] = 2.21259E-04; // Zr-94
    map_numden[40096] = 3.56255E-05; // Zr-96
    map_numden[92234] = 1.14510E-08; // U-234
    map_numden[92235] = 8.01574E-05; // U-235
    map_numden[92238] = 1.48852E-04; // U-238
    ScaleSTL::map_to_vectors( map_numden, &ids, &numden );
```

$^{7}LiF-BeF_2-ZrF_4-UF_4$
(65-29-5-1 mole %)

35% enriched ^{235}U

exNEAMS.cpp
Modifying for Molten Salt Simulation

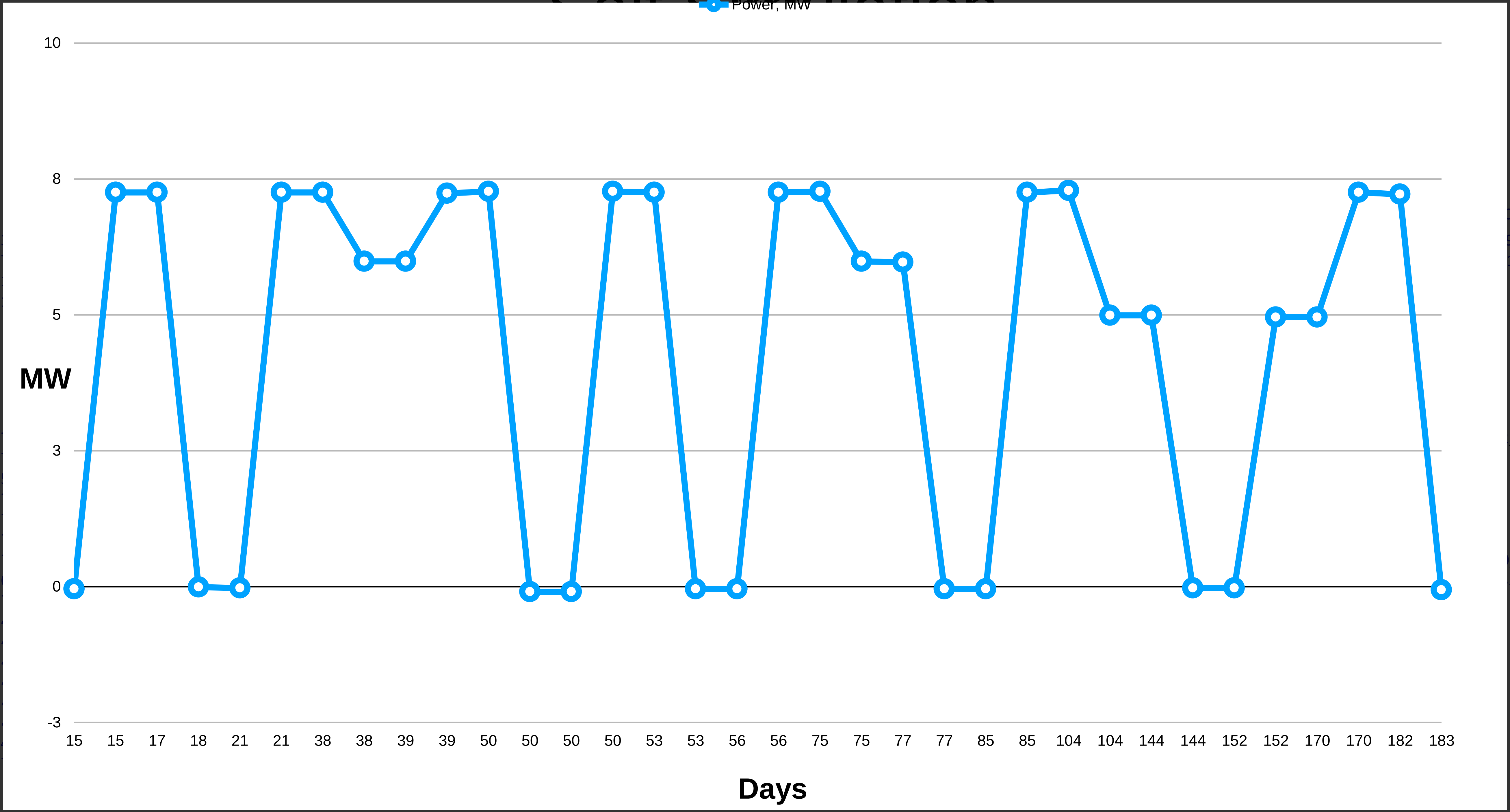
```
  std::vector<double> time{0.52, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 14.9, 15, 15, 16, 17, 17.34, 17.59, 18, 19, 20, 20.89, 20.99, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33, 34, 35, 36, 37, 37.75, 37.8, 38, 38.92, 38.97, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 49.44, 49.69, 49.74, 49.89, 50, 51, 52, 52.64, 52.79, 53, 54, 55, 56, 56.14, 56.15, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70,
71, 72, 73, 74, 74.88, 74.93, 75, 76, 76.36, 76.41, 77, 78, 79, 80, 81, 82, 83, 84, 85, 85.04, 85.09, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 103.73, 103.88, 104, 105, 106, 107, 108, 109, 110, 111,
112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 143.86, 143.91, 144, 145, 146, 147, 148, 149, 150, 151, 152,
152.04, 152.05, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 169.97, 169.98, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 181.95, 182, 182.36, 182.37};

   //std::vector<double> flux{1, 1, 1, 0, 0};
   std::vector<double> power{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7.251971791624240, 7.25197179162425, 7.25197179162426, 7.25197179162427,
7.251971791624240, 0, 0, 0, 0, 0, 7.251971791624240, 7.25197179162425, 7.25197179162426, 7.25197179162427, 7.25197179162428, 7.25197179162429, 7.2519717916243, 7.25197179162431, 7.25197179162432,
7.25197179162433, 7.25197179162434, 7.25197179162435, 7.25197179162436, 7.25197179162437, 7.25197179162438, 7.25197179162439, 7.2519717916244, 7.25197179162441, 7.251971791624240,
5.9849753563303500, 5.98497535633036, 5.9849753563303500, 7.234850218174320, 7.23485021817433, 7.23485021817434, 7.23485021817435, 7.23485021817436, 7.23485021817437, 7.23485021817438,
7.23485021817439, 7.2348502181744, 7.23485021817441, 7.23485021817442, 7.23485021817443, 7.269093365074150, 7.269093365074150, 0, 7.269093365074150, 7.269093365074151, 7.269093365074152,
7.269093365074153, 7.269093365074155, 0, 0.00000000000000, 0.00000000000000, 0.00000000000000, 0.00000000000000, 0, 7.25197179162425, 7.25197179162426, 7.25197179162427, 7.25197179162428,
7.25197179162429, 7.2519717916243, 7.25197179162431, 7.25197179162432, 7.25197179162433, 7.25197179162434, 7.25197179162435, 7.25197179162436, 7.25197179162437, 7.25197179162438,
7.25197179162439, 7.2519717916244, 7.25197179162441, 7.25197179162442, 7.25197179162443, 7.269093365074150, 5.9849753563303500, 5.98497535633036, 5.98497535633037, 5.967853782880430, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 7.251971791624240, 7.25197179162425, 7.25197179162426, 7.25197179162427, 7.25197179162428, 7.25197179162429, 7.2519717916243, 7.25197179162431, 7.25197179162432, 7.25197179162433,
7.25197179162434, 7.25197179162435, 7.25197179162436, 7.25197179162437, 7.25197179162438, 7.25197179162439, 7.2519717916244, 7.25197179162441, 7.25197179162442, 7.286214938524070,
4.991924096235140, 4.99192409623515, 4.99192409623516, 4.99192409623517, 4.99192409623518, 4.99192409623519, 4.9919240962352, 4.99192409623521, 4.99192409623522, 4.99192409623523,
4.99192409623524, 4.99192409623525, 4.99192409623526, 4.99192409623527, 4.99192409623528, 4.99192409623529, 4.9919240962353, 4.99192409623531, 4.99192409623532, 4.99192409623533,
4.99192409623534, 4.99192409623535, 4.99192409623536, 4.99192409623537, 4.99192409623538, 4.99192409623539, 4.9919240962354, 4.99192409623541, 4.99192409623542, 4.99192409623543,
4.99192409623544, 4.99192409623545, 4.99192409623546, 4.99192409623547, 4.99192409623548, 4.99192409623549, 4.9919240962355, 4.99192409623551, 4.99192409623552, 4.99192409623553,
4.99192409623554, 4.991924096235140, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4.957680949335310, 4.95768094933532, 4.95768094933533, 4.95768094933534, 4.95768094933535, 4.95768094933536, 4.95768094933537,
4.95768094933538, 4.95768094933539, 4.9576809493354, 4.95768094933541, 4.95768094933542, 4.95768094933543, 4.95768094933544, 4.95768094933545, 4.95768094933546, 4.95768094933547,
4.95768094933548, 4.957680949335310, 7.251971791624240, 7.25197179162425, 7.25197179162426, 7.25197179162427, 7.25197179162428, 7.25197179162429, 7.2519717916243, 7.25197179162431,
7.25197179162432, 7.25197179162433, 7.25197179162434, 7.25197179162435, 7.25197179162436, 7.217728644724400, 7.2177286447245, 0};
```

Power history taken from: M. Rosenthal, R. Briggs, P. Haubenreich, al. Molten salt reactor program semiannual progress report for period ending august 31, ORNL-4622. USA: Oak Ridge National Laboratory, 1970: 38-41, (1968)

exNEAMS.cpp

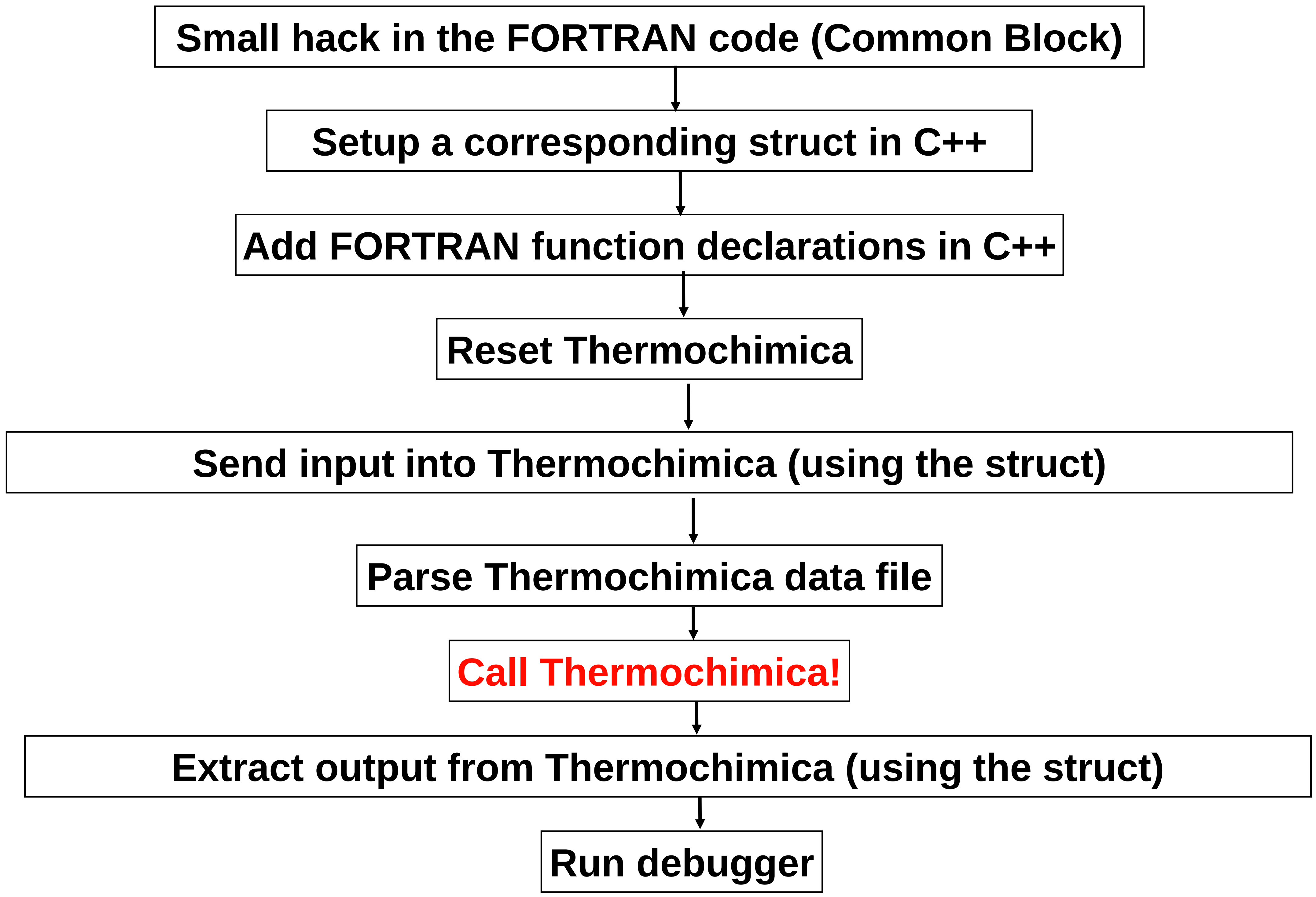
Modifying for Molten Salt Simulation



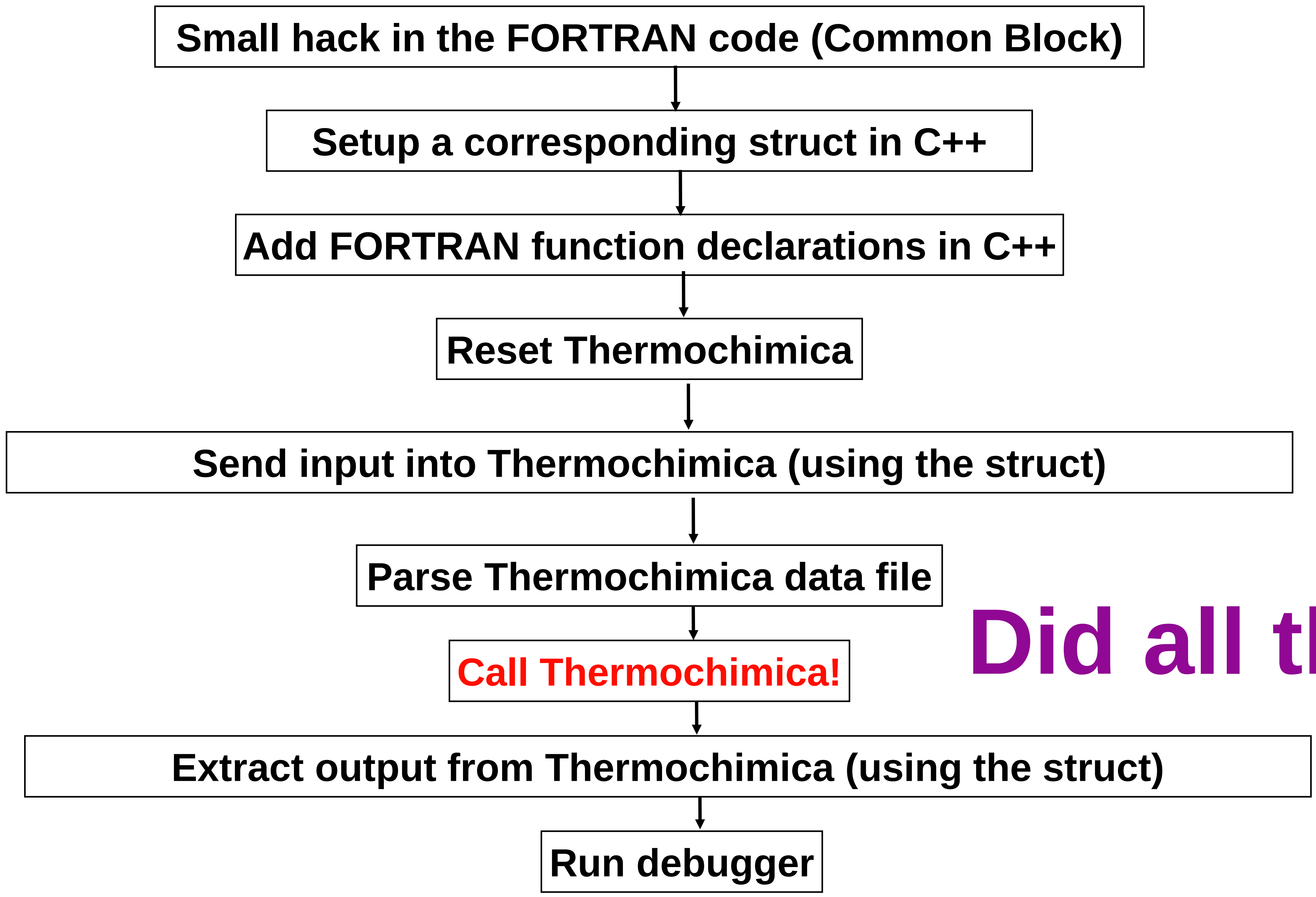
Haubenreich, al. Molten salt reactor program semiannual progress report for period ending august 31, ORNL-4622. USA:

Let’s bring in the thermochemistry to our mini-MSRE experiment.

Integrating Thermochimica into C++ Code


Small hack in the FORTRAN code (Common Block)
Setup a corresponding struct in C++
Add FORTRAN function declarations in C++
Reset Thermochimica
Send input into Thermochimica (using the struct)
Parse Thermochimica data file
Call Thermochimica!
Extract output from Thermochimica (using the struct)
Run debugger

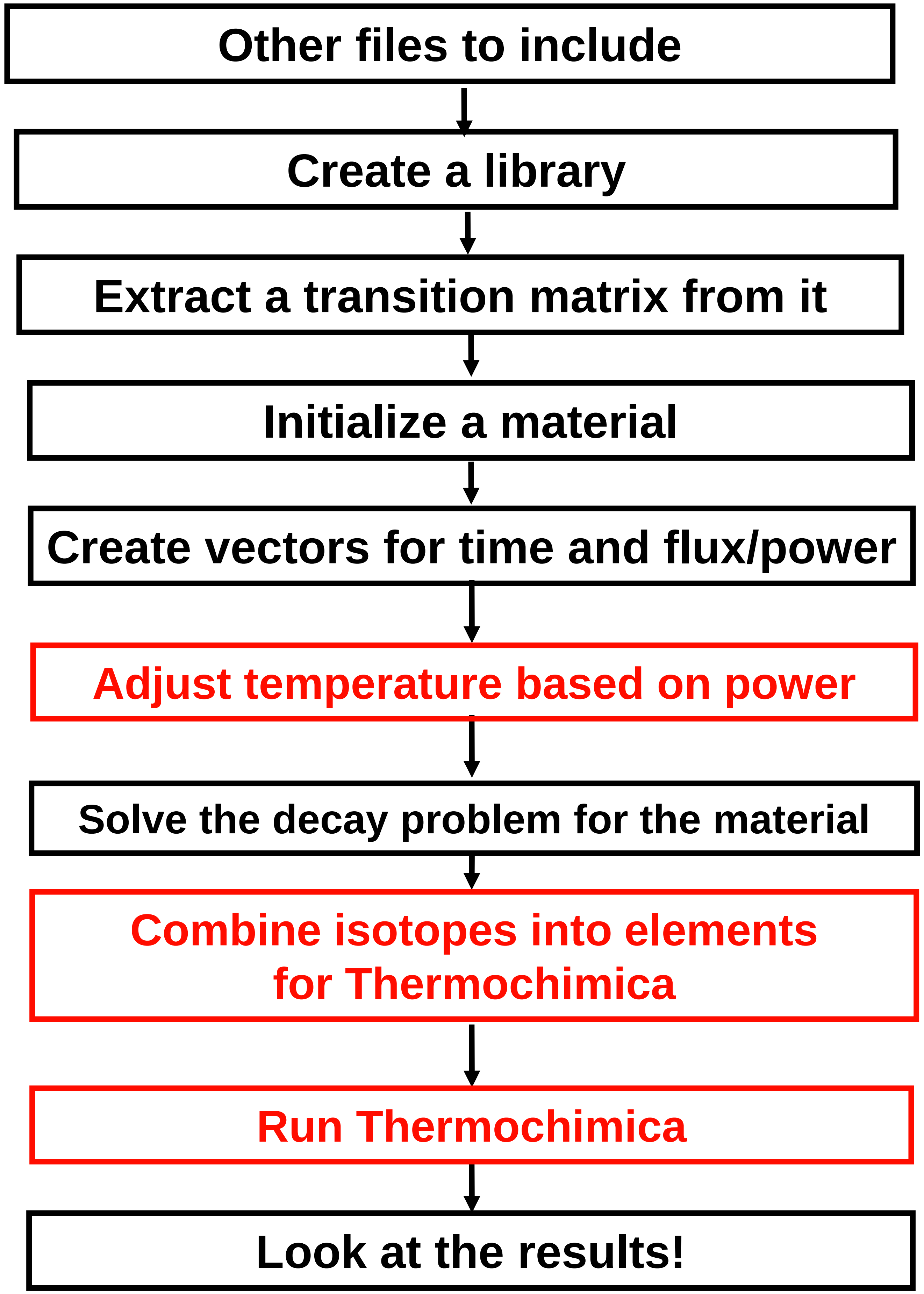
Integrating Thermochimica into C++ Code


Small hack in the FORTRAN code (Common Block)
Setup a corresponding struct in C++
Add FORTRAN function declarations in C++
Reset Thermochimica
Send input into Thermochimica (using the struct)
Parse Thermochimica data file
Call Thermochimica!
Extract output from Thermochimica (using the struct)
Run debugger

Did all that.

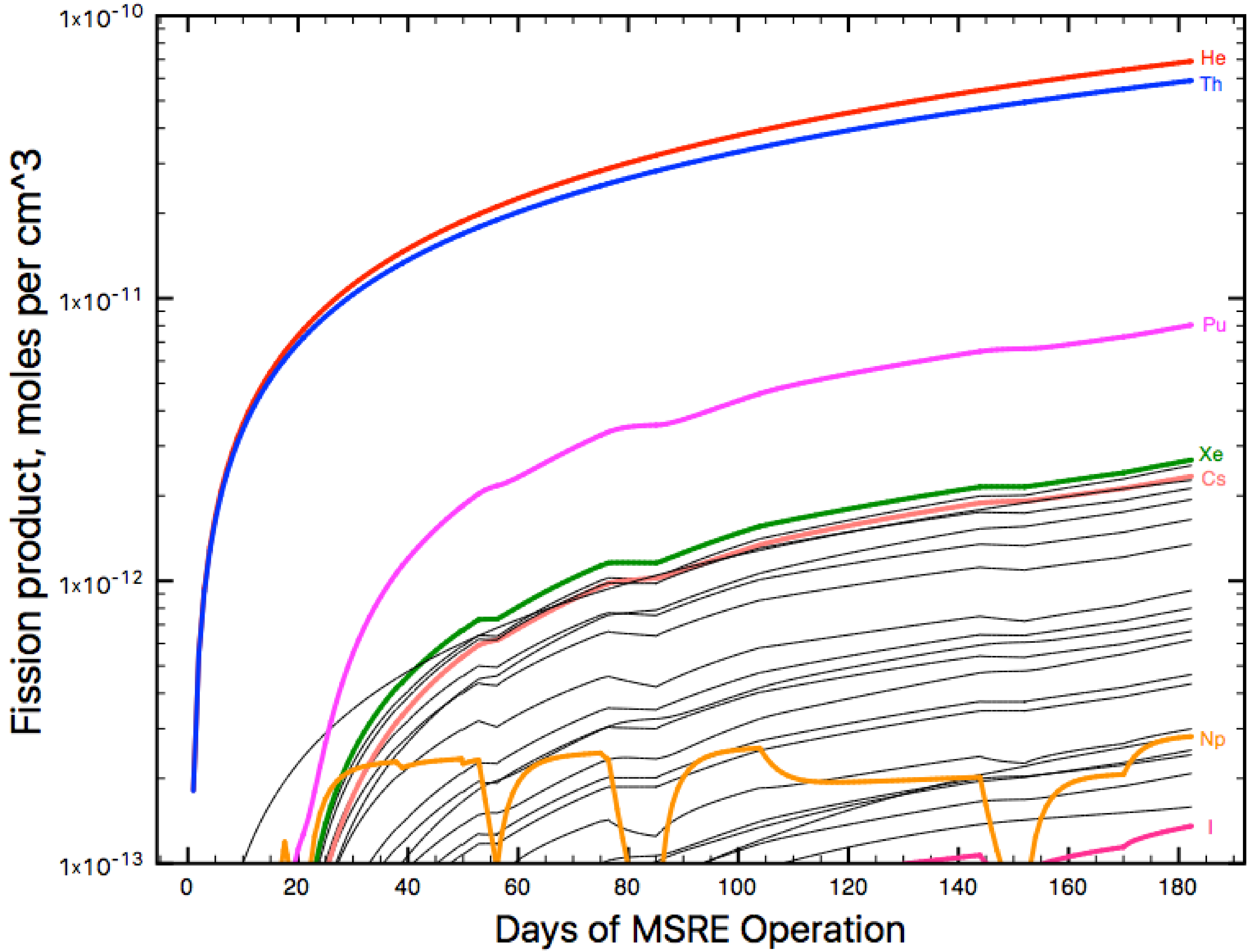
exNEAMS.cpp - Test2

an Introductory Example


Other files to include
Create a library
Extract a transition matrix from it
Initialize a material
Create vectors for time and flux/power
Adjust temperature based on power
Solve the decay problem for the material
Combine isotopes into elements for Thermochimica
Run Thermochimica
Look at the results!

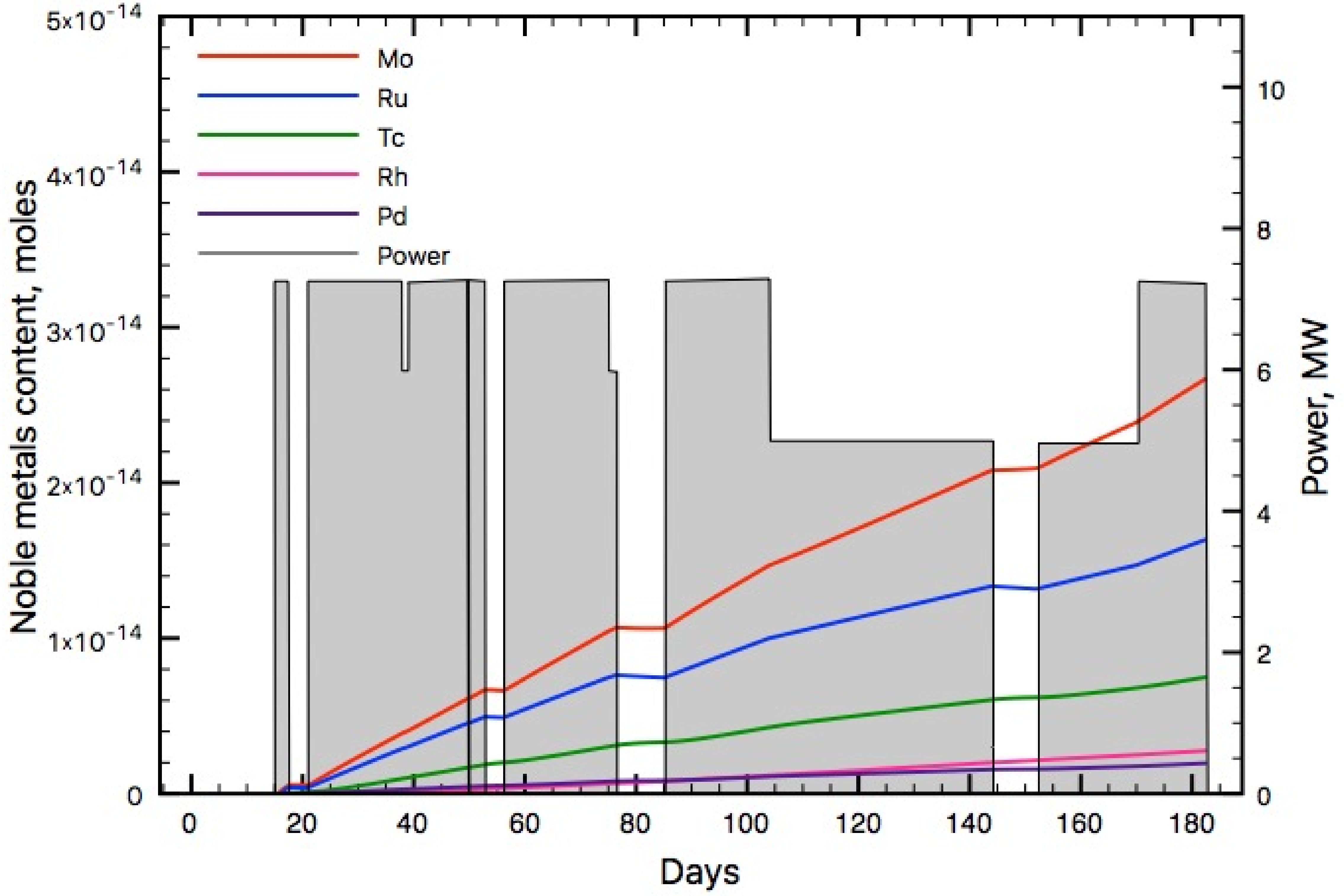
Results

Fission Product Formation During Second Campaign of MSRE



Results

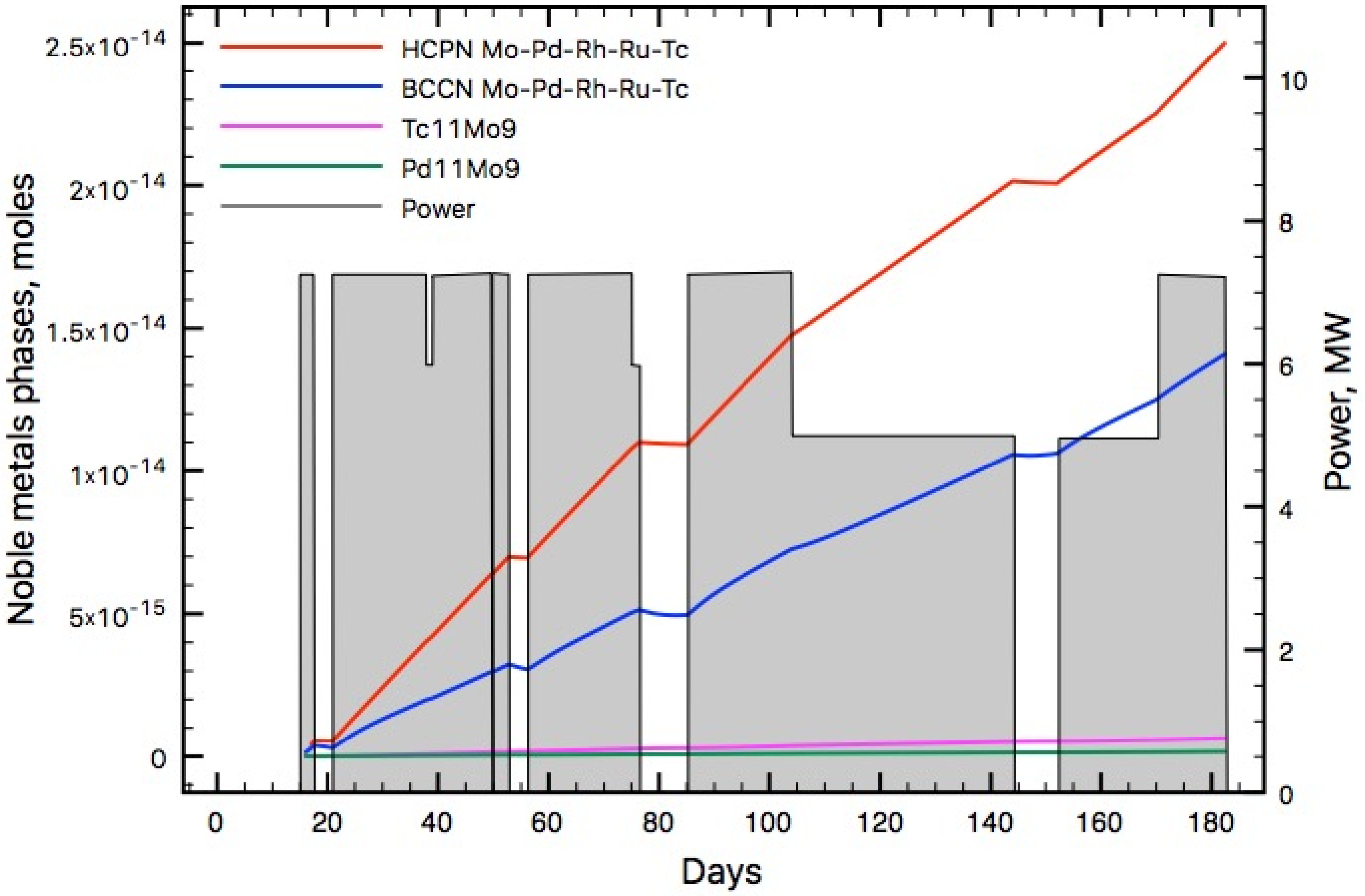
Noble metals formation from ORIGEN-API, for MSRE



Power history taken from: M. Rosenthal, R. Briggs, P. Haubenreich, al. Molten salt reactor program semiannual progress report for period ending august 31, ORNL-4622. USA: Oak Ridge National Laboratory, 1970: 38-41, (1968)

Results

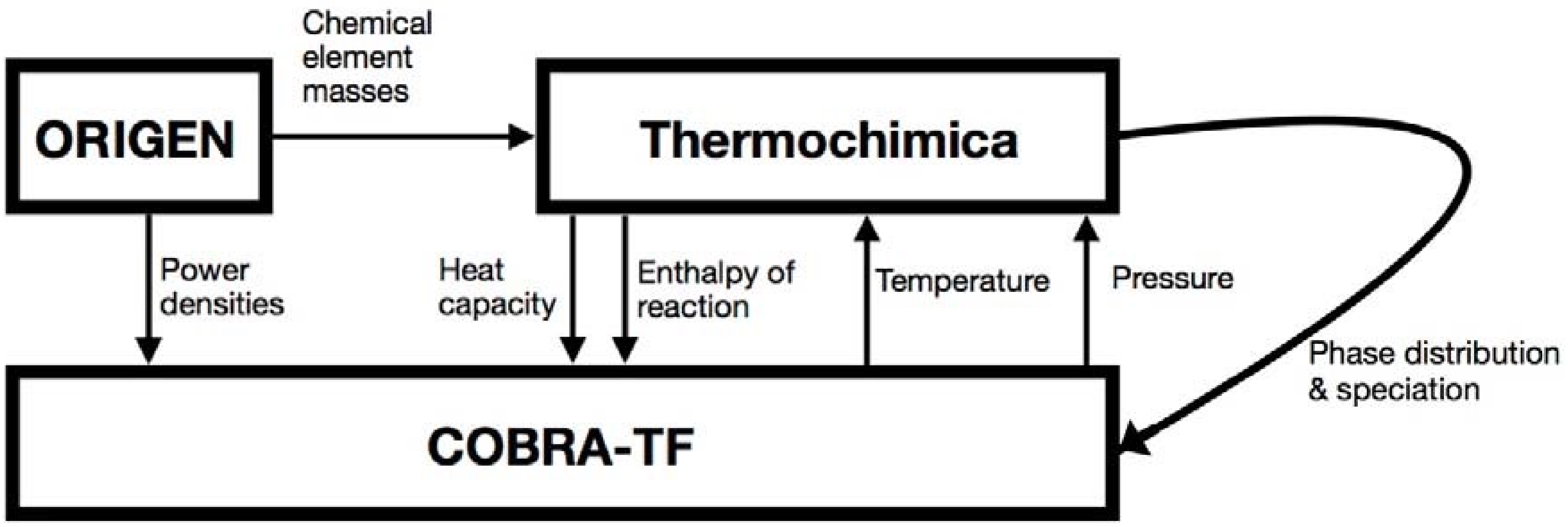
Phase evolution of noble metals formation from ORIGEN-API / Thermochimica coupling, for MSRE



Power history taken from: M. Rosenthal, R. Briggs, P. Haubenreich, al. Molten salt reactor program semiannual progress report for period ending august 31, ORNL-4622. USA: Oak Ridge National Laboratory, 1970: 38-41, (1968)

Future Work

Incorporating thermal-hydralics for temperature, pressure, and transport



B.W.N. Fitzpatrick et al., Proceedings of the ANS Annual Meeting, Philadelphia, 2018.

Future Work

Expanding the thermodynamics database through experimental work

Netzsch Jupiter STA 449 F1: DSC + TGA for thermodynamic measurements on salts

Glovebox for fabricating salts

Future Work

Viscometer?

Expanding the thermodynamics database through experimental work

Netzsch Jupiter STA 449 F1: DSC + TGA for thermodynamic measurements on salts

Glovebox for fabricating salts

Acknowledgements

- This research was undertaken, in part, thanks to the funding from the Canada Research Chairs (950-231328) and Discovery Grant programs of the Natural Sciences and Engineering Research Council of Canada.

- Special thanks to Ben Collins, Robert Salko, Robert Taylor, Jake McMurray, Ted Besmann, Z. Taylor for all the useful discussions