IAEA
International Atomic Energy Agency

HCP Tutorial 1 - Basics

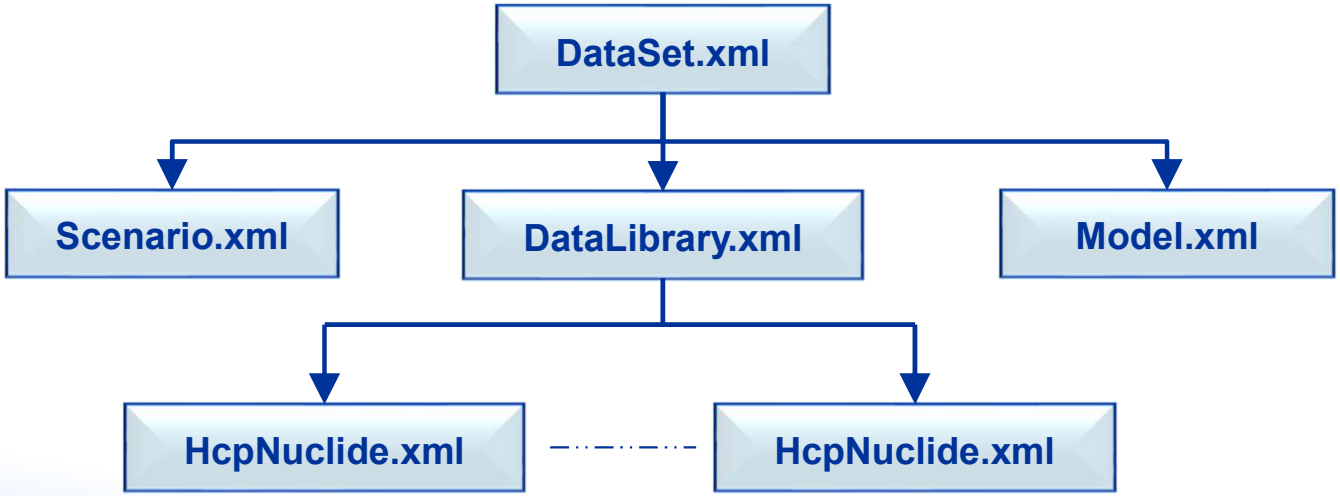
Dr.-Ing. André Xhonneux

Joint IAEA-ICTP Workshop on Reactor Physics, Thermal Hydraulics and Plant Design Engineering of Small Modular Reactors

13 April 2025, International Centre for Theoretical Physics (ICTP), Trieste

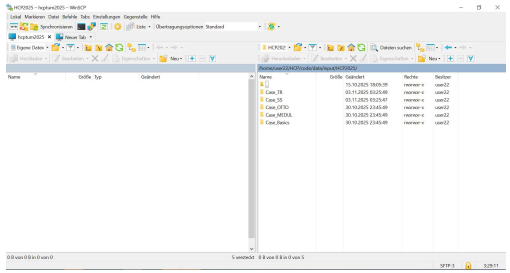
HCP Input Concept - Overview of Files

- **DataSet.xml** Master file which connects all other files
- **Model.xml** Geometry, materials, fuel elements, …
- **Scenario.xml** Time control, time-dependent data, settings
- **DataLibrary.xml** Nuclear database for all code modules
- **HcpNuclide.xml** Nuclide-specific data, one file per nuclide



Making use of Tools to work with HCP

- **WinSCP is used to transfer data between your local machine and the server.**
- **Run the code via the console using Putty (if you are not running the code locally).**
- **You can view/adapter the files in multiple ways**
  - Open file in ***WinSCP*** (no colorful highlighting)
  - Open file in console (***Putty)*** using ***nano***, ***pico***, ***vi***, … (limited colorful highlighting)
  - Create a local copy on your machine, and open with e.g. ***Notepad++*** (better highlighting)
- **WinMerge is a proper tool to compare files** (e.g. to compare your solution vs. the reference solution)



Running the code

1) **During the ICTP workshop in Trieste:**

   runproto $INPUT/CASE/DataSet.xml

1) **On your own Linux machine (e.g. after the workshop):**

   - Navigate to: **HCP/code/*bin/Linux***
   - *Type:* ***./proto relative_path_to_DataSet.xml***

     For instance: ***./proto ./../data/input/HCP2026/Case_Basics/Task1/DataSet.xml***

   This way allows also to easy execute the cleanup command in between two runs.

Use of Cleanup Command

- A part of the output is stored where the binary is located **HCP/code/bin**.

- These can be erased all at once by running **cleanup.sh** (also located in that folder), by typing in the console:

  ./cleanup.sh

Known Bug

HCP might crash with the message:

At line 53 of file /home/ntecher/HCP/code/src/modules/mgt/mgtc/fortran/PNDWriter.f90 (unit = 99, file = '../../data/input/HCP2026/Case_XX/Task_YY/data.pnd')

Fortran runtime error: End of file

In that case, delete the pnd-file in the folder specified in the error message and run the code again.

(the failure is caused by the fact that if this file already exists from an earlier run of HCP, it is not able to overwrite)

XSD-Schemes

- **Since there is no user input manual yet, the xsd-schemes might be of additional help, see:**

  HCP/code/data/xmlschemes

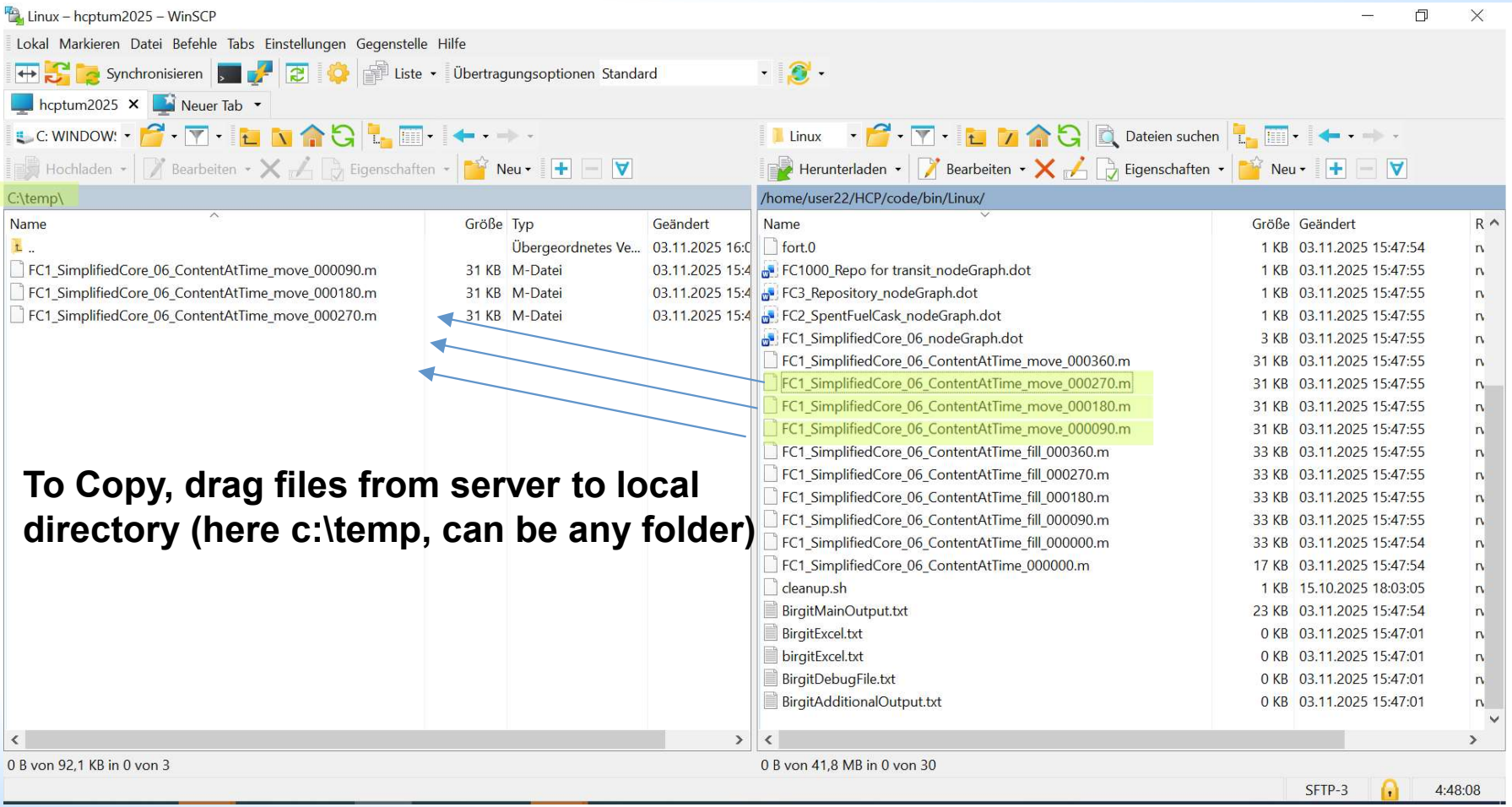
Valuation using Octave - Installation

- **Install Octave on your local machine (for visualization).**
  You might need administration rights for this!

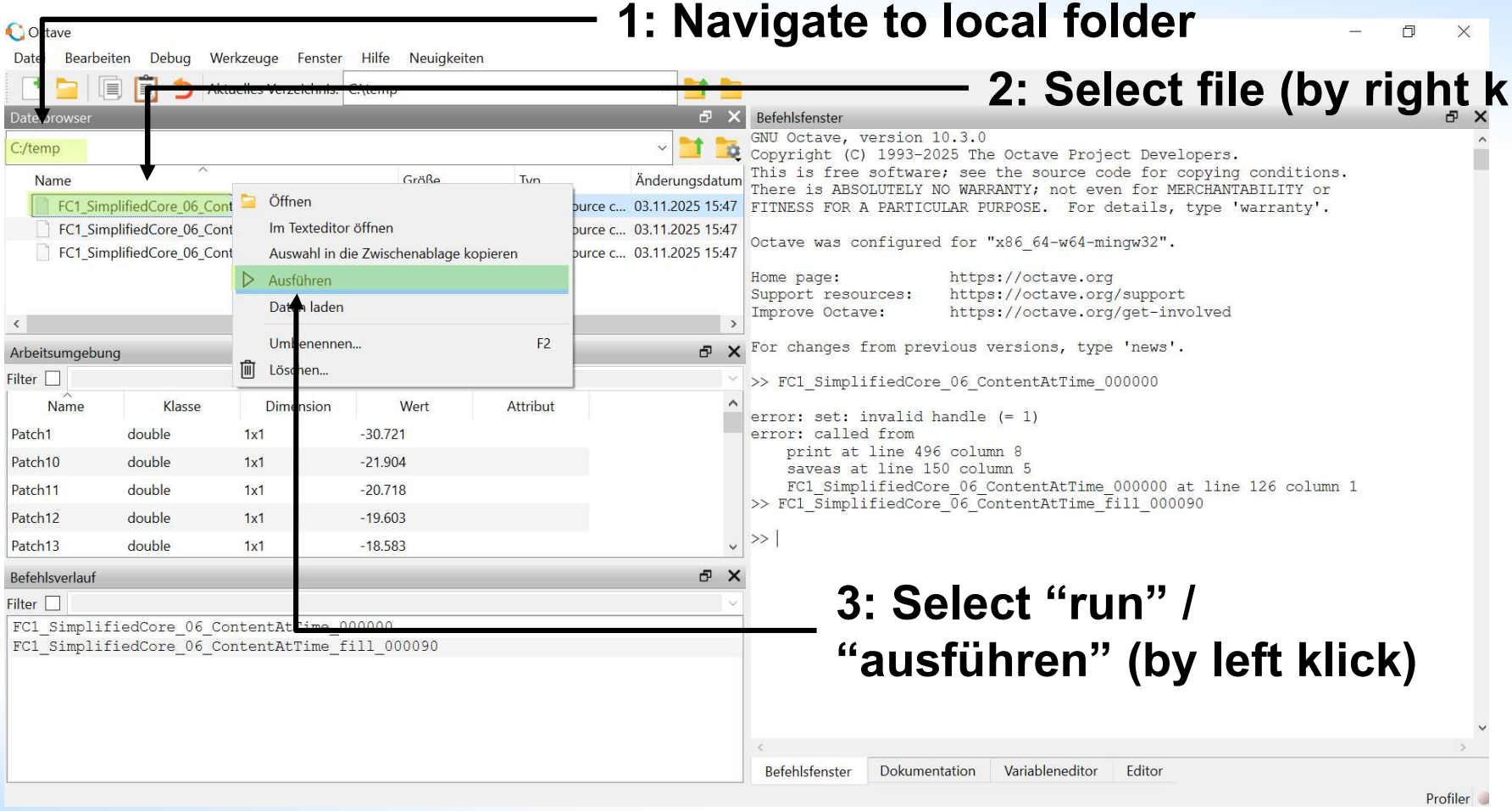
  https://octave.org/download

- **If you have Matlab installed, there is no need to install Octave.**
  You can use Matlab instead of Octave.

Valuation using Octave – Copy Files



Valuation using Octave – Run files



Tutorial 1: Preparation

In most Subfolders, there are two datasets:

1) DataSet.xml

- Model.xml
- Scenario.xml
- ../../../../../libs/Linux/endfb70_regtest_new.bin

2) DataSet_Solution.xml → This file

- Model_Soltution.xml (reference solution)
- Scenario_Solution.xml (reference solution)
- ../../../../../libs/Linux/endfb70_regtest_new.bin (same)

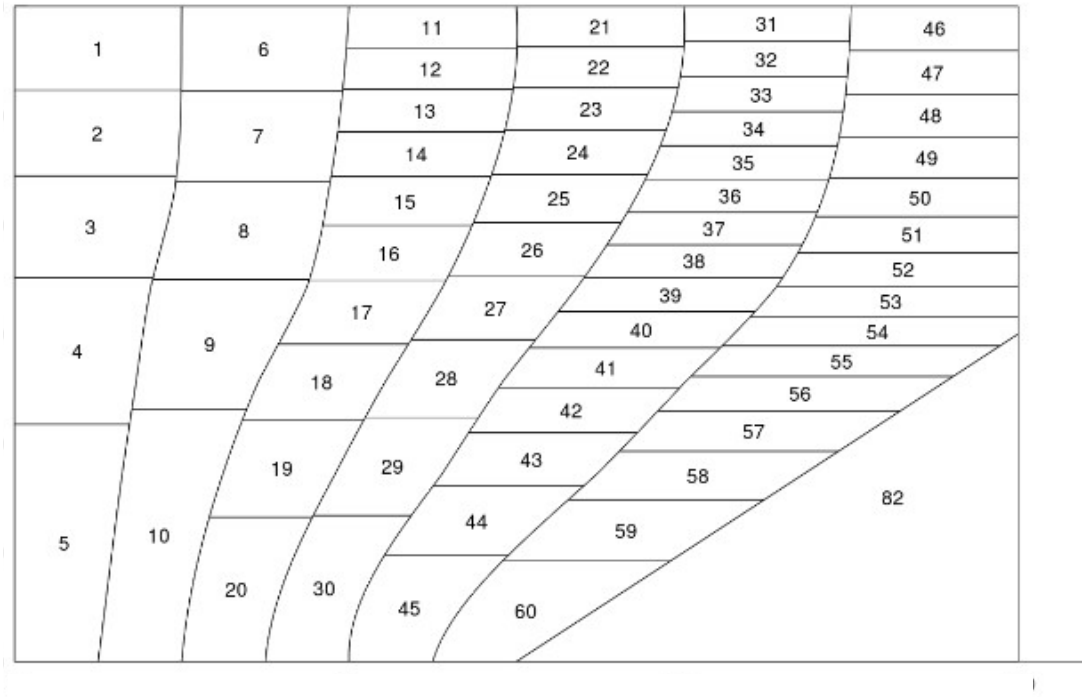
Either the Model.xml and/or the Scenario.xml has to be extended/modified to fullfill the task. Running the code while using the DataSet_Solution.xml should always work!

Tutorial 1: Learning Outcomes

- Basic understanding of xml files in general
- Basic understanding of Model.xml structure
- Setting up a coated particle
- Setting up a fuel element

Tutorial 1: Learning Outcomes

- Setting up dedicated node grid for fuel shuffling for a pebble-bed reactor (Core-Model with Cone and **O**nce **T**hrough **T**hen **O**ut (OTTO) / single core pass fuel shuffling)
- Setup initial core filling
- Setup shuffling instructions



Copy files

- In **putty**, please type:

scp -r /home/SharedFolder/Case_Basics/ /home/**userXX**/HCP/code/data/input/HCP2026/

were **userXX** ist your username, e.g. **user01**

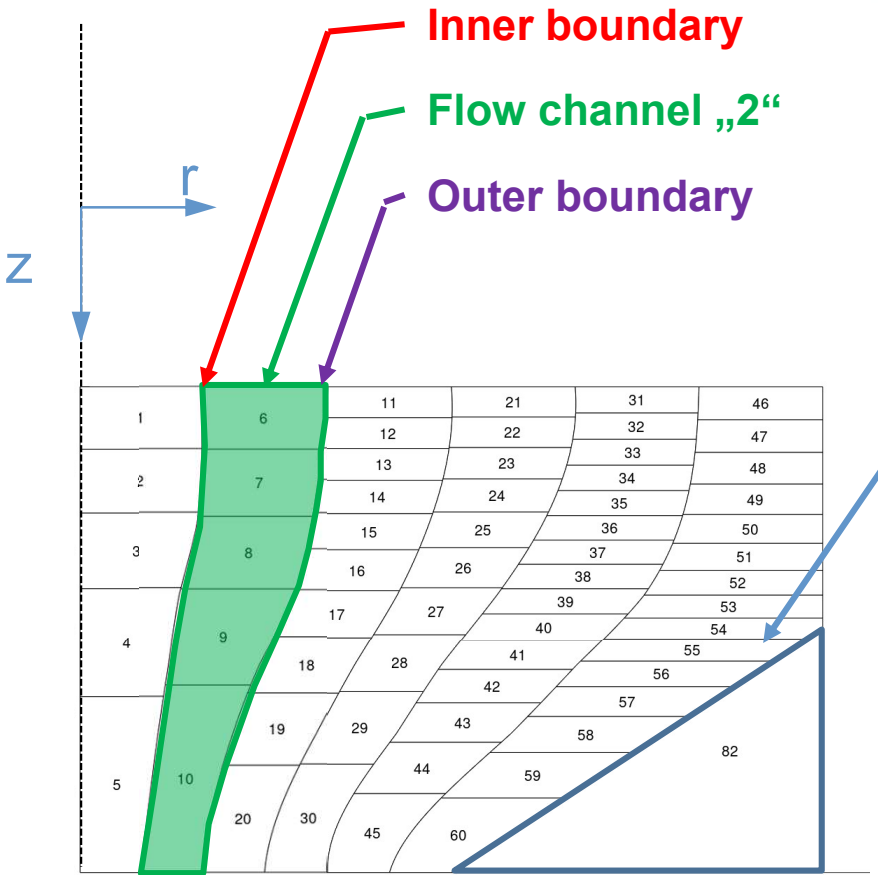
Task 1

- Navigate toHCP/code/***data/input/HCP2026/Case_Basics/Task1*** and open **Model.xml**
- Check within the **Model.xml** for the definition of the **CoatedParticle** and how it‘s linked to **GeoObjects**
- As you can see, the model does not contain a full TRISO particle yet (some layers are missing).
  Add the missing pieces to the input file to make a full TRISO by using the following data:

	„Dimension“	Material
Kernel	**Diameter** = 500 µm	UO2
1st layer	Layer thickness = 90 µm	Buffer
2nd layer	Layer thickness = 40 µm	PyC
3rd layer	Layer thickness = 35 µm	SiC
4th layer	Layer thickness = 40 µm	PyC

- Add the **attribute** fuelZone to the definition of the kernel and set it to “true”
- Check the coated particle composition and set the number of particles to 15000
- Check whether you can run the code.

Introduction NodeGrid Definition

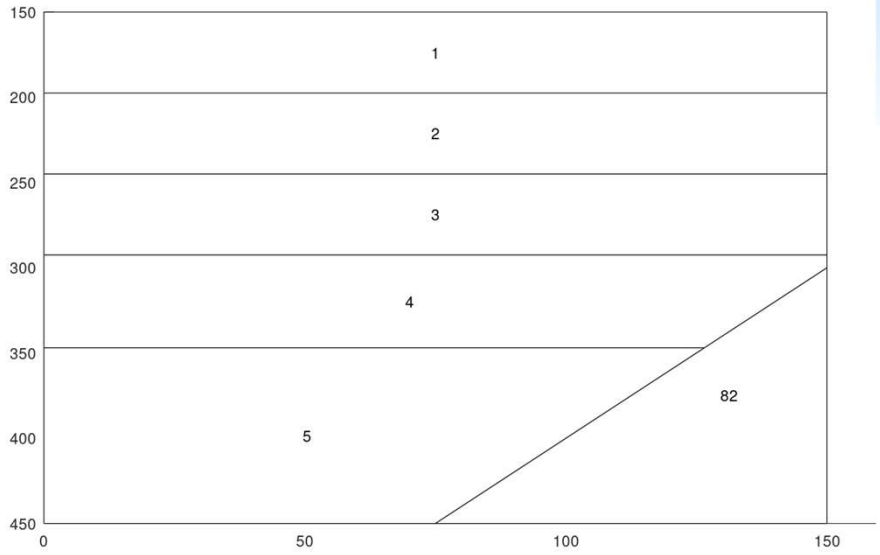


- z-Axis is always pointing downwards
- Flow channels are defined from the symmetrical axis outwards towards the reflector
- Even a **cone** is a flow channel (although not part of the pebble bed and content is non-movable)
- Each **flow channel** is limited by two flow lines
  - 1st FlowLine id: inner boundary
  - 2nd FlowLine id: outer boundary
- **FlowLines** are defined by multiple points, from top to bottom, where “r” and “z” coordinates are separated:

Task 2: Exercise

Introduction

- In the initial model provided for tutorial 5, the core contains the pebble-bed and an outer core.
- The pebble-bed is **only sub-divided in one flow channel**. This channel contains **5 nodes**, with **NodeIds = 1, 2, 3, 4 and 5** (see figure).



Task 1:

Please navigate to

HCP/code/data/input/HCP2026/Case_Basics/Task2
open the file ***model.xml*** and retrieve from the definition of this FlowChannel the **radius of the pebble bed.**
(you can find the answer on next slide).

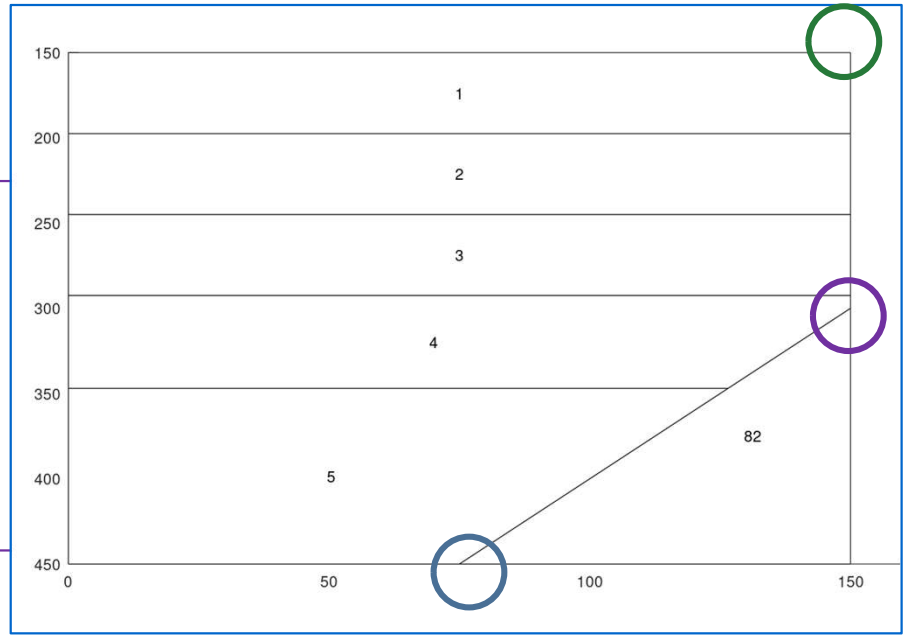
Task 2: Answer

The FlowChannel is limited in radial direction by FlowLine 1 and FlowLine 4, where FlowLine 1 defines the inner boundary and FlowLine 4 the outer boundary:

```
<FlowChannel id="1" flowLine="1 4" nNodes="5" pebbleBed="true">
  <Node id="1-5" fillingFactor="0.61"/>
</FlowChannel>
```

FlowLine 4:

```
<FlowLine id="4" spline="false" pebbleBedBoundary="true">
  <PointSet>
    <R n="3" unit="cm">
      150.0 150.0 75.0
    </R>
    <Z n="3" unit="cm">
      0.0 150.0 300.0
    </Z>
  </PointSet>
</FlowLine>
```



Answer: Radius of the pebble-bed = 150 cm

Task 3: Sub-divide core in Channels

Within this tutorial, we prepare the model for the simulation of the fuel shuffling.
First, we sub-divide the pebble-bed in multiple FlowChannels.

1. Please **navigate** to HCP/code/***data/input/HCP2026/Case_Basics/Task3***
   and **open** the file ***model.xml***

2. Besides FlowLine 1 and 7 (which you have seen during task 1), other flow lines are already given in the Model.xml (but not used so far).
   Please **check** the definition of these other flow lines.

3. Please **adjust** the outer flow line of **FlowChannel 1** (check section FlowLine="**1 4**")
   by referring here to the flow line (already defined), which starts at radius r = 50 cm (instead of flow line 4, which starts at radius = 150 cm) for the outer limiting flowline of the first channel).

Task 3: Sub-divide core in Channels

4. **Add two additional flowChannels in the pebble-bed by doing the following:**
   - First, copy the definition of the already existing pebble-bed flow channel to form the basis for the other two flow channels and paste them directly after the first flow channel.
   - Give the newly created flow channels **id = 2 respectively 3** (ids need to be unique!)
   - Set the “reference” to the inner and outer flow lines for the newly created flow channels, making use of the flow lines already provided. For FlowChannel with id 2 to “2 3”, for FlowChannel with id 3 to “3 4”
   - Please define ***unique Node-ids*** for the nodes in both newly created flow channels.
     - Flow channel 2 should contain 10 nodes: **NodeId 6, 7, … 15**
     - Flow channel 3 should contain 15 nodes: **NodeId 16, 17, … 30**

Task 3: Sub-divide core in Channels

- Please keep in mind, that the newly defined nodes have to be ***filled*** with fuel. Check the section **<Nodes>** and adapt this section in such a way, that also newly created nodes **6 … 30** are filled with fuel.

Task 3: Sub-divide core in Channels

5. Please go to in the console (Putty) and run the HCP-Proto for this task (see slide 3 for options).

6. If everything was done correctly, the pre-processing is executed and after that HCP is ended (you see “**run finished on ….”** in Console).

7. Visualize node grid by:
   - copying **vsopGrid.m** from **HCP/code/bin/Linux** to a folder on your local machine (see slide before for more explanations).
   - run **vsopGrid.m** in Octave.

Task 3: Solution Part 1 (Model.xml)

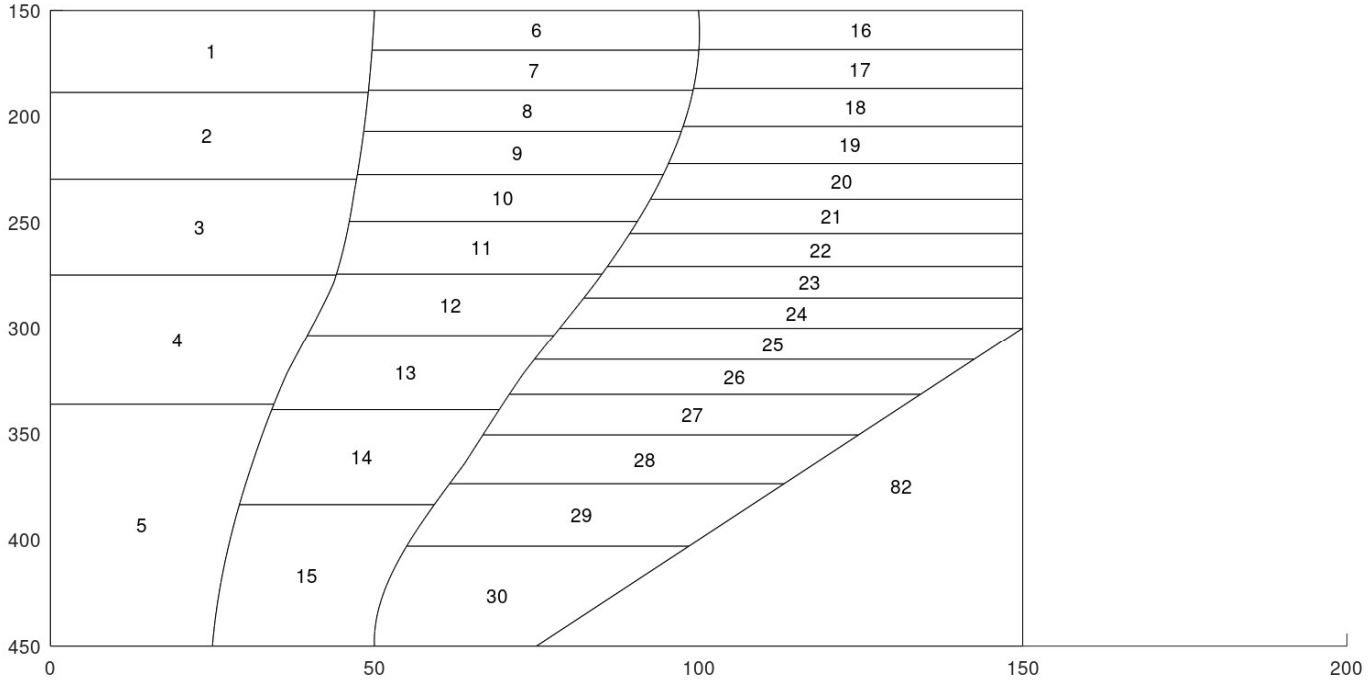
```
<FlowChannelSets n="1">
  <FlowChannelSet id="1" phi="0 360.0" unit="deg">

    <FlowChannel id="1" flowLine="1 2" nNodes="5" pebbleBed="true">
      <Node id="1-5" fillingFactor="0.61"/>
    </FlowChannel>

    <FlowChannel id="2" flowLine="2 3" nNodes="10" pebbleBed="true">
      <Node id="6-15" fillingFactor="0.61"/>
    </FlowChannel>

    <FlowChannel id="3" flowLine="3 4" nNodes="15" pebbleBed="true">
      <Node id="16-30" fillingFactor="0.61"/>
    </FlowChannel>

    <FlowChannel id="4" flowLine="4 5" nNodes="1" pebbleBed="false">
       <Node id="82" fillingFactor="1.0"/>
    </FlowChannel>

  </FlowChannelSet>
</FlowChannelSets>
```

Task 3: Solution Part 2 (Model.xml)

```
<Nodes>
  <!--
  Core nodes are filled by an action in the scenario, so fill="auto"
        For consistency all node ID must be given here which shall
  -->
  <Node id="1-30" fill="auto"/>
```

Task 3: Solution Part 2

(Visualized by Octave)



Task 3 cont‘d (no reference solution)

- So far, the filling factor of all nodes being part of the pebble-bed is equal to 0,61.

- Try to change the filling factor for the nodes of the innermost fuel channel in such a way, that the nodes of these FlowChannel have each their unique filling factor.

  Example:
  <Node id="1-5" fillingFactor=“0.2 0.3 0.4 0.5 0.6"/>

- Check the effect on the NodeGrid by visualizing it.

Task 4: Activate refilling during Shuffling

In this case, we start adding fuel to the core

1. Please navigate to HCP/code/***data/input/HCP2026/Case_Basics/Task4*** and open the file ***Scenario.xml***
2. Search/look for to **<Settings type="FuelManagement">**
3. Please check**“<Setting type="Scheme">** and try to understand the scheme being defined.

Task 4: Activate refilling during Shuffling

5. run HCP for task 4 (see slide 3 for options).
   (the execution of the code will end with Number of batch fractions is equal to zero! This is in this case OK!)
6. Navigate toHCP/code/**bin/Linux** to locate the files containing m-files (core contens / batch visualisation)
7. **Move the following files to your local machine**
   - **FC1_SimplifiedCore_06_ContentAtTime_move_000090.m**
   - **FC1_SimplifiedCore_06_ContentAtTime_move_000180.m**
   - **FC1_SimplifiedCore_06_ContentAtTime_move_000270.m**
8. Visualize grids with **Octave by moving active folder in Octave** to the local folder you have copied the files to in the last step.
9. After that, run the following files in Octave (one after another):
   - **FC1_SimplifiedCore_06_ContentAtTime_move_000090.m**
   - **FC1_SimplifiedCore_06_ContentAtTime_move_000180.m**
   - **FC1_SimplifiedCore_06_ContentAtTime_move_000270.m**
10. **Check what happens (the core should be drained stepwise)**

Task 4: Activate refilling during Shuffling

11. Go to “<Settings type="FuelManagement">”

12. Please activate in “**The fill section**” the following section (so far commented out, thus not executed)

```
<Fill fuelCollection="1">
  <Level from="450.0" to="150.0" unit="cm"/>
  <FuelComposition n="1">
    <Fraction value="1.00" unit="norm1" fuelElement="1"/>
  </FuelComposition>
</Fill>
```

Task 4: Activate refilling during Shuffling

13. Repeat the HCP run.

14. After that, run the following files in Octave:

    - **FC1_SimplifiedCore_06_ContentAtTime_move_000090.m**
    - **FC1_SimplifiedCore_06_ContentAtTime_fill_000090.m**
    - **FC1_SimplifiedCore_06_ContentAtTime_move_000180.m**
    - **FC1_SimplifiedCore_06_ContentAtTime_fill_000180.m**

15. **And check what happens:**
    - **you can see the two steps, first discharge (move), then “fill” for each Action**
    - **the filling level should stay constant after the “fill”**