PENTrack and Cryogenic Simulation Procedures

Sean Lan, Ruediger Picker rpicker@triumf.ca

May 8, 2025

Abstract

PENTrack is a Monte Carlo simulation software that we run on remote Compute Canada servers for simulating UCN lifetimes in theoretical configurations. The process of preparing an initial SolidWorks file, submitting jobs remotely, and analyzing the outputs can be rather involved and lengthy. This document explains the general process, some of S.Sidhu's workflow, and some of the author's workflow aimed at simplifying the process. This document also records everything the author knows about working with the simulations and should be provided to co-op students working with these simulations for the first time, even if they do not use S.Sidhu's workflow. Note that the author is neither the maintainer of PENTrack nor the analysis scripts, who are instead W. Schreyer and S. Sidhu, respectively.

Contents

Revision History

- **April 27, 2020** - Initial version by Sean Lan
- **May 8, 2020** - Adapting .bashrc to make PENTrack compile on Cedar
- **May 21, 2020** - Added some useful tools for Windows and how to use Compute Canada Resource Allocation
- **May 27, 2021** - Added some clarifications which parts apply to PENTrack in general and which to the nEDM workflow
- **December 13, 2023** - S. Sidhu added appendix and instructions on how to get the hepak-tools package working on a Windows computer. Also added a section on calculating material properties for the TUCAN source and EDM experiment.
- **May 5, 2025** - K. Drury added an example job script in Section 4.3.3, updated Section 4.4, added Section 4.5, and Section 13 on using a Toy Model for learning how to use PENTrack.

GitHub Links

- **nEDMsensitivity**
- **PENTrack**
- **Kyle's Thesis and Toy Model**

1 Overview

Our goal in using PENTrack is to simulate different conditions for the ultracold neutron (UCN) and neutron electric dipole moment (nEDM) to optimize our physical setup to maximize experimental sensitivity. We can also use these simulations to verify our experiments in reality; it's always good to compare reality to theory.

This document is aimed solely at the practical process and technical details of carrying out simulations, but it does not cover any of the theory. Some portions of this document apply to all PENTrack simulations, but some portions apply only to S.Sidhu's workflow, which is what the author followed. You can find the design notes for the theory behind the simulation and analysis Where?. If nothing else, you can ask S. Sidhu for the design notes.

2 Summary of Process

The general pipeline for simulating a scenario from start to finish following S. Sidhu's workflow is as follows. Sections in blue apply only to this specific workflow, everything else is common to all PENTrack simulations.

1. Prepare a SolidWorks model of the simulation environment.
2. Generate STL files of the model in a few different configurations to represent the different steps in the experiment of Filling, Storing, and Emptying.
3. Upload the STL files to the remote servers and process them to remove all spaces from the filenames.
4. Set up the configuration and material files for PENTrack. The configurations will be different for each step of Filling, Storing, and Emptying.
5. Send in a job to the Compute Canada server to run PENTrack and perform simulations using the STL files.

6. Run a script that merges the *.out output files into an out.root file, which can be processed by CERN's ROOT libraries. Alternatively, you can configure PENTrack to print all output directly to ROOT files.

7. Verify that the experiment was properly set up by running some checks on this out.root file for any immediate errors.

8. Further process the Storing, Emptying, and Filling out.root files with some analysis scripts.

9. Run the another script daystoreach.py which takes in the Filling out.root as well as the processed Storing and Emptying files to produce your final data and some graphs.

Many of these steps have been automated with scripts. The later sections will go into further detail for each step.

While the previous list is specific to S.Sidhu's analysis workflow (indicated in blue), this document also covers how to use PENTrack for general use. At its core, PENTrack is the software doing the actual simulation. Steps 8 and 9 are analysis steps and not specific to PENTrack. Separating the experiment into several stages of Filling Storing and Emptying is also not necessary, but is used to improve S.Sidhu's analysis.

3 Software and workflow setup

The author prefers to use Windows OS for everything, although the established workflows are for Linux OS. The main benefit of Windows is the ability to run SolidWorks and work on the Compute Canada servers on the same PC, as well as user familiarity. One cannot run SolidWorks on Linux, but the setup for everything else is much easier. To use Linux, at minimum dual boot is required; one with Windows to work with SolidWorks, and another to use the Linux terminal.

This software and workflow section has portions that are relevant for Windows only as well as portions that are relevant for both Windows and Linux.

3.1 Windows Exclusive

The main difficulty in working with Windows is the lack of Linux terminal and X11 forwarding.

The author used Windows 7 due to availability, but if you have access to Windows 10 you can install a Linux terminal which is cleaner than Git. To install a Linux subsystem in Windows 10, you can follow this guide.

If you're stuck using Windows 7 like me, or even worse Windows 8, you need to install a Linux terminal to SSH into the remote server. I suggest using Git for Windows because it also comes with Git, which is useful.

However, Windows lacks a critical feature; X11 support. This is necessary to display graphs and histograms from the remote server. You can follow this tutorial to install XLaunch(vcxsrv) which handles the X11 forwarding. The author has also tried using Cygwin/X and found that it works, although the setup time is far longer. When opening XLaunch, make sure to check "Disable Access Control." Once installed and running, if you run `$ module load root` and `$ root` from the remote server, the splash should appear and once you hover over the XLaunch icon in the taskbar it should say "1 client". This will only work on a login node and not a compute node until properly configured, as will be covered later.

As a Windows plebian, we like our user interfaces, so I installed WinSCP which allows for UI based navigation, file management, and file editing of the remote server. It's a lot more intuitive than using a terminal.

An alternative is MobaXterm. It combines SSH, file access through SFTP, and an X server in one convenient tool.

Along with WinSCP you will want to install a nicer text editor than the Windows Notepad. The author used Sublime Text 3, although Visual Studio Code or Atom or anything other than Notepad will also improve your experience.

3.2 Windows and Linux

Regardless of your OS of choice, you will need to install SolidWorks on Windows. SolidWorks should already be on the Windows computers within the UCN team. SolidWorks is the Computer-Assisted Design (CAD) software of choice in industry and you should already know how to use it if you're replacing me. If not, go figure it out. Watch videos on YouTube or something.

4 Getting Started with Compute Canada

You will need to open a Compute Canada account under the UCN group. Ask Ruediger Picker for further details because he will need to sponsor you to open the account.

In addition to the information found in this document, documentation is also maintained online.

4.1 Connecting to Cedar

Once you have your Linux terminal or equivalent open, you can SSH into Compute Canada, specifically the Cedar servers with:

```
$ ssh -Y <account_name>@cedar.computecanada.ca
```

For example, in place of `<account_name>` the author would use:

```
$ ssh -Y slan@cedar.computecanada.ca
```

You would then enter the password you used for your Compute Canada account. This account can be used for all Compute Canada computing servers, although the author only used Cedar.

The argument `-Y` is necessary to enable the aforementioned X11 forwarding and allow the graphs and histograms from ROOT to properly display. If you are finding `-Y` does not work, you can try `-X` instead, although I don't actually know the difference.

4.2 Familiarizing yourself with Cedar's File System

Cedar has two main file systems, project and scratch. The project filesystem is where you would install local software and store any final results from the experiment for long term safekeeping. The scratch filesystem is for generating and logging data temporarily, because Compute Canada will purge files on the scratch drive every 60 days. There is 1000GB of storage in the project drive and 20TB of storage in the scratch drive, which you can check at any time using the `quota` command.

4.3 Compute Canada's Job and Node System

4.3.1 Nodes

When you first SSH into Cedar, you will be on a login node. This is different from a compute node which is also different from an interactive compute node.

- Login nodes are lightweight and will kill your processes if you run anything too CPU or memory intensive on them, but they are always there to handle your SSH connection. You will connect to Compute Canada and start on a login node. If on Windows, Putty is probably the go-to SSH terminal: https://docs.computecanada.ca/wiki/Connecting_with_PuTTY, VcXsrv works well on Windows 10.
- Compute nodes do the bulk of the work in Cedar but cannot be directly accessed. Instead, you need to submit a job into the job scheduler and wait until the compute node is open and processes your batch file. Furthermore, since you cannot access these compute nodes, generally the job you submit should not require any user input and should run by itself start to finish.
- Interactive compute nodes have the power of a compute node with the interactivity of a login node but also require you to wait for an allocation before you can connect to them.

- The job scheduler will allocate compute nodes based on your priority score, which is determined by the amount of jobs you have submitted and your account privileges. Recently, we applied for a Resource Allocation for the whole UCN group, which should give our accounts higher priority. You can use it by replacing `--account=def-rpicker` with `--account=rrg-rpicker` in the commands used below.

You can request an interactive compute node with the `salloc` command followed by a whole bunch of additional parameters which are difficult to remember. An example usage would be:

```
$ salloc --time=2:0:0 --ntasks=1 --account=def-rpicker --mem=4G --x11
```

This command would request 1 task on the "def-rpicker" account, an allocation of 4G of memory on an interactive compute node for 2 hours, enable X11 forwarding for all nodes (so you can run graphical applications). If you exceed the memory limit running some process, the process (not the allocation) will be killed. You will also be kicked off the interactive compute node once the time limit is reached. This command is used frequently but the parameters are so difficult to remember so it is best to create an alias for it in your `.bashrc` which we will do later.

You *must* have `--x11` in order for you to use X11 forwarding on an interactive compute node. That is, if you do not have this option enabled, even if you have an X11 server running on Windows, you will not be able to interact with ROOT and other ui on the interactive compute node. This was a great pain for me until I found out this option existed. This is important.

4.3.2 Jobs

To use the majority of Compute Canada's processing power you must submit a job into a queue and wait until compute nodes are available to process your request. The official documentation may explain this better than me. This is done by creating a bash script that the compute nodes can execute and submitting the bash script with `sbatch`:

```
$ sbatch doSomething.sh
```

Within this `doSomething.sh` bash script (the job script, in thsi example) there needs to be a few additional lines so the job scheduler Slurm knows your settings for the script. A simple bash script that runs PENTrack with some additional parameters for the job scheduler is as follows:

```
#!/bin/bash
#SBATCH --time=00:15:00
#SBATCH --account=def-rpicker
echo 'Hello, world!'
sleep 30
```

You can submit this script and use the `sq` command to check the progress of all your jobs in the job scheduler. Because these files are executed remotely, instead of `echo` or other outputs printing to your terminal, it gets output into a Slurm output file (typically ending in `.out` unless reconfigured). These `.out` files are buffered and are typically only produced once your job is finished running.

All the lines beginning in `#` are commands that the job scheduler can read, so be careful with your bash scripts because `#` is also used for comments. The job scheduler will only check for parameters until it reaches executable code, however, so all comments past the start are safe. You can also just use `##` for comments.

Thus to construct a bash script for Slurm it must begin with `#!/bin/sh` to indicate to the system where to find the bash shell. The following lines with `#SBATCH` are parameters passed to the job scheduler. These two parameters `time` and `account` are the minimum that Slurm requires to schedule a job. Instead of being included in the file the settings could have also been given as command-line arguments to `sbatch` as in:

```
$ sbatch --time=00:15:00 --account=def-rpicker doSomething.sh
```

4.3.3 Example Job Script

Below is a batch script I wrote by making adding some additional code to "`slurm.sh`," found in GitHub PENTrack repository.

```
#!/bin/bash

# how to run PENTrack on a computing cluster using the SLURM task scheduler

#SBATCH --array=1-20   # run multiple simulations in parallel
#SBATCH --time=4:00:00 # set maximum time simulations are allowed to run)
#SBATCH --mem=4000M                    # reserve additional memory
#SBATCH --account=def-rpicker          # set an account

# combine Job ID and array task ID to a unique ID that is passed to PENTrack
ID=$SLURM_ARRAY_TASK_ID  # Slurm array task index
JOB=$SLURM_ARRAY_JOB_ID  # Slurm job ID

mkdir ./out/$JOB  # make a dir in out folder named after the job

# run PENTrack with the created ID, the default config, and the default output directory
./PENTrack $JOB$ID ./in/config.in ./out/$JOB

mkdir ./out-slurm/$JOB      # make a dir to put the slurm-out files in once the job is done
mv *.out ./out-slurm/$JOB   # move the slurm out files into the dir
```

Assuming you have a directory in your workspace called "`/slurm-out`" (I created one), the terminal output will be saved in files stores in "`/slurm-out`," and the actual data will be stored in "`/out`." The subdirectory containing the actual files is named after the job number.

4.4 Configuring your environment

We will now edit the `.bashrc` file to automatically load certain modules upon logging in and to make our life easier in other ways. The `.bashrc` file is found in your home directory, which you can reach by using `cd` with no arguments or by navigating in WinSCP. Here is my `.bashrc` file as an example and we will cover the purpose of each section:

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
. /etc/bashrc
fi

# User specific aliases and functions
alias gohome="cd ~/projects/def-rpicker/kdrury"
alias goscratch="cd ~/scratch"
alias scancelAll="scancel -t PENDING -u $USER"
alias sallocDefault="salloc --time=2:0:0 --ntasks=1 --account=def-rpicker --mem=4G --x11"
alias dir="ls"
alias ll="ls -l"

# Load Compute Canada modules. these have been tested with PENTrack on May 8, 2020

module load nixpkgs/16.09
```

```
module load scipy-stack/2019a
module load gcc/7.3.0
module load boost/1.68.0
#module load intel/2016.4
module load root/6.14.04

# Configure Compute Canada batch variables
export SLURM_ACCOUNT=def-rpicker
export SBATCH_ACCOUNT=$SLURM_ACCOUNT
export SALLOC_ACCOUNT=$SLURM_ACCOUNT
```

4.4.1 Aliases

Typically, most of your work will be done in your project directory, so it's nice to set up an alias to make it easier to get there. Here you can see the `gohome` alias is used to move to the project directory and the `goscratch` alias for the scratch directory.

Using Slurm and its command-line arguments is terribly confusing when often all you want to do is simply cancel all current pending jobs while testing things or allocate an interactive node, which is what `scancelAll` and `sallocDefault` do, respectively.

4.4.2 Loading modules

To analyze the PENTrack output files we require CERN's ROOT libraries, which is already installed on Compute Canada. We could run `$ module load root` every time we log in, but it's easier to have it on run on startup by including it in the `.bashrc`. This applies to scipy and gcc as well. Python 2.7 is automatically loaded as a ROOT dependency.

4.4.3 Batch variables

Previously we discussed that `account` was one of the parameters that must be passed to the Slurm job scheduler for every single job. However, your work will typically only be carried out on a single account. Adding these few lines to the `.bashrc` removes the need to specify the account when submitting jobs. This allows for batch scripts to be more reusable between different people since they don't have a specific account.

4.5 Setting up SFTP

Setting up a Safe File Transfer Protocol (SFTP) is recommended for transferring files from Cedar to your local machine. It is very simple to set this tool up on MobaXterm:

1. Right click on the left side of the screen under "User sessions."
2. Click "New session."
3. Click the SFTP icon. It looks like a yellow Earth.
4. Under "remost host," type "cedar.computecanada.ca." Under "Username," type your ComputeCanada username. You will need to enter your password and use 2FA to login.

Using SFTP, you can easily download and upload files from and to your Cedar directory.

5 Installing and Running PENTrack

5.1 Running PENTrack

You will need to install PENTrack on the Compute Canada server in your project directory. You can find the GitHub repository for PENTrack here. PENTrack can be installed anywhere in your project directory, so once you have navigated to your desired directory clone the repository:

```
$ git clone https://github.com/wschreyer/PENTrack.git
```

We want to build PENTrack, but to do so we need to load an additional module and enter the PENTrack folder to build:

```
$ cd PENTrack
$ cmake .
$ make
```

Then you can return to the directory you cloned in PENTrack and try running it to see if it's installed properly:

```
$ cd ..
$ ./PENTrack/PENTrack 1 PENTrack/in/config.in .
```

At which point you should see:

```
########################################################################
###                      Welcome to PENTrack,                        ###
### a simulation tool for ultracold neutrons, protons and electrons ###
########################################################################
```

You can exit with Ctrl C before you produce too many output files in your directory.

5.2 How To Access and Analyze your Data

One can elect to output data in the form of a text file. in this case, multiple `.out` files can be read into Pandas dataframes and combined for easy analysis.

6 Preparing a SolidWorks File for PENTrack Simulations

This section assumes you already know how to use SolidWorks. If not, go watch YouTube videos and play around a bit, it's not hard.

The end goal in using SolidWorks is to produce an STL file of essh -Yach part in the assembly with the correct relative coordinates for use in PENTrack. Because the analysis is carried out in several stages of Filling, Storage, and Emptying, your assembly should support multiple configurations so that the valves are in the correct orientations for each stage. It is fine if you do not use configurations and instead manually add and break mates to change the position of the valves, but this is more prone to errors and takes more work if you need to recreate your STL files. PENTrack also supports ignoring certain parts for a specified interval, so another trick is to have multiple parts overlapping and have PENTrack ignore them in order to simulate a valve opening and closing during the simulation.

To reiterate: We are using SolidWorks solely as a tool for producing STL files. Whether you use configurations within SolidWorks to export different STL files or use the PENTrack configuration file to ignore files during certain periods is entirely up to you as long as the resulting simulation makes sense.

Each simulation in PENTrack typically goes through three stages, mimicking reality; Filling, Storing, and Emptying. The storing and emptying are further separated into the two cells of the nEDM chamber, the top and bottom cells. This is because they are at vertical different positions and will have different energies due to gravity. We thus have 5 simulations, 1 for filling, 2 for storing, and 2 for emptying.

We typically choose a solid to represent the starting location of the UCN during each simulation, so we would have one at the target where the beam produces the UCN, and then another in the EDM chamber for both storing and emptying.

Aside from this general theory, the actual specifics of the model depend on what exactly you are simulating and should be provided for you.

The general process for exporting STL files from an assembly is as follows:

1. Open the assembly. Click "File" in the toolbar.

2. Click "Save As."
3. In the window that pops up, at the bottom, check the box that says "Include all referenced components."
4. Under "Save As type," select STL.
5. Select "Options.."
6. Verify that the criteria listed below are met.
7. Save.

Criteria for exporting STL files:

- Suppressed models will **not** be exported to STL
- Hidden models will **not** be exported to STL
- Transparent models will be exported to STL
- Try to suppress or hide models that aren't seen by the UCN to reduce upload times and file space
- Once you select Save As, make sure you choose the following settings in Options:
  - Units: Meters
  - Output as: Binary
  - Resolution: Fine
  - Show STL info before file saving
  - Do NOT translate STL output data to positive space
  - Output coordinate system - Coordinate System 1, which represents distance along one of the guides (not the default origin) - Most important here is that the positive z axis points in the opposite direction of gravity and that any field input files are oriented correctly with respect to the STL coordinate system.
  - Everything else should be left unchecked

To note:

- Remember to export an STL file of the entire assembly for each stage of the simulation. RP: don't know what that means... WS: me neither
- MeshLab is a useful tool to inspect and repair STL files.
- It is often useful to combine many small parts into a single part, if they are the same material in PENTrack. To do this in SolidWorks, select multiple part files in the object tree, then click "Form New Assembly." Then, right click on the new assembly in the object tree and click "Save Assembly(in External File)." Now you can open the assembly in a new SolidWorks window and save the assembly as a SLDPRT file.

7 Preparing PENTrack Simulations

Now is a good time to discuss file organization on the Compute Canada servers. We already discussed the different project and scratch drives, but how you manage the files for simulations is very important to maintaining a good workflow. The author has an established file structure that is specific for S. Sidhu's workflow for which the scripts work and the author hopes the file structure is modular enough to be reused for future simulations.

Each simulation study should have its own folder, and within each simulation there should be a folder for each stage of the simulation. You will also want a folder to keep all the simulation studies together

Thus far, you might have a structure like thus:

```
/
└── varyingDiameters
    ├── 8.5cm
    │   ├── filling
    │   ├── topStorage
    │   ├── bottomStorage
    │   ├── topEmpty
    │   └── bottomEmpty
    └── 9.5cm
        └── ...
```

There being 5 stages means that PENTrack will need to be run 5 times, each time with its own STL folder and configuration file, `config.in`. PENTrack also requires a material properties file, `materials.in`. This materials file can be shared between all 5 stages, but the configuration file should probably be separate for each file. Each STL folder may contain different configurations of the model. The filling stage may also require a magnetic field file but I don't know enough about this to confirm, it would be best to ask S.Sidhu or W.Schreyer. Thus, a more detailed structure might be as such:

```
/
└── varyingDiameters
    ├── 8.5cm
    │   ├── materials.in
    │   ├── filling
    │   │   ├── STL
    │   │   ├── config.in
    │   │   └── SCMatOriginAlongX_hires_CGS.table
    │   ├── topStorage
    │   │   ├── STL
    │   │   └── config.in
    │   ├── bottomStorage
    │   │   ├── STL
    │   │   └── config.in
    │   ├── topEmpty
    │   │   ├── STL
    │   │   └── config.in
    │   └── bottomEmpty
    │       ├── STL
    │       └── config.in
    └── 9.5cm
        ├── materials.in
        └── ...
```

The location of `materials.in` can be on the same level as the simulation stages or one level higher if you want all the different simulation studies to refer to the same materials. It doesn't really matter and you can do both if you want since the path of `materials.in` is specified by each configuration file.

```
/
└── varyingDiameters
```

```
├── materials.in
├── 8.5cm
│   └── ...
└── 9.5cm
    └── ...
```

Now that you have the STL files you will need to upload it onto the Compute Canada remote servers. How you do this is up to you, but at least on Windows you can easily drag in your STL files in WinSCP. They should be placed within the STL folder for each simulation as mentioned before. You will also need to remove all the spaces from the names of the STL files otherwise the configuration file for PENTrack will not work. Don't do this manually. There is a script for this and its use will be explained later.

You will need to set up each configuration file manually. It is mostly self explanatory if you take your time to read the file. Typically after you set up the first most of it can be recycled for the next one as long as the STL file names are the same. Configuration files will be explained in the next section, and reading the documentation within the configuration itself is also helpful.

You will also need to set up the material files, although there should be nothing to change from what your supervisor provides unless you are specifically experimenting with different material properties.

8 Configuration Files for PENTrack

PENTrack contains an internal configuration file along with its installation, but this is typically not very useful. Your configuration file should be specific to the simulation being studied, just as your STL files should represent the geometry for what you want to study, not for a house or something else. Together the STL and configuration file for each simulation dictate what geometry and properties PENTrack sees.

The main portion of `config.in` that you need to change for each simulation is the STL file portion. Here, you can add or remove STL files from the PENTrack simulation by specifying a path for the file. The higher the ID number, the higher the priority. That is, when an UCN ends up in two parts at the same time, the part with the higher priority will be the one to reflect the UCN. Typically you would still try to minimize the amount of intersection in your model to accurately reflect reality. These STL files must not have any spaces in their names, and so because SolidWorks outputs the files with spaces, we have a script to go through and erase the spaces. Again, all STL files must be in the STL folder because the paths have been specified `STL/<filename.STL>` but this could be changed with enough effort.

For each STL file, you must set a material, and an optional ignore time. The list of materials should be set up in your `materials.in` file, and the path to this `materials.in` file is specified near the beginning of the `config.in` file.

Finally, there are some other parameters for the simulation in the `config.in` file, but those are typically left untouched, or at least unless you know what you're doing there.

9 Using Scripts to Submit a Job to Compute Canada

NOTE FROM KYLE: when you use a script to submit a job, the jobs will be queued. The contents of the config.in file when the JOB ACTUALLY GETS EXECUTED (NOT WHEN YOU QUEUED IT) are what is used in the simulation. So let's say you queued up sims, and then copied some new numbers into your config.in file, and queued up some more. If one of the original sims hasn't started, then when it does, it will use the wrong config.in file, because it'll use the contents you pasted in the file for the second run instead of the first. This messed me up royally for a while.

Once the previous sections involving setting up the STL and `config.in` files are complete, the rest of the work is largely automated as long as the directory structure has been followed correctly. From here, just follow the instructions to run the scripts and produce your desired output data. The scripts will be explained in detail in a later section, but everything works fine if you just run everything in the correct order.

1. Set up STL files, `config.in`, `materials.in`, magnetic field, and exact directory structure
2. Open `config.sh` to adjust the paths and directories to match your account. This should only need to be done once, but the batch settings inside may need to be changed more frequently if some simulations

are more CPU time intensive than others.

3. Run `removeAllSTLSpaces.sh` saying `y` to any studies/stages you want to remove spaces for
4. Run `testPENTrack.sh` to see if properly configured. `removeAllSTLSpaces.sh` must have already been run. Say `y` to any studies/stages you want to analyze
5. Run `generateBatchFiles.sh` saying `y` to any studies/stages you want to analyze
6. Run `submitTasks.sh` and accept auto merge unless you want to run the manual merge later. Say `y` to any studies/stages you want to analyze. All previous scripts must have been run for this work.
7. Wait for all tasks to be complete, using `sq` to check the status periodically. You can check the PENTrack output files once a simulation is done to see how long it took so you can estimate how much time you need to allocate to your tasks.
8. If you did NOT select auto merge earlier, now is the time to `mergeAllScratch.sh` which will submit merge tasks to the job scheduler. You will then need to wait for these merge tasks to finish, using `sq` to check as before.
9. Run `verifyRoot.sh` which will look at all the new merged out.root files to see if they make sense. Say `y` if you want to look (generate a plot) of any specific stage. After the canvas is generated, you can use `.q` in the terminal to quit the canvas and move onto the next one. It is at this stage that you want to check for any leaks in your models. A small amount of leakage is to be expected for certain interfaces between STL models if they aren't triangulated perfectly.
10. At this point all further analysis is specific to S.Sidhu's workflow, which requires the simulation to be split into stages for his `daystoreach.py` python script.
11. Run `processOutputs.sh` which will go through the PENTrack output and ask if it is an emptying, storage, or filling stage. Do this for all 5 stages of each simulation in preparation for the next step.
12. You will need to manually edit the `daystoreach.py` file to point to the correct `.root` files, the ones that were produced by the previous `processOutputs.sh` script and not the ones that were produced by PENTrack. Talk to S.Sidhu for an introduction to the `daystoreach.py` script. Keep in mind that the root files you are looking for are the one that are 1 directory above the ones produced by merging the PENTrack output files.
13. Finally run `daystoreach.py` (which is a python file, so use `$ python daystoreach.py`) to produce plots of the days to reach among other data for each simulation. You may want to be on an interactive compute node for this, as it may take an hour or two if you have 8 simulations (times 5 for the 5 stages) so there is some heavy computing involved.

By the end of running all these scripts you should have a few pdf files which you can view. Other useful data can be found in a `.txt` file in the same directory as the PDFs.

10 How the Scripts Work

This section provides a high level overview of the input and output to each script to automate S. Sidhu's workflow, and its role in the overall process. A detailed explanation of the code is in the script itself, as it is fully commented with all the weird bash hacks used. To understand how the scripts work, first we must again go over the pipeline for the simulation process, not skipping over intermediate output files:

1. Manually set up STL files, `config.in`, `materials.in`, magnetic field, and exact directory structure
2. Run `removeAllSTLSpaces.sh` which removes spaces from the STL files
3. Run `testPENTrack.sh` to run PENTrack locally and see if the configuration files are correct

4. Run `generateBatchFiles.sh` which goes through each folder and generates a single batch file which will execute PENTrack with the proper settings and configuration file.
5. Run `submitTasks.sh` which will go through each folder and submit the batch files. It will also submit `mergeTask.sh` as a dependency if automerge is enabled.
6. PENTrack will produce `.out` files which need to be processed.
7. If you did not select auto merge earlier, `mergeAllScratch.sh` will submit `mergeTask.sh` to the job scheduler. `mergeTask.sh` collects all the `.out` files produced by PENTrack into a single `out.root` file for analysis.
8. Run `verifyRoot.sh` which will look at all the newly merged `out.root` files and run `openDrawing.c` to generate the plots.
9. At this point all further analysis is specific to S.Sidhu's workflow, which requires the simulation to be split into stages for his `daystoreach.py` python script.
10. Run `processOutputs.sh` which will go through the PENTrack output and ask if it is an emptying, storage, or filling stage. For filling stages it just copies the `out.root` file one directory up and renames it. For storage it runs `cellStorage.C` and for emptyingit runs `cellEmpty.C`, then copies it one directory up and renames it.
11. You will need to manually edit the `daystoreach.py` file to point to the correct `.root` files, the ones that were produced by the previous `processOutputs.sh` script and not the ones that were produced by PENTrack.
12. Finally run `daystoreach.py` (which is a python file, so use `$ python daystoreach.py`) to produce plots of the days to reach among other data for each simulation.

With a general idea of how the scripts work on a higher level, now is the time to open up each of the scripts in a text editor and have a look through them to understand how they all work if you want to make any changes.

11 How to Use ROOT

CERN's ROOT libraries are responsible for a lot of the data processing and heavy lifting going on in the background. Ideally you don't need to work directly with ROOT, but it may be good to know what it is and some basic functionality. For additional help, ask W. Schreyer directly, as he is very familiar with ROOT within the project team. ROOT is also an interactive C++ compiler, making use of Clang.

The main functionality you need to know is `$ root` will open up ROOT. use `$ .q` to quit ROOT. if you use `$root out.root` ROOT will open up the out.root file. You can use `TBrowser t` to open up a UI and look inside this root file. You can use `$ neutronend->Draw("zend:xend")` to produce basically a cross sectional view of the geometry and figure out where the UCN reached. For other ways of using ROOT, just consult W.Schreyer.

12 Troubleshooting

Here are some notes from the author on troubleshooting

- It says a .root file is still open : a process is probably still running, check with sq. If it isn't running, try making the out.root file again with the manual merge script
- The top and bottom simulations both have the same output : Change the configuration file carefully
- ROOT canvas not opening : Check that your X11 server is opened with access controlled disabled. The server needs to be open before you open your terminal client and sign into Cedar. If this problem occurs on an interactive compute node, check that you have X11 forwarding set up correctly in your salloc command (go back and read carefully)

13 A Simple Toy Model for Testing PENTrack

13.1 Preparing STL Files

To get a grasp on the fundamentals of PENTrack, we recommend doing a basic exercise. Download the toy model I made and run a simulation (see "Kyle's Thesis and Toy Model" on page 3):

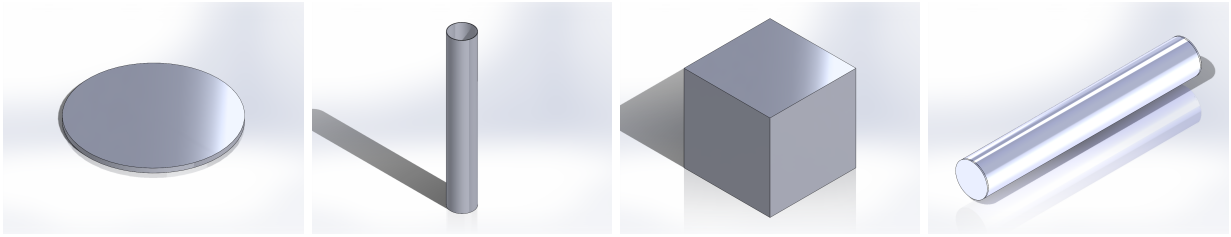

Figure 1: Right to left: flange, detector, guide, source, and the entire Toy Model assembly created in SolidWorks.

The guide tube is a hollow cylinder with inner and outer radii of 0.145 m and 0.150 m, respectively. The flanges have a radius of 0.150 m and a thickness of 0.005 m. The source is a 0.01 m cube with its bottom face coincident with the top face of the bottom flange of the guide tube at $z = 0$. The guide tube is 1 m in height, and the top flange is made of the pseudo-material 'UCNdet' in the PENTrack config file, which means that all the UCNs that hit the top flange will be absorbed. By following the steps to export the SolidWorks assembly, these files will be saved using the same coordinate system, so they are assembled correctly in PENTrack.

After exporting the entire assembly using SolidWorks' "Pack and Go" feature, one can double-check that the geometry is consistent by dragging and dropping the STL files into Meshlab, which displays the STL files together:

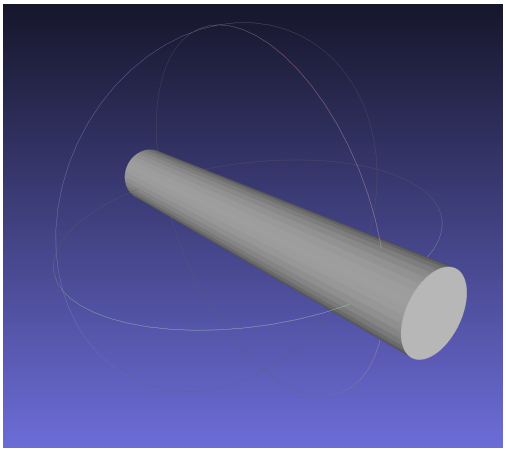

Figure 2: The Toy Model as viewed in Meshlab.

13.2 Setting up the Config File

Most of the `config.in` file should be configured properly, but to implement the Toy Model geometry, you will need to change some things:

```
[GEOMETRY]
############ Solids the program will load ###############
#  Each solid has to be assigned unique ID and a material from above.
# IDs have to be larger than 0, ID 1 will be assumed to be the default medium which is always present.
# Particles absorbed in a solid will be flagged with the ID of the solid.
# The ID also defines the order in which overlapping solids are handled (highest ID will be considered first).
# If paths to StL files are relative they have to be defined relative to this config file.
# Ignore times are pairs of times [s] in between the solid will be ignored, e.g. 100-200 500-1000.
#ID     STLfile    material_name    ignore_times
1       ignored                 default
2       bottom_flange.STL       DLC
3       top_flange.STL          UCNdet
4       guide.STL               DLC

[SOURCE]
########### sourcemodes ##############
# STLvolume: source volume is given by a STL file, particles are created in the space completely enclosed in the STL surface
# boxvolume: particle starting values are diced in the given parameter range (x,y,z) [m,m,m]
# cylvolume: particle starting values are diced in the given parameter range (r,phi,z) [m,degree,m]
# Volume source produce velocity vectors according to the given angular distributions below.
# If PhaseSpaceWeighting is set to 1 for volume sources the energy spectrum is interpreted as a total-energy spectrum.
## The probability to find a particle at a certain initial position is then weighted by the available phase space,
## i.e. proportional to the square root of the particle's kinetic energy.
#
# STLsurface: starting values are on surfaces in the given STL-volume
# cylsurface: starting values are on surfaces in the cylindrical volume given by parameter range (r,phi,z) [m,degree,m]
# Surface sources produce velocity vectors cosine(theta)-distributed around the surface normal.
# An additional Enormal [eV] can be defined. This adds an additional energy boost to the velocity component normal to the surface.
#########################################

sourcemode      STLvolume

STLfile         source.STL      # STL volume used for STLvolume/STLsurface source, path is assumed relative to this config file
```

Figure 3: The geometry and source specifications in the config file. 'DLC' stands for diamond-like carbon, and 'UCNdet' is a dummy material with a high imaginary Fermi potential to act as a detector by absorbing all neutrons. ID=1 corresponds to the medium the simulation takes place in; in this case, air. The source type has been chosen to be a volume source.

13.3 Validating Results

I set up the config file to take snapshots every 10 seconds of the simulation over 1000 s (the length of the simulation). The snapshot data is not shown in this guide, but the start and end data is shown in Figure 4, along with plots of neutron trajectory lengths, neutron lifetimes, and plots of neutron lifetimes over initial kinetic energy and the number of neutrons over time (these last two plots are fitted to exponential curves).

14 Conclusion

Hopefully by the end of this document, you have a good idea of how to use PENTrack in conjunction with Compute Canada for UCN simulation. If nothing else, just know how to set up the STL and configuration files, the scripts will do the rest. W. Schreyer will be your best resource for any further questions regarding PENTrack simulations.

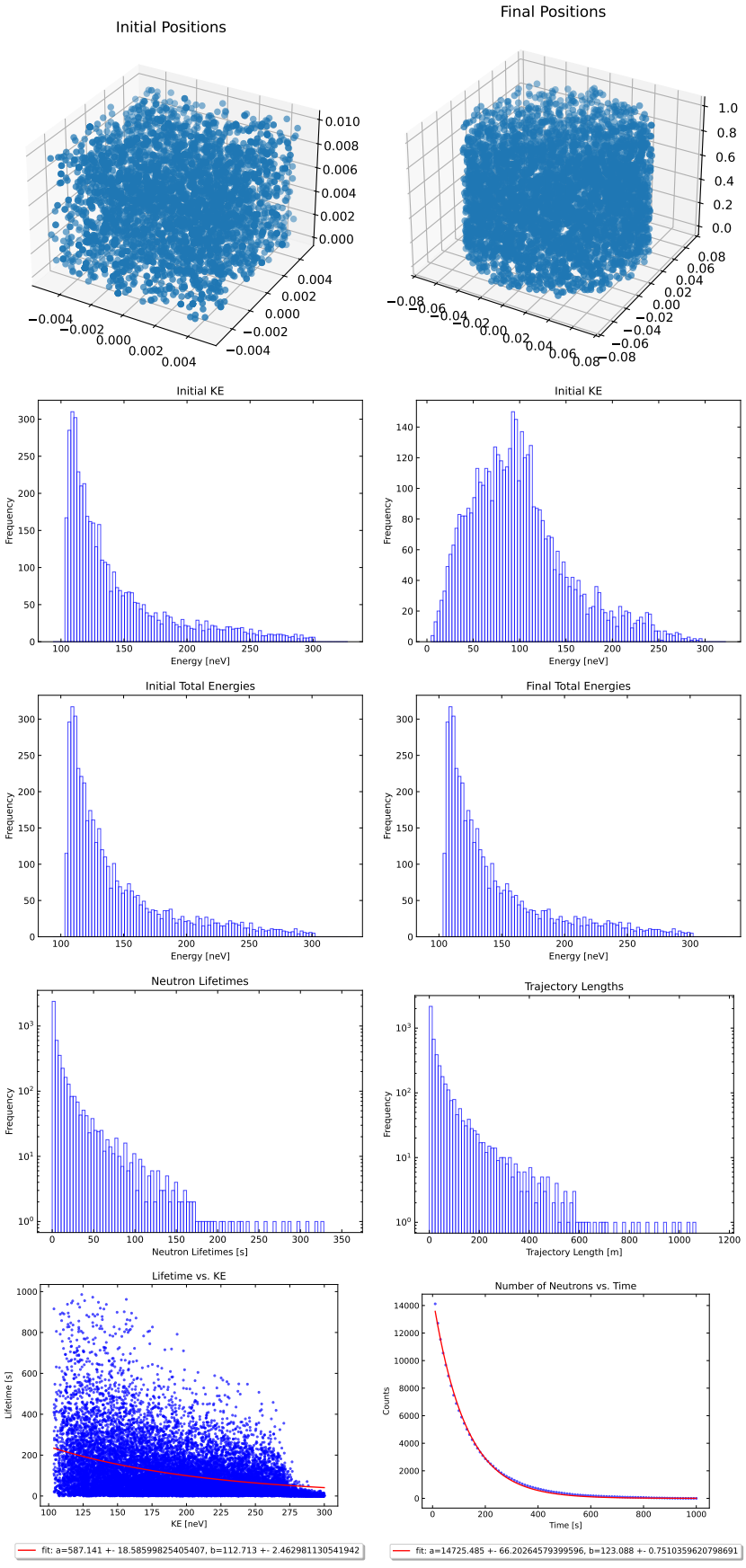

Figure 4: Data from the 20,0000 neutrons in the toy model simulation.

A Material properties

A.1 Fermi potential database

Sample calculation

A.2 Table of values used in simulations

List of materials from literature and experiment

B Hepak-tools package

With the hepak-tools repository cloned and dependant-software installed, the various UCN source scripts can be used. These scripts can be used to determine various parameters of the UCN source, which can then be used to determine the temperature profile of isopure ^{4}He modelled in the TUCAN source simulations.

B.1 HEPAK.xla

HEPAK Version 3.40 is an Excel add in. Functions can be called within cells to return experimental parameters based on a series of arguments. The focus of this section will be instructing the reader on how to successfully integrate the add-in with Excel, because by default, Excel doesn't trust HEPAK and won't allow its features to be used.

The .xla file along with the HEPAK user guide can be found here. Once the file is installed, open Excel, and navigate to File > Options > Trust Center > Protected View. You should also see Macro Settings within the Trust Center; configure the settings for each as shown:

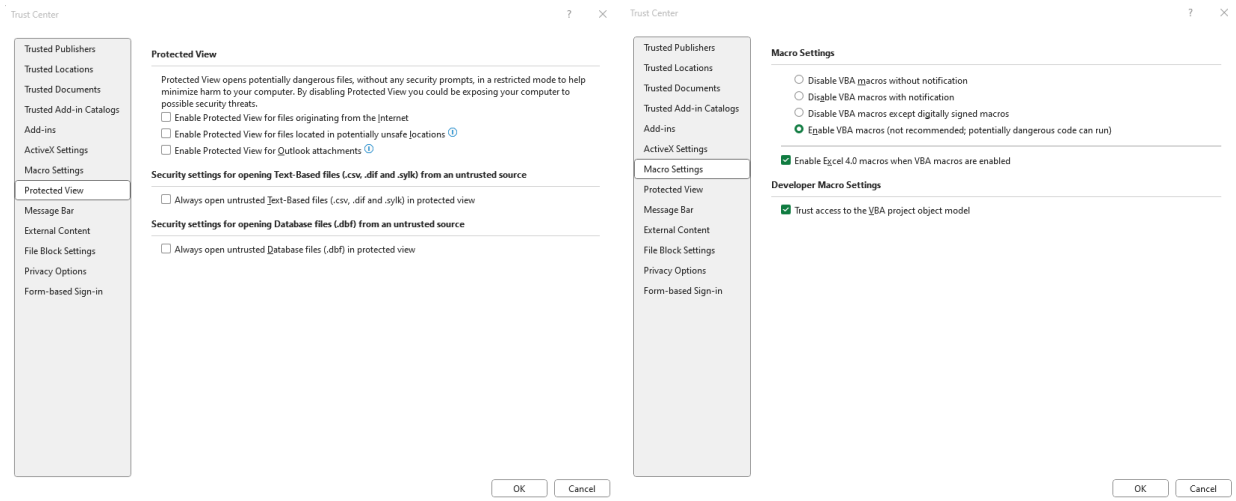

Figure 5: Protected View (left) and Macro settings (right) for using the HEPAK Excel add-in.

Navigate to the HEPAK.xla properties and make sure that Read-Only and Hidden are unchecked. Within Excel, go back to options > Add-ins. At the bottom of the panel, you'll see a drop down menu labeled "Manage." Click on the "Go..." button, then click browe. Find HEPAK.xla and open it. You might get some kind of error. If you do, restart Excel, and then go back to Add-ins, click "Go..." again, and you should see Hepak within the list of available Add-ins. Make sure to check the box. Restart if necessary. You can now use the HEPAK add-in in Excel.

B.2 How to install the package, HEPAK and, HE3PAK

The python script used to calculate TUCAN source parameters and the temperature profile of isopure He-II requires a Windows operating system and a 32-bit python environment. The README for the script can be found on https://github.com/wschreyer/hepak-tools and contains more information.

To install and get this scrip running, the following steps should be taken:

1. Clone the repository hepak-tools from GitHub from a terminal, be sure that you are in an appropriate folder:

```
git clone https://github.com/wschreyer/hepak-tools.git
```

2. Download and place hepak and he3pak[1] dll files into the same folder. They can be found on plone: https://ucn.triumf.ca/vertical-ucn-source/thermodynamics/hepak, https://ucn.triumf.ca/vertical-ucn-source/thermodynamics/he3pak.

3. Create a 32-bit python environment. Miniconda (or Conda) is recommended for this.

   (a) The Windows installer can be found on the miniconda website: https://docs.conda.io/projects/miniconda/en/latest/index.html.

   (b) Once installed, open a miniconda terminal to see that the base environment is indicated:

```
(base) C:\Users\User>
```

   (c) To list different python environments installed, use the command:

```
conda env list
```

   (d) To create a 32-bit python environment, use the following commands:

```
conda create -y -n py32;
conda activate py32;
conda config --env --set subdir win-32;
conda install -y python
```

   where the name of the environment name is chosen by the user to be py32.

   (e) Install necessary packages:

```
conda install -y scipy matplotlib pywin32
```

4. Check to see that the script works by navigating into the hepak-tools folder and listing all files in the directory using the command dir. The file UCNsource.py should be listed. Try running this file using the command

```
python UCNsource_HeIIsegments.py
```

   Often there is an error that looks like this:

```
Rebuilding cache of generated files for COM support...
Checking 00020813-0000-0000-C000-000000000046x0x1x9
Could not add module (IID('{00020813-0000-0000-C000-000000000046}'), 0, 1, 9) -
<class 'AttributeError'>: module
```

```
'win32com.gen_py.00020813-0000-0000-C000-000000000046x0x1x9'
has no attribute 'CLSIDToClassMap'
Done.
module
'win32com.gen_py.00020813-0000-0000-C000-000000000046x0x1x9'
has no attribute 'MinorVersion'
Try clearing contents of C:\Users\<username>\AppData\Local\Temp\gen_py
to fix this error
```

this can be fixed by deleted this file, for example,

```
rmdir /s C:\Users\User\AppData\Local\Temp\gen_py
```

where <username> here is "User". There is a prompt to acknowledge the use of hepak. This is an Excel pop-up, and the use can clear it by clicking the 'Ok" button. If there are no errors, then the script is now ready to use.

5. The python environment is now installed. After opening a new miniconda terminal, to activate the environment, use the commands

```
conda env list
```

to list the environments. To change environments, use the command

```
conda activate py32
```

and to deactivate

```
conda deactivate
```

Finally, to delete an environment, use the command

```
conda env remove --name py32
```

The hepak-tools package has now been successfully installed.

There are many scripts in this package, and they can be edited using a text editor. The primary script in the hepak-tools package is the UCNsource.py script that provides a model of the ^{3}He fridge for the UCN source. The UCNsource_parameterSweep.py performs a scan of all experimental parameters and plots the resulting temperatures and flows in the fridge. The UCNsource_HeIIsegments.py calculates the temperature profile in the Gorter-Mellink channel and prints out a list of temperatures, UCN storage lifetimes, and imaginary Fermi potential averaged over segments along the channel.

The UCNsource_parameterSweep.py and UCNsource_HeIIsegments.py scripts utilize the UCN source model in the UCNsource.py script and require a list of input parameters provided by the user (this is a list of descriptions, selected variable names, assumed baseline values, units),

- Heat load from the target: Beam heating, 8.1 W
- The static heat load to the isopure ^{4}He : He-II static heat, 0.25 W
- The static heat load to the funnel at the bottom of the S-bend: He-II funnel heat, 1.25 W

- Properties of the ^{3}He pumps, either:
  - The pressure drop in the ^{3}He pumps: 3He pressure drop, 100 Pa, or
  - The pumping speed based off the 2-stage or 3-stage configuration: 3He pumping speed, 4300 m^3/h
- Properties of the ^{4}He pumps, either:
  - The pressure drop in the ^{4}He pumps: He pressure speed, 100 Pa, or
  - The pumping speed: He pumping speed, 2000 m^3/h
- The 4 K pot inlet temperature: He reservoir inlet temperature, 10 K
- The static heat load to the 4 K pot: He reservoir static load, 0.6 W
- The 1 K pot static heat load: 1K pot static load, 0.05 W
- The HEX4 exit temperature 2.8 K
- The HEX5 exit temperature 2.8 K, which is equivalent to the 1 K pot inlet temperature
- The ^{3}He inlet pressure: 3He inlet pressure, 50×10^3 Pa
- The diameter of the Gorter-Mellink channel: Channel diameter, 0.148 m
- The diameter of the Gorter-Mellink channel: Channel length, 2.356 m
- The diameter of HEX1: HEX1 diameter, 0.148 m
- The length of HEX1: HEX1 length, 0.6 m
- The area of ^{3}He on the surface of HEX1: HEX1 3He surface, 1.3999 m^2/m
- Thermal conductivity of HEX1: HEX1 conductivity, 9.6/0.07/0.6 W/m K
- The Kapitza coefficient for HEX1 K_G: HEX1 Kapitza kG, 35*0.61 $W/m^2/K^4$
- The overfill level of ^{4}He in the S-bend: HeII overfill, 0.05 m
- The pressure in the 4 K pot:He reservoir pressure, 1.3×10^5 Pa
- The flow of isopure ^{4}He: Isopure He flow, 0.0 kg/s
- The vapor pressure of isopure ^{4}He: Isopure He pressure 100×10^2 Pa
- The temperature of the 20 K shield: 20K shield temperature, 20 K
- The temperature of the 100 K shield: 100K shield temperature, 100 K

B.3 UCNsource.py

The UCNsource.py script calculates many parameters of the UCN cryostat that can be used to determine the lowest temperature of the ^{4}He in the UCN source. It requires the pumpdata.py and HEXdata.py. The pumpdata.py script is used to calculate the pumping curves of the Busch 2-stage and 3-stage system. The HEXdata.py script is used to interpolate data from measured ^{3}He boiling curves in a UCN fridge, and can be modified to interpolate ^{4}He boiling curves if experimental data is available. Both the pumpdata.py and HEXdata.py scripts use CSVs generated from plots using https://plotdigitizer.com/.

Although this script contains many functions, the two that are most commonly used to return values are,

1. the equationSet function calculates and returns:
   - The ^{3}He flow rate (kg/s)
   - The 1 K pot temperature (K)

- 4 K pot temperature (K)

2. the calcUCNSource function calculates and returns the following important properties:

   - The 4 K (He reservoir) pot inlet temperature (K)
   - The 4 K pot flow rate (kg/s)
   - The 1 K pot flow rate (kg/s)
   - The helium consumption of the TUCAN cryostat (L/h)
   - The 20 K shield flow rate (kg/s)
   - The 100 K shield flow rate (kg/s)
   - The helium consumption of the shields (L/h)
   - The temperature of ^{3}He above HEX1 (K)
   - The vapor pressure of ^{3}He above HEX1 (Pa)
   - The liquid fraction of ^{3}He above HEX1
   - The temperature of HEX1 on the ^{3}He side (K)
   - The temperature of HEX1 on the isopure ^{4}He side (K)
   - The vapor pressure of isopure ^{4}He (Pa)
   - The pressure head of isopure ^{4}He (Pa)
   - The coldest temperature of isopure ^{4}He T_HeII_low at the center of HEX1 (K)
   - The temperature of isopure ^{4}He T_HeII_high in the production volume (K).

The TUCAN source is cooled by a Joule-Thomson cryostat. See Fig. 6 for a cut-through plot. Pumping on a cryogenic liquid lowers the vapor pressure above it, which results in a lower temperature of the liquid. The cooling power Q can be calculated by the latent heat of condensation L (J/mol) and the helium mass flow rate n/t (mol/s) of the liquid,

$$Q = \frac{n}{t} L. \tag{1}$$

a. The cooling load (of a JT system) is calculated using the function HeCoolingLoad by a multiplying a given flow of helium through the JT vessel by the difference in outlet and inlet enthalpies.

b. The cooling flow (of a JT system) is calculated using the function HeCoolingFlow by dividing a given heat load (to the JT vessel) by the difference in outlet and inlet enthalpies.

c. The evaporation rate of helium in the 4 K pot while ^{3}He is flowing through it (assuming that the ^{3}He is cooled to the 4 K pot temperature) is calculated by diving the total heat load to the pot (from ^{3}He, increase in vapor pressure of helium, and static head load) by the latent heat of the helium in the 4 K pot,

$$\frac{n}{t} = \frac{Q}{L}. \tag{2}$$

d. The helium flows through the 20 K and 100 K shield are calculated with the HeCoolingLoad function and the heat load from isopure ^{4}He flows and static heat loads to the shields.

e. The 1 K pot and ^{3}He temperatures are taken from hepak and he3pak using the sum of the inlet pressure of the pumps and the pressure drop in the pumping ducts. The inlet pressures are interpolated from Busch pumping curves using the helium flow rates.

f. The lowest temperature T_HEX1_low of HEX1 occurs at the top where it is in contact with ^{3}He. This temperature, HEX1TemperatureLow, is determined by interpolating measured ^{3}He boiling curves. The heat flux q is assumed to be evenly distributed across the HEX1 surface.

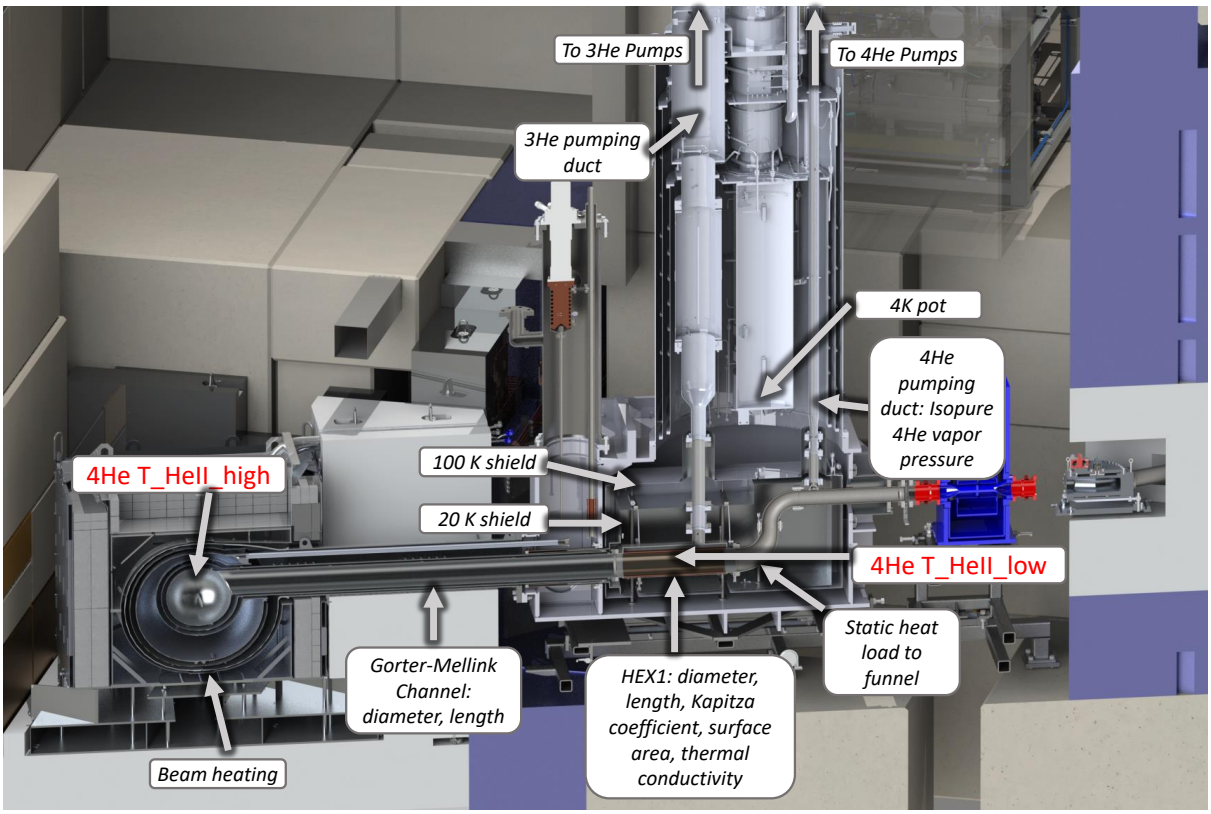

Figure 6: Cut-through of the TUCAN source and cryostat. The locations of important parameters used in the hepak-tools package are indicated on this plot.

g. The highest temperature T_HEX1_high of HEX1 occurs on the inside of HEX1 where the metal is in contact with isopure ^{4}He. 2D FEMM simulations were used to determine the temperature gradient across the copper to be 0.07 K. This was used to determine the thermal conductivity of HEX1. The assumed baseline value is 9.6/0.07/0.6 W/m K, which includes 9.6 W heat load coming from the isopure ^{4}He distributed over the 0.6 m length of HEX1.

h. The coldest isopure ^{4}He temperature T_He-II_low occurs inside HEX1 and is calculated by dividing T_HEX1_high by the heat load per surface area and the Kaptiza conductance,

$$\text{T\_He-II\_low} = \text{T\_HEX1\_high} + \frac{Q_{^4\text{He}}}{A}\frac{1}{h_K}, \tag{3}$$

where the Kapitza conductance is calculated from the equation,

$$h_K = K_G\, 20\, T_{\text{Cu}}^3. \tag{4}$$

i. The position x_0 inside HEX1 where the heat flux is zero (assuming constant heat flux is removed along HEX1) is determined by the function HeIIlowestTemperaturePosition, which balances the heat flux coming from upstream and downstream of HEX1. The position x_0 is determined by multiplying the midpoint of HEX1 by the difference in downstream heating and upstream heating is divided by the total heat spread through the length (which includes radiation heat coming from the S-bend and He-II funnel heat).

j. The heat flux in isopure ^{4}He in the He-II state is determined using the function HeIIheatFlux. For each position x along the length of the UCN source (from the bottom of the S-bend to the end of the helium bottle), the heat flux is calculated by dividing the heat input by area (the cross-sectional area times the length of x) at that position.

k. The temperature profile of isopure ^{4}He in the He-II state from inside HEX1 to the end of the helium bottle is calculated by the function GorterMellink. It uses the Gorter-Mellink equation, starting with the temperature T_He-II_low at the position x_0, using either the HEPAK or Van Sciver model for the thermal conductivity of He-II. This function integrates the ODE given by,

$$\frac{dT}{dx} = k(T)\left(\frac{q}{A}\right)^3, \tag{5}$$

where $k(T)$ is the thermal conductivity parameter. The values of $k(T)$ are taken directly from HEPAK data when using the HEPAK model. The thermal conductivity parameter[1] is calculated when using the Van Sciver model from the equation,

$$k(T) = g(T_\lambda)\left[t^{5.7}\left(1 - t^{5.7}\right)\right]^3, \tag{6}$$

where $g(T_\lambda) = \rho^2 s_\lambda^4 T_\lambda^3 / A_\lambda$, $t = T/T_\lambda$, and the entropy $s_\lambda = 1.559$ J/kg $\cdot$ K, the Gorter-Mellink mutual friction parameter $A_\lambda = 1450$ m s/kg, the helium density $\rho = 145.2$ kg/m^3, and the temperature at the lambda point $T_\lambda = 2.172$ K.

l. The HeIITemperatureHigh function determines T_HeII_high using the GorterMellink function, for both HEPAK and Van Sciver models.

B.4 Temperature profile for PENTrack simulations of the TUCAN source

The temperature profile of isopure ^{4}He in inside the TUCAN source, both in the liquid and vapor state, can be estimated using the UCNsource_He-IIsegments.py script. The temperature at each stage must be converted to a Fermi potential for PENTrack simulations, and the script generates a text that can be copied and pasted directly into the materials.in file of a source simulation. The script uses a function called

[1] In other documents this is often denoted to as $f^{-1}(T)$ for the Van Sciver model.

refinedHeatFluxModel, which calls the UCNsource.py script, to calculate the temperature profile of liquid isopure ^{4}He from the end of the helium bottle to the liquid overfill level inside the S-bend (gravity riser).

The Fermi potentials are calculated using segmentedLiquidTemperature function. This function requires a parameter called liquidSegmentation which separates the He-II into segments and should equal to the number of helium fills in the Solidworks model (79 in most cases). The user must input an assumed value of the Yoshiki B parameter. The mean helium density ρ_{He}, mean temperature T, in each segment is calculated using data from hepak. For each segment, the UCN storage lifetime is calculated using the equation,

$$\tau_{\text{up}} = \frac{1}{BT^7}. \tag{7}$$

The real and imaginary parts of the Fermi potential $U_F = V_F - iW_F$ are then calculated using the equations,

$$V_F = \frac{2\pi\hbar^2}{m_n}\frac{\rho_{He}}{m_{He}b_{He}}, \tag{8}$$

where $m_n = 1.6749 \times 10^{-27}$ kg is the mass of the neutron, $m_{He} = 6.6465 \times 10^{-27}$ kg is the mass of helium, ρ_{He} is the helium density, and $b_{He} = 3.26 \times 10^{-15}$ m is the neutron scattering length on helium, and

$$W_F = \frac{\hbar}{2\tau_{up}}. \tag{9}$$

See Sect. xxx for a sample calculation of the real part of the Fermi potential for nickel phosphorus.

The real part of the Fermi potential for the helium vapor remains the same, given by Equation (9), but the imaginary part must be calculated separately. This is done similarly as the He-II by creating segments and taking the mean temperature and density of each segment. The gas upscattering lifetime τ_{vap} of UCNs can be calculated from the equation

$$\tau_{\text{vap}} = \frac{1}{\frac{\rho_{He}}{m_{He}}\sigma_{He}\overline{v}_{He}}, \tag{10}$$

where $\sigma_{He} = 8.5486 \times 10^{-29}$ m^2 is the free cross-section of helium,

$$\overline{v}_{He} = \sqrt{\frac{8k_BT}{\pi m_{He}}} \tag{11}$$

is the average temperature-dependent velocity of the helium gas atoms, and where $k_B = 1.3806 \times 10^{-23}$ J/K is Boltzmann's constant. All of these steps are performed in a function called segmentedVaporTemperature.

References

[1] Cryodata.Inc. *HEPAK User's Guide*. Horizon Technologies, 2005.