

# Locust Simulation Tutorial: RF Signal Generation from Trapped Charged Particles

P. L. Slocum

NNPSS

Wright Laboratory and YCRC

Yale University

28 June 2018

# Outline

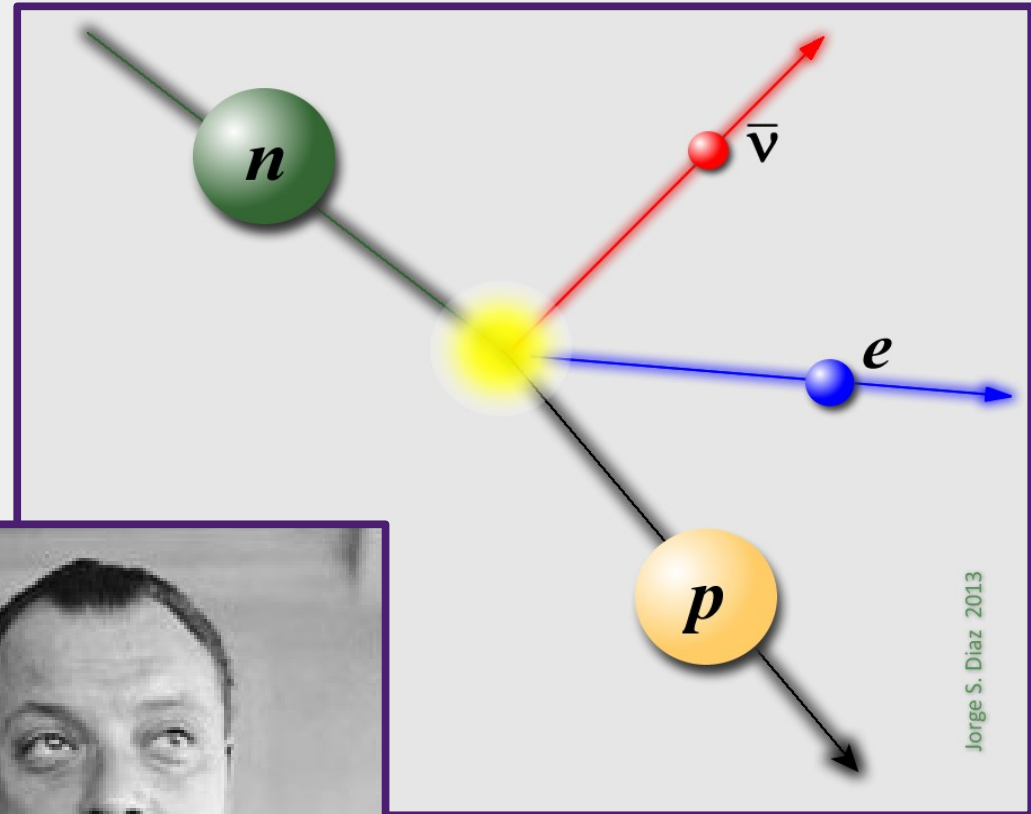
- Motivation: Project 8 experiment.
- Why do we need a new simulation? Compare Locust with existing particle physics simulation packages.
- Purpose of the simulation
  - Generate the same data that we should see from the real experiment.
  - This allows us to test the feasibility and systematics of the real experiment.
- Logistics and examples.
  - Generating and processing a signal.
  - Running the simulation on HPC.
- This pdf sits online at  
[https://github.com/project8/locust\\_mc/blob/develop/Config/NNPSSTutorial/NNPSSTutorial.pdf](https://github.com/project8/locust_mc/blob/develop/Config/NNPSSTutorial/NNPSSTutorial.pdf)

# Neutrinos in radioactive decay

- Neutrino first postulated in 1930 by Wolfgang Pauli to preserve energy conservation in radioactive decays.
- Integral part of Fermi's theory of beta decay in 1933, precursor to theory of the weak interaction.
- Neutrinos first detected in 1956 by Reines and Cowan.



Wolfgang Pauli



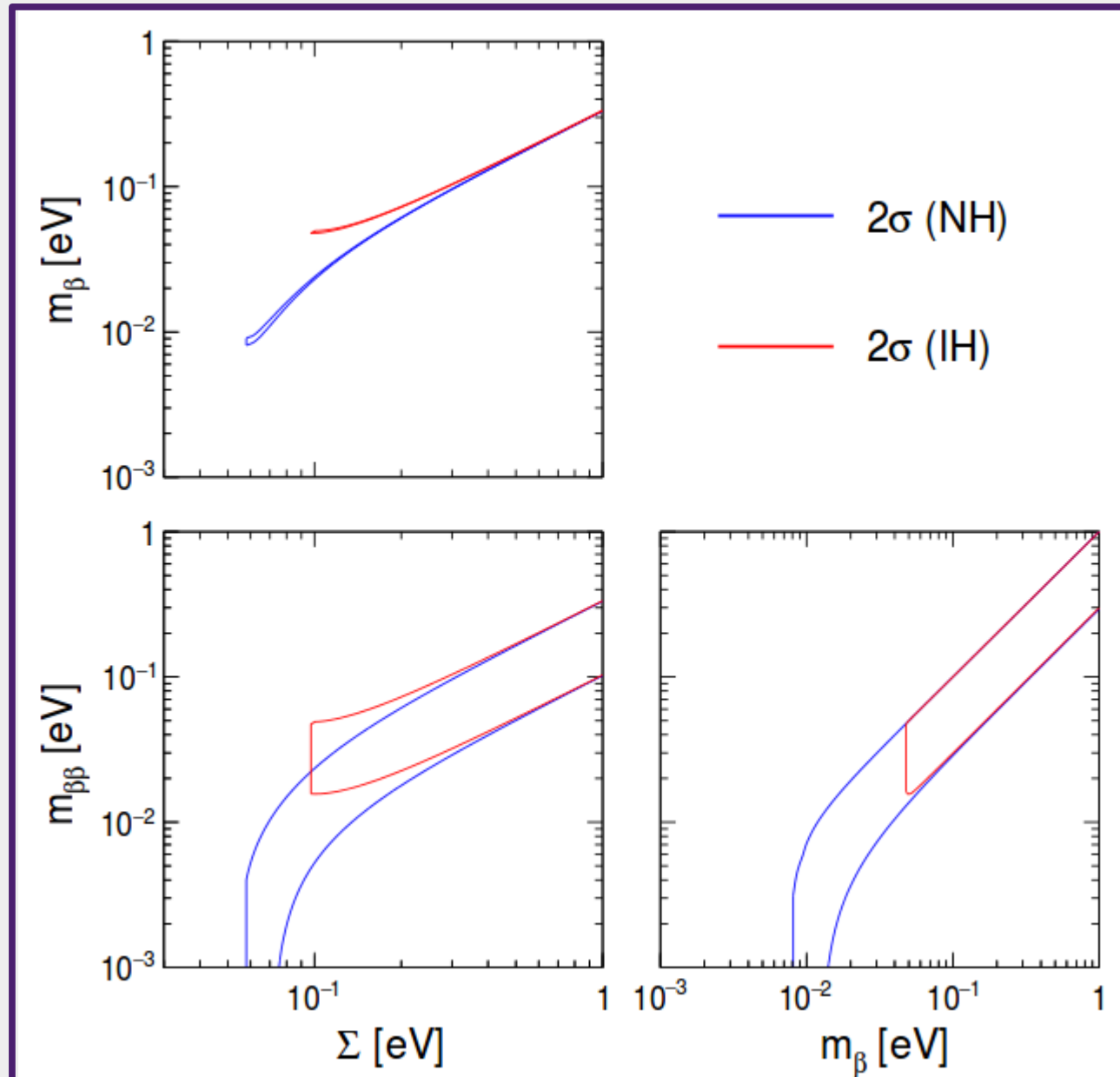
# Constraints on effective masses

$$m_\beta = \sum_i m_{\nu i}$$

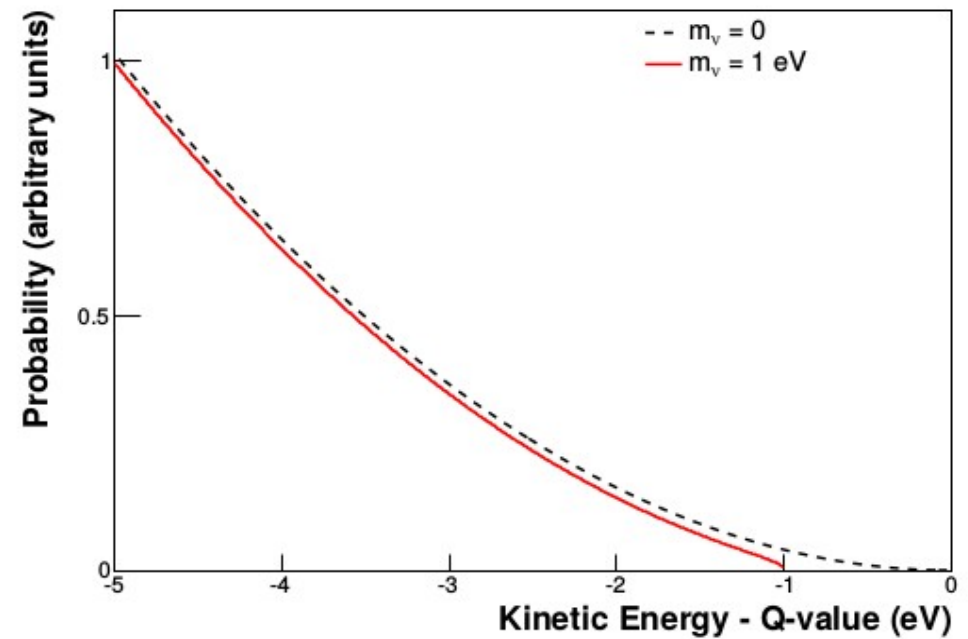
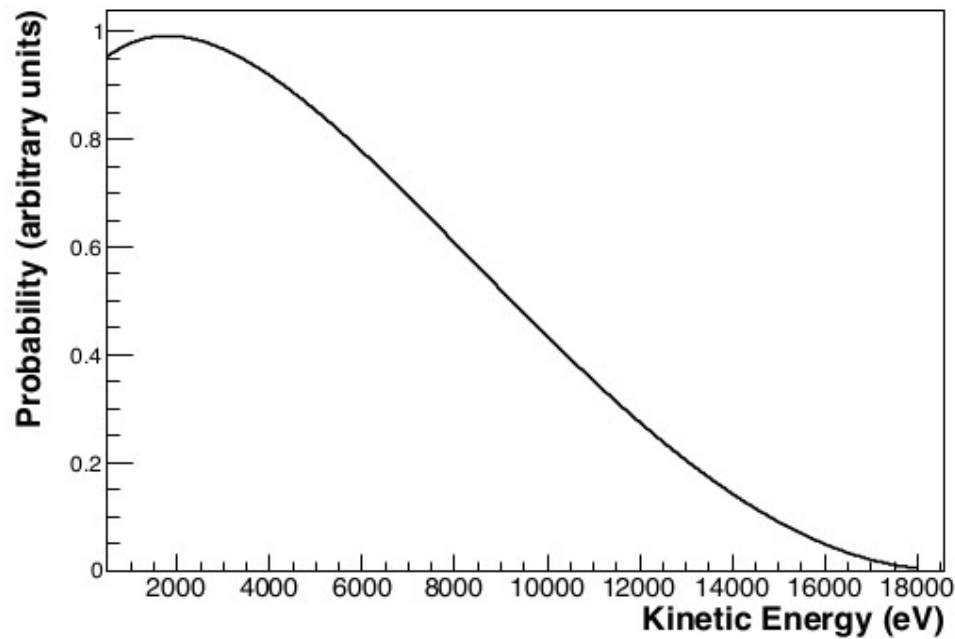
$$m_\beta^2 = \sum_i |U_{ei}|^2 m_{\nu i}^2$$

$$\langle m_{\beta\beta} \rangle = \left| \sum_i |U_{ei}| m_{\nu i} e^{i\alpha_i} \right|$$

Constraints on masses from cosmology and from direct measurements can be compared.



# T<sub>2</sub> beta decay



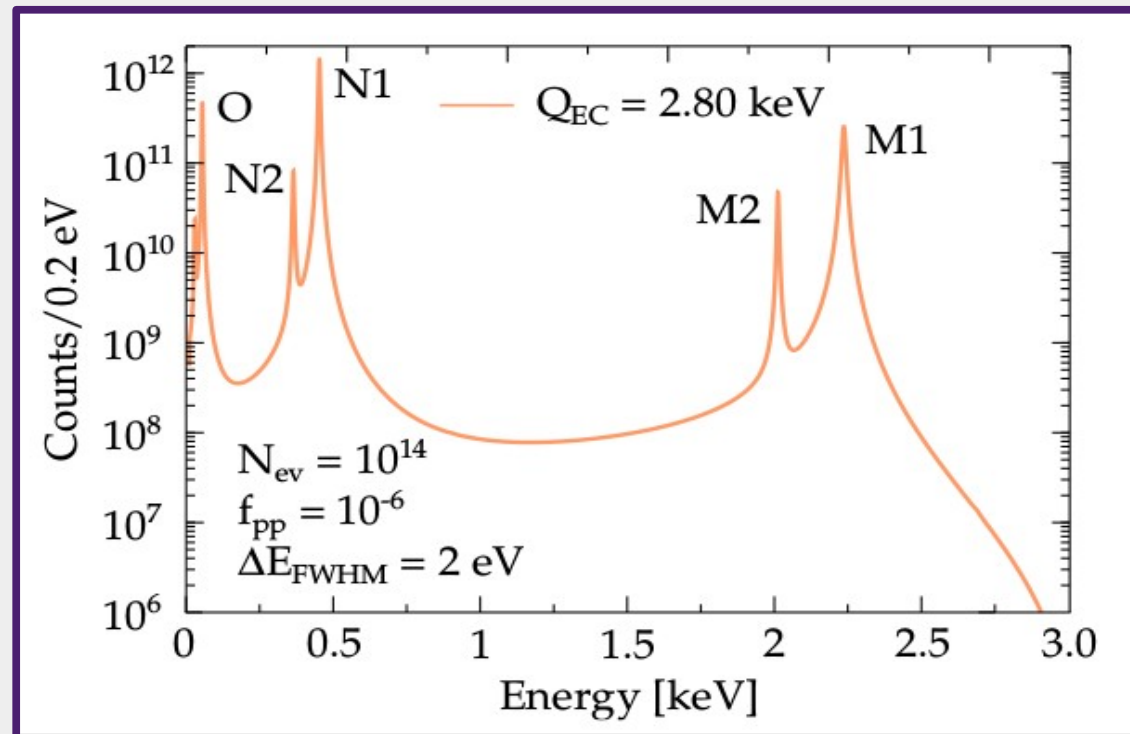
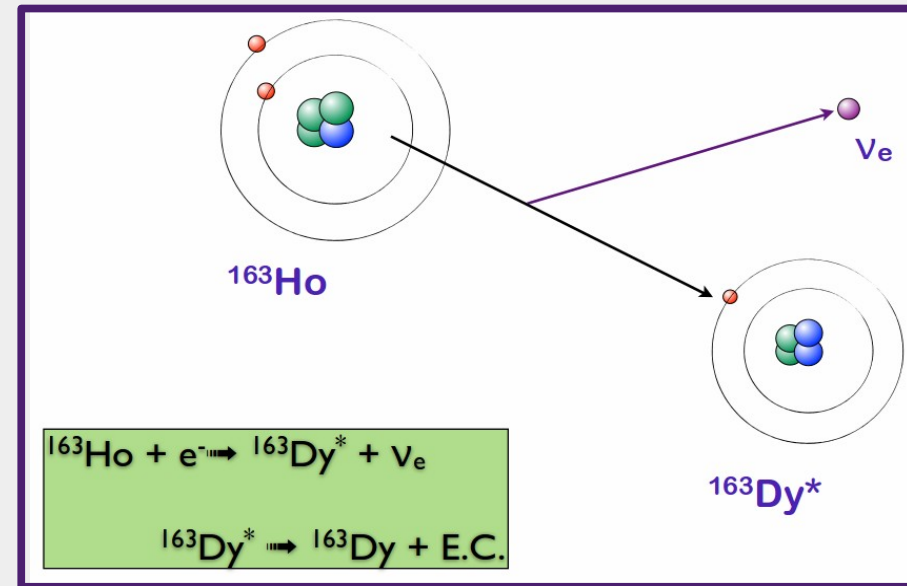
$$\begin{aligned} \frac{dN}{dE_e} = & \frac{G_F^2 m_e^5 \cos^2 \theta_C}{2\pi^3 \hbar^7} |M_{\text{nuc}}|^2 F(Z, E_e) p_e E_e \\ & \times \sum_{i,k} |U_{ei}|^2 P_k(E_{\text{max}} - E_e - V_k) \sqrt{(E_{\text{max}} - E_e - V_k)^2 - m_{\nu i}^2} \\ & \times \Theta(E_{\text{max}} - E_e - V_k - m_{\nu i}) \end{aligned}$$

# Direct measurements of effective neutrino mass

- Electron capture from  $^{163}\text{Ho}$ 
  - Holmes, Echo, NuMecs
- $T_2$  beta decay
  - Mainz/Troitsk
  - KATRIN: Karlsruhe Tritium Neutrino Experiment
  - [Project 8](#)

# Decay of $^{163}\text{Ho}$ by electron capture

- Proposed in 1982 by DeRejula and Luisignoli.
- $^{163}\text{Ho}$  decays to  $^{163}\text{Dy}$  by EC. De-excitation emission is measured by xray calorimetry.
- Holmes, ECHo, NuMECS experiments.
- Measurement of distortion of EC xray spectrum at end point due to  $m_\nu$ .
- Resolution is presently defined by xray calorimetry and atomic calculations.



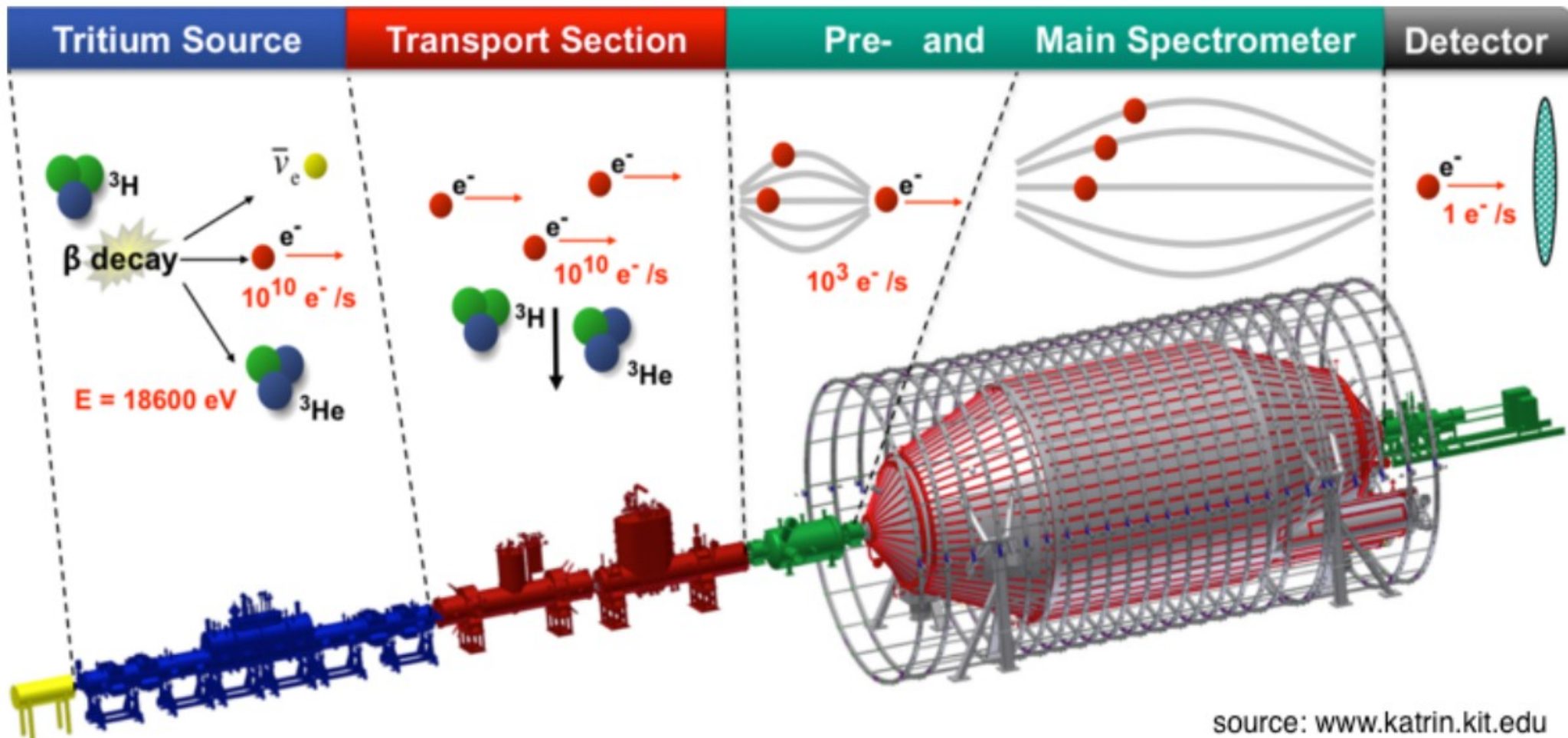


# KATRIN

Diameter of spectrometer is 10 m.

Required vacuum is  $10^{-11}$  mbar.

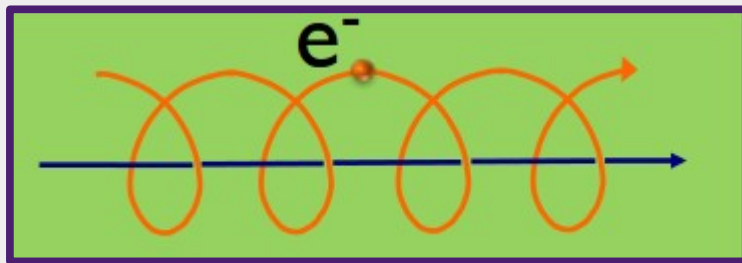
Projected sensitivity is  $m_\nu < 0.2$  eV.





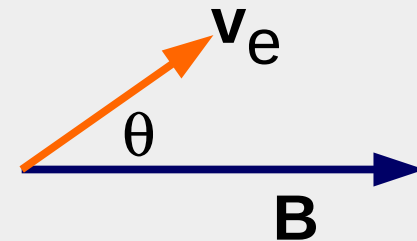
# Project 8: Cyclotron Radiation Emission Spectroscopy (CRES)\*

Pioneered by the Project 8 collaboration in 2015. Measure energy of single electrons indirectly by detecting boosted cyclotron frequency:



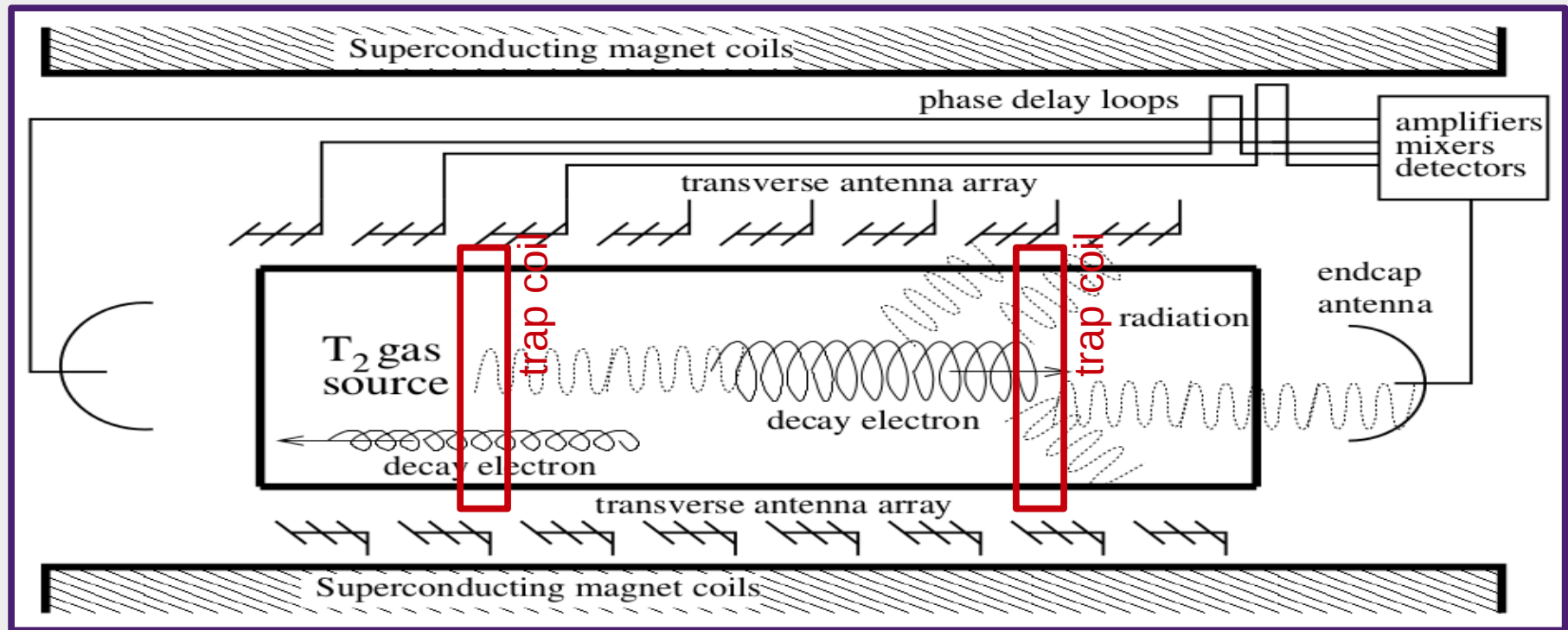
$$f_{\gamma} \equiv \frac{f_c}{\gamma} = \frac{eB}{2\pi\gamma m_e}$$

$$P(\gamma, \theta) = \frac{1}{4\pi\epsilon_0} \frac{2}{3} \frac{e^4}{m_e^2 c} B^2 (\gamma^2 - 1) \sin^2 \theta$$



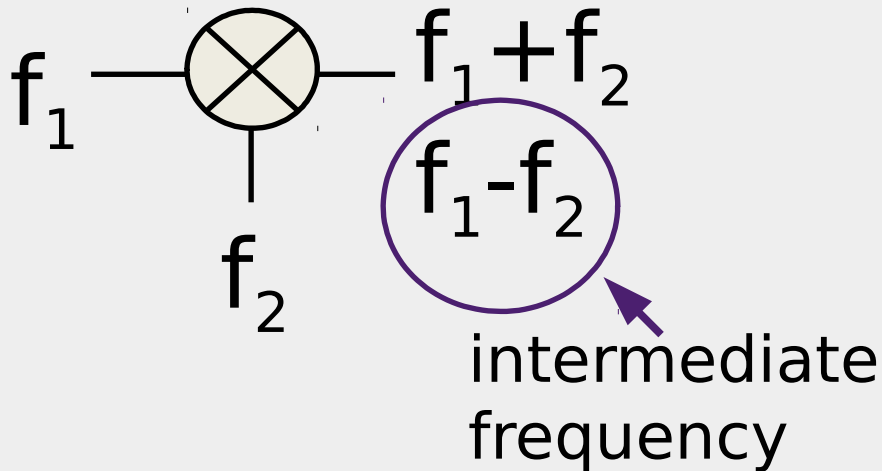
Emitted power depends on pitch angle  $\theta$  but cyclotron frequency does not.

# Cyclotron Radiation Emission Spectroscopy (CRES)



- Trap electrons emitted from a radioactive source gas in a 1 T magnetic field and collect the cyclotron emission.
- Voltages induced in the antennas are filtered and digitized.
- Time of measurement determines frequency resolution.

# Heterodyne detection



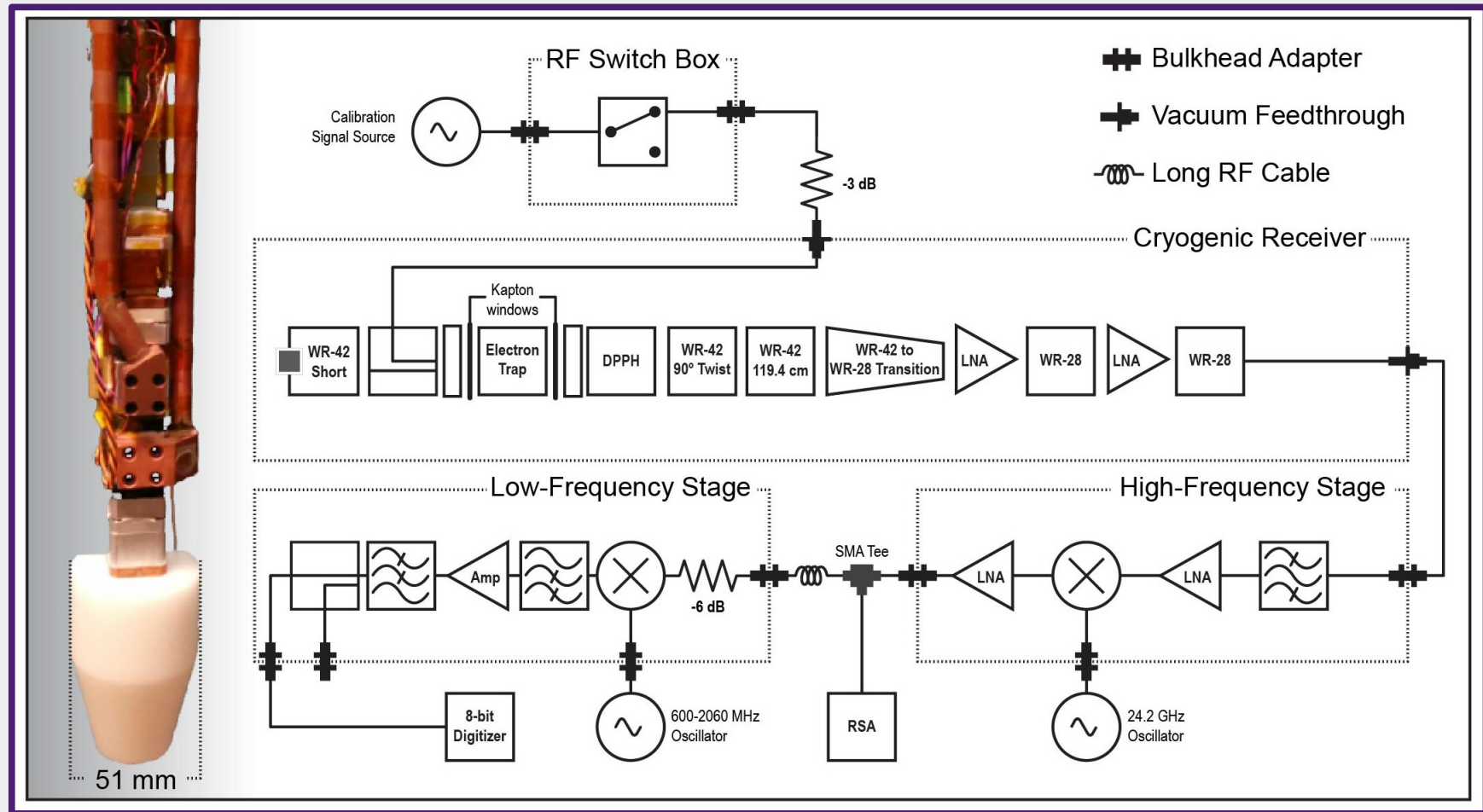
“Never measure anything but frequency.”



Dr. Arthur Schawlow

- Voltages are sampled in time after mixing down from 26 GHz to tens of MHz.
- Frequency spectrum is calculated by FFT.
- 18 keV electrons in 1 T gives SNR of  $\sim 10$  in waveguide.  
Frequency resolution =  $1/\tau$ .

# Project 8 Phase 1 prototype schematic



How can we simulate the  
RF signal generation?

# Status of existing particle physics simulation software packages

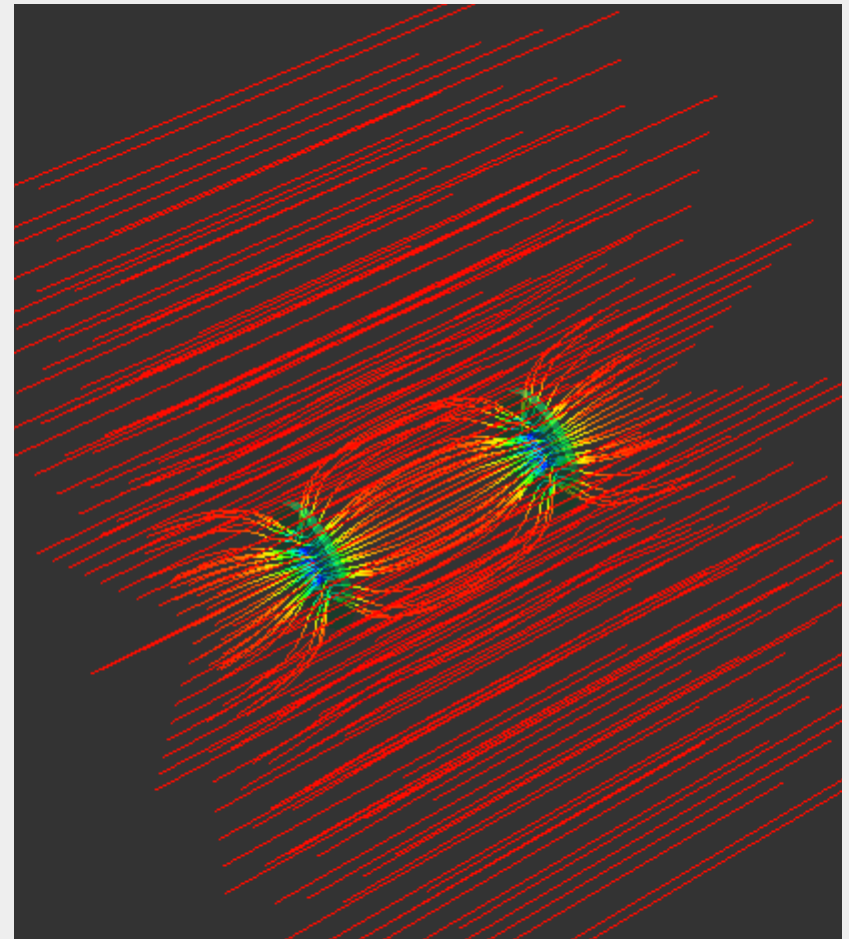
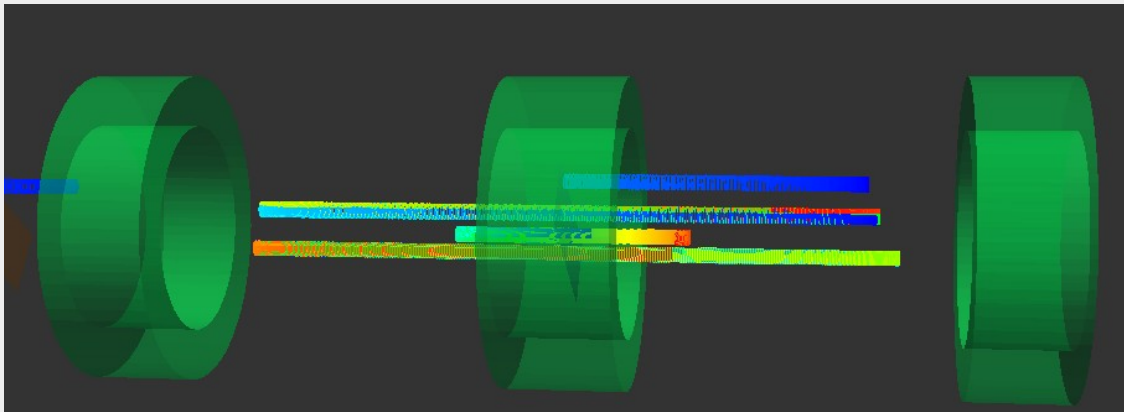
- Geant4: Particle tracking and kinematics
- Comsol and HFSS: EM field solutions
- Kassiopeia: Flexible, modular in-situ field solutions with simultaneous particle trajectory calculations (Furse et al., NJoP 2017).
- Locust: New simulation interface with Kassiopeia to model RF signal generation from particle trajectories, developed for Project 8.

# Trajectory calculations in EM field solutions with Kassiopeia simulation

Simultaneous field solutions and trajectory calculations.

Magnetic bottle created by pinch coils in a constant magnetic background field.

↓  
Trajectories of five 18 keV electrons, four trapped in a magnetic bottle. Field lines not shown. Range of initial  $(x_0, y_0, z_0)$ .



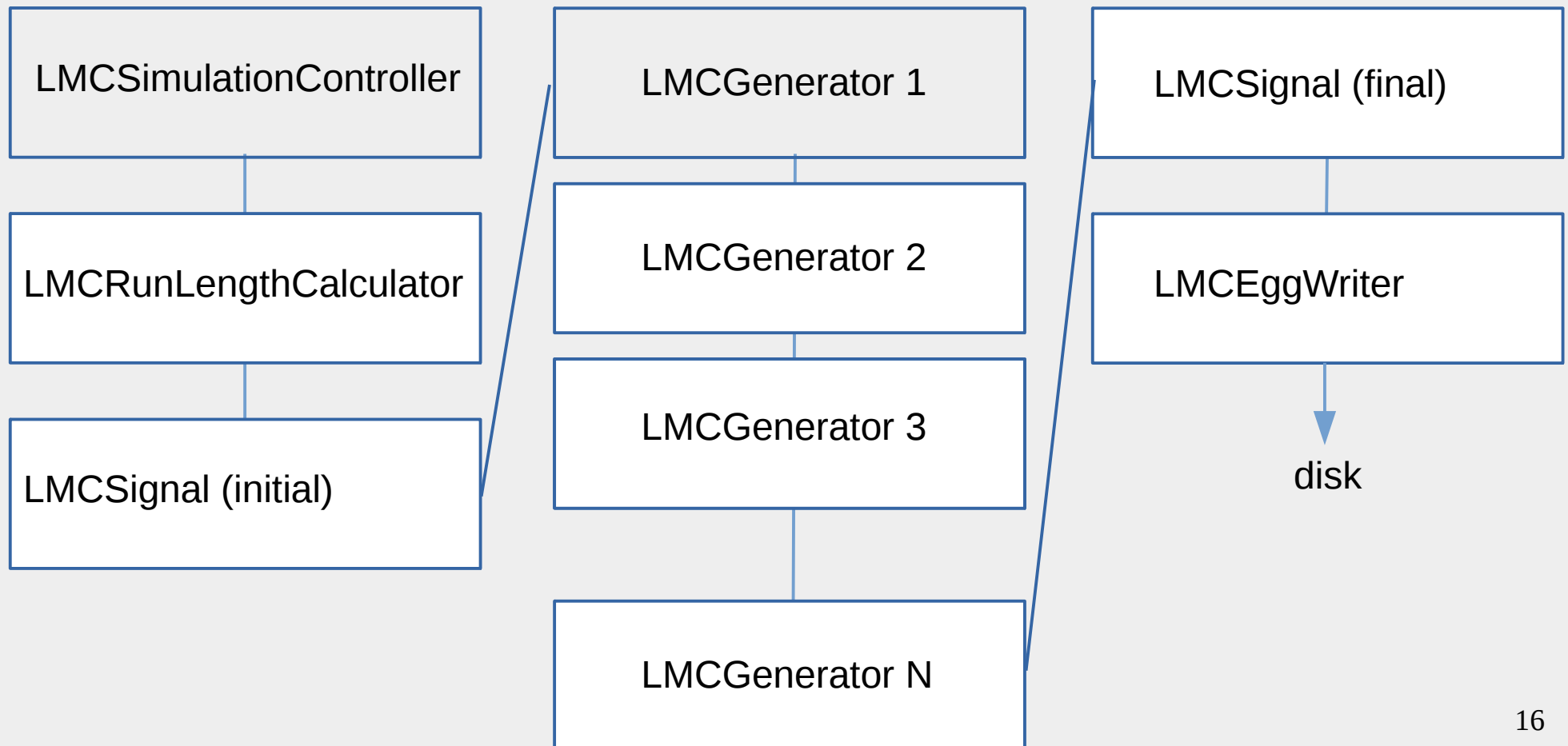


# Locust simulation software block diagram

Configuration parameters are read from the json file.

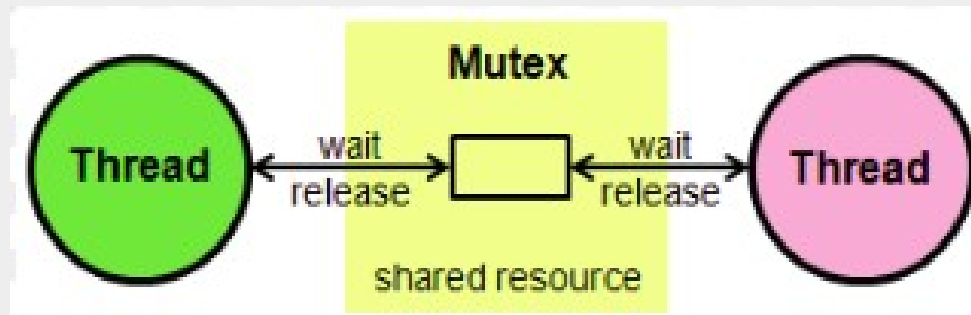
Generators populate and modify the LMCSignal object.

Final LMCSignal object is a time series of voltages to be written to file.



# How do the Kassiopeia and Locust simulations interact?

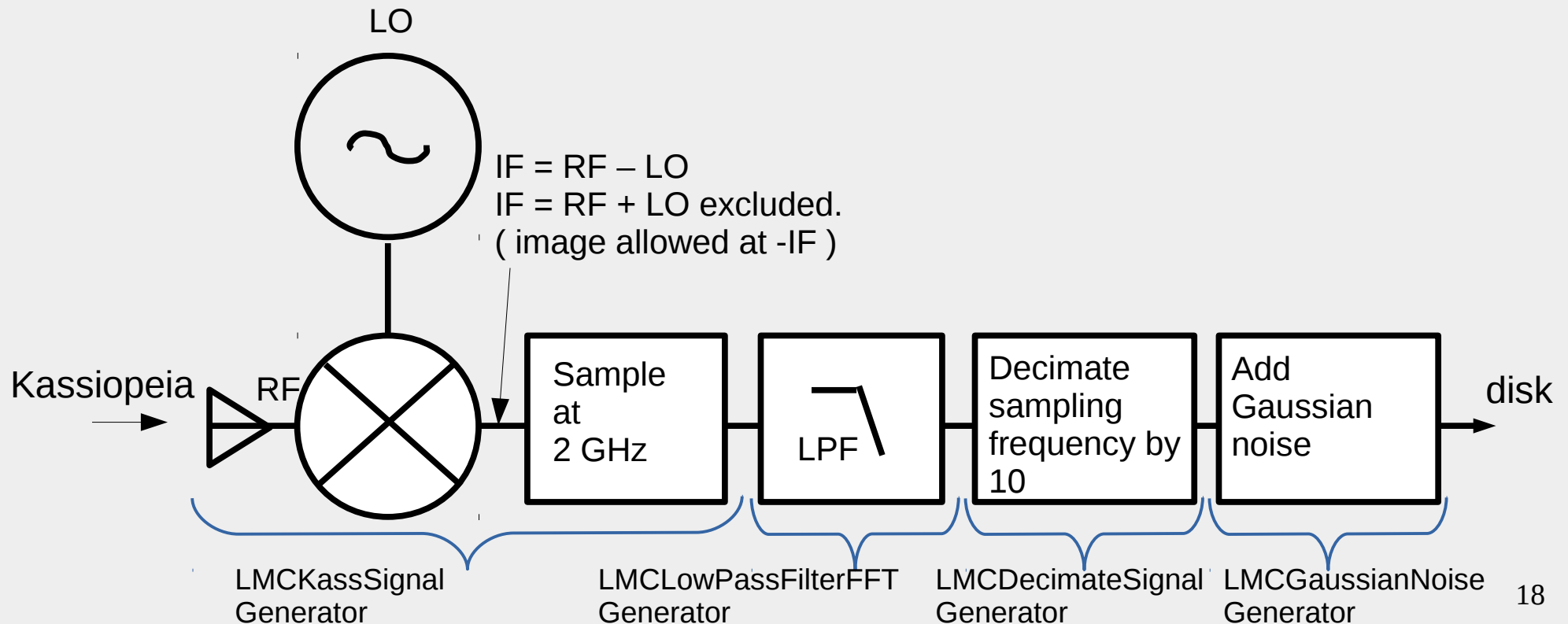
- They are compiled into one executable.
- Only one of them runs at a time, while the other locks control of a `std::mutex` object.
- Kassiopeia runs for e.g. 0.5 ns at a time, then unlocks the mutex. Locust takes control of the mutex, calculates one voltage, and then hands control of the mutex back to Kassiopeia.



# Locust simulation example

## RF block diagram

- Use the energy losses reported by Kassiopeia to guide input.
- Generate antenna voltages at RF sampling frequency. Mix down to baseband with algorithmic receiver.



# Why do we need the RF receiver block diagram in the simulation?

- Lab experiment has an RF receiver. The main purpose is to lower the signal frequency from e.g. GHz to MHz for efficient sampling and data management.
- Signal generated with Locust should have the same format as the empirical data. This provides a comparison for lab data analysis.
- Signal post-processing needs a signal as input. Signal properties constrain information that can be derived from signal processing.
  - e.g. Information that might be in the signal: voltage amplitude, voltage phase.
  - Information that might not be in the signal: large bit depth for higher accuracy, phase continuity, missing energy.
  - Signal properties (or missing properties) can help tell us whether the experiment will work (or not).

# List of examples in this tutorial

- Visual Tool Kit (VTK) output.
- Single electron in a waveguide with one antenna.
- Driving the simulation with a tunable test signal.
- Larger number of single electrons in a waveguide, run in parallel on hpc.
- Single electron in an array of multiple antennas.
- Checking the magnetic field map.

# Steps to get started

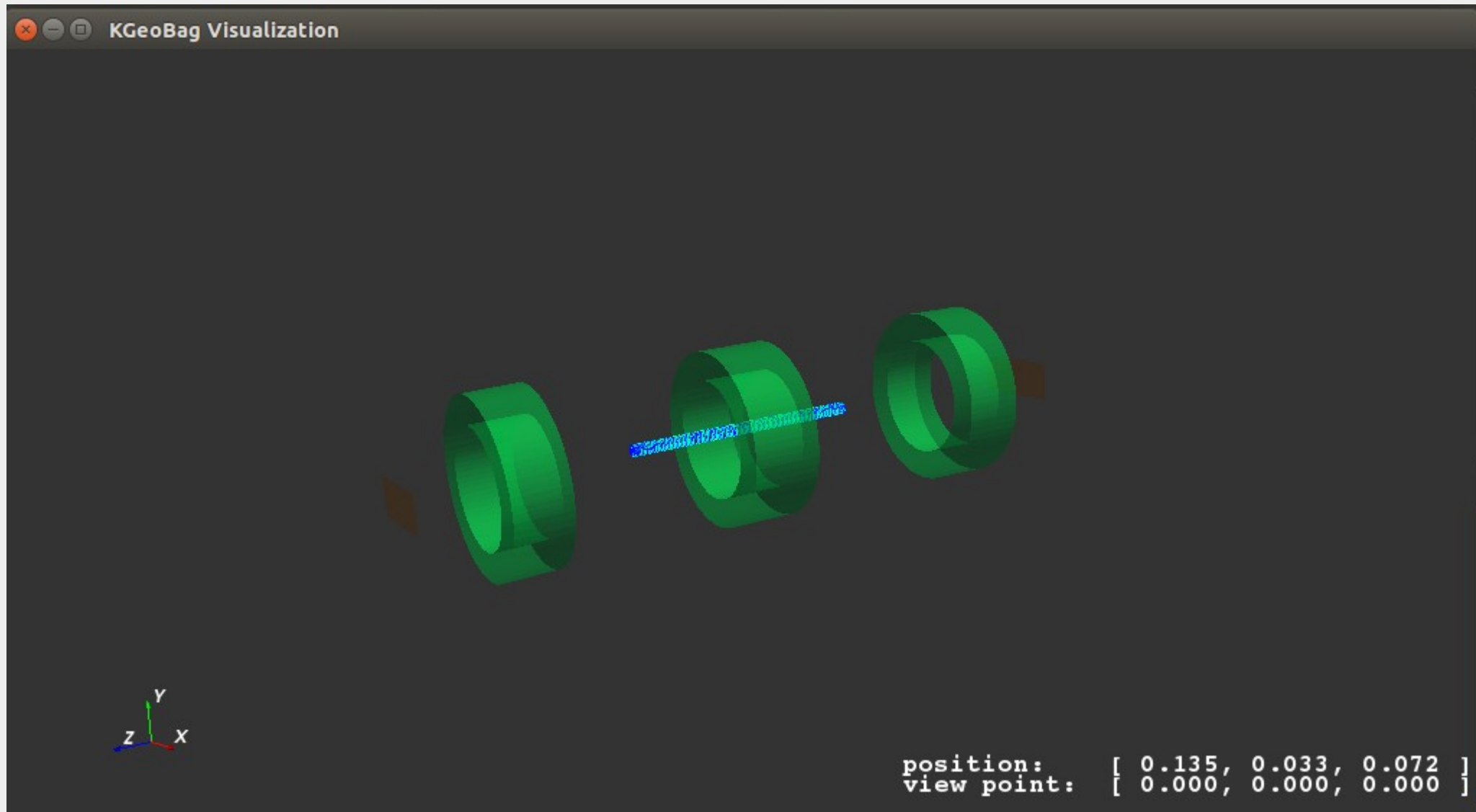
- `ssh -Y netID@grace.hpc.yale.edu`
- `module load Tools/nnpss`
- `setup_dirs.sh`
- `cd project8/manageTutorial`
- `cp /home/hep/baker/ps48/project8/manageTutorial/* .`
- `cp /home/hep/baker/ps48/locust_mc/Config/NNPSSTutorial/*  
~/locust_mc/Config/NNPSSTutorial/`

# Example 0: Look at VTK output of a trajectory calculation with Kassiopeia

- 1) ssh to Grace cluster: `ssh -Y netID@grace1.hpc.yale.edu .`
- 2) Log in to compute node: `srun --x11 --pty -c 4 -p interactive bash`
- 3) `LocustSim config =`  
`~/locust_mc/Config/NNPSSTutorial/LocustTutorialTemplate.json`
- 4) Output should appear as on next slide if:
  - Kassiopeia has been compiled with VTK.
  - The vtk window has been instantiated in the xml file.
  - OpenGL version on your laptop is compatible with that on the cluster.
  - Otherwise this example is more suitable to run on a standalone laptop.



# Example 0: Look at VTK output of a trajectory calculation with Kassiopeia, cont.



# Logistics and examples of simulated RF signal generation

# Questions before and after Locust

- Questions to ask before running Locust:
  - How much of the physics should appear in the generated signal?
  - Is there other physics that will not appear in the generated signal?
  - How will we tell the difference?
- Questions to ask after running Locust:
  - Did we measure the physics that we needed?
  - If not, how should we change the detection hardware? Describe the hardware change algorithmically in Locust, and generate a new signal.

# Some more detailed questions before/after running Locust

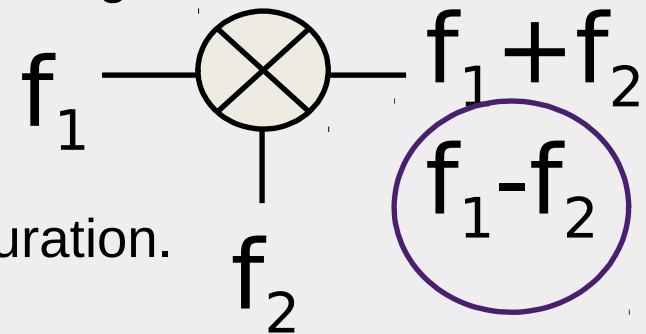
- Did we detect the signal?
  - Is the local oscillator tuned to put the signal in our baseband window spanning from  $-f_{\text{Nyquist}}$  to  $+f_{\text{Nyquist}}$ ?
  - An easy way to check this is to interrupt the simulation and print the RF frequency to the terminal.  $\text{IF} = \text{RF} - \text{LO}$ .
  - Otherwise check the processed data to look for the signal.

- Was the digitizer range set correctly?

- Locust prints ~100 digital voltages at completion.
- Look for values of 0 and 255. These indicate saturation.
- Too many 127 and 128s mean too much range.
- Adjust the digitizer range in the json file “digitizer” parameters.

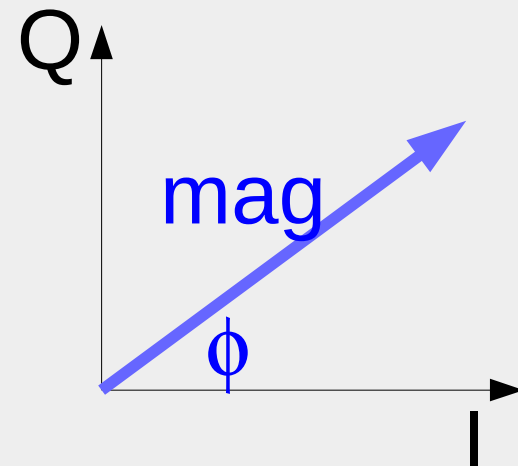
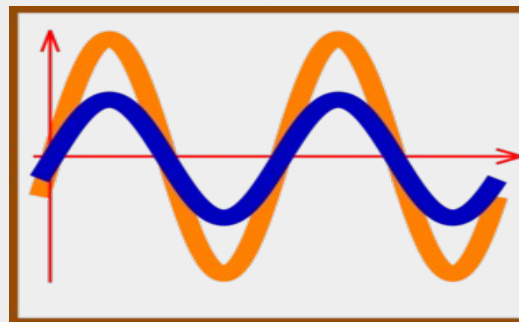
- What is the digitizer bit depth?

- It is hard-wired to 8 bits at the moment.



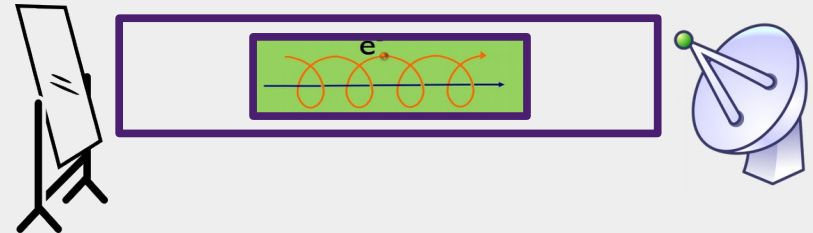
# What are we actually going to do?

- Generate a time series of complex voltages (LMCSignal object).
- Each voltage sample will have a real part (I) and an imaginary part (Q).
- First we calculate the voltage I sampled at time t with phase  $\phi(t)$ , and then we derive Q from it by lagging the voltage phase  $90^\circ$ .
- In the offline data stream we will have the voltage magnitude and phase information.  $\phi = \text{atan}(Q/I)$ .



# Locust example 1: Magnetically trapped electron radiates in a waveguide with one antenna

- What is the signal?
  - Power radiated by the electron.
- How will we detect it?
  - Generate voltages on an antenna by, e.g.,  $P = V^2/R$ .
- What is the voltage amplitude  $V$ ?
  - Consider the power in the waveguide mode(s) that propagate to the antenna.
  - $V = \sqrt{50\Omega P}$ , if it is changing with time we have AM.
- What is the voltage phase  $\phi$ ?
  - Require  $\phi$  to advance continuously as  $\phi(t) \pm f'(t)$ ; it changes as FM.
  - $f'$  is the frequency observed right at the antenna, including Doppler shift.
- What physics is not in the signal?
  - Waveguide modes that do not propagate.
  - Frequency of radiation observed anywhere but at the antenna.
  - We might reconstruct the missing physics with post-simulation analysis.



# Locust example 1: Magnetically trapped electron radiates in a waveguide with one antenna, cont.

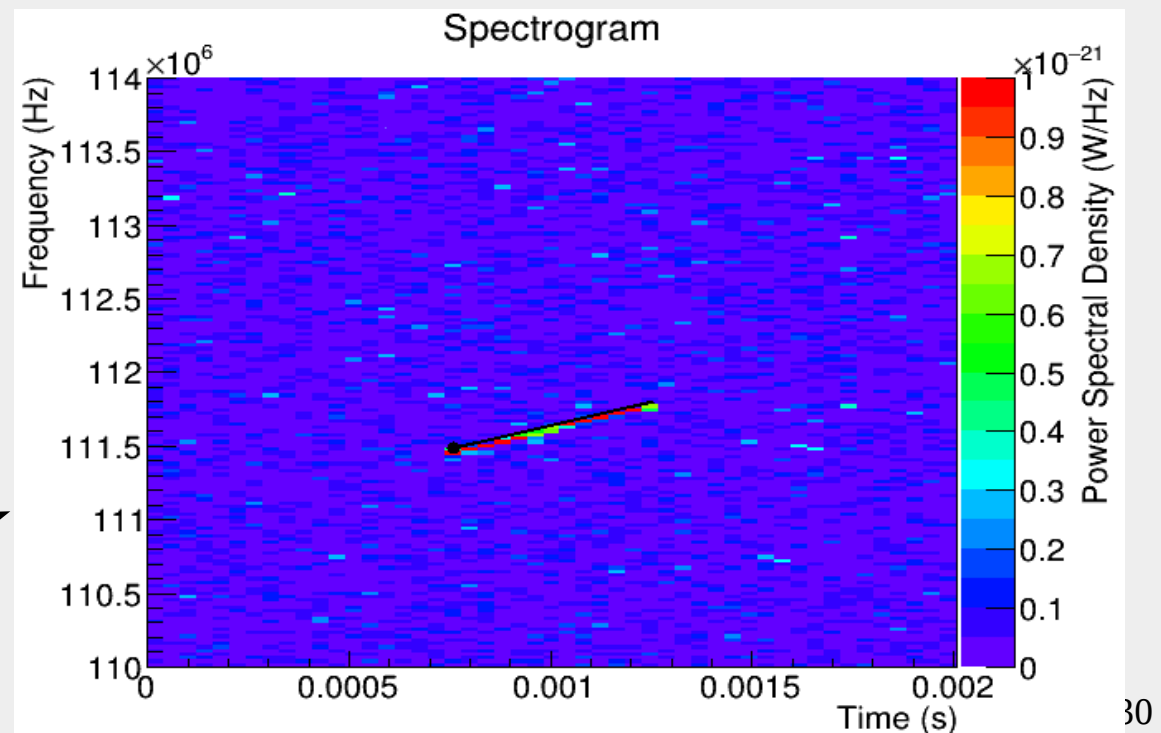
- How do we implement this in software?
  - Inside the Locust generator LMCKassSignalGenerator we have populated the LMCSignal object with a time series of complex voltages.
    - $V_I(t) = V_0(t)\cos(\phi(t)); V_Q(t) = V_0(t)\sin(\phi(t))$
    - $V_0(t)$  and  $\phi(t)$  are calculated as on previous slide.
    - Sampling rate is chosen according to bandwidth needs.
- How do we run this on Yale HPC?
  - `cd ~/project8/manageTutorial`
  - `emacs SimulateSeed &`, note 4 cpus, output file.
  - `./SimulateSeed`
  - Wait 30 minutes. Check that the file `locust_jobSeed55` is growing.



# Locust example 1: Magnetically trapped electron radiates in a waveguide with one antenna, cont.

- Check to see whether the slurm job has finished, and that there is an egg file in ~/data/SimulationTutorial.
  - `squeue -u netID`
  - `ls -l ~/data/Simulation/Tutorial/*.egg`
- Log on to a compute node: `srunch --x11 --pty -c 4 -p interactive bash`
- `cd ~/project8/manageTutorial`
  - `./ProcessOneSpectrogram`
  - `root -l`
  - `.L PlotExample()`
  - `PlotTrack()`
  - `PlotStartFreqs()`
  - `PlotSlopes()`

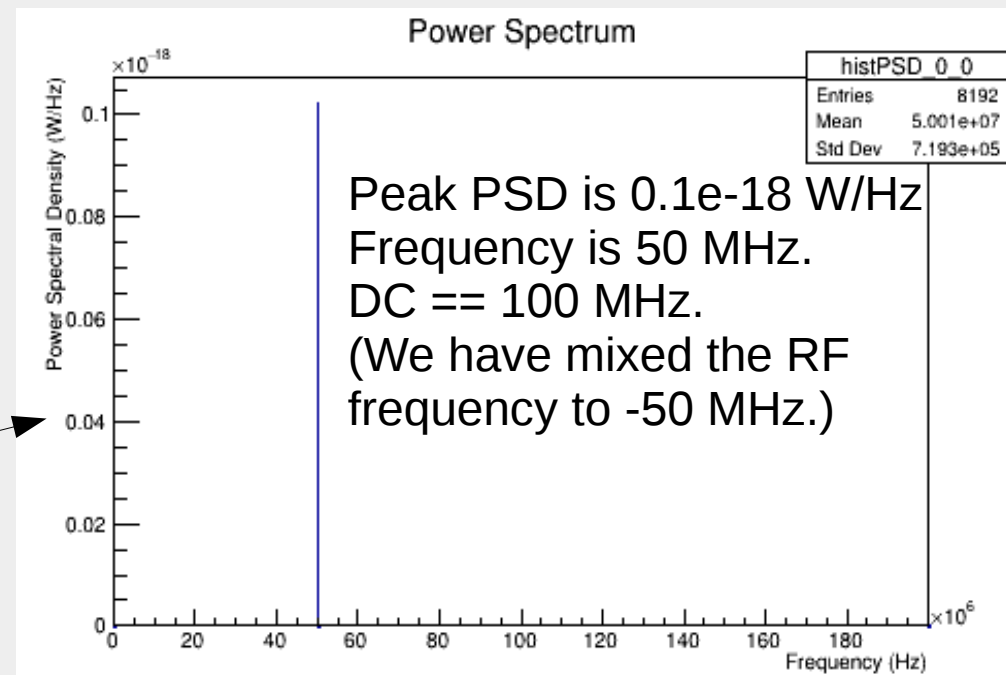
output



# Locust example 2: Drive the receiver with a complex sine wave

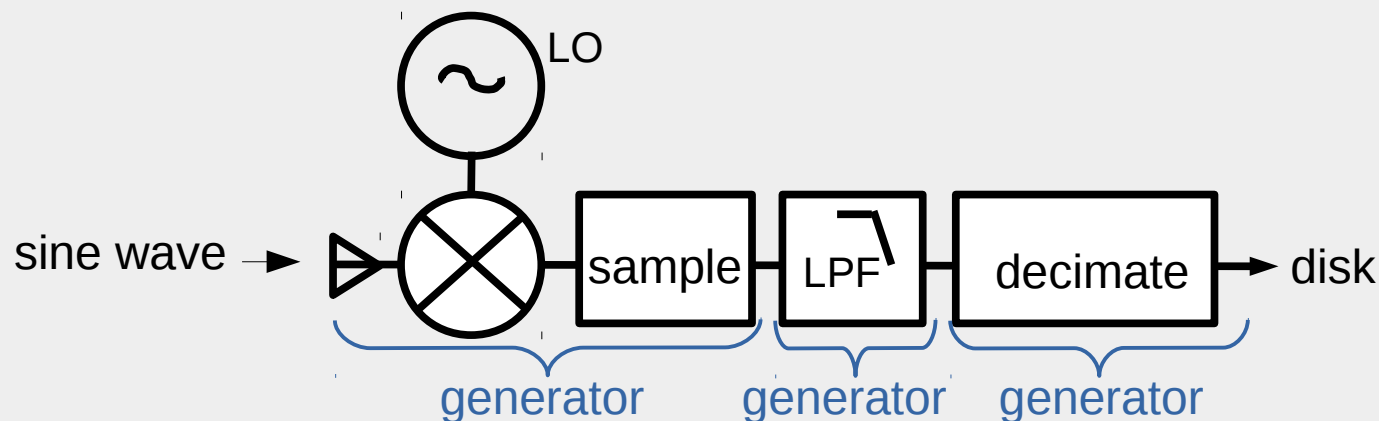
- Log on to a compute node: `srun --x11 --pty -c 4 -p interactive bash`
- Open the sine wave config file: `emacs ~/locust_mc/Config/NNPSSTutorial/LocustSineWave.json` &. Check which generators are listed, see that “lo-frequency” = 20.15 GHz.
- Open the sine wave generator: `emacs /home/hep/baker/ps48/locust_mc/Source/Generators/LMCTestSignalGenerator.cc` , scroll to `DoGenerateTime()`. Find `test_frequency = 20.1e9 Hz`.
- `LocustSim config=~/locust_mc/Config/NNPSSTutorial/LocustSineWave.json`
- `Katydid -c ~/locust_mc/Config/NNPSSTutorial/katydid_basic.json`
- `cd ~/project8/manageTutorial`
  - `root -l`
  - `.L PlotExample.c`
  - `PlotTestSignal()`

output



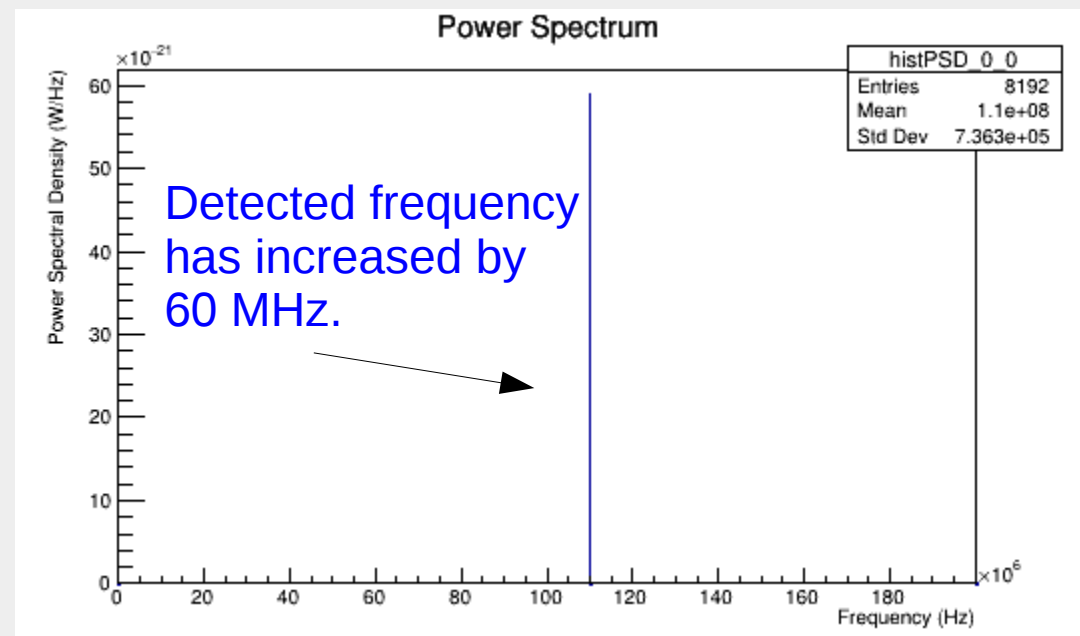
# Locust example 2: Check the frequency and normalization of the complex sine wave

- Open the generator: [emacs](#)  
[/home/hep/baker/ps48/locust\\_mc/Source/Generators/LMCTestSignalGenerator.cc](#) &
- Scroll to DoGenerateTime() function and look at this line: `aSignal->LongSignalTimeComplex()[ch*aSignal->TimeSize()*aSignal->DecimationFactor() + index][0] += sqrt(50.)*5.e-8*cos(voltage_phase-LO_phase);`
- We are modifying the (multichannel, complex, oversampled, in-phase) LMCSignal object by adding a voltage to it. The antenna impedance is  $50\Omega$ , the voltage amplitude is  $5.e-8$  volts, and the frequency after downmixing will be (test\_frequency - fLO\_frequency).
- Power in the sine wave should be  $V^2/R = (5.e-8)(5.e-8)/50. = 2.5e-15/50. \text{ W}$ . Checking the PSD histogram we see  $\text{PSD} = 0.1e-18 \text{ W/Hz}$  with a bin width of  $200.e6/8192$ , for a total power of  $2.44e-15 \text{ W}$ . The  $50\Omega$  has been applied in Katydid post-processing.
- Signal frequency is 50 MHz, which in this post-processing configuration, is 50 MHz below DC. The frequency span shown corresponds to -100 MHz through 100 MHz.



# Locust example 2: Try tuning the LO (complex sine wave, continued)

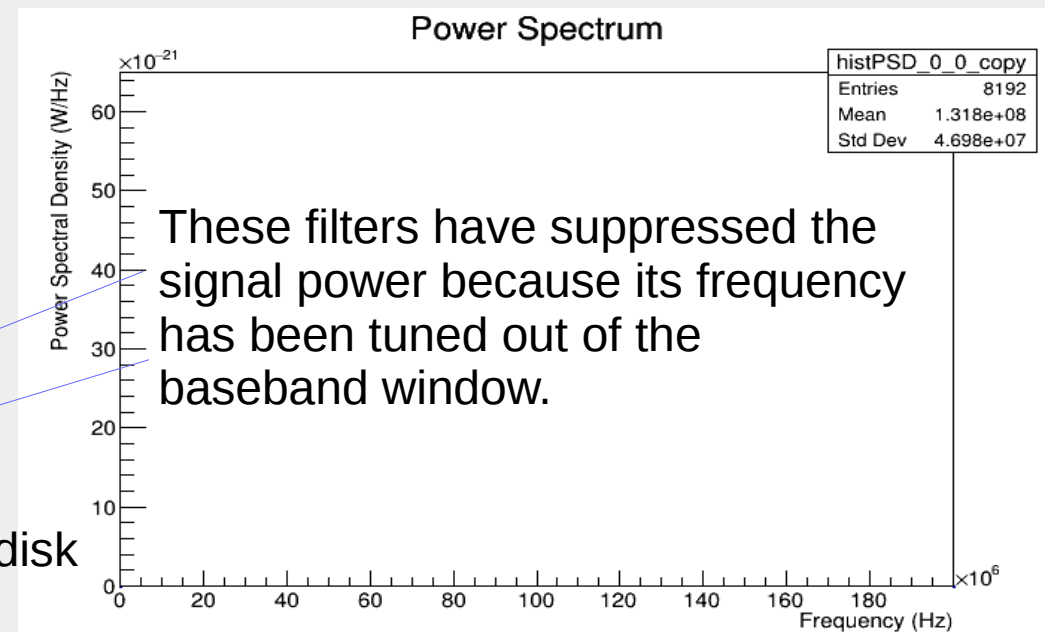
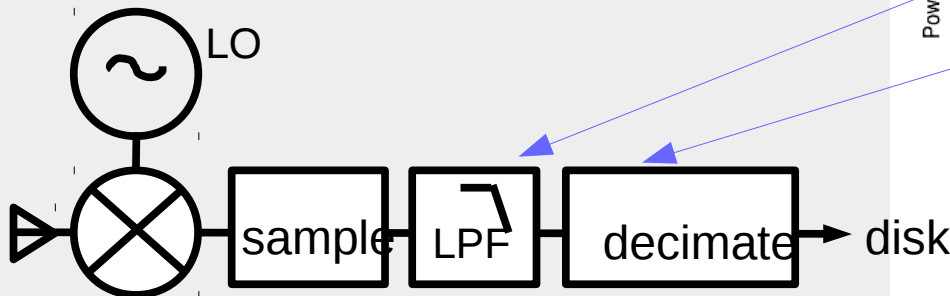
- `emacs ~/locust_mc/Config/NNPSSTutorial/LocustSineWave.json`
  - Look at the config parameters in the “test-signal” generator. Decrease the “lo-frequency” by 60 MHz from 25.15e9 to 25.09e9 Hz.
  - Save the file.
- `LocustSim`  
`config=~/locust_mc/Config/NNPSSTutorial/LocustSineWave.json`
- `Katydid -c ~/locust_mc/Config/NNPSSTutorial/katydid_basic.json`
- `cd ~/project8/manageTutorial`
  - `root -l`
  - `.L PlotExample.c`
  - `PlotTestSignal()`



# Locust example 2: Now tune the signal all the way out of the window (sine wave, continued)

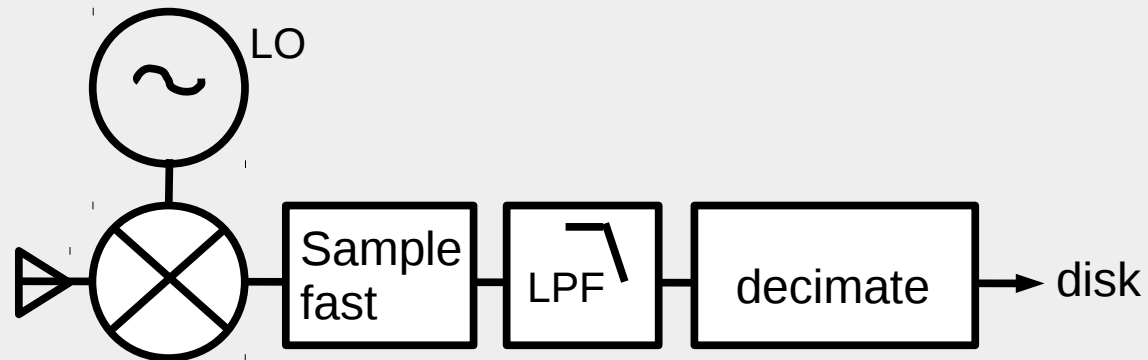
- `emacs ~/locust_mc/Config/NNPSSTutorial/LocustSineWave.json`
  - Look at the config parameters in the “test-signal” generator. Decrease the “lo-frequency” by another 100 MHz from 20.09e9 to 19.99e9 Hz.
- `LocustSim`  
`config=~/locust_mc/Config/NNPSSTutorial/LocustSineWave.json`
- `Katydid -c ~/locust_mc/Config/NNPSS/katydid_basic.json`
- `cd ~/project8/manageTutorial`

- `root -l`
- `.L PlotExample.c`
- `PlotTestSignal()`

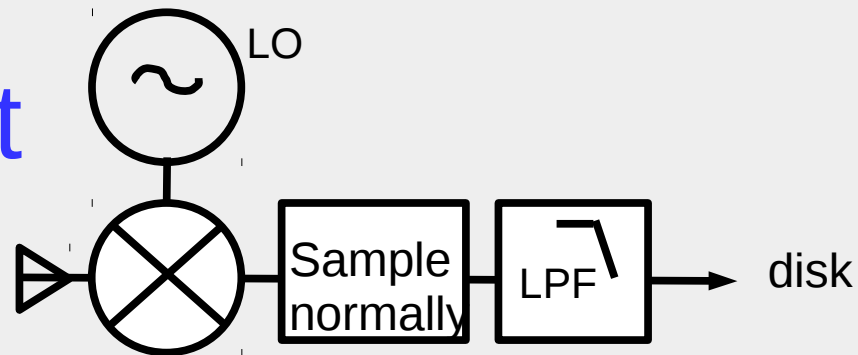


# About the filtering (1)

Why  
this?

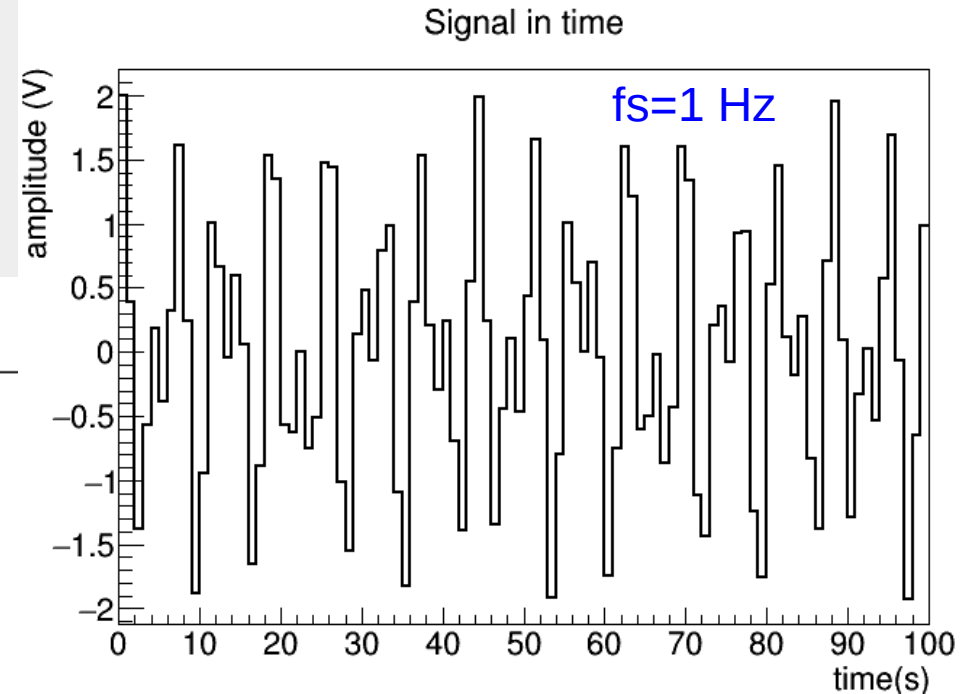
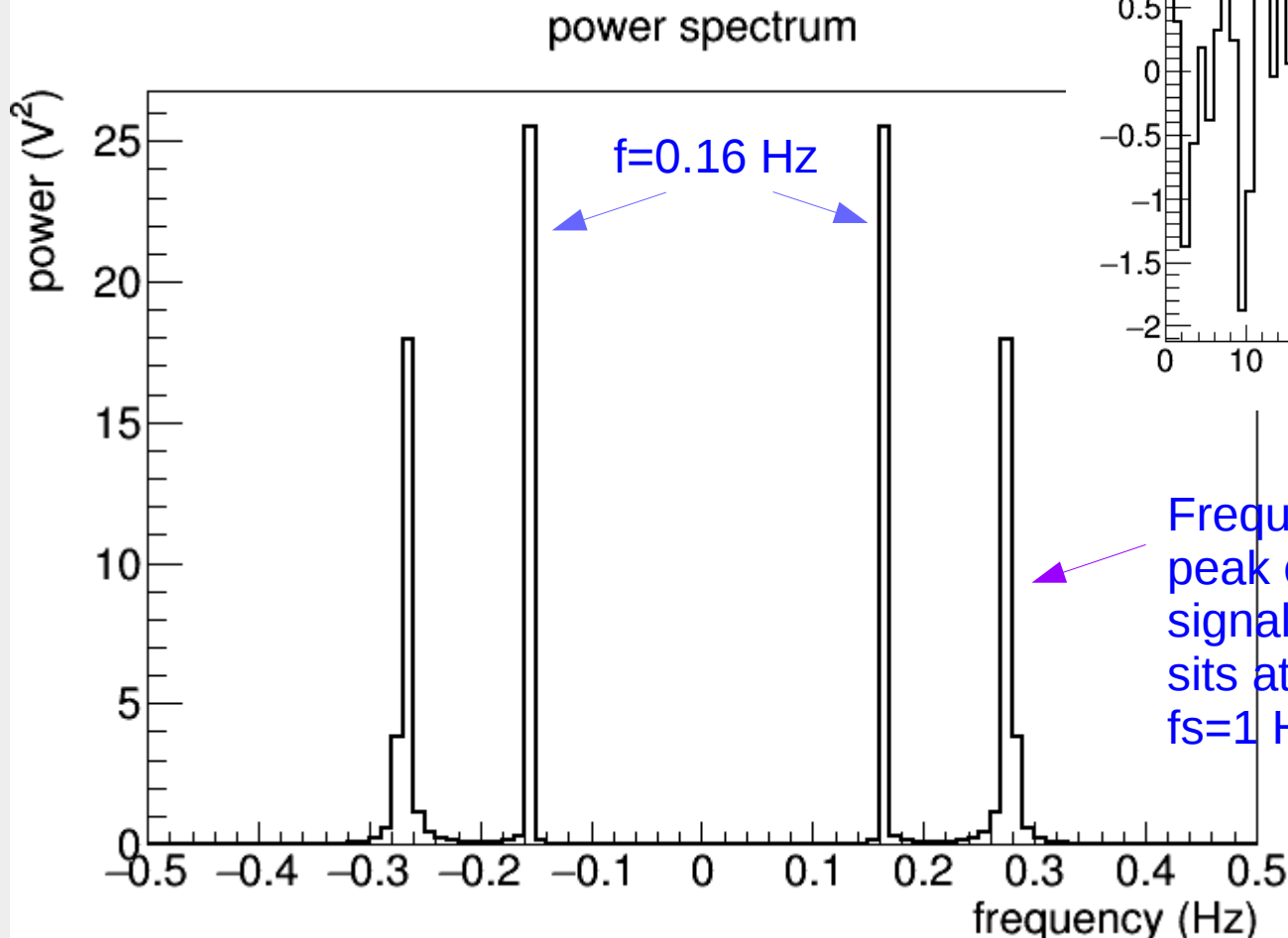


And not  
this?



# About the filtering (2)

Using a test signal with a low ( $1/2\pi$ ) and high frequency ( $8/2\pi$ ) component  
 $V(t) = \cos(t) + \cos(8t)$  we can see what happens to the high-f signal for  $f_s=1$  Hz.

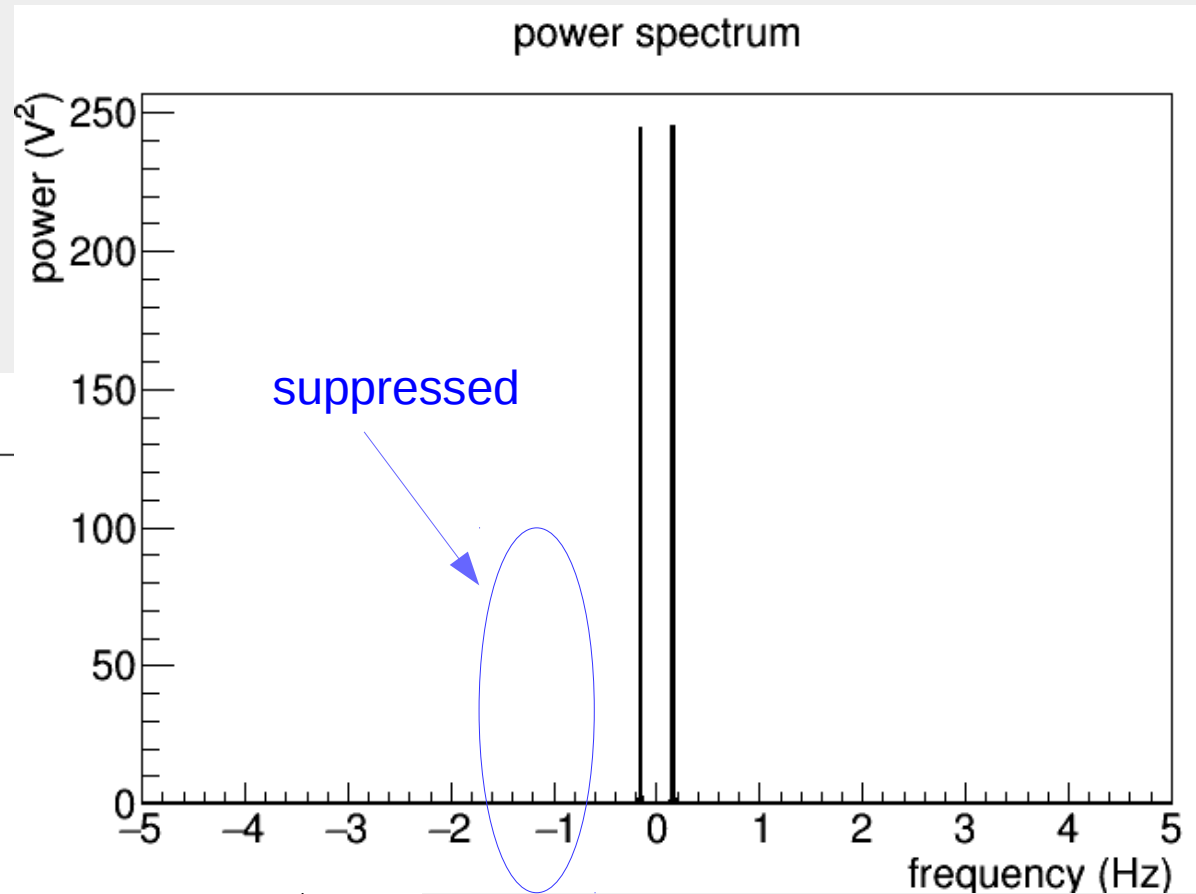
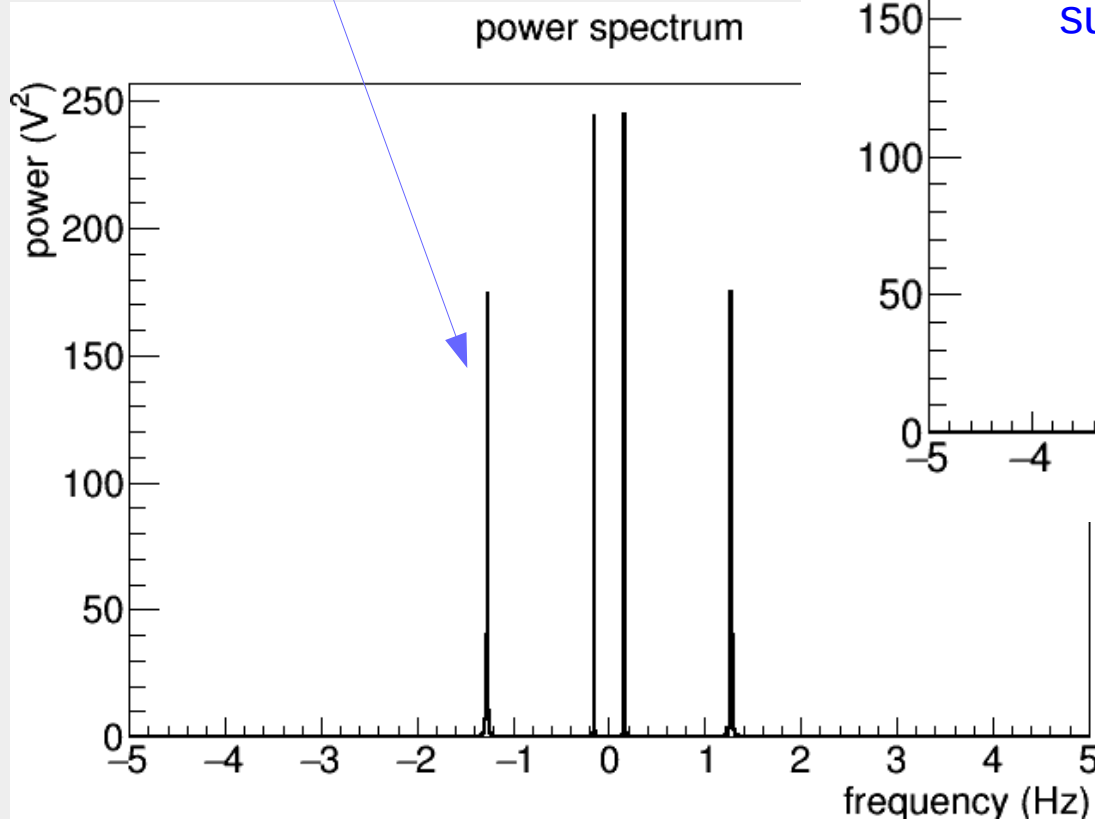


Frequency aliasing: This peak corresponds to the signal with  $f = 1.27$  Hz but sits at  $\sim 0.3$  Hz because  $f_s=1$  Hz is too slow.



# About the filtering (3)

Sampling faster: Now the signal with  $f=(8/2\pi)=1.27$  Hz is measured correctly and can be suppressed with a LPF in the frequency domain at 0.5 Hz.

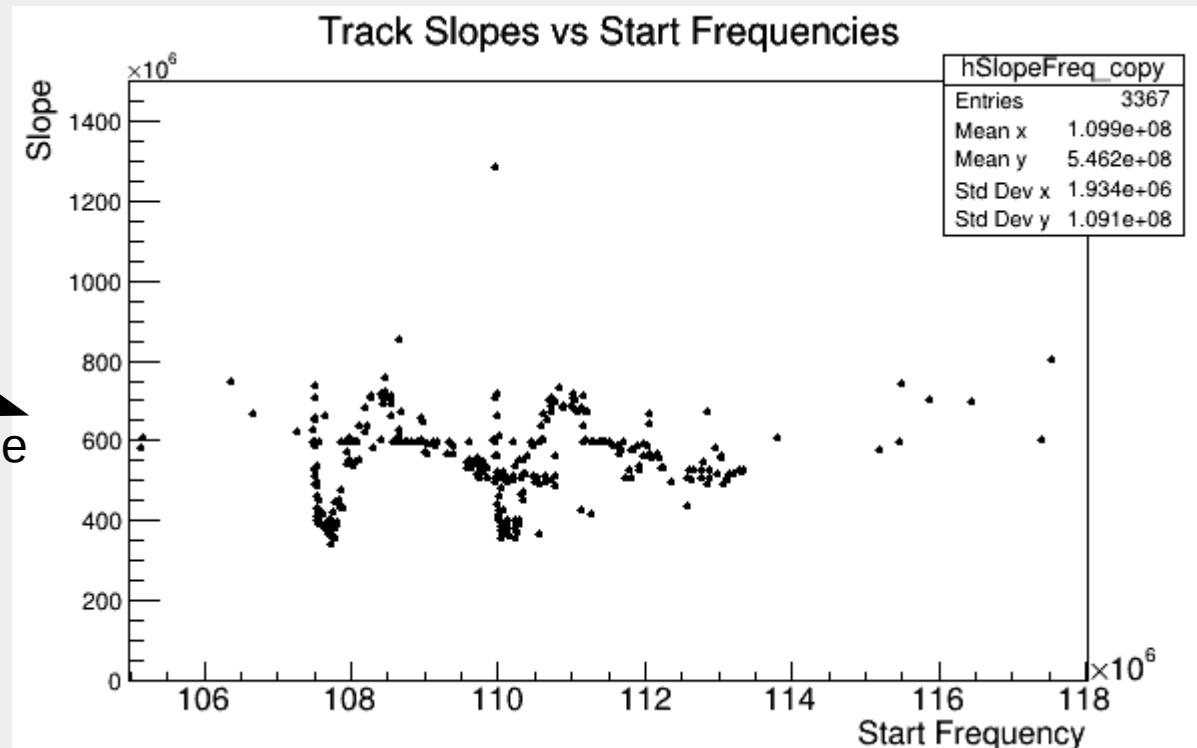


Return to Example 1 to process  
signal and plot results

# Locust example 3: Generating more statistics

- `cd ~/project8/manageTutorial/`
- `emacs SimulateSeed &`
- Look at the range of seeds to be submitted as slurm jobs. Pick 56 through 62. Exit and submit the job: `./SimulateSeed`
- It has to run for 3 hours, but we can process some pre-existing data:  
`sbatch ProcessEggFilesBatch`, then
  - `root -l`
  - `.L PlotSpectrum.c`
  - `PlotKrypton()`

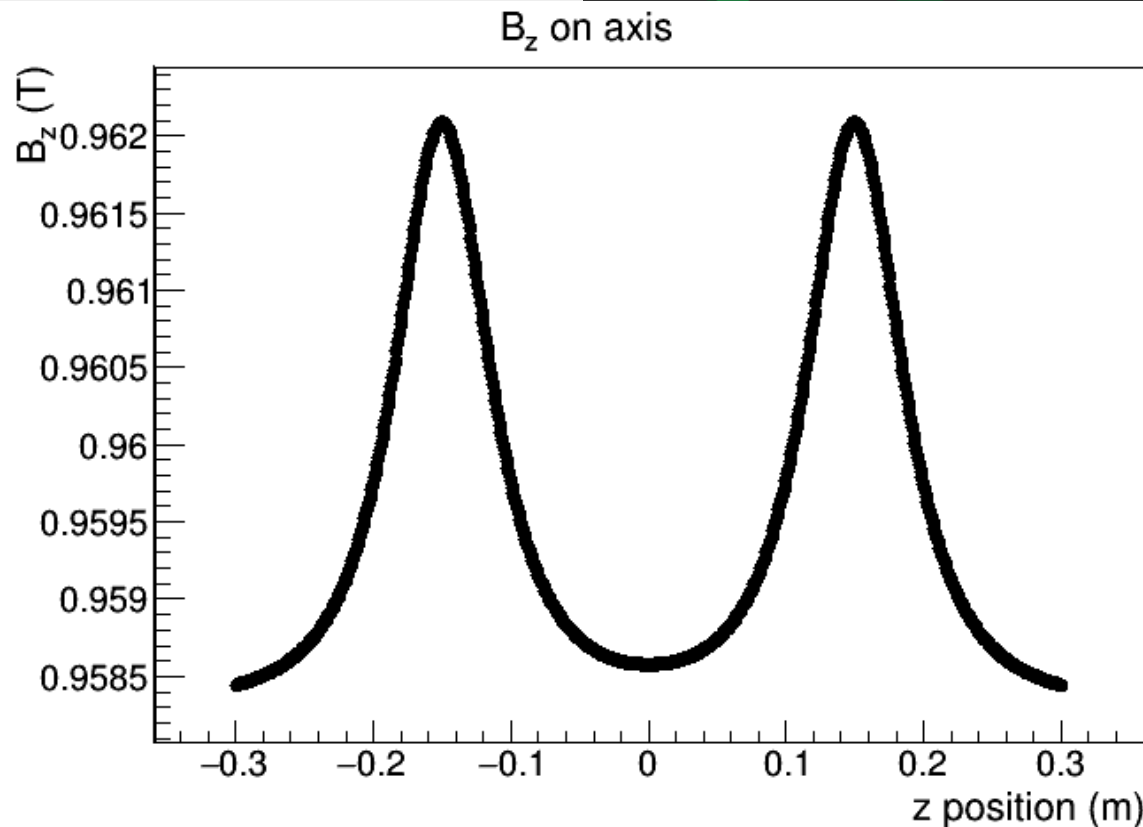
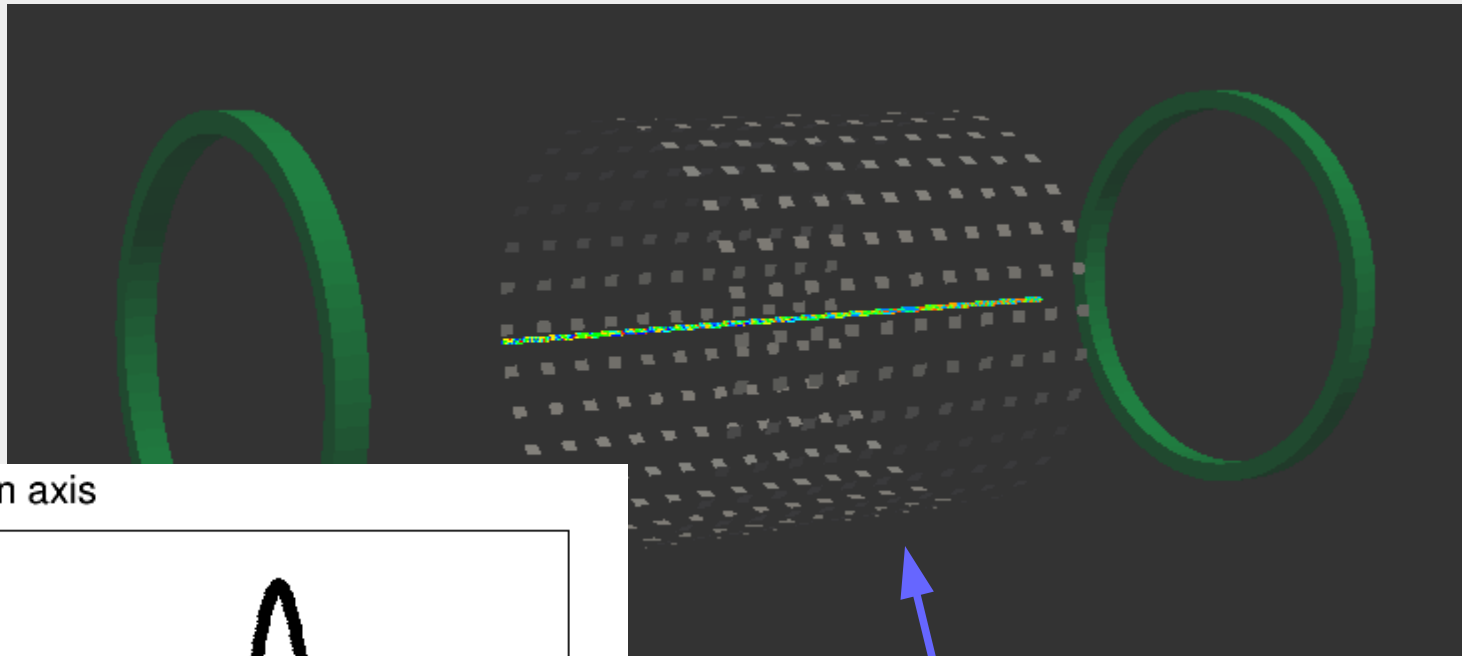
Each point represents one processed track in frequency-time space.



# Locust example 4: Magnetically trapped electron radiates in free space with 30 channels

Patch array radius  
5.16 cm.

Spacing 0.0108 m.



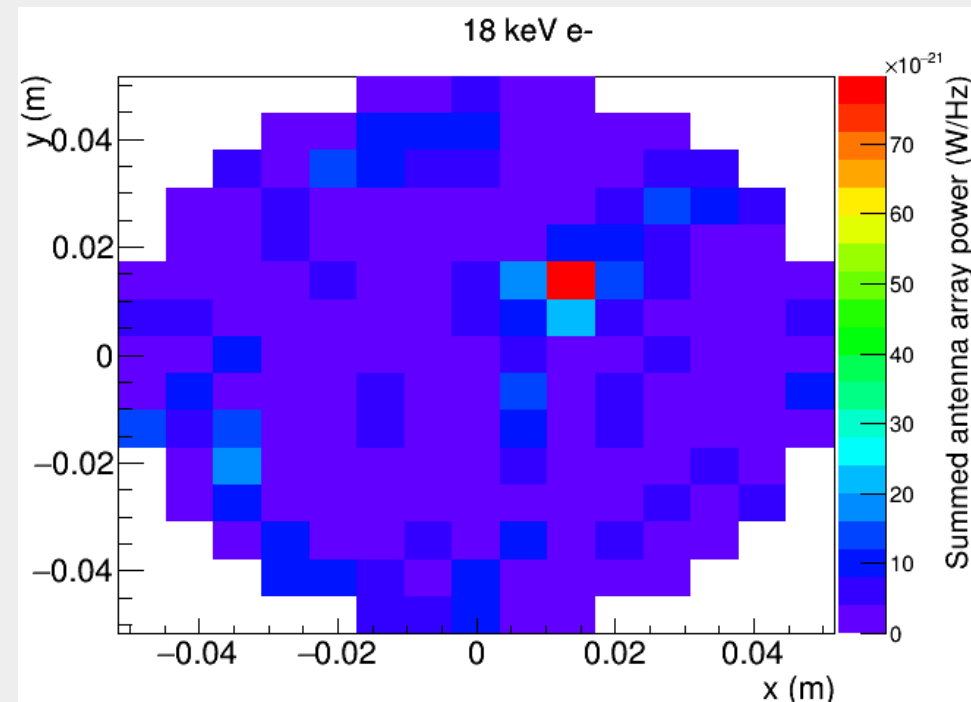
One amplifier per longitudinal strip for phased summing in time domain.

# Locust example 4: Multiple receiver channels

- Check the Locust config file:
  - `emacs ~/locust_mc/Config/NNPSSTutorial/LocustPhase3Template.json`
  - Find the “simulation” config parameters, where “n-channels” is set to 30.
  - Check which generators will be called: “patch-signal”, “lpf-fft”, “decimate-signal”, “digitizer”.
- Check the slurm bash script:
  - `emacs ~/project8/manageTutorial/SimulateMultichannel`
  - Notice we request 12 cpus, which is more than for the single channel simulation.
  - Note ntask=1 still, even with multiple threads and channels.

# Locust example 4: Magnetically trapped electron radiates in free space with 30 antennas

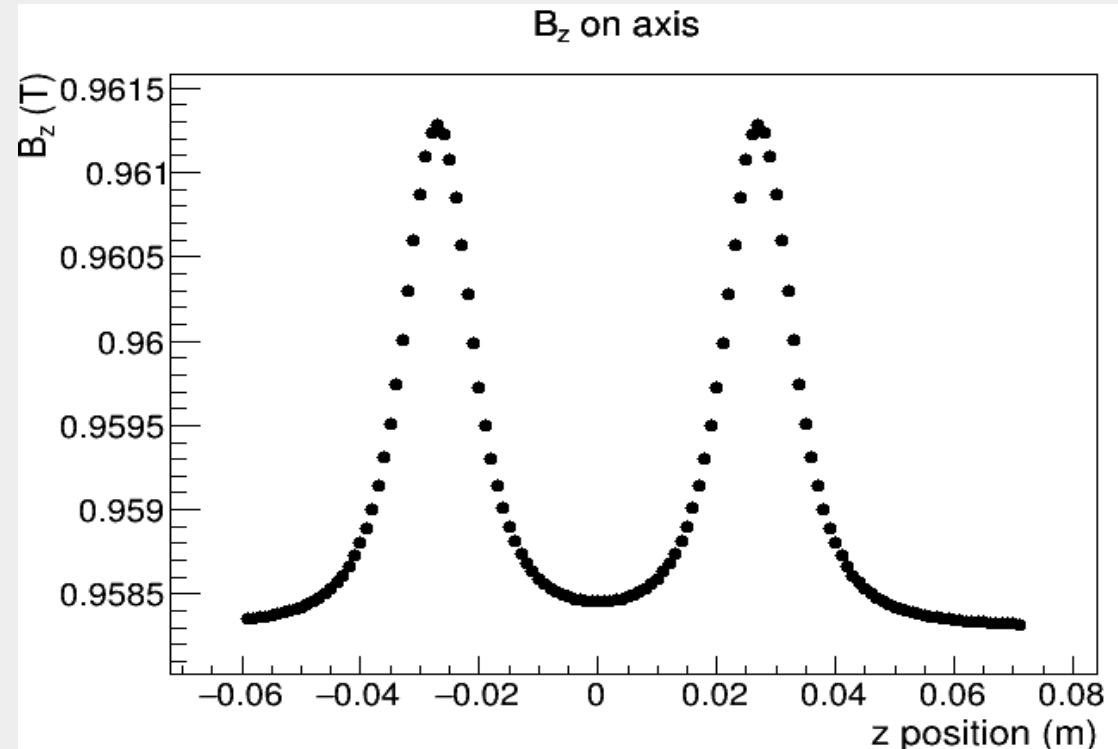
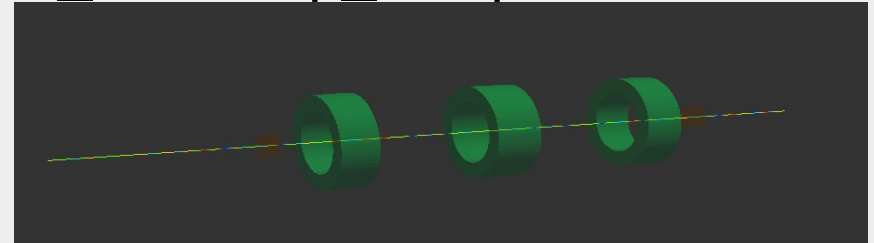
- Run the simulation:
  - `cd ~/project8/manageTutorial`
  - `./SimulateMultichannel`
  - `queue -u netID`
  - ~3 hours later it will finish.
- Process pre-existing data
  - `./ProcessMultichannelEggFiles`
  - `root -l`
  - `.L ChannelSumming.c`
  - `beamform()`



Reconstructed electron position  
from phase-sensitive sum

# Locust example 5: Check the magnetic field map

- Kassiopeia contains a trajectory `gen_bfieldlines` for a charged particle moving along a selected field line.
- This is implemented in `Project8Tutorial_FieldMap_Template.xml`, called by `Locust_FieldMap.json`
  - `cd ~/project8/manageTutorial`
  - `srun --x11 --pty -c 4 -p interactive bash`
  - `LocustSim config=~/locust_mc/Config/NNPSSTutorial/LocustFieldMap.json`
- Plot the field map:
  - `root -l`
  - `.L PlotFieldMap.c`
  - `fieldmap()`



# Tutorial and scripts in repositories on Github

- Simulation
  - [https://github.com/project8/locust\\_mc.git](https://github.com/project8/locust_mc.git)
  - Configuration files are in the same directory referenced in this tutorial: `~/locust_mc/Config/NNPSSTutorial/` .
- Interpretation scripts
  - <https://github.com/project8/scripts.git>
  - Files in repository directory `~/scripts/NNPSSTutorial/` are the same as those used in this tutorial, in the directory `~/project8/manageTutorial/` .



# Summary

- RF signal generation can be simulated with the Locust simulation.
  - A time series of induced voltages can be calculated one sample at a time.
  - AM and FM modulation can be applied to the voltages.
- Processing the generated signal tells us what information we have detected about an experiment.
  - It's important to know which information we have not detected.
  - It's important to know whether (or not) we can reconstruct any missing information from the signal analysis.