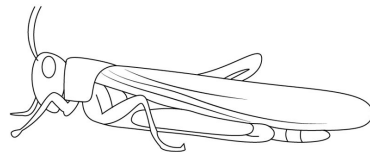# Locust v2.5.1 notes

P. L. Slocum
Aug. 26, 2023

Outline

- <mark>Brief introduction and instructions for setting up</mark>

- Tutorial examples

- Intermediate checks

- Development options

- Config files and hpc cluster tools (presently partially internal).

Setting up

1. Install docker as in the instructions, https://docs.docker.com/engine/install/ .
2. Pull the Locust container: docker pull ghcr.io/project8/locust_mc:latest
3. Pull the p8compute container: docker pull project8/p8compute:latest
4. Create a directory in your home directory, e.g.

   mkdir ~/p8tutorial

   (Steps 5-7 are optional for plotting Root histograms):

5. Pull the p8compute-jupyter container: docker pull project8/p8compute-jupyter
6. Start p8compute-jupyter and leave it open for the duration of the tutorial:
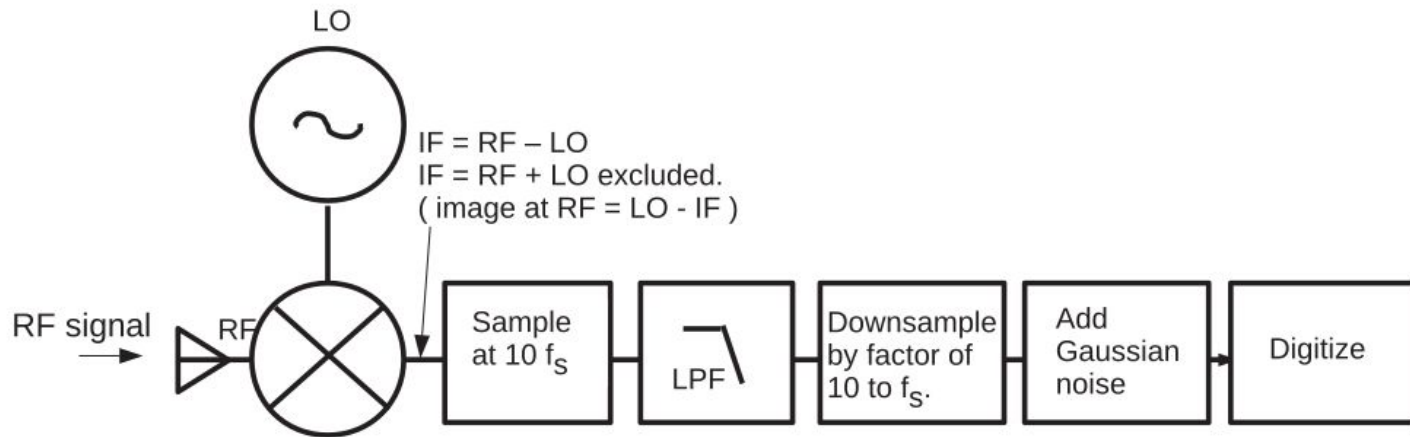
   ```
   docker run -p 8888:8888 -v ~/p8tutorial:/tmp project8/p8compute-jupyter
   ```

7. Open a browser tab using one of the links provided in the resulting terminal output.
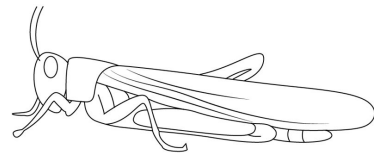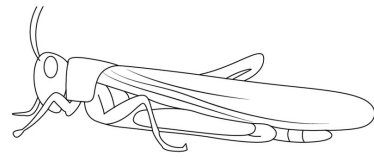
Locust* simulation software

- Developed in C++ within Project 8.
- Integrated with Project 8 DAQ libraries.
- Functionality models an RF receiver and digitizer.
- Interfaces are modular and flexible to allow for arbitrary input signals.

LO

IF = RF − LO
IF = RF + LO excluded.
(image at RF = LO - IF)

RF signal → RF

Sample at 10 f_S

LPF

Downsample by factor of 10 to f_S.

Add Gaussian noise

Digitize

*Project 8, New J. Phys. 21 113051   4

Kassiopeia* simulation software

- Developed by the KATRIN collaboration.
- Highly advanced calculations of electron trajectories and energy losses in EM fields.
- Adaptable EM field solutions with configurable source geometries.
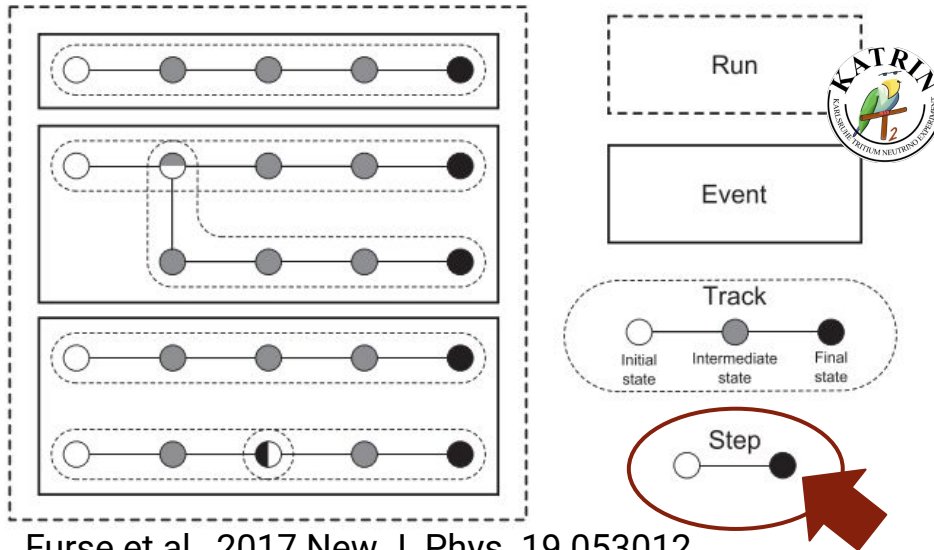- Modular and flexible; C++ based code.

(a) Exact trajectory

(b) Adiabatic trajectory
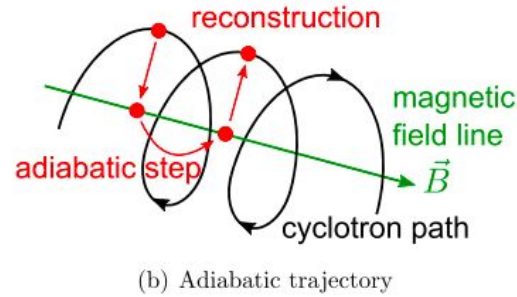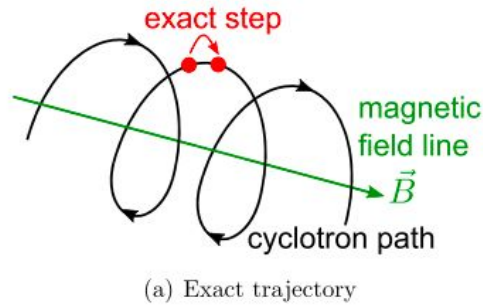
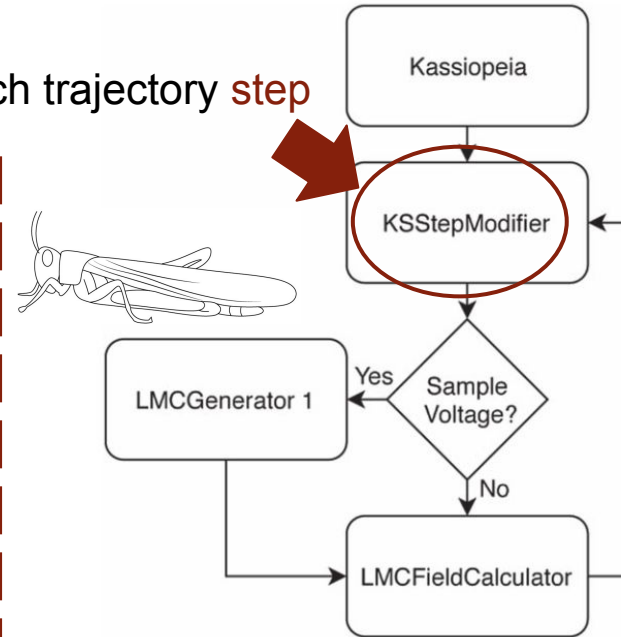*Furse et al., 2017 New J. Phys. 19 053012

Locust-Kassiopeia interface
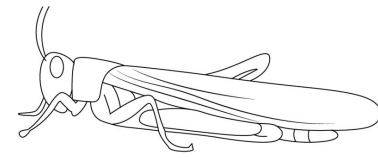
- Modularity in both Locust and Kassiopeia supports tight integration of Locust-Kassiopeia in the time domain.

- Cross-package communication happens after each trajectory step in Kassiopeia.

Furse et al., 2017 New J. Phys. 19 053012
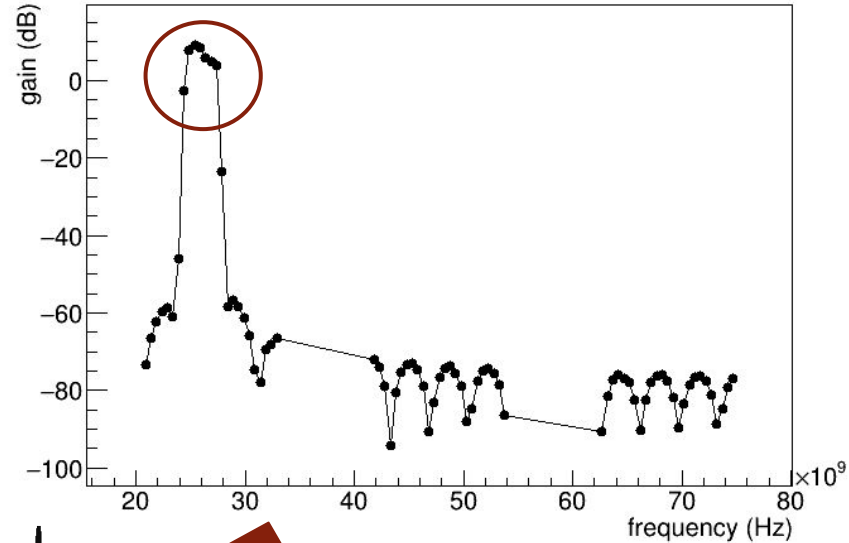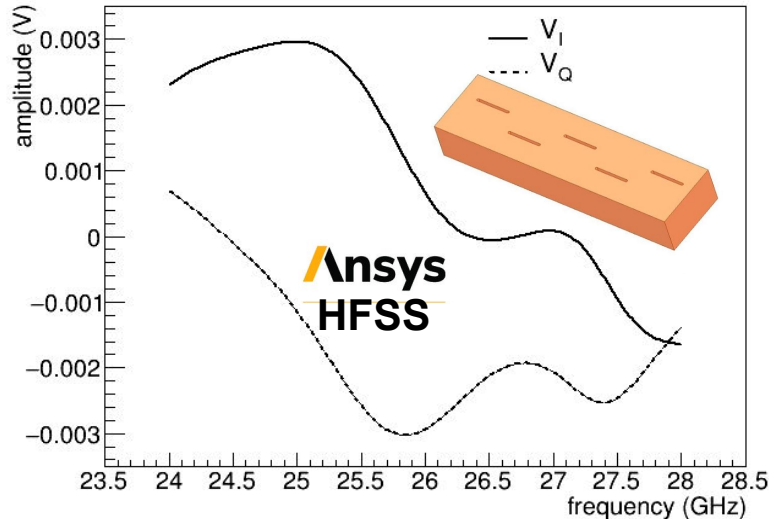
Project 8, 2019 New J. Phys. 21 113051

6

Locust-HFSS interface

Locust antenna

- Interface relies on Linear Time-Invariant (LTI) system theory.

- LTI antenna model is configured in HFSS.

- Locust EM fields drive the LTI antenna FIR.

FIR

Locust work flow diagram
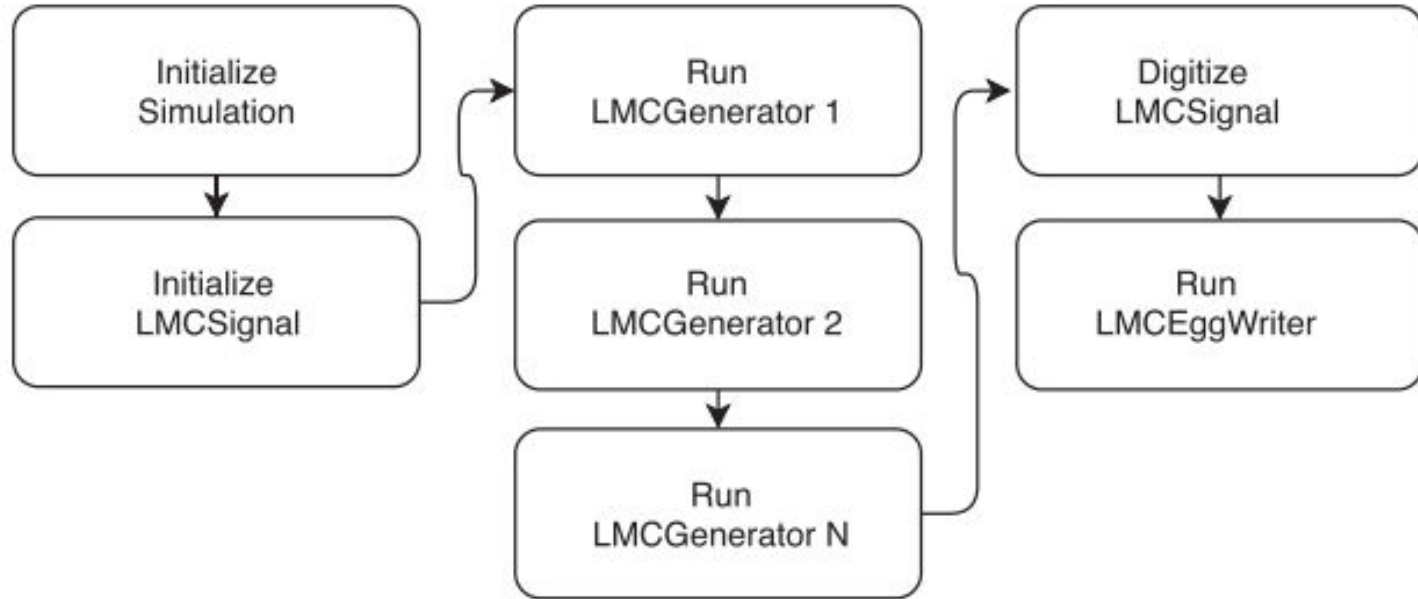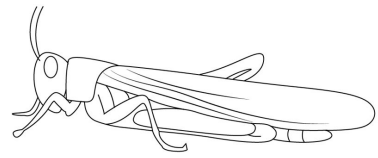
Locust work flow diagram

Cavity and waveguide calculations are handled here.

Initialize Simulation

Initialize LMCSignal

Run LMCGenerator 1

Run LMCGenerator 2

Run LMCGenerator N

Digitize LMCSignal

Run LMCEggWriter

New generator ("cavity-signal") and its main configuration options:
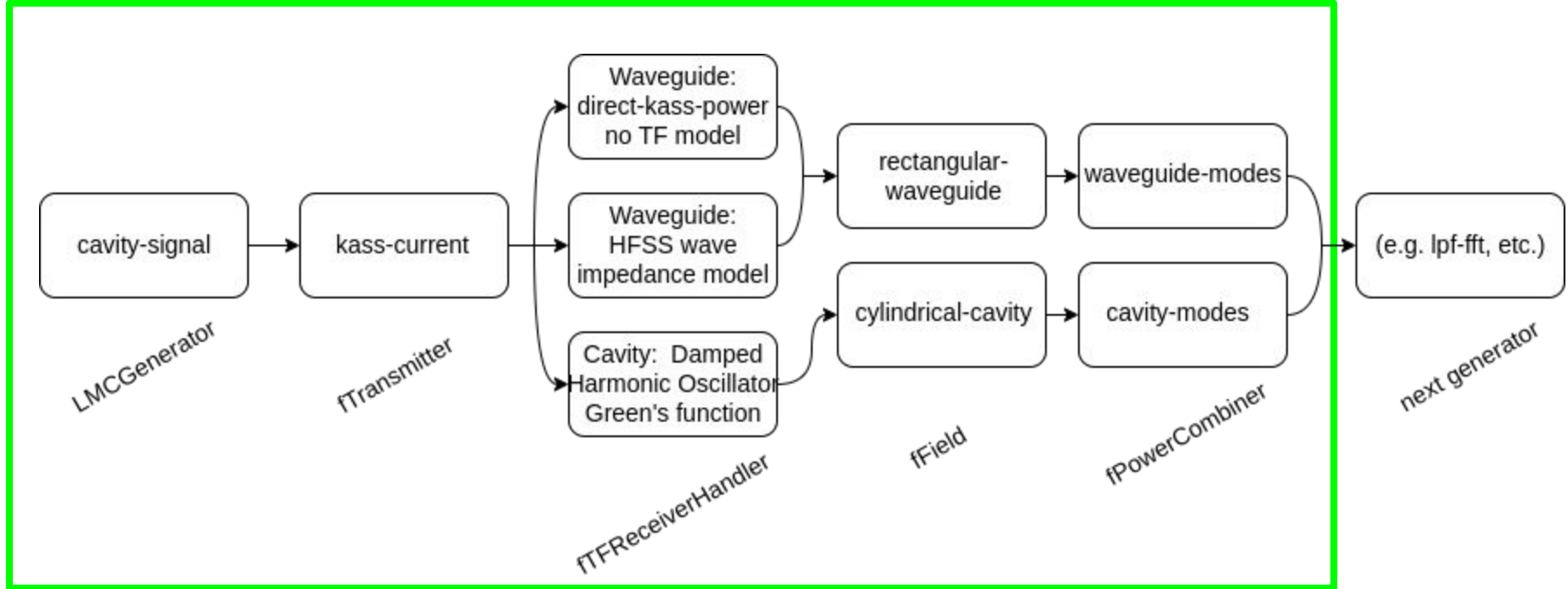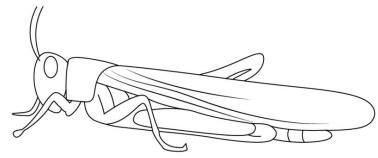
Basic Locust json file: List of generators to be run, followed by parameter definitions
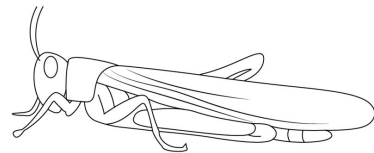
```
{
  "generators":          Generator 1:
  [                      Selected by
  "test-signal",         application.
  "lpf-fft",
  "decimate-signal",     Generators
  "digitizer"            2+: Receiver
  ],                     chain

  "test-signal":
  {
  "rf-frequency": 20.7e9,
  "lo-frequency": 20.65e9,
  "amplitude": 5.0e-8
  },
```
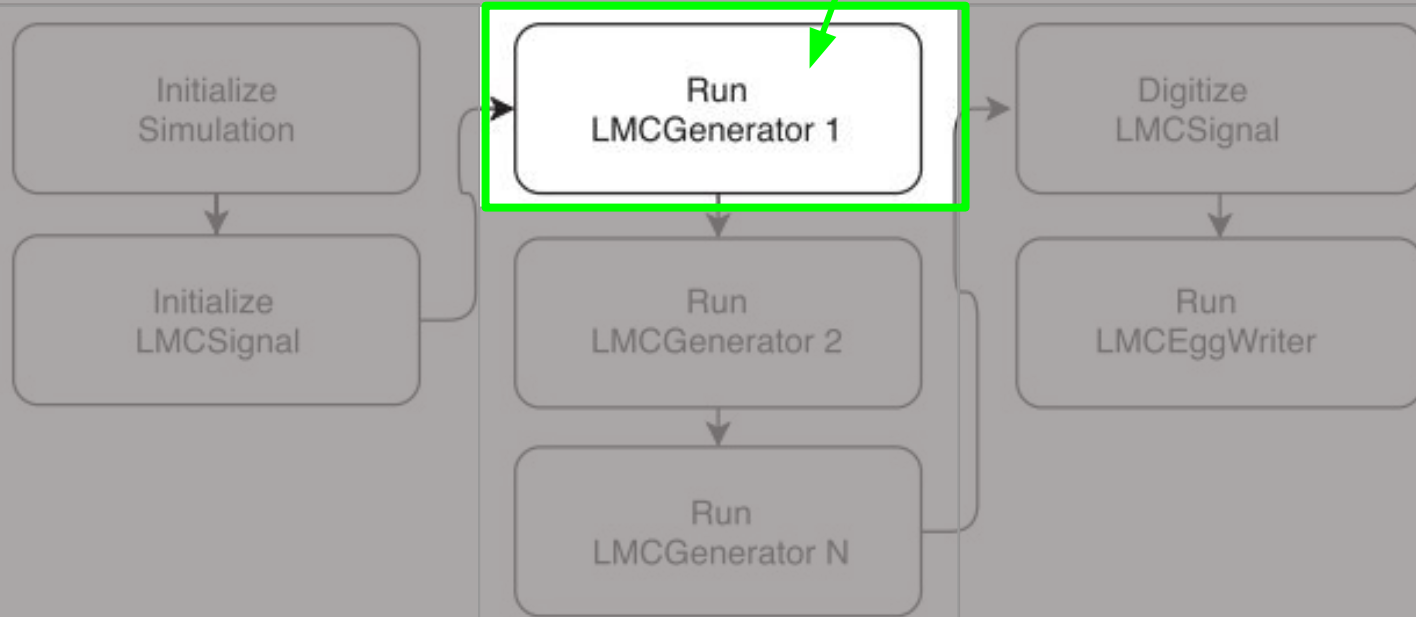
```
  "simulation":
  {
      "egg-filename": "/usr/local/p8/locust/v2.1.1/output/locust_mc.egg",
      "n-records": 1,
      "n-channels": 1,
      "record-size": 8192
  },


  "gaussian-noise":
  {
  "noise-floor-psd": 4.0e-22,
  "domain": "time"
  },
```

```
  "digitizer":
  {
  "v-range": 8.0e-6,
  "v-offset": -4.0e-6
  }

}
```

Outline
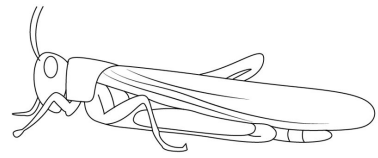
- Brief introduction and instructions for setting up

- <mark>Tutorial examples</mark>

- Intermediate checks

- Development options

- Config files and hpc cluster tools (P8-internal).

Running the tutorial examples
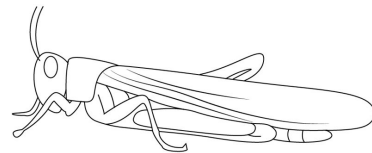
1. In the ~/p8tutorial directory, clone the hexbug and locust-tutorial repos:

    cd ~/p8tutorial
    git clone git@github.com:project8/hexbug
    git clone git@github.com:project8/locust-tutorial

2. cd into to ~/p8tutorial/locust-tutorial/scripts:

    cd ~/p8tutorial/locust-tutorial/scripts

3. Open tutorial_v2.5.1_Locustscript.sh and tutorialKatydidscript.sh with a text editor.
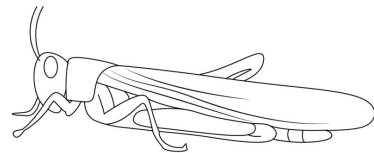
Example #1: Locust test signal plus noise

- Drive 50 ohm antenna with a sinusoid in LMCTestSignalGenerator:

  - (uncomment the command below #Example 1 in tutorialLocustscript_v2.5.1_.sh, then):
    `./tutorialLocustscript _v2.5.1_.sh`
  - (or interactively, in container):
    `LocustSim config=/path/to/config/LocustSignalPlusNoise.json`
  - Process egg file with `./tutorialKatydidscript.sh`

```
"test-signal":
{
"rf-frequency": 20.7e9,
"lo-frequency": 20.65e9,
"amplitude": 5.0e-8
}

"gaussian-noise":
{
"noise-floor-psd": 4.0e-22,
"domain": "time"
}
```

This is in LocustSignalPlusNoise.json

Example #1: Locust test signal plus noise, cont.
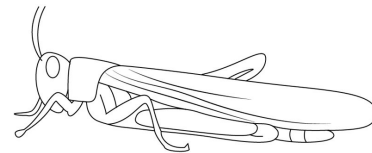
- Open the file output/basic.root and plot histPSD_0_0, as in ->

- Or, in the jupyter browser tab, navigate with single clicks to /tmp/locust-tutorial/scripts/plotting/Plot PSD.ipynb

- Click the ▶▶, then click "Restart and run all cells".

- This plot should appear -> .

Power Spectrum

IF = RF-LO = 50 MHz,
where 100 MHz corresponds to 0 Hz

Check normalization:
|amplitude|^2 = 2.5e-15 W
2.5e-15/(200.e6/8192) = 1.e-19 W/Hz
✔

histPSD_0_0	
Entries	8192
Mean	1.012e+08
Std Dev	5.705e+07
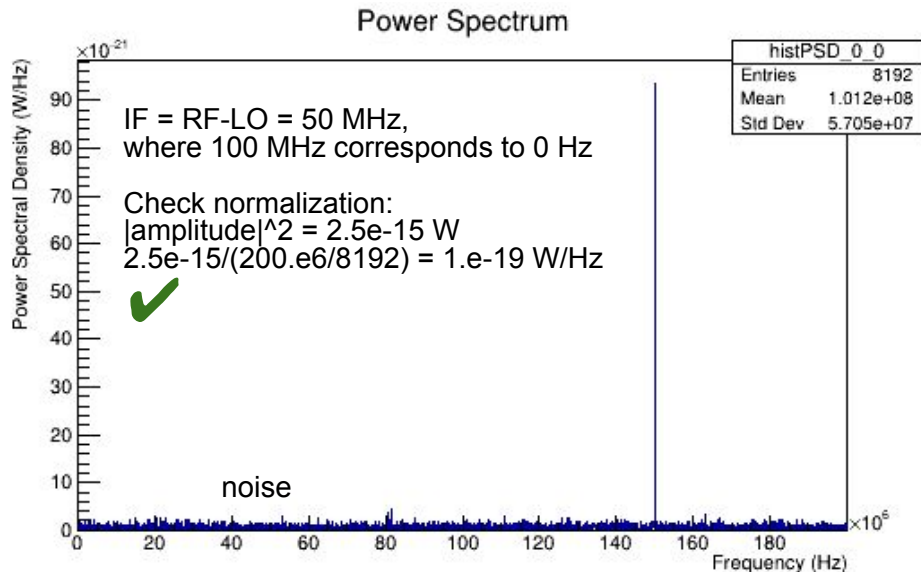
noise

Example #1: Locust test signal plus noise, cont.

- In the jupyter browser tab, navigate with single clicks to /tmp/locust-tutorial/scripts/plotting/PlotLogPSD.ipynb

- Click the ▸▸, then click "Restart and run all cells".

- The mean noise power is as configured, at   log10(4.e-22 W/Hz)=-21.4.
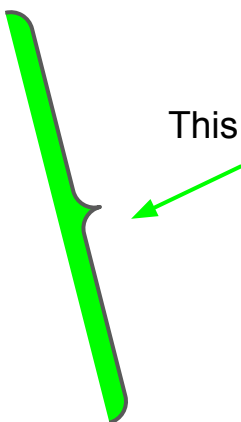
Example #2: 25.9 GHz cavity example

- Calculate an unnormalized signal from a single-mode resonant cavity, driven with a trapped Kassiopeia electron.

  - (uncomment the command below #Example 2 in tutorialLocustscript_v2.5.1_.sh, then):
    `./tutorialLocustscript _v2.5.1_.sh`
  - (or interactively, in container):
    `LocustSim config=/path/to/config/LocustCavityCCA.json`
  - Process egg file with `./tutorialKatydidscript.sh`

"cavity-signal":

```
{
    "transmitter": "kass-current",
    "cavity-radius": 0.007,
    "cavity-length": 0.1,
    "back-reaction": "true",
    "dho-cavity-frequency": 25.9e9,
    "dho-time-resolution": 9.0e-11,
    "dho-threshold-factor": 0.01,
    "event-spacing-samples": 10,
    "rectangular-waveguide": false,
    "voltage-check": "false",
    "lo-frequency": 25.9602e9,
    "xml-filename": "/home/penny/locust_mc/cbuild/config/LocustKass_Cavity_CCA.xml"
},
```
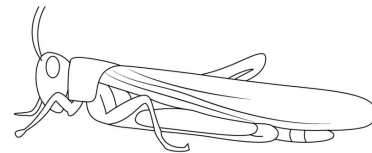
This is in LocustCavityCCA.json
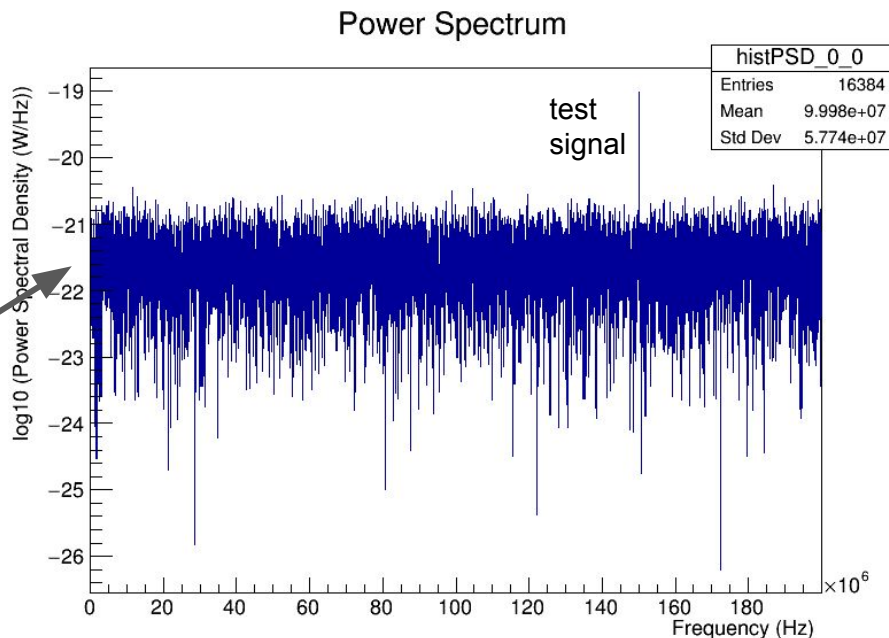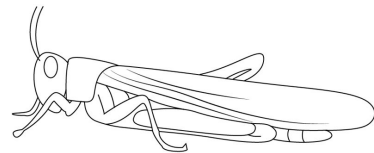
Example #2: 25.9 GHz cavity example, cont.

- Open the file output/basic.root and plot histPSD_0_0, as in ->

- Or, in the jupyter browser tab, navigate with single clicks to /tmp/locust-tutorial/scripts/plotting/Plot PSD.ipynb

- Click the ▶▶, then click "Restart and run all cells".

- This plot should appear -> .

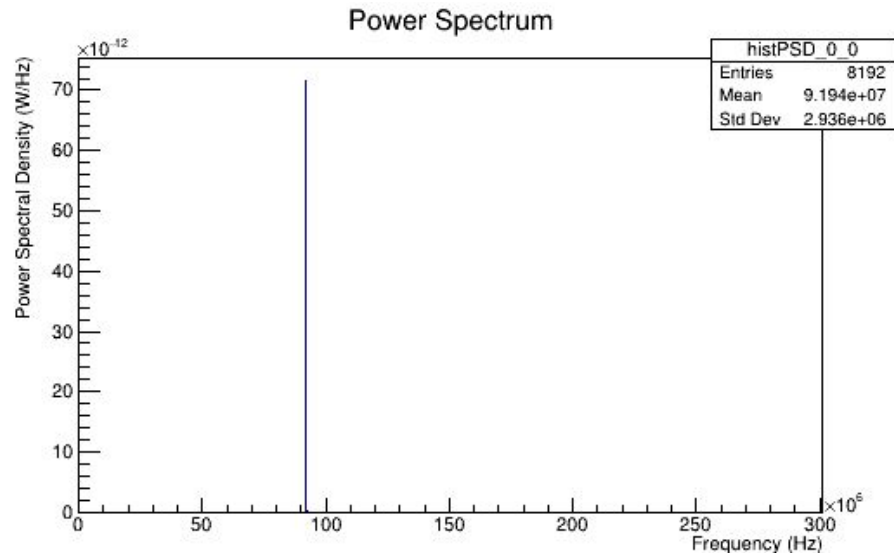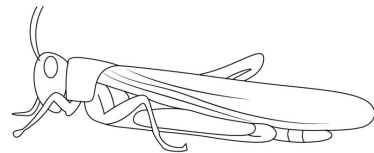# Example #3: WR42 waveguide

- Calculate a normalized signal from a TE10 mode WR42 waveguide, driven with a trapped Kassiopeia electron.

  - (uncomment the command below #Example 3 in tutorialLocustscript_v2.5.1_.sh, then):
    `./tutorialLocustscript_v2.5.1_.sh`
  - (or interactively, in container):
    `LocustSim config=/path/to/config/LocustWaveguideTemplate.json`
  - Process egg file with `./tutorialKatydidscript.sh`

```
  "cavity-signal":
{
  "rectangular-waveguide": true,
  "direct-kass-power": false,
  "tf-receiver-filename": "/path/to/data/WEGA_Impedance_Center.txt",
  "tf-receiver-bin-width": 0.01e9,
  "transmitter": "kass-current",
  "waveguide-x": 0.010668,
  "waveguide-y": 0.004318,
  "waveguide-z": 10.0,
  "center-to-short": 0.05,
  "center-to-antenna": 0.05,
  "waveguide-central-frequency": 1.63e11,
  "back-reaction": "true",
  "event-spacing-samples": 10,
  "voltage-check": "false",
  "lo-frequency": 25.9602e9,
  "xml-filename": "/home/penny/locust_mc/cbuild/config/LocustKass_Waveguide_Template.xml"
},
```

Set "direct-kass-power"=true to skip convolution with HFSS impedance.

Default: Convolve Kass current with complex impedance from HFSS

reflecting short

This is in LocustWaveguideTemplate.json
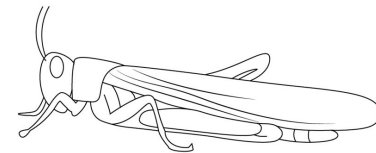
Example #3: WR42 waveguide, cont.

- Open the file output/basic.root and plot histPSD_0_0, as in ->

- Or, in the jupyter browser tab, navigate with single clicks to /tmp/locust-tutorial/scripts/plotting/Plot PSD.ipynb

- Click the ▸▸, then click "Restart and run all cells".

- This plot should appear -> .

Example #3: WR42 waveguide normalization check
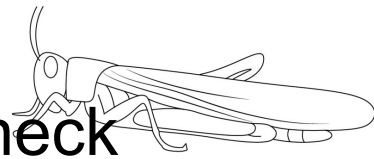
Remove the reflecting short from the simulation in the config file or on the command line as in

```
LocustSim -c
config/LocustWaveguideTemplate.json
"cavity-signal.waveguide-short"=false
["cavity-signal.direct-kass-power"=true]
```

Process with Katydid.

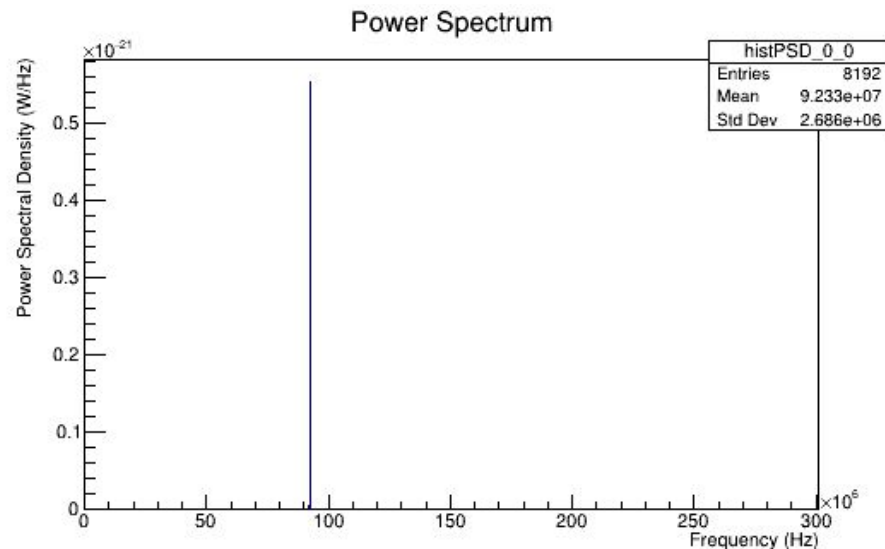Total power radiated by the electron is ~1.e-15 W. Roughly half the power radiated by the electron, scaled (~0.4x) with expected mode power coupling near x=0, should appear in the output -> .

(1.e-15 W) * (0.5) * (0.4) = 2.e-16 W

Power Spectrum

histPSD_0_0	
Entries	8192
Mean	9.23e+07
Std Dev	1.33e+06

(4.e-21 W/Hz)(301.e6 Hz/8192)
= 1.5e-16 W

~ ✔
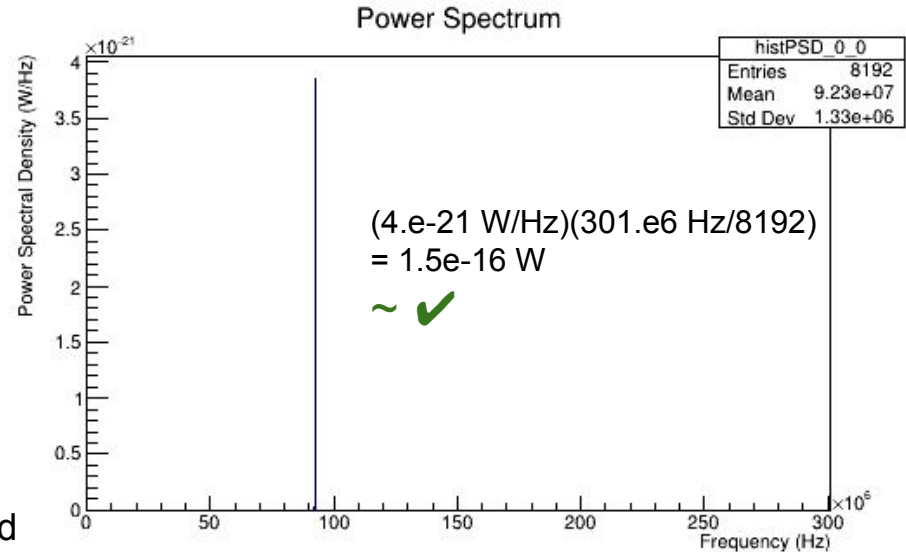
Example #3: WR42 waveguide (cont)
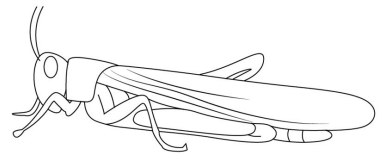
- For a square grid of electron starting positions (typically run on an HPC cluster), we can plot the detected power in the carrier line, as in the plot below.

- Config files are as compiled in the Locust repo. (Cluster job workflow as defined in hexbug.)

- (Plotting macro will be available in hexbug soon.)

Carrier power
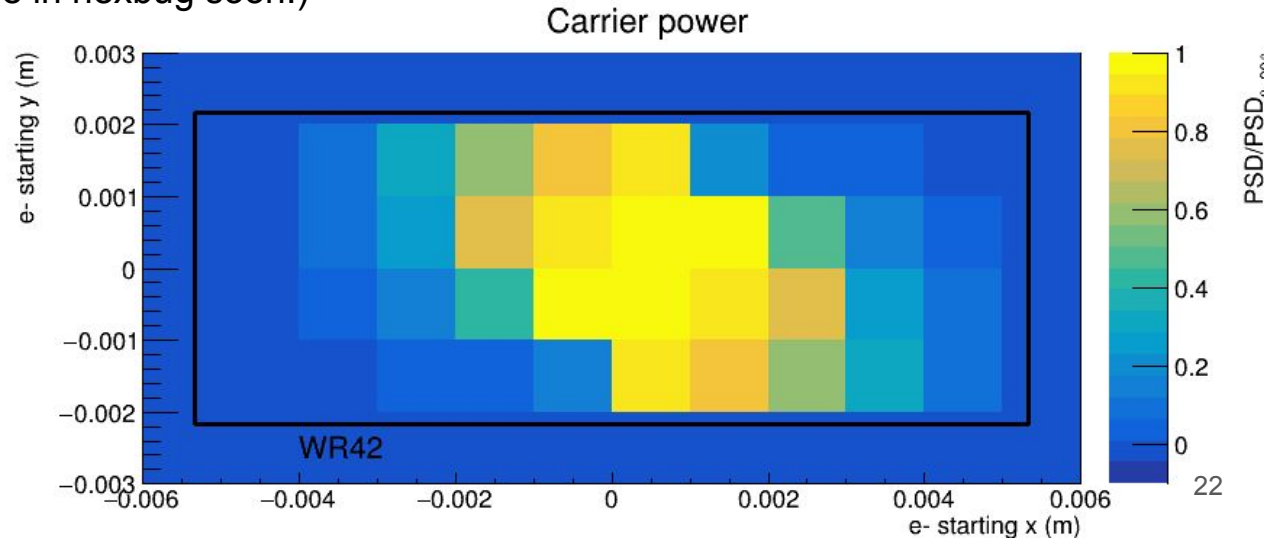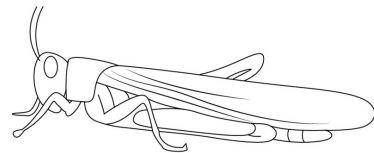
Example #5: How to plot an example magnetic trap axial field
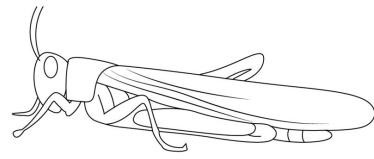
- Use Kassiopeia to calculate B field values along field lines in a placeholder trap:

  - (uncomment the command below #Example 5 in tutorial_v2.5.1_Locustscript.sh, then):
    `./tutorial_v2.5.1_Locustscript.sh`
  - (or interactively, in container):
    `/path/to/LMCKassiopeia /path/to/config/JustKassFieldMap.xml`
  - Plot the Root output using the Jupyter browser:

    - Navigate with single clicks to
      /tmp/locust-tutorial/scripts/plotting/PlotFieldMap.ipynb
    - Click the ▸▸, then click "Restart and run all cells".

magnetic_field_z:position_z
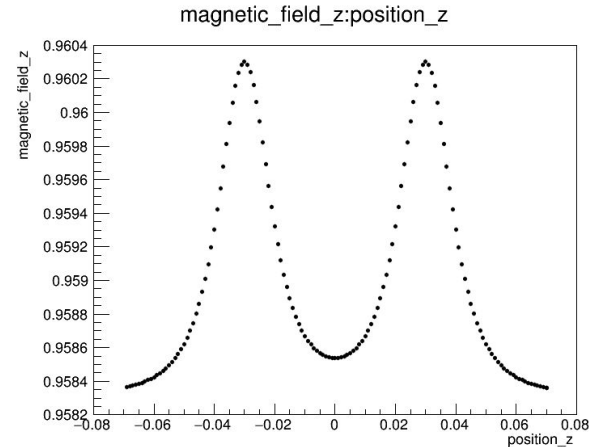
Example #5A: How to plot a different trap, e.g. one selected from the P8 hexbug repository

- We are going to need to edit the Kassiopeia xml file:

  - cd ~/p8tutorial/locust-tutorial/output

  - Start the container interactively, and mount your local ~/p8tutorial/locust-tutorial/output directory to the container like this: docker run -it --rm -v ~/p8tutorial/locust-tutorial/output:/usr/local/p8/locust/current/output ghcr.io/project8/locust_mc:latest /bin/bash

  - Inside the container, copy the container's xml file to your (now) locally mounted directory (all one line): cp /usr/local/p8/locust/current/config/JustKassFieldMap.xml /usr/local/p8/locust/current/output/JustKassMyNewFieldMap.xml and then exit the container.

  - On your local machine, open JustKassMyNewFieldMap.xml with a text editor.

    - Find this line referencing the default trap, and delete it: `<include name="[config_path]/FreeSpaceGeometry.xml"/>`

    - Replace it with a reference to the 1 GHz hexbug trap, e.g.: `<include name="/tmp/hexbug/Phase3/Trap/CavityGeometry_V00_00_00.xml"/>`

Example #5A continued: How to plot a different trap, e.g. one selected from the P8 hexbug repository

○ (Editing JustKassMyNewFieldMap.xml, continued:)

- ■ Find these two lines:
  ```
  <ksterm_min_z name="term_min_z" z="-.07"/>
  <ksterm_max_z name="term_max_z" z=".07"/>
  ```
- ■ Edit them to be:
  ```
  <ksterm_min_z name="term_min_z" z="-2.0"/>
  <ksterm_max_z name="term_max_z" z="2.0"/>
  ```
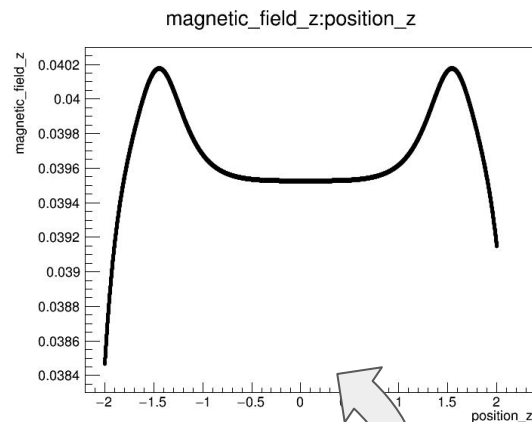- ■ Find this line: `<z_list add_value="-0.07"/>`
- ■ Edit it to be: `<z_list add_value="-2.0"/>`
- ■ Save the modified JustKassMyNewFieldMap.xml file.

● cd ~/p8tutorial/locust-tutorial/scripts

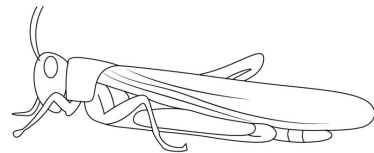● Uncomment Example 5A, and then run `./tutorial_v2.5.1_Locustscript.sh`

● Plot the result in PlotFieldMap.ipynb with the ▶▶, and "Restart and run all cells"

magnetic_field_z:position_z

25

Example #5A, follow-up: How to upload a new trap to hexbug

- cd ~/p8tutorial/hexbug

- Create a new branch of the hexbug repo: git checkout -b feature/myNewBranch

- cd Phase3/Trap

- Create a new file with your trap coil(s) and current(s). Aim to follow the file names and formats used by the other traps in the same directory. Plot your new field map as in Example 5A for a hexbug trap.

- Optionally, generate a vtk output file for visualizing with Paraview software. Uncomment the `<vtk_window> </vtk_window>` section by deleting the `<!--` and `-->` before and after it. Set `enable_display="false"`. Paste `path="[output_path]"` into the `<vtk_geometry_painter/>` section.

  - Run the script `./tutorial_v2.5.1_Locustscript.sh`
  - The output geometry file Project8.vtp can be visualized with Paraview.

- After testing/debugging, commit your new trap to feature/myNewBranch, push to github, do a pull request to develop, and follow up on Slack/etc.

Example #6: Plotting imported HFSS results (waveguide)

- When using LTI filtering in Locust, it is often useful to generate intermediate plots for checking.

- Check: Interrupt the waveguide simulation and plot HFSS impedance and derived FIR to Root histograms:

  - (uncomment the command below #Example 6 in tutorialLocustscript_v2.5.1_.sh, then): `./tutorialLocustscript_v2.5.1_.sh`

  - (or interactively, in container):
    ```
    LocustSim -c config/LocustWaveguideTemplate.json
    "cavity-signal.print-fir-debug"=true
    ```
    (Follow instructions in the terminal.)

Example #6: Plotting imported HFSS results (waveguide, cont.)

- Option 1: Interactively open the file output/TFhisto.root and plot histogram "Real" or "Imag", as in ->

- Option 2: In the jupyter browser tab, navigate with single clicks to /tmp/locust-tutorial/scripts/plotting/PlotTF.ipynb

  - Click the ▸▸, then click "Restart and run all cells".

  - Two plots should appear, showing the HFSS impedance, as imported. ->

Coefficients

Real	
Entries	401
Mean	199.7
Std Dev	89.55

Coefficients

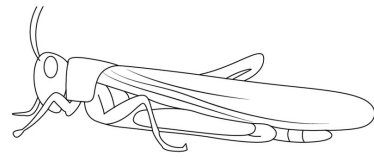| Imag	
Entries	401
Mean	151.3
Std Dev	36.91

A. Telles
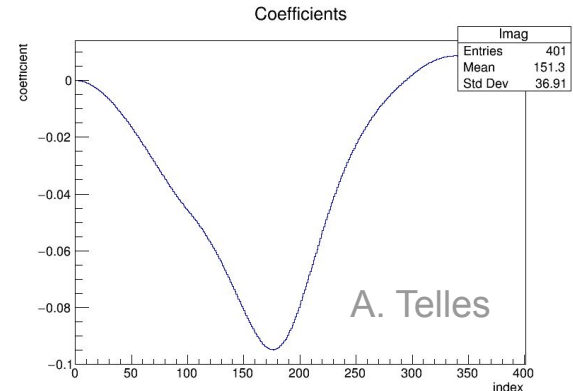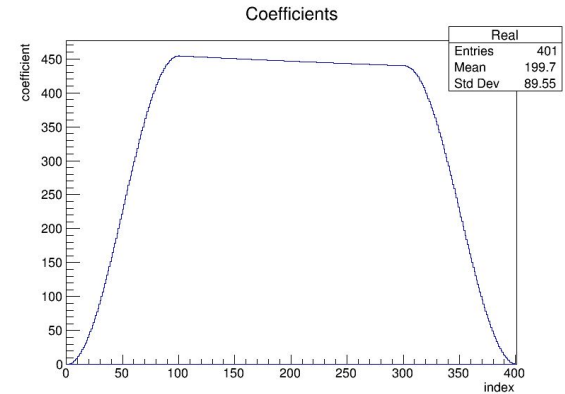
Example #6: Plotting FIR filter derived from imported HFSS results (waveguide, cont.)
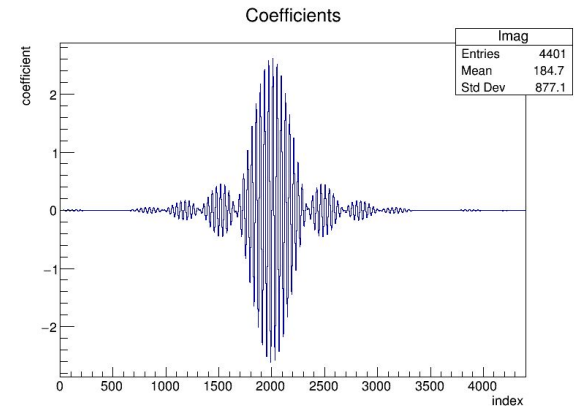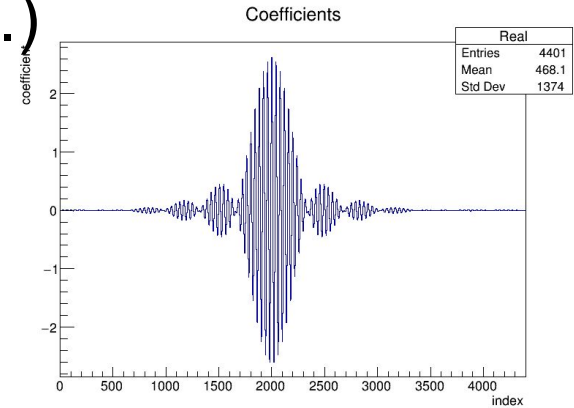
- Option 1: Interactively open the file output/FIRhisto.root and plot histogram "Real" or "Imag", as in ->

- Option 2: In the jupyter browser tab, navigate with single clicks to /tmp/locust-tutorial/scripts/plotting/PlotFIR.ipynb

  - Click the ▸▸, then click "Restart and run all cells".

  - Two plots should appear showing the FIR filter derived from the HFSS impedance. ->

Example #7: Plotting an analytic Green's function (cavity)
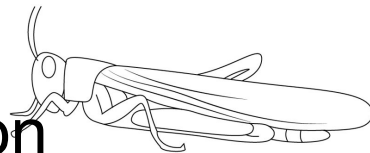
- Similar to the waveguide in #6, the analytic cavity Green's function can be written to FIRhisto.root . The Green's function can also be checked with the unit test `bin/testLMCCavity`.

- Check: Interrupt the cavity simulation and plot the calculated Green's function to Root histograms:

    - (uncomment the command below #Example 7 in tutorialLocustscript_v2.5.1_.sh, then): `./tutorialLocustscript_v2.5.1_.sh`

    - (or interactively, in container):
      `LocustSim -c config/LocustWaveguideTemplate.json`
      `"cavity-signal.print-fir-debug"=true`
      (Follow instructions in the terminal.)

Example #7: Plotting an analytic Green's function (cavity, cont.)
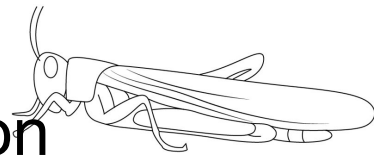
- Option 1: Interactively open the file output/FIRhisto.root and plot histogram "Real" or "Imag", as in ->

- Option 2: In the jupyter browser tab, navigate with single clicks to /tmp/locust-tutorial/scripts/plotting/PlotFIR.ipynb

  - Click the ►►, then click "Restart and run all cells".

  - Two plots should appear showing the real and imaginary parts of the resonant Green's function. ->
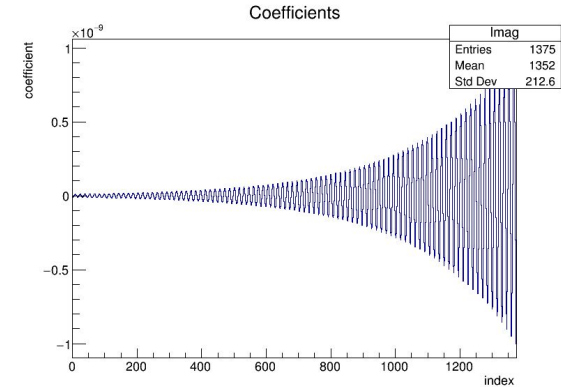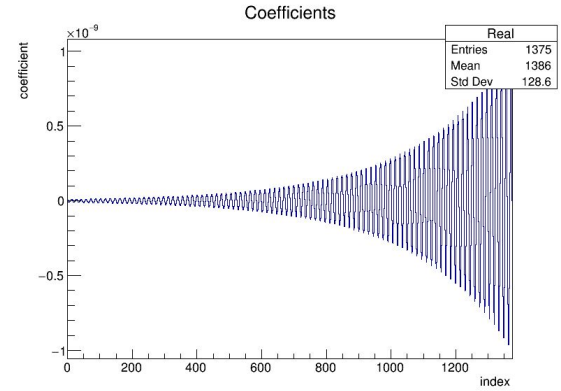
Example #8: Plotting a mode map (cavity/waveguide)

- Onboard mode maps can be checked with a command line flag. With the flag, the simulation is interrupted and the mode maps are written to file.

- Check: Interrupt the waveguide simulation and plot the mode maps to a Root file:

  - (uncomment the command below #Example 8 in tutorialLocustscript_v2.5.1_.sh, then): `./tutorialLocustscript_`v2.5.1_`.sh`

  - (or interactively, in container):
    `LocustSim -c config/LocustWaveguideTemplate.json`
    `"cavity-signal.plot-mode-maps"=true`
    (Follow instructions in the terminal.)

32

Example #8: Plotting a mode map (cavity/waveguide, cont.)
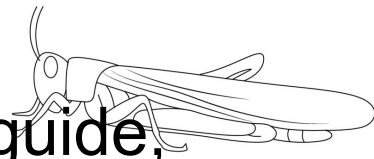
- Option 1: Interactively open the file output/ModeMapOutput.root and plot one of the contained histograms, as in ->

- Option 2: In the jupyter browser tab, navigate with single clicks to /tmp/locust-tutorial/scripts/plotting/PlotModeMap.ipynb

  - Click the ▸▸, then click "Restart and run all cells".

  - This plot should appear showing a selected mode map. ->
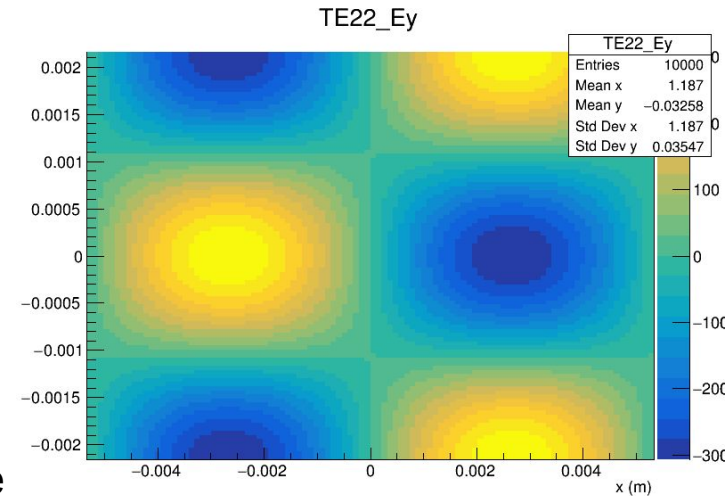
Outline

- Brief introduction and instructions for setting up

- Tutorial examples

- Intermediate checks

- Development options

- Config files and hpc cluster tools (presently partially internal).

Some helpful unit tests

bin/testLMCCavity [-h]: Check the cavity resonance configuration, and optionally plot the output to a histogram in a Root file. This test is also run automatically by the LMCCavitySignalGenerator.

bin/testAliasing [-h]: Check the frequencies of RF harmonics including n=0,1,2 and determine whether they will appear in the measurement window. This test is also run automatically by the LMCCavitySignalGenerator.
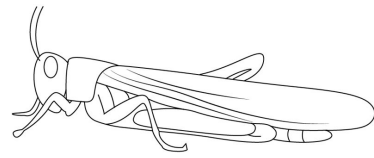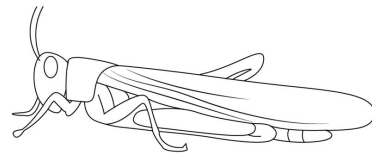
Outline

- Brief introduction and instructions for setting up

- Tutorial examples

- Intermediate checks

- Development options

- Config files and hpc cluster tools (presently partially internal).

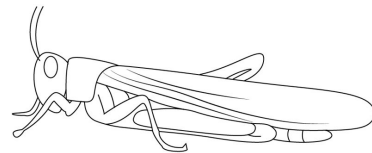# How to obtain Locust and run some examples

1. docker pull ghcr.io/project8/locust_mc:latest
2. Start the container: docker run -it –rm ghcr.io/project8/locust_mc:latest /bin/bash
3. And/or, to mount a local directory to the docker output directory:
   (all one line): docker run -v /path/to/mydirectory:/usr/local/p8/locust/current/output -it –rm ghcr.io/project8/locust_mc:latest /bin/bash

Inside the container:

1. source /usr/local/p8/locust/current/setup.sh
2. source /usr/local/p8/kassiopeia/current/bin/kasperenv.sh
3. cd /usr/local/p8/locust/current/
4. LocustSim -c config/LocustCavity1GHz.json (1 GHz cavity example)
   or
   LocustSim -c config/LocustCavityCCA.json (25.9 GHz cavity example)
   or
   LocustSim -c config/LocustWaveguideTemplate.json (WR42 waveguide example)

Output *.egg files will be in the directory /usr/local/p8/locust/current/output .
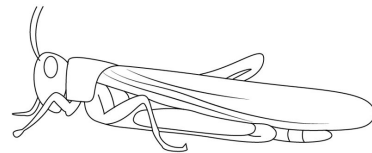
How to develop (Option #1)

Option #1: (Re)build a local Docker container for each modification.

1. git clone git@github.com:project8/locust_mc.git

2. git submodule update –init –recursive

3. Make your code modifications etc.

4. docker build -t mylocustcontainer .

5. Run your new docker container as in previous slides.

6. Evaluate, return to Step #3.
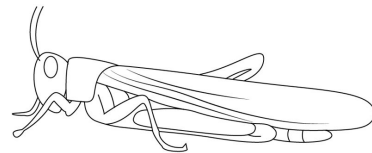
How to develop (Option #2)

Option #2: Build from source in the kassiopeia_builder container.

First, clone locust and submodules as in Option #1.

1. docker pull ghcr.io/project8/kassiopeia_builder:v3.8.2-dev
2. docker run -it -v /path/to/your/locust_mc:/locust_mc --rm ghcr.io/project8/kassiopeia_builder:v3.8.2-dev /bin/bash
3. source /usr/local/p8/kassiopeia/current/setup.sh
4. source /usr/local/p8/kassiopeia/current/bin/kasperenv.sh
5. cd /locust_mc
6. mkdir cbuild
7. cmake -Dlocust_mc_PREBUILT_KASS_PREFIX=${KASS_PREFIX} -Dlocust_mc_BUILD_WITH_KASSIOPEIA=ON ../
8. make install
9. Make code modifications locally, then make install
10. Run LocustSim in the kassiopeia_builder container, evaluate, return to Step #8.

You can close the container and reopen as above, then continue working.

How to compile from source (not typically recommended for new users)

Check system requirements at https://github.com:project8/locust_mc.git

1. git clone git@github.com:project8/locust_mc.git
2. cd locust
3. git submodule update –init –recursive
4. mkdir build
5. cd build

6. ccmake ../
   or
7. ccmake -Dlocust_mc_BUILD_WITH_KASSIOPEIA=ON ../

8. make install

Steps to extend Locust-Kass

1. Develop a new unique LMCGenerator class and type its name in all files where other LMCGenerators are listed. For example, P8 cavity and waveguide signal simulations are run with the LMCCavitySignalGenerator.

2. Develop or select an LMCTransmitter to transmit Kass info to the Locust signal classes, e.g.:
   a. LMCKassCurrentTransmitter: Kass electron kinematics (Presently used in P8 cavity/waveguide simulations).
   b. LMCKassTransmitter: LW free-space field solutions.

3. After the 1st LMCGenerator as in #1, other classes are used for low-pass filtering, digitization, and optionally, broadband white noise. See the example json files.

Outline

- Brief introduction and instructions for setting up

- Tutorial examples

- Intermediate checks

- Development options

- <mark>Config files and hpc cluster examples (P8-internal).</mark>

Hexbug repo: Config files and hpc cluster scripts
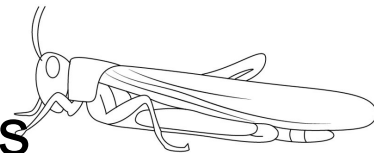
Hexbug is a private repository internal to Project 8.

On an hpc cluster (like Grace at Yale) that runs Slurm for job management, clone hexbug as git clone git@github.com:project8/hexbug.git .

cd hexbug

Create an output directory ("outputdir") for your work, as in e.g.
mkdir /gpfs/gibbs/pi/heeger/[yourInitialsEtcHere]
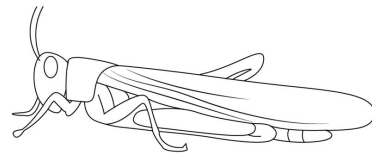
Edit clusterScripts/clusterCavity.cfg to configure outputdir

cd clusterScripts/[choiceOfDemonstrator]

python3 GenerateCavitySims.py or python3 GenerateCavitySims_dSQ.py (the latter is preferred).

Follow terminal instructions. Edit scripts as needed to parametrize the job array.
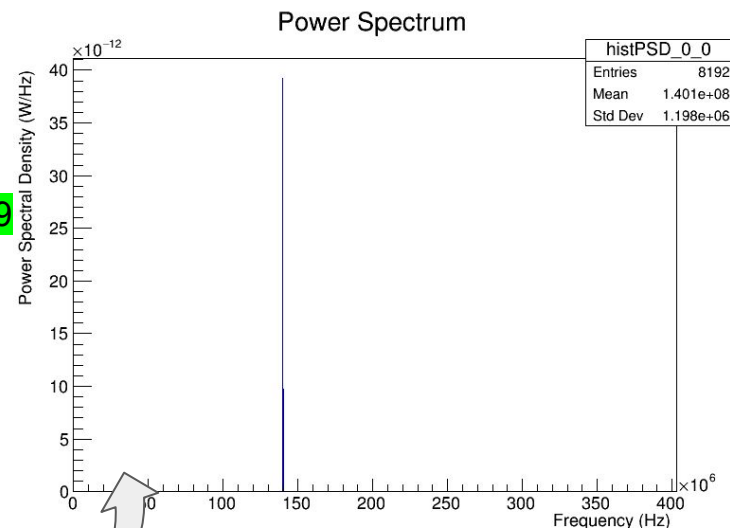
HPC cluster example I: P8-CCA (internal)

1. Clone the hexbug repo as in the previous slide.

2. Create an output directory for your work, as in e.g.
   mkdir /gpfs/gibbs/pi/heeger/[yourInitialsEtcHere]

3. cd hexbug/clusterScripts/

4. Edit the file clusterCavity.cfg -> Find the definition for "outputdir" and replace it with your output directory path from #2 above. Save the file and close it.

5. We will simulate Locust-Kass with the trap defined in the file*:
   hexbug/Phase3/Trap/CavityGeometry_VCCA_trap_LD_8_with_extent_V01.xml .
   cd hexbug/clusterScripts/Cavity_VCCA_trap_LD_8_with_extent_V01

6. python3 GenerateCavitySims_dSQ.py , and follow terminal instructions. The array of 11 jobs should finish in ~15 minutes, with results in the directory [outputdir]/results/ . Output files should include:
   egg files (raw simulation output),
   basic*.root files (frequency spectra),
   Katydid*.root files (frequency-time spectra).
   Kass*.root files (Kass trajectory information).

*Calculations by R. Reimann

44

HPC cluster example I: P8-CCA (internal, continued)

7. Let's look at a frequency histogram for one Katydid time slice, in e.g. the Katydid-processed file [outputdir]/results/Seed600_Angle90.00_Pos0.004/basic_Angle90.00 _Pos0.004.root .

8. On your local machine, cd ~/p8tutorial/locust-tutorial/output/

9. Move the file from the HPC cluster to your laptop (all one line): scp [yourNetID]@grace.hpc.yale.edu:[outputdir]/results/Seed600_Angle9 0.00_Pos0.004/basic_Angle90.00_Pos0.004.root .

10. Either open the file with Root and plot histPSD_0_0, or

11. In your jupyter browser tab, navigate to the file /tmp/locust-tutorial/scripts/plotting/PlotPSD.ipynb . Edit "basic.root" to read "basic_Angle90.00_Pos0.004.root", press the ▶▶, and "Restart and run all cells". You should see this plot:

Power Spectrum

histPSD_0_0	
Entries	8192
Mean	1.401e+08
Std Dev	1.198e+06

# HPC cluster example II: Waterfall plot (internal)
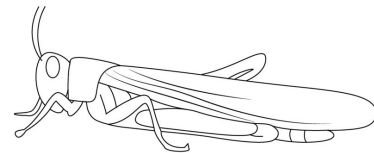
1. Similar to the previous example (I), start in the same directory: `cd hexbug/clusterScripts/Cavity_VCCA_trap_LD_8_with_extent_V01` .

2. Edit the file Kass_config_P8_Cavity_Template.xml. Find the line `<ksterm_max_time name="term_max_time" time="1.e-4"/>` , and change 1.e-4 to 1.e-3 . Save the file and close it. This will lengthen the duration of the e- simulation from 0.1 ms to 1 ms.

3. Start the same job array as in (I) with the GenerateCavitySims_dSQ.py script. The default simulations (with θ>89°) will take ~2 hours to finish.

4. From your local machine, move a Katydid-processed waterfall file from the HPC cluster to your laptop as in:

   cd ~/p8tutorial/locust-tutorial/output/

   scp [yourNetID]@grace.hpc.yale.edu:[outputdir]/results/Seed600_Angle90.00_Pos0.004/KatydidOutput_Angle90.00_Pos0.004.root .

# HPC cluster example II: Waterfall plot (internal, continued)
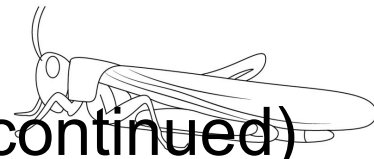
5. Either open the file with Root and plot PSDSpectrogram_0_0, or

6. In your jupyter browser tab, navigate to the file /tmp/locust-tutorial/scripts/plotting/PlotWaterfall.ipynb. Check that the default filename "KatydidOutput_Angle90.00_Pos0.004.root" matches your local file. Press the ▶▶, and "Restart and run all cells". You should see this plot:

7. Notes on the axes:

   ● The frequency axis maps from -fs/2 to fs/2, where fs is the sampling frequency used in the simulation (defined in your config file). This is true even though the labels span from 0->fs.

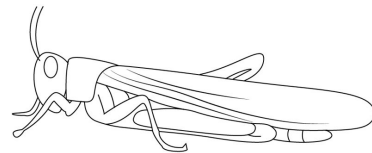   ● The time axis is as simulated.

Spectrogram
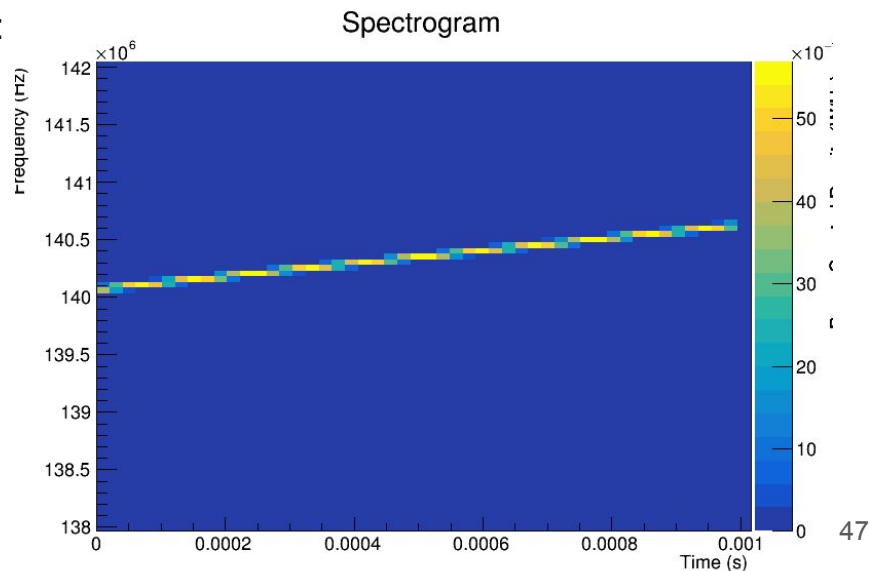
Some next steps in Locust-Kass development
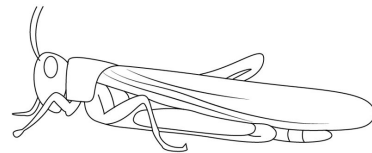
- We are working to provide access to relevant Kass parameters in the Locust config file. This will minimize the need to manage 2 different config files.

- Pileup of simultaneous electrons in one time series.

- Multiple cavity modes with independent frequencies and Q values.

- New resonant noise generator.

-

Troubleshooting

- No egg file was written.

  - The digitizer range was exceeded, and an exception was thrown. (Exit code = 1)
  - A trapped electron hit the walls of the cavity or waveguide very quickly, and a signal was not generated properly. See https://github.com/project8/locust_mc/issues/290 .

- Histograms and time series are empty.

  - Digitizer range is too large.
  - The electron was not well trapped, ending the run early.
  - Record length is too short (e.g. record ended before electron started).
  - "event-spacing-samples" has delayed the event start time(s) past the end of the record.
  - LO is tuned such that the signal is out of the window.
  - Katydid n-slices is too small and so the Locust signal was not processed.
  - Katydid was not run at all.
  - B field is higher/lower than expected, moving signal out of window.

Troubleshooting (continued)

- Unexpected high-power artifacts

  - Digitizer range is too small (or possibly too large).

- Other

  - Switch on the verbose <mark>"voltage-check": "true"</mark> flag to check for reasonable voltage values while simulation is running.
  - Look at the Katydid time series of voltages: check for clipping and quantization.