Gnuradio and Radio Astronomy
Part I: Basics

Marcus Leech
Canadian Centre for Experimental Radio Astronomy
http://www.ccera.ca

SDR and Radio Astronomy

- LONG road to acceptance in both professional and amateur communities
- First paper on SDR and Radio Astronomy:
  - “New Directions in Amateur SETI Receiver Design” EuroSETI, 2004
  - For over a decade, I was the only person using SDR techniques in radio astronomy.
  - RTL-SDR started to change that—now several sources for Radio Astronomy software for cheap SDRs:
    - VIRGO, WVURAIL, CCERA, others
    - Early GnuRadio had in-tree RA applications

Gnu Radio overview

- First released in 2001 as offshoot of MIT project
- Grown hugely in use and capabilities since then
- Dozens of developers contributing to it
- Hundreds of different applications written with it
- Thousands of users of GR-based applications
  - GQRX
  - Gr-inspector
  - Dozens of others
    - CCERA has quite a number: https://github.com/ccera-astro/

Gnu Radio what is it?

- Primarily it's a specialized *Development Framework* for DSP applications
- Often mis-perceived as "A radio with a lot of knobs".
- Gnu Radio programs are *Directed Graphs* of individual DSP processing blocks.
  - Majority of "blocks" are written in C++
  - Many of those blocks are ruthlessly optimized and use Vector capabilities of underlying CPU.
  - Python provides a **control plane** layer
- Dozens of different *blocks* included in base code.
  - MOST involve "the works of man"--telecom, wireless, radar, etc.

Why use *Gnu Radio* for RA?

- At its core, GR is a *multi-threaded scheduler* and *ring-buffer manager*.
  - You'd have to write those yourself anyway
- Plenty of standard blocks that are useful for RA
- OOT "plug ins" and embedded Python blocks make it easy to write domain-specific blocks.
- *Gnu Radio Companion* (GRC) provides a nice graphical flow layout tool.
  - Analogous to *schematic capture* tools in the hardware world.
  - Makes quick experiments painless.

Gnu Radio Companion (GRC)

- A GUI tool for laying-out Gnu Radio *flow-graphs*
- Vaguely-similar to both SimuLink (Mathworks) and LabView (National Instruments).
- Use sa *Data Flow* model of computation.
  - Procedural "stuff' generally hidden inside GR blocks.
  - Has a powerful *dependency evaluator* that allows relationships between block parameters and variables to be maintained with dynamic consistency**.**
  - You don't **need** GRC for making GR *flow-graphs*, but it's awfully convenient.
  - *GRC* entered the scene several years AFTER *Gnu Radio* was already under significant development and distribution.

Gnu Radio Companion--cont'd

- For RA, "abusing" the dependency evaluator in GRC is really useful.
- Emergence of *Embedded Python Blocks* within a GRC-based flow-graph makes some of that abuse no longer necessary.
- Even-newer versions of GR (3.9 and later) allow insertion of so-called "code snippets" into the generated code—relaxing even further the "pure" *Data Flow* model previously enforced by GRC.
- We are using GR 3.8.2 for this seminar, since that is what is packaged for Ubuntu 21.04.
- A GUI representation of the flow is really useful, particularly for those coming from a hardware/analog background.

What is an *amateur* radio telescope?

- A typical amateur radio telescope can be thought of as a single-pixel **radiometer** and sometimes also a **spectrometer**.
- More advanced might venture into **interferometry** or even **pulsar** detection.
- These are **ALL** relatively easy to implement using GR.
  - **BUT** you still need some non-trivial programming and DSP knowledge.
  - Remember: GR is a **programming framework**

Software Defined Radios

- Typical *Software Defined Radio* (SDR)
  - RF gain
  - Down-conversion to I/Q (complex) base-band
  - High-speed I/Q sampling
  - Bandwidth defined by sample-rate (more or less)

- GR uses I/Q (complex base-band) signals almost everywhere.
- See: https://www.pe0sat.vgnet.nl/sdr/iq-data-explained/

A *Radiometer* in software

- A hardware/analog radiometer “back end”:
  - Some RF/IF gain
  - A diode detector
  - An Op-Amp integrator
  - Some DC gain
  - Probably an A/D or maybe (retro) a chart recorder
- Radiometer in software
  - Use complex-to-magnitude^2
  - Average (integrate/low-pass filter)
  - Decimate and log

A *Spectrometer* in software

- Hardware spectrometer
  - Filter-bank with LOTs of parallel detectors.
  - Swept IF or RF with a single detector
    - Spectrum analyzers have used this approach for decades
- SDR/DSP *spectrometer*
  - Collect samples into discrete *windows*
  - Apply window-shaping function
  - Compute FFT on that sample window
  - Compute PSD with complex-to-mag^2
  - Average
  - Scale/decimate/log

Why do we square?

- Complex-to-mag^2 common sub-theme in radiometer and spectrometer—why?
  - I/Q samples are **voltage** linearly proportional to antenna voltage
  - FFT output in **same units** as input
  - *Ohm's Law* tells us power is proportional to V^2
- Ultimately, we want **power** estimates
- Remember that a diode is used as a **square law** detector—it converts input voltage to a power estimate (over a fairly narrow range of linearity-in-power)

Why do we average or integrate?

- Statistics tells us that *variance* in a data-set is reduced by averaging over a larger number of samples
- Our “signals” are often less than the variance, so integration is mandatory to reduce variance.
- In RA, we achieve that in **two dimensions**
  - S ~= *Tsys/sqrt(Bw * Ti)*
    - *Ti* = integration (averaging) time
    - *Bw* = signal bandwidth
- This would be implemented with an R-C+OpAmp integrator in hardware.

Averaging in DSP

- Gnu Radio has a large number of blocks that can be used for averaging
  - *Single Pole IIR filter*
  - *FIR Low Pass filter*
  - *Average*
  - *Integrate*
- They all perform nearly-identically, with subtleties around both performance and the achievable *transfer function*.
- I personally prefer *Single Pole IIR filter*
  - *High performance and able to do very long integrations*

Terminal

File Edit View Search Terminal Help

```
mleech@mleech:~$ git clone https://github.com/ccera-astro/ra_funcs
Cloning into 'ra_funcs'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 24 (delta 4), reused 20 (delta 3), pack-reused 0
Receiving objects: 100% (24/24), 4.80 KiB | 4.80 MiB/s, done.
Resolving deltas: 100% (4/4), done.
mleech@mleech:~$ cd ra_funcs
mleech@mleech:~/ra_funcs$ sudo make install
[sudo] password for mleech:
cp ra_funcs.py /usr/local/lib*/python3.7*/*-packages
cp: cannot create regular file '/usr/local/lib*/python3.7*/*-packages': No such
file or directory
make: [Makefile:2: install] Error 1 (ignored)
cp ra_funcs.py /usr/local/lib*/python3.9*/*-packages
cp ra_funcs.py /usr/local/lib*/python3.8*/*-packages
mleech@mleech:~/ra_funcs$
```

Terminal

File Edit View Search Terminal Help

```
mleech@mleech:~/ra_funcs$ git clone https://github.com/ccera-astro/baa_seminar
Cloning into 'baa_seminar'...
remote: Enumerating objects: 80, done.
remote: Counting objects: 100% (80/80), done.
remote: Compressing objects: 100% (54/54), done.
remote: Total 80 (delta 49), reused 53 (delta 25), pack-reused 0
Receiving objects: 100% (80/80), 40.96 KiB | 1.86 MiB/s, done.
Resolving deltas: 100% (49/49), done.
mleech@mleech:~/ra_funcs$
```

Testing setup we'll be using today.

- 30x60cm para-grid
- SawBird H1 LNA
- AirSpyMini
- Zenith Pointing
- 35deg FOV
- 30m USB Extender to house
- 3m coax to isolate USB/SDR