Beacon Signal Acquisition and Processing Using Software Radio

Harendra Guturu

University of California, Berkeley

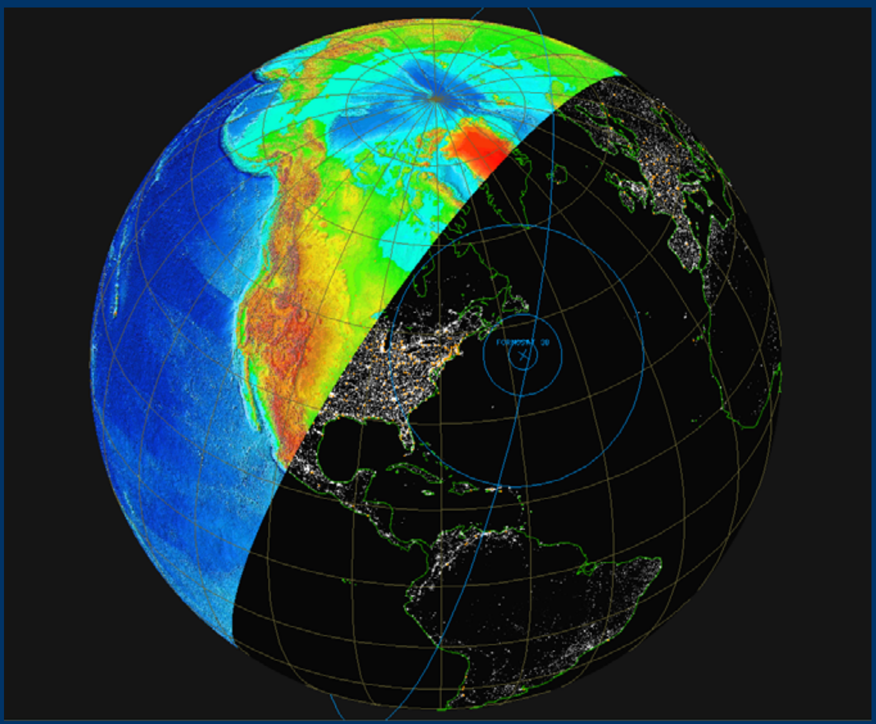
MIT Haystack, REU Summer 2007

Mentors: Phil Erickson & Bill Rideout

What do we want from data?

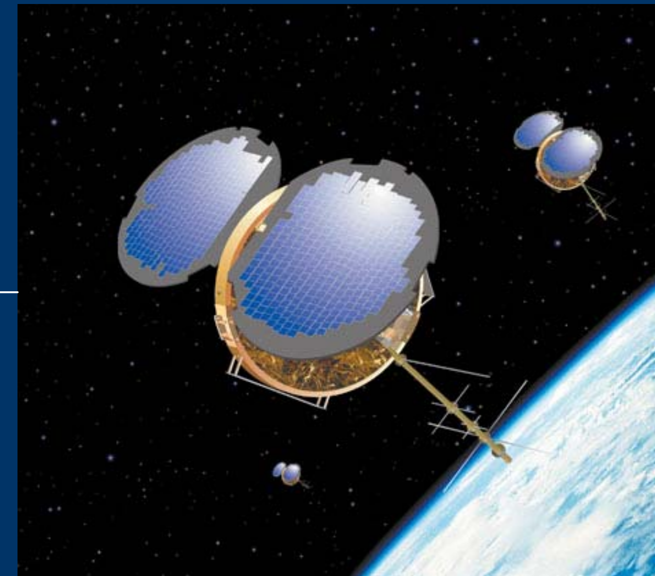
- S4
  - Normalized standard deviation of signal power
- Sigma Phi
  - Standard deviation of phase
- Total Electron Content (TEC)
  - The number of electrons between receiver and beacon
- Parameters are useful to understand the ionosphere during the pass

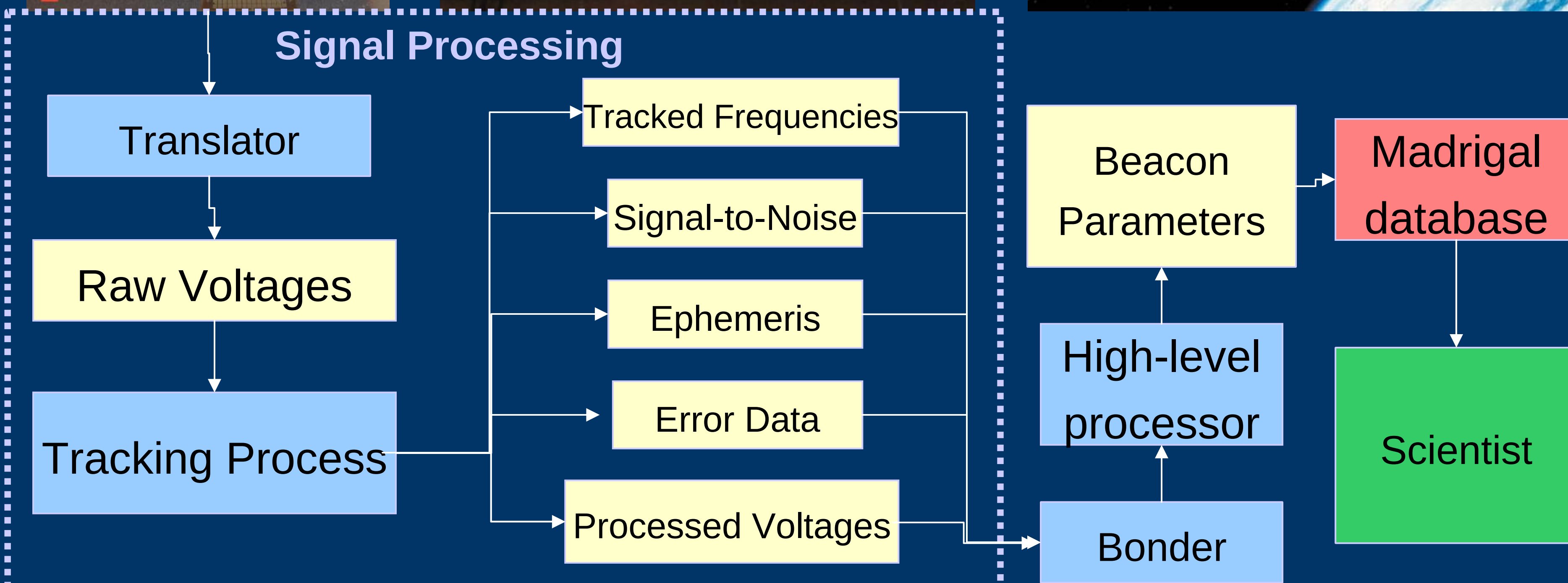
What is getting in the way?



- Problem:
  - Satellite pass causes Doppler shift (change) in its tone
  - Frequency needs to be removed to analyze the data
- Solution:
  - Track (find) the frequency as it changes
  - Using the tracked frequency to shift the signal to baseband


Intercepted Signals For Ionospheric Science




Signal Processing
Translator
Raw Voltages
Tracking Process
Tracked Frequencies
Signal-to-Noise
Ephemeris
Error Data
Processed Voltages
Bonder
High-level processor
Beacon Parameters
Madrigal database
Scientist

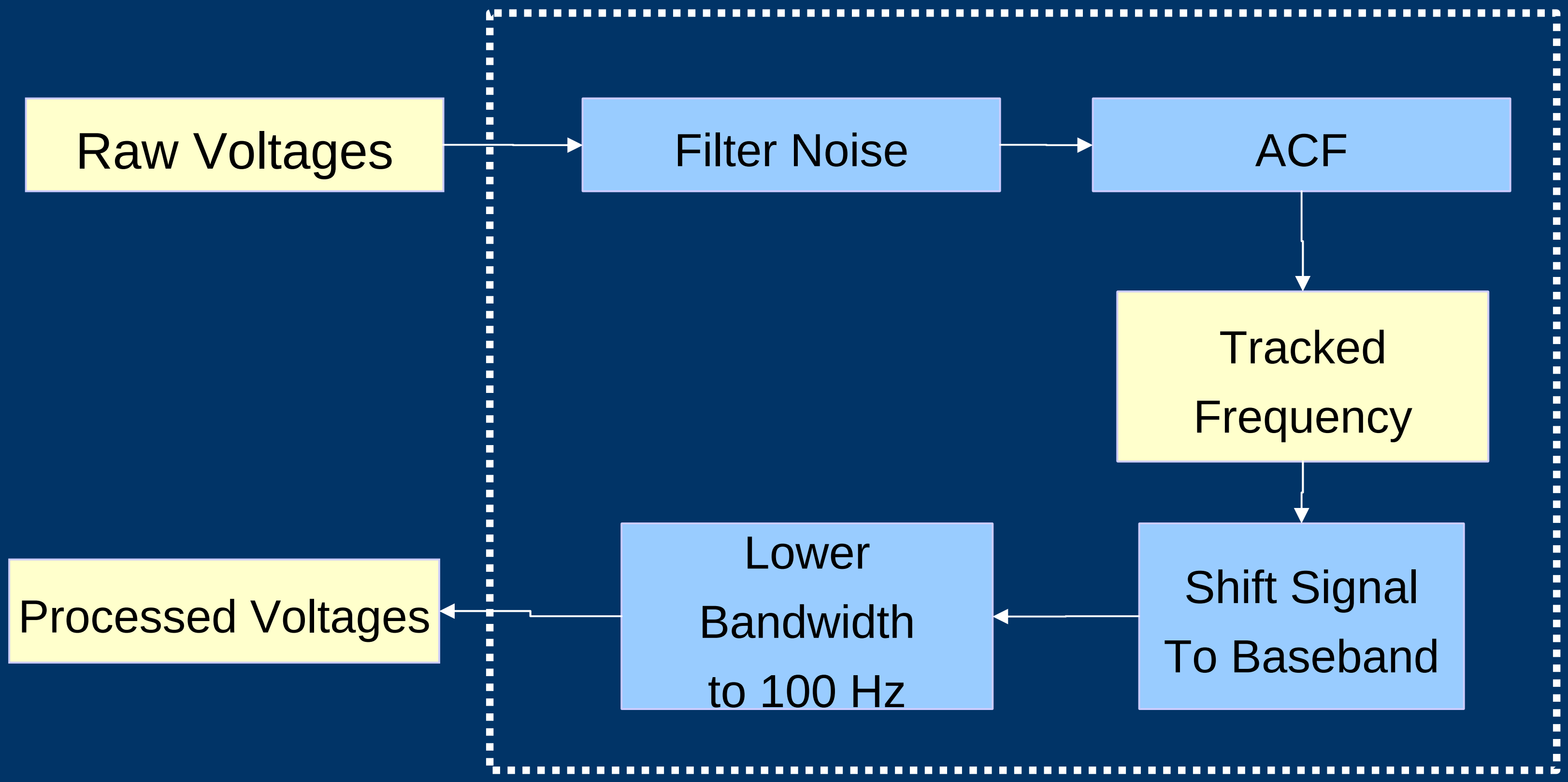
Step 1: Translator

- Purpose:
  - Create HDF5 (a scientific data format) file from the incoming MidasW file format
  - Formatting the voltages from the digital receiver
  - Add additional configuration information regarding the receiver and pass

Step 2: Frequency Tracking Process

- The core of the software beacon receiver
- Replaces dedicated signal processing hardware
- Uses autocorrelation function (ACF) to track the frequency
- Prepares the data for scientific analysis

Step 2: Frequency Tracking Process



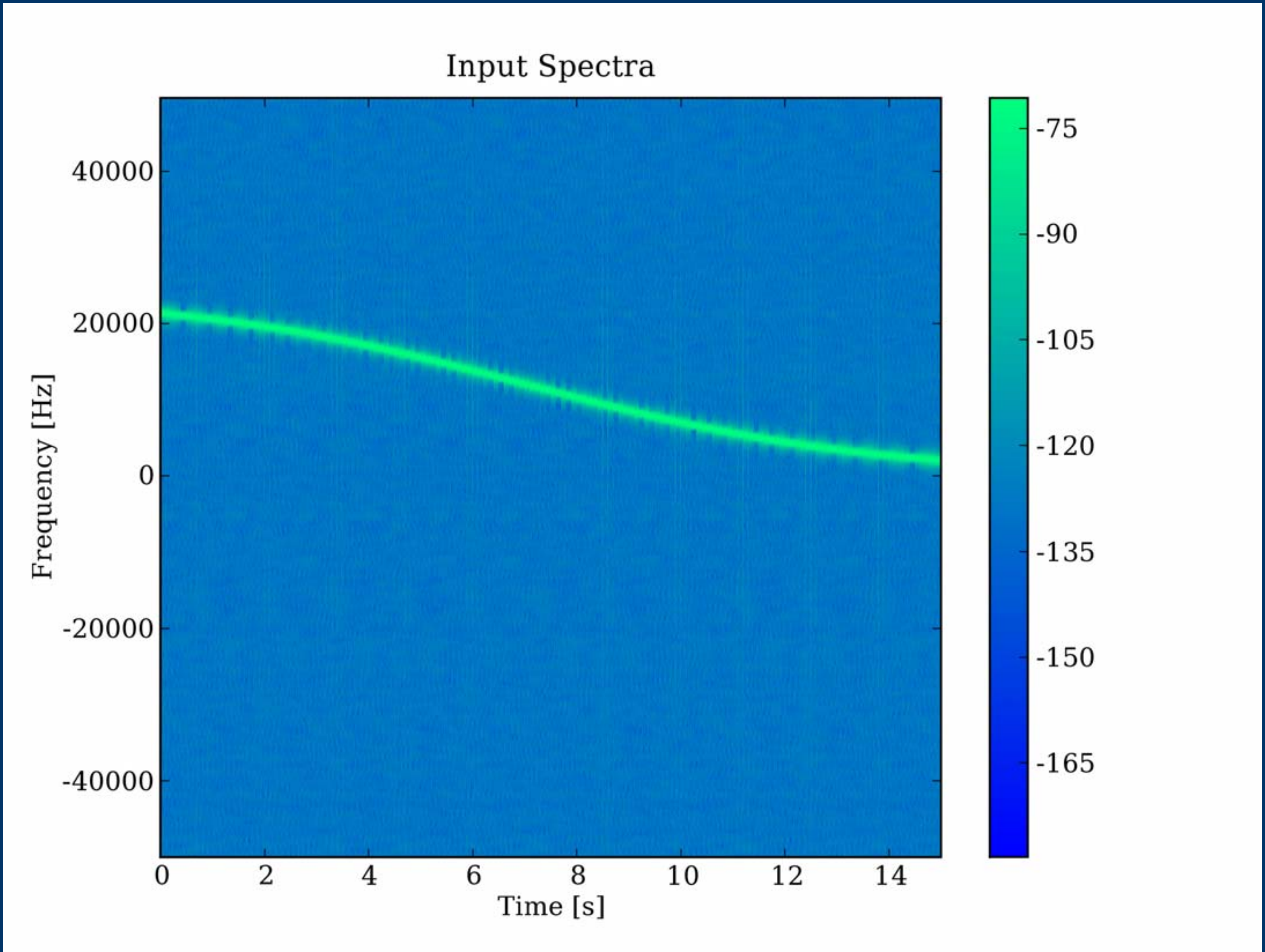
Step 3: Bonder

- Purpose:
  - Combines all data from one pass into one file
  - Ex: Some satellite emit multiple frequencies and more than one frequency is required for calculating TEC
  - Optional since one pass may not necessarily have multiple files
  - Maintains HDF5 format

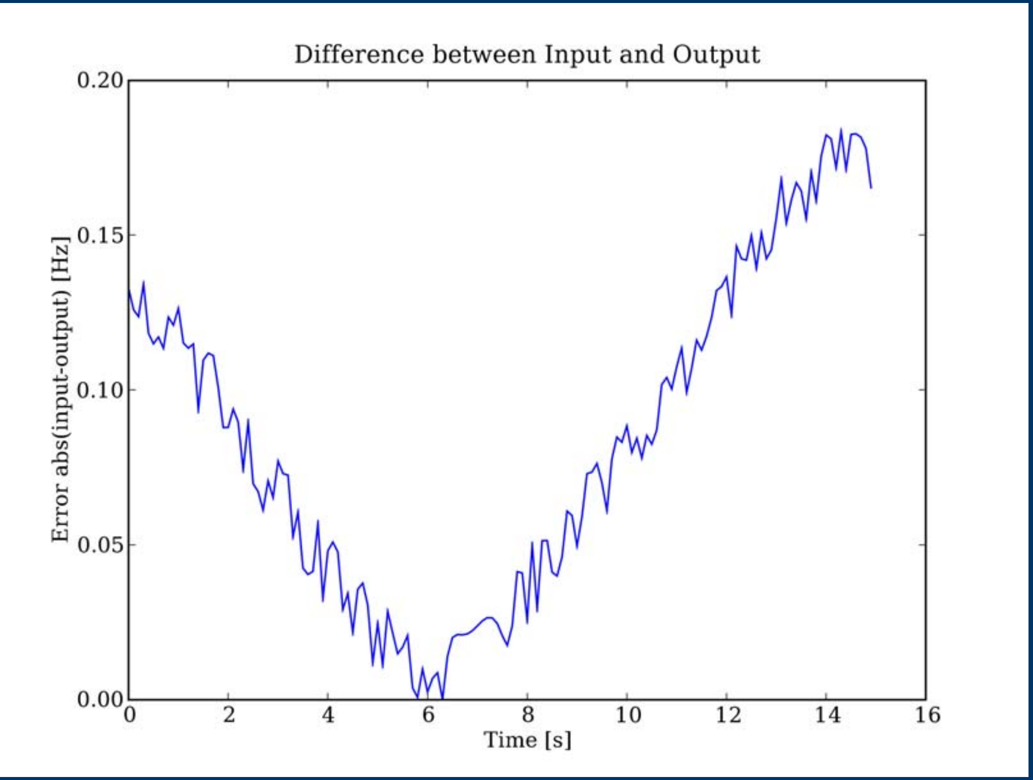
Frequency Tracking Challenges

- Dealings with high volume of data
  - Ex: 100 kHz sampling over ~10 min pass equals ~1GB of data using 128 bit complex values.
- Radio Frequency Interference (RFI)
- Designing good filters

Tracking Results (Synthetic Input Data)

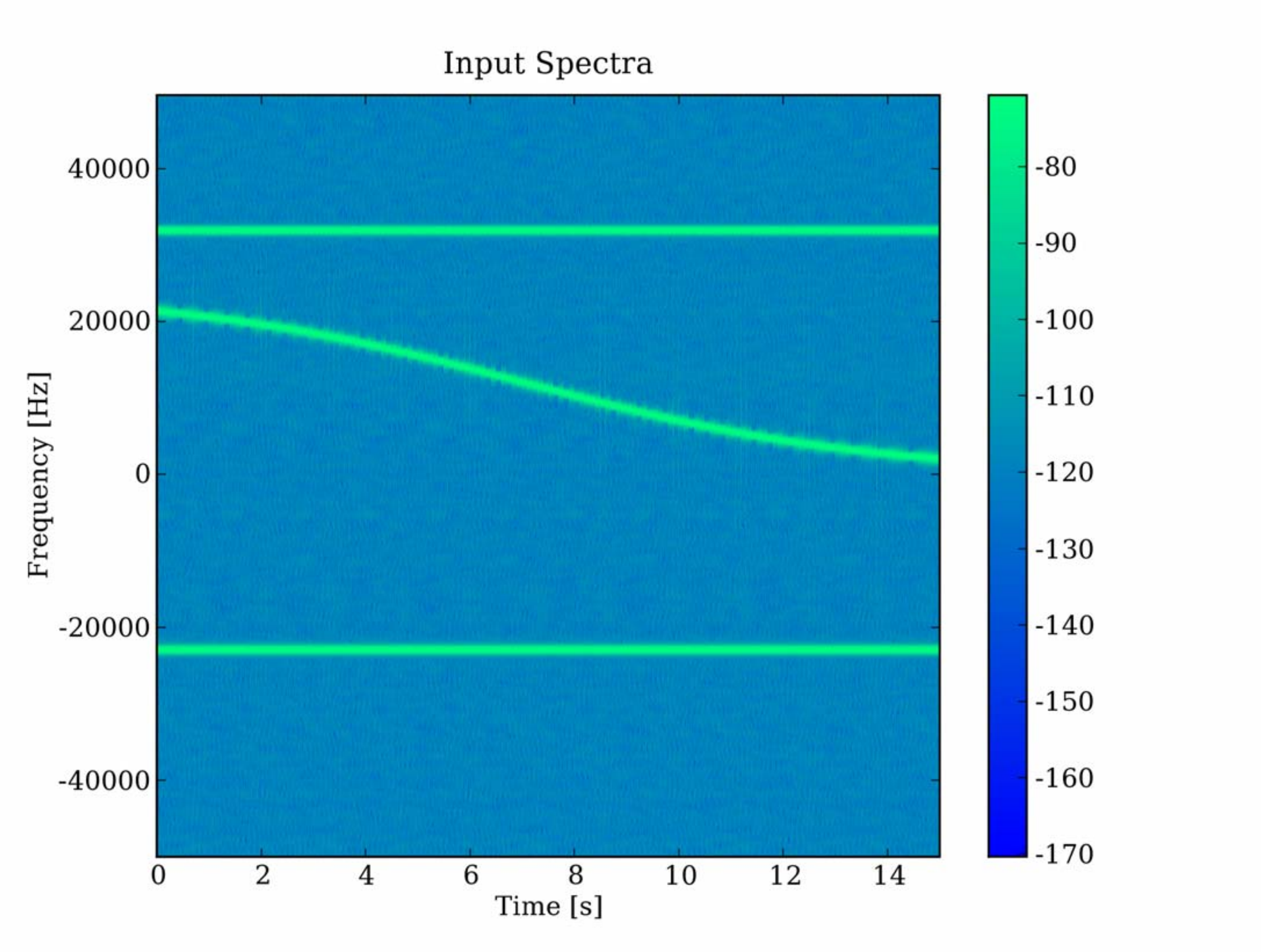


Tracking Results [abs(expected-tracked)]

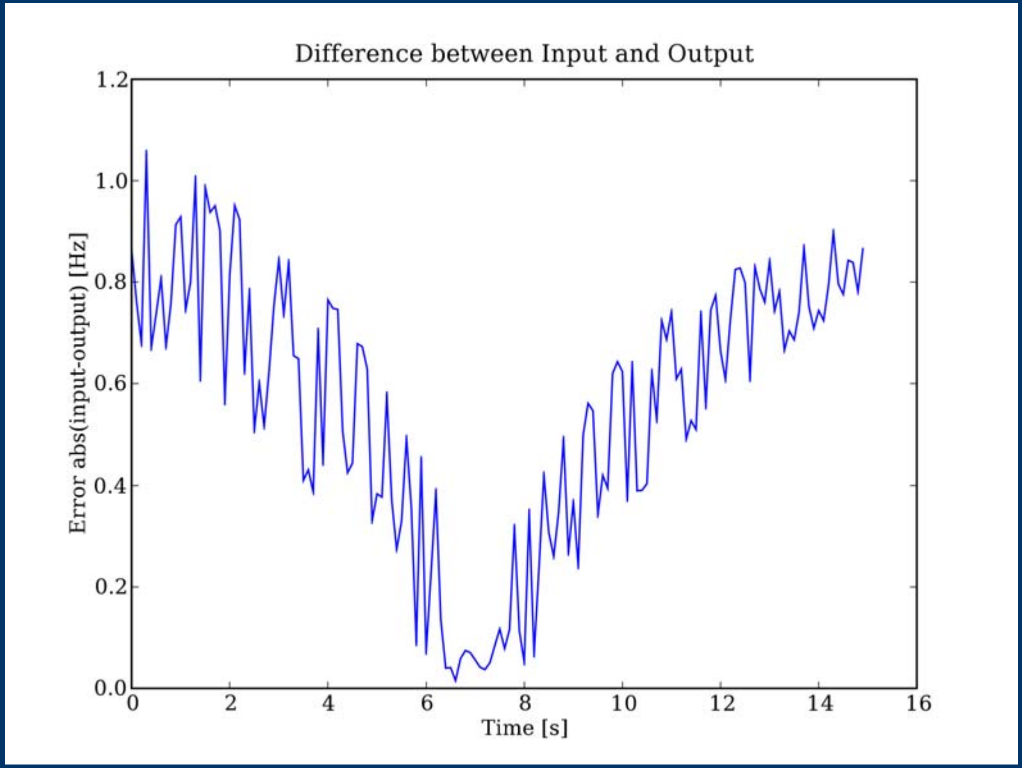


- Max error is ~0.18Hz
- Very good compared to input bandwidth of 24kHz
- Good level for subsequent high-level processing

Tracking Results (Synthetic Input Data)



Tracking Results [abs(expected-tracked)]



- Max error is ~1.1Hz
- Good compared to input bandwidth of 24kHz
- Still good for subsequent high-level processing

Things Discovered

- Tracking satellites is pretty hard
- Tracking bandwidth filtering is necessary
- RFI is a bigger problem than expected
- Biggest bottleneck is loading and saving data files rather than processing

Status of Project

- Tracks the frequency
- Lowers Bandwidth from 100kHz to 100Hz
- Calculates Signal-to-Noise ratio (SNR), Ephemeris, & Error values
- Documented and stored in revision control for future modifications
- Integrates with the hardware end and high level processing end

Future Work

- Tracking
  - Stress testing
  - Advance filtering techniques
- Overall
  - Master control script for automation
  - Profile and optimize software

Acknowledgements

- Project Mentors
  - Phil Erickson
  - Bill Rideout
  - Frank Lind
  - Jim Marchese
  - Anthea Coster
- Others:
  - Ching Lue
  - Paul Bernhardt
- My teammates:
  - Damian Ancukiewicz
  - William Harmon