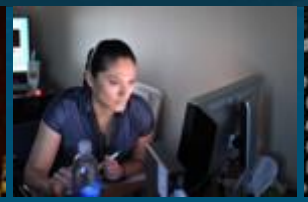
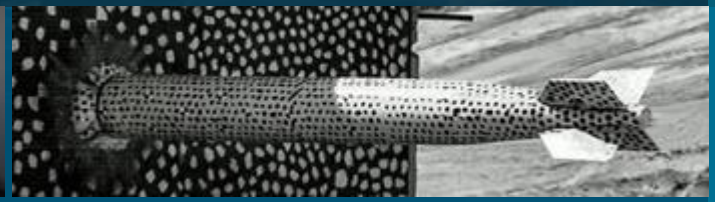
SAND2022-6001C

Sandia National Laboratories

Geophysical Monitoring System (GMS)
Moment Tensor Prototyping





PRESENTED BY

Christian Poppeliers

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2022-xxxx

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Moment Tensor Prototyping Motivation and Plan

MOTIVATION

- Provide a well-structured, well-documented roadmap for developers
- Provide more utility/customization than previous tools
- Include a Graphical User Interface (GUI)

PLAN

- Based on Gene Ichinose's MTINV toolbox and Andrea Chiang's TDMTPY package
- Uses Bob Herrmann's Green's function calculations
  - Initial focus on inversion with precomputed library of GFs
  - Implemented GFs "on-the-fly" calculation during the processing workflow (v2.0)
- Prototyping is done entirely in Python

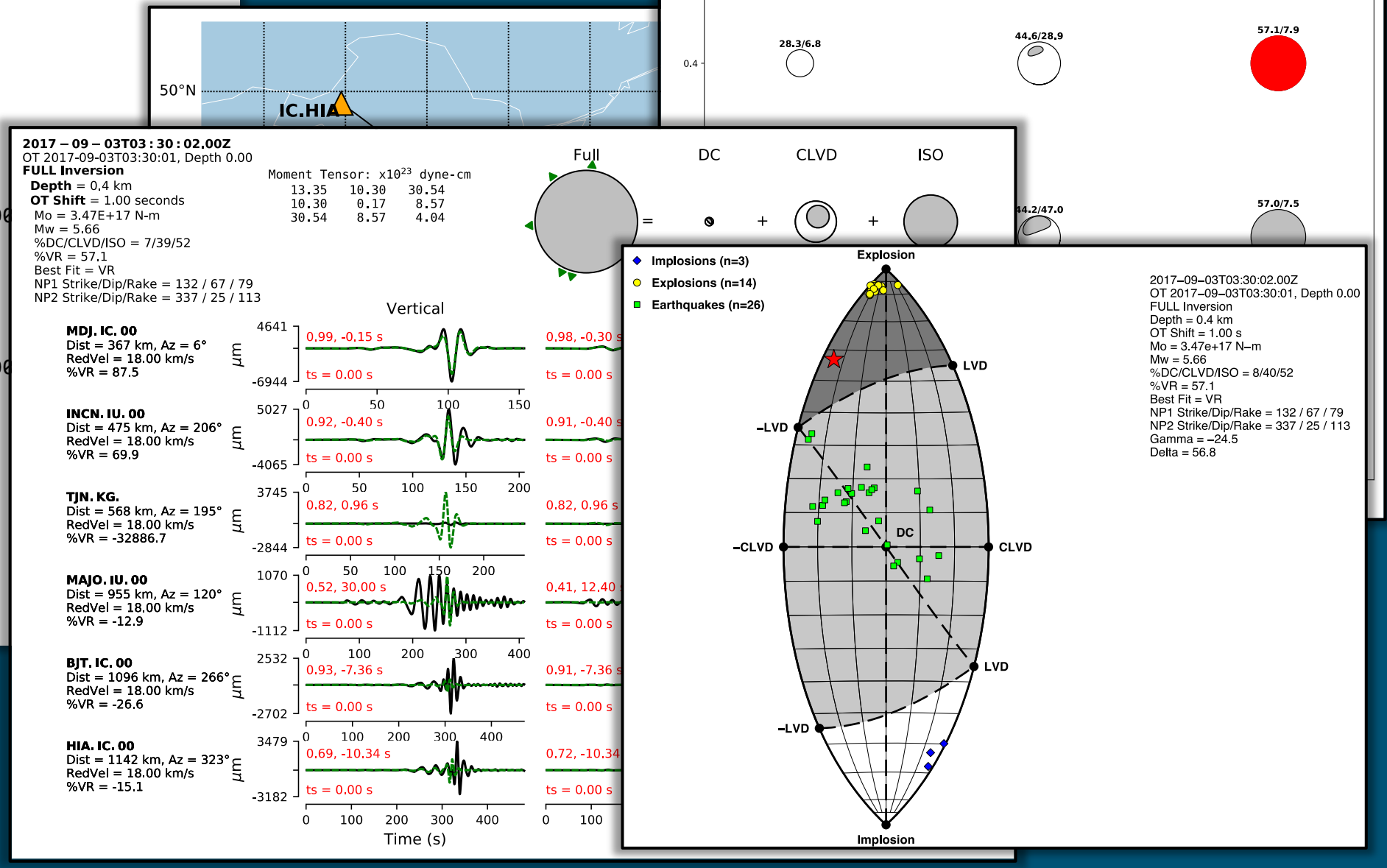
Moment Tensor Prototyping Accomplishments

Deliveries

- Nov 2020 – Version 1.0, backend inversion code run from command line
- Mar 2021 – Version 1.5, included a graphical user interface (GUI)
- Mar 2022 – Version 1.7, cleanups including minor bug fixes, improved visualizations, added user-requested processing parameters, more robust waveform/database readers, etc.

MT Prototype – Version 1.0, command line operation

```
(momenttensor) > mt-inv mtinv.in

 For t0 = -1.0 and depth = 0.4 km
   Full Moment Tensor Inversion
   Mw = 6.04
   Percent DC/CLVD/ISO = 6/43/50
   VR = 28.29%

 For t0 = -1.0 and depth = 0.600000000
   Full Moment Tensor Inversion
   Mw = 5.89
   Percent DC/CLVD/ISO = 8/43/47
   VR = 27.28%

 For t0 = -1.0 and depth = 0.800000000
   Full Moment Tensor Inversion
   Mw = 5.78
   Percent DC/CLVD/ISO = 11/43/45
   VR = 26.77%

 For t0 = 0.0 and depth = 0.4 km
   Full Moment Tensor Inversion
   Mw = 5.72
   Percent DC/CLVD/ISO = 28/33/37
   VR = 44.56%
```



MT Prototype – Version 1.7, GUI Operation

Run Parameters | Event / Station Parameters | Waveform Fits | Solution | Visualizations | Report

Inversion Parameters

Inversion Type: Deviatoric
Green's Function Type: Herrmann
Green's Function Location: HerrmannGFlibraries Browse...
Base Moment: GF Default 1e+20 dyne-cm
Correlate?
Correlation Tolerance: 0.85
Maximum Shift 10.00
t0 0.00
User SNR Defining? Minimum SNR: 3.0
Best Fit: VR VR+PDC

Solution Search Parameters

Origin Time (trange):
Min: 0.00 sec Max: 0.00 sec Step: 1.00 sec
Depth (zrange):
Min: 24.02 km Max: 24.02 km Step: 1.00 km

Write Out Parameters

Write out Green's functions
Write out Synthetic Seismograms

Plotting Parameters

Plot stations on focal mechanism
Plot reference populations on Lune diagram
Plot cross-correlation parameters (maxcorr/lag
Plot component time-shifts
Plot non-defining components

Load Defaults

Defaults File:
No file selected Browse...
Load File

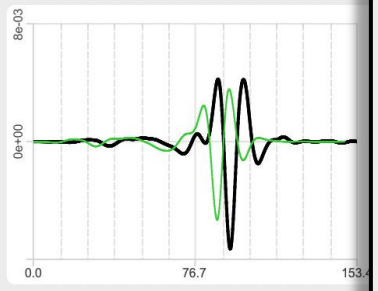
Select Configuration Default Directory

Run Parameters | * Event / Station Paran

Vertical

Station Defining
MDJ.IC.00.BH?
Dist: 367.9 km, Az: 6.0*
RedVel: 12.00 km/sec
AmpFac: 1.00
%VR (Station): 80.22

Component Defining
Shift: 40 samp
6.00 sec
MaxCorr: 0.98
Lag: -3 samp / -0.45 sec
%VR: 98.67



Station Defining
INCN.IU.00.BH?
Dist: 475.4 km, Az: 206.0*
RedVel: 12.00 km/sec
AmpFac: 1.00

Component Defining
Shift: 0 samp
0.00 sec
MaxCorr: 0.91
Lag: -7 samp / -1.40 sec

Run Parameters | Event / Station Parameters | Waveform Fits | Solution | Visualizatio

Observed Data Directory: Data Browse... Response File Directory: Resp Browse

Initial Station Parameters

Reduction Vel: 18.0 km/s
Shift: 0.0 sec
Number of Points (npts): 1024
Sampling Interval (DT): Distance Dependent
0.00 sec/sample
Minimum Defining Distance: Magnitude Dependent
0.00 km
Maximum Defining Distance: Magnitude Dependent
1000.00 km
Max Defining Stations: 8 stations
Motion: d - Displacement
Weight Scheme: None
Velocity Model: wus

Filter Params

Filter: BP - Bandpass
# of Corners (#C): 3
# of Passes (#P): 2
Low Corner (LC): 0.0200 Hz
High Corner (HC): 0.1000 Hz

Reset Initial Station Parameters

Event Parameters

Comment: OT 2015-06-24T19:12:55, Depth 33.7
Origin Time: 2015-06-24T19:12:55
Latitude: 41.827
Longitude: 88.478
Depth (km): 33.7
Magnitude: 5.0
Event Info File: 015Earthquake/event.info Browse.
Load File

Run Setup

Defining	Station	Dist	Az	RedVel	AmpFac	npts	DT	Motion	Weight	Model	Z	R	T	Shift Z	Shift R	Shift T
☑	WMQ.IC.00.BH?	229.60	344.00	12.00	1.00	256	0.780000	v - Velocity	1.00	litho_42N_88E_modified	☑	☑	☑	-3.9	-3.9	-4.68
☑	AT01.XL..HH?	571.20	44.00	12.00	1.00	256	1.170000	v - Velocity	1.00	litho_42N_88E_modified	☑	☑	☑	-4.68	-4.68	-11.7
☑	AT12.XL..HH?	688.00	60.00	12.00	1.00	256	1.170000	v - Velocity	1.00	litho_42N_88E_modified	☑	☑	☑	-2.34	-2.34	-3.51

Moment Tensor Prototyping Upcoming Work

Refining Version 1.7

- Fixing minor bugs
- Responding to feedback, adding minor features

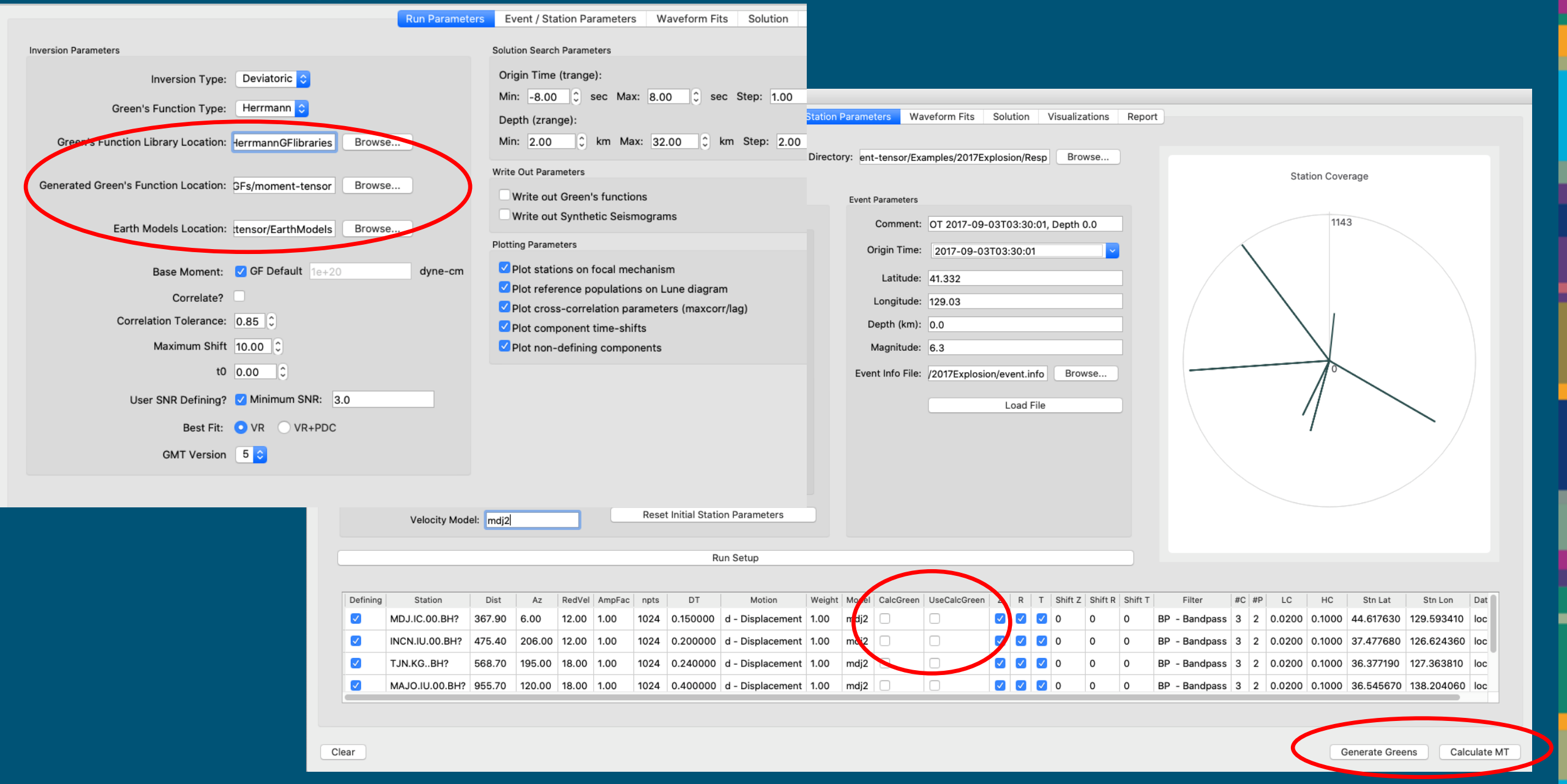
Developing Version 2.0: Adding functionalities

- Can calculate Green's functions (GFs) during processing workflow ("on-the-fly")
- Choice of velocity model, Greens function type (1D), and whether to use library GFs

On-going improvements

- Single-station mixed database-local usage
- Forward calculation
- Fixed isotropic depth
- Additional visualization tabs

MT Prototype – Version 2.0



Moment Tensor Prototyping Summary

- Python-based, moment tensor inversion prototype based on Gene Ichinose's MTINV delivered
  - Version 1.0 – Command line
  - Version 1.7 – GUI

- Work is continuing
  - Refining/cleaning up current delivered version; v1.7
  - Finalizing additional functionalities (e.g., Green's function calculation on-the-fly); v2.0

Geophysical Monitoring System (GMS)
Generalized F-detector Prototyping

Julia A. Sakamoto

May 6, 2022

SAND2021-xxxx

Introduction

- USDC implementation (incorporating Selby's Fortran code into DFX) would not effectively guide GMS development team who will code Gen-F.
- SNL's pure Python version has improved and simplified structure, elaborate commenting, more screen outputs during runtime, and an accompanying user guide.
- We have validated our implementation using test datasets and processing results to check against Selby's code.
- Delivered to USNDC to be used for continued testing and tuning of Gen-F (complements ops DFX version).

From Traditional F-detector to Generalized F-detector

- The F-detector is generalized to utilize *a priori* information about signal and noise. Required modifications are conceptually simple: assume the noise is correlated, model the noise, account for that model in calculations of the beam and corresponding F-statistic.
- Yet the theoretical framework is quite complex, involving statistics, probability theory and physical models of noise and signals. Three papers by N. D. Selby describe the theory and methodology:
  - 2008 and 2011 papers pertain to frequency-domain implementation.
  - 2013 paper implements a hybrid frequency-time-domain multiple-filter approach.
- Advantages:
  - Arrays are treated equally and objectively.
  - Much less time-consuming tuning is required.
  - Number of false detections is greatly reduced. More candidate associations (compared to traditional signal detection methods used at the IDC) are made despite the total number of F detections reduced by a factor of two.
  - F is statistically convenient: it follows a well-known distribution and can thus be converted to a probability of detection for a given *a priori* SNR.

Gen-F Prototyping Accomplishments

- Previously:
  - In-depth understanding of Selby's papers (2008, 2011, 2013) and comparison of complex theoretical framework with vast Fortran code invoking discussions with N. D. Selby and Jeff Given of Leidos.
  - Delivery of Gen-F 1.0, a fully functional prototype built from the ground up using Python OOP.
  - Meticulous testing of algorithm demonstrated highly consistent results with Fortran implementation.
  - Edits to address issues experienced by Leidos (e.g., program crashing, bizarre results for certain arrays).
- Recently:
  - Augmented algorithm to process data from CSS database tables using database reader codes (A. Conley, B. Young).
  - Major restructuring of algorithm into well-organized subdirectories, e.g., *genf* library of source code.
  - Modification to critical subroutines to process both original IMS format datasets from Fortran implementation and now waveform data (and metadata) and response data from database tables.
  - Software packaging by our support team for slick installation, easier operation of Gen-F tool via command-line interface, and other enhancements and functionalities.
  - Preliminary processing of ARCES array via database tables to replicate key figures in Selby's papers.
  - Delivery of Gen-F 2.0 in March 2022.

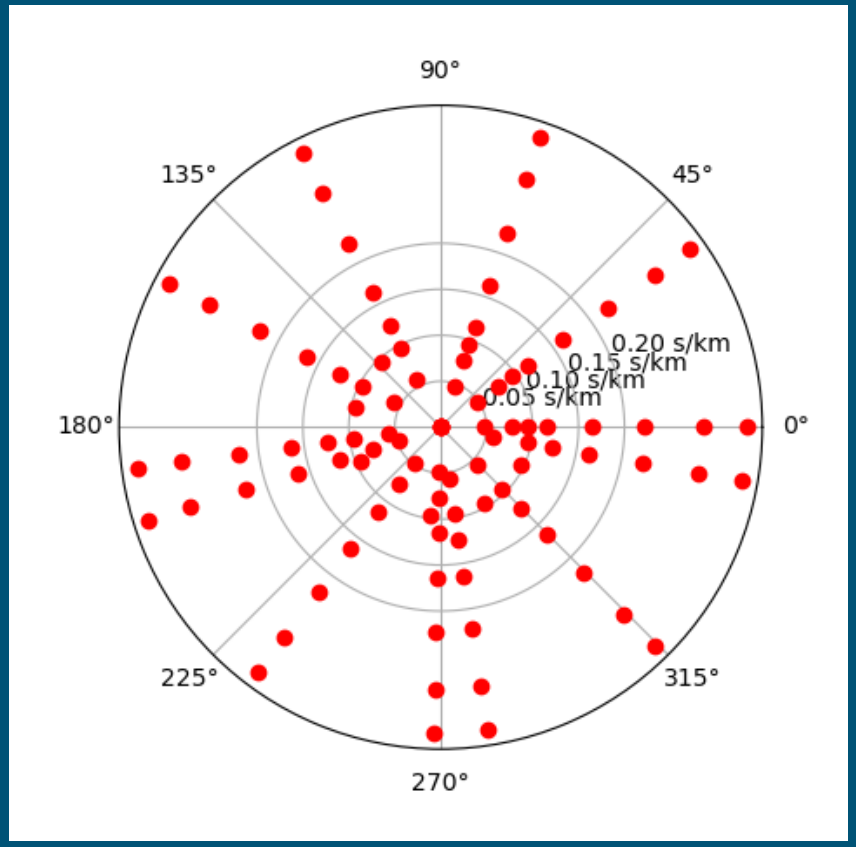
Gen-F Prototype Code Sample

- Example of simple function for reconstructing detection beams.
- Extensive commenting incorporated throughout code:
  - Provides clear definitions of variables and purpose of functions.
  - Describes the methodology in the papers with citations and equation numbers.
  - Clearly explains various operations from both physical and numerical perspectives.
  - Notes modifications from the Fortran implementation.

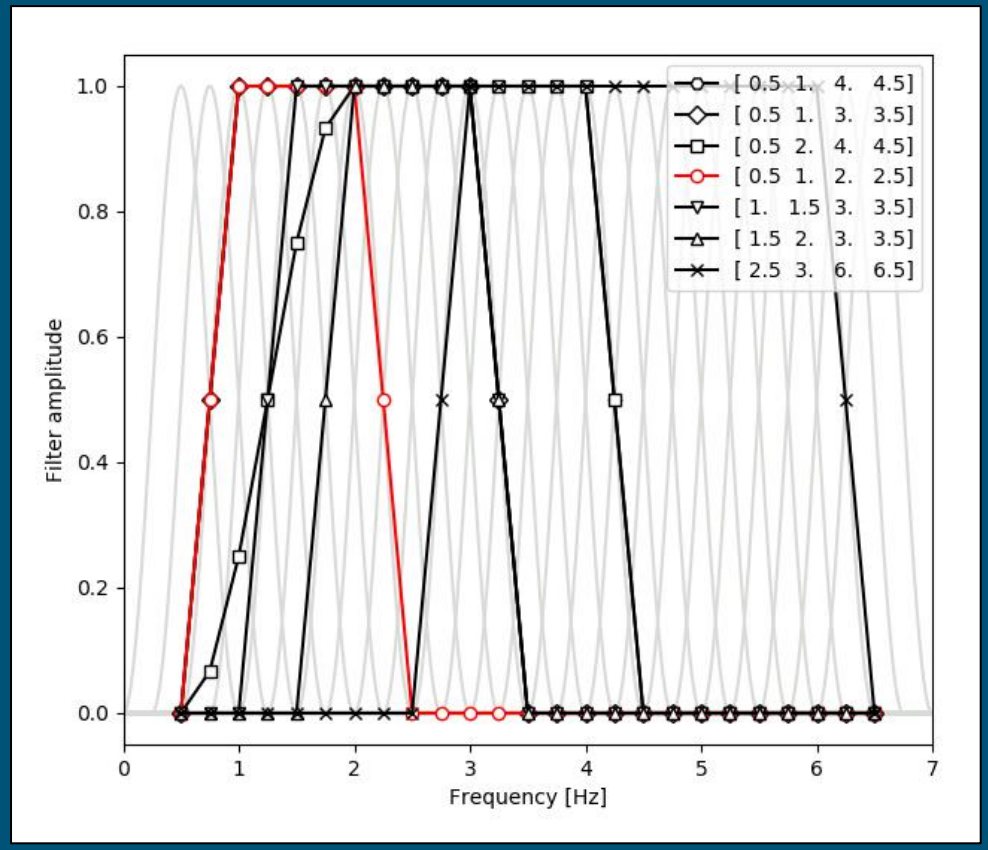
```python
def detBeamRecon(fta, samplingRate, maskArray, detection):
    """
    Reconstructs the detection minimum-power beam described in Eqn. 7 of Selby, 2011. Waveform data have
    already been shifted in the methods of the 'MinimumPowerBeam' class, prior to the detection phase, but
    without proper beam weighting or normalization. The latter is applied by this function, post-detection.

            'fta' : 'FrequencyTimeAnalysis' object
    'samplingRate' : Sampling rate [Hz] (real float)
       'maskArray' : Power mask per channel per sample point [npts, numChan] (boolean: 0 or 1)
       'detection' : 'Detection' object
       'beamTrace' : Reconstructed minimum-power beam for given detection (real float)
    """

    # Extract the narrowband-filtered waveforms. Array size of 'fta.value' is [npts, numFreq, numChan], where
    # the number of frequencies equals the number of narrowband filters. Values are complex.
    ftaValue = fta.value

    # Convert 'dt' per channel to integer shifts per channel. Selby does type conversions here with NINT, which
    # rounds to the nearest integer: for a > 0, NINT(a) = INT(a + 0.5). Need a loop for now, since 'np.int'
    # works only on scalars.
    shift = np.zeros(numChan, dtype=int)
    # Loop over channels
    for k in range(np.shape(ftaValue)[2]):
        shift[k] = np.int(np.round(-1. * samplingRate * detection.dt[k]))

    # Normalization per frequency in the MP beam equation
    norm = np.sqrt(detection.noisePower * detection.posterior)

    # Loop over frequencies
    for n in range(np.shape(ftaValue)[1]):
        # Shift and normalize the filtered waveforms and apply the detection passband filter for the nth
        # frequency.
        ftaValue[:,n,:] = np.roll(ftaValue[:,n,:] * maskArray, shift, axis=0) * detection.filter[n] / norm[n]

    # Detection beam weights are real with an array size of [numFreq, numChan]. Stack them 'npts', i.e.,
    # shape(ftaValue[0]) times. (Note that while the beam weights are complex-valued when originally computed
    # by the 'MinimumPowerBeam' class, Selby only keeps the real part in the 'Detection' object.)
    weightsStack = np.dstack([detection.weights] * np.int(np.shape(ftaValue)[0]))
    # Apply proper beam weights per frequency per channel for each sample point in the fta waveforms. 3D array
    # 'weightsCopy' must be transposed to match the dimensions of 'ftaValue'.
    ftaValue *= weightsStack.transpose(2, 0, 1)

    # Marginalize the frequency dimension from the weighted fta waveforms, i.e., sum over frequencies (inner
    # sum), reducing array size to [npts, numChan]. Apply the shifted power mask per channel. Sum over channels
    # (outer sum) to form the final reconstructed beam.
    beamTrace = np.sum(np.sum(ftaValue, axis=1), axis=1)

    # Selby keeps only the real part
    return beamTrace.real
```

Gen-F Prototype Results: Test Beamset & Passband Filters

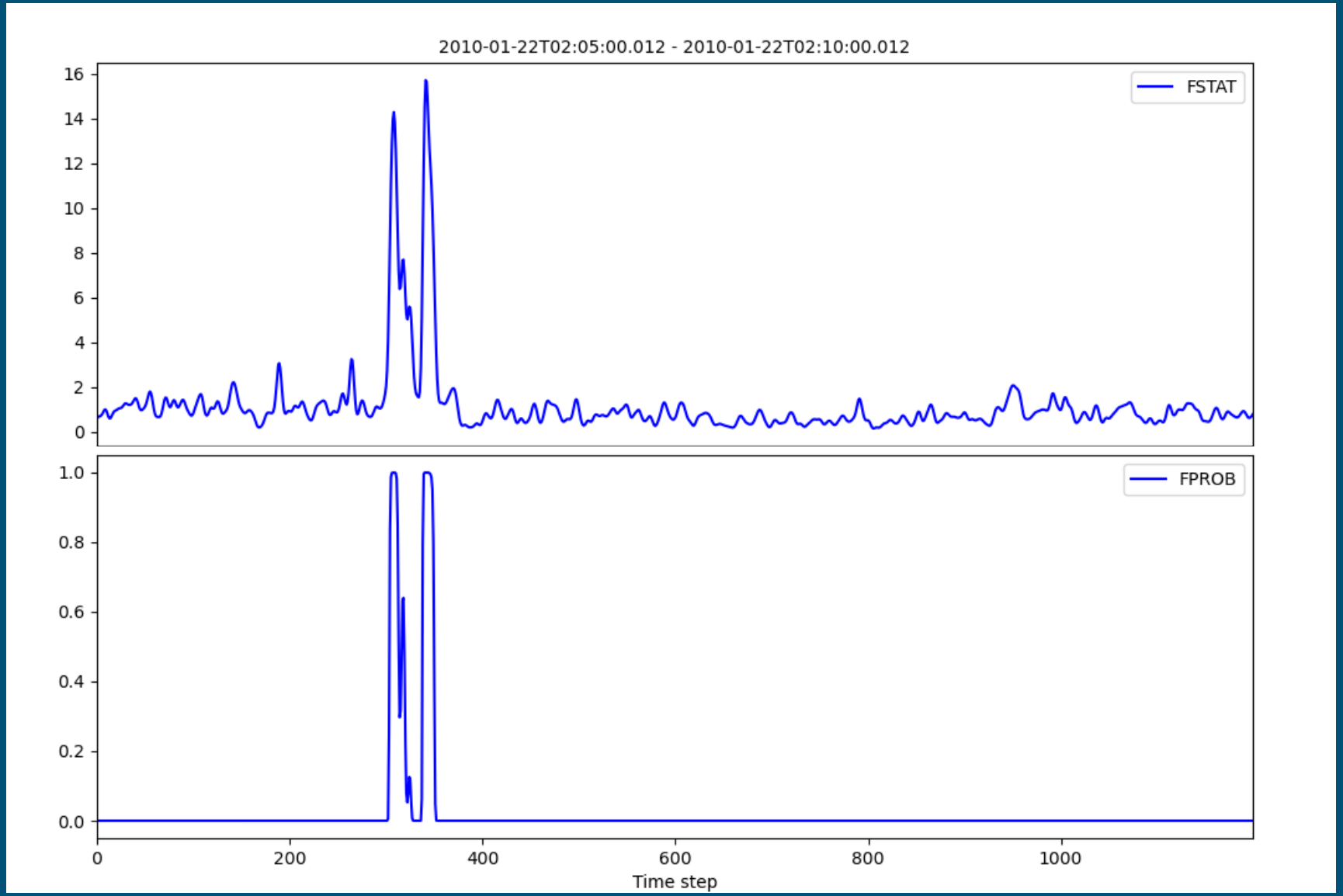
ARCES Beamset



Passband Filters



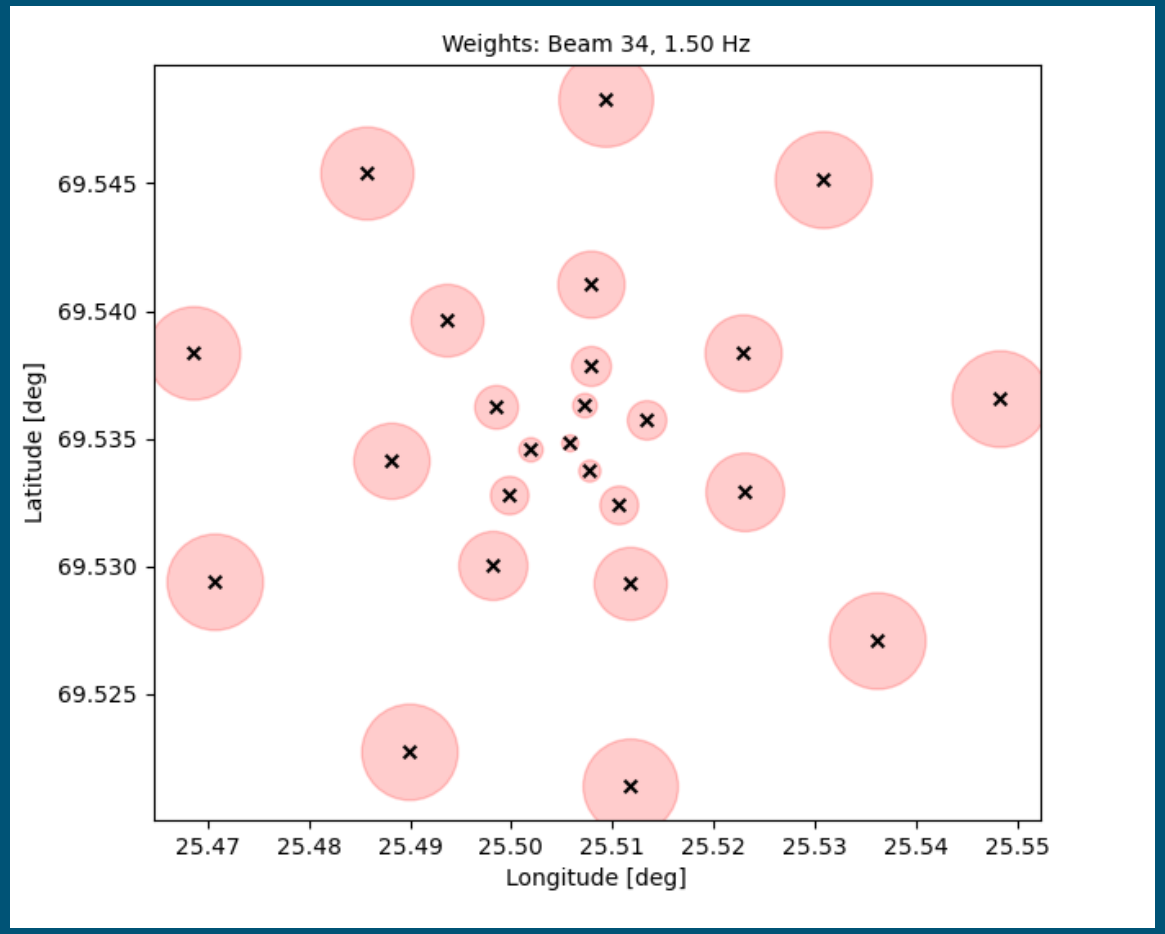
Gen-F Prototype Results: F-statistic & F-probability Traces



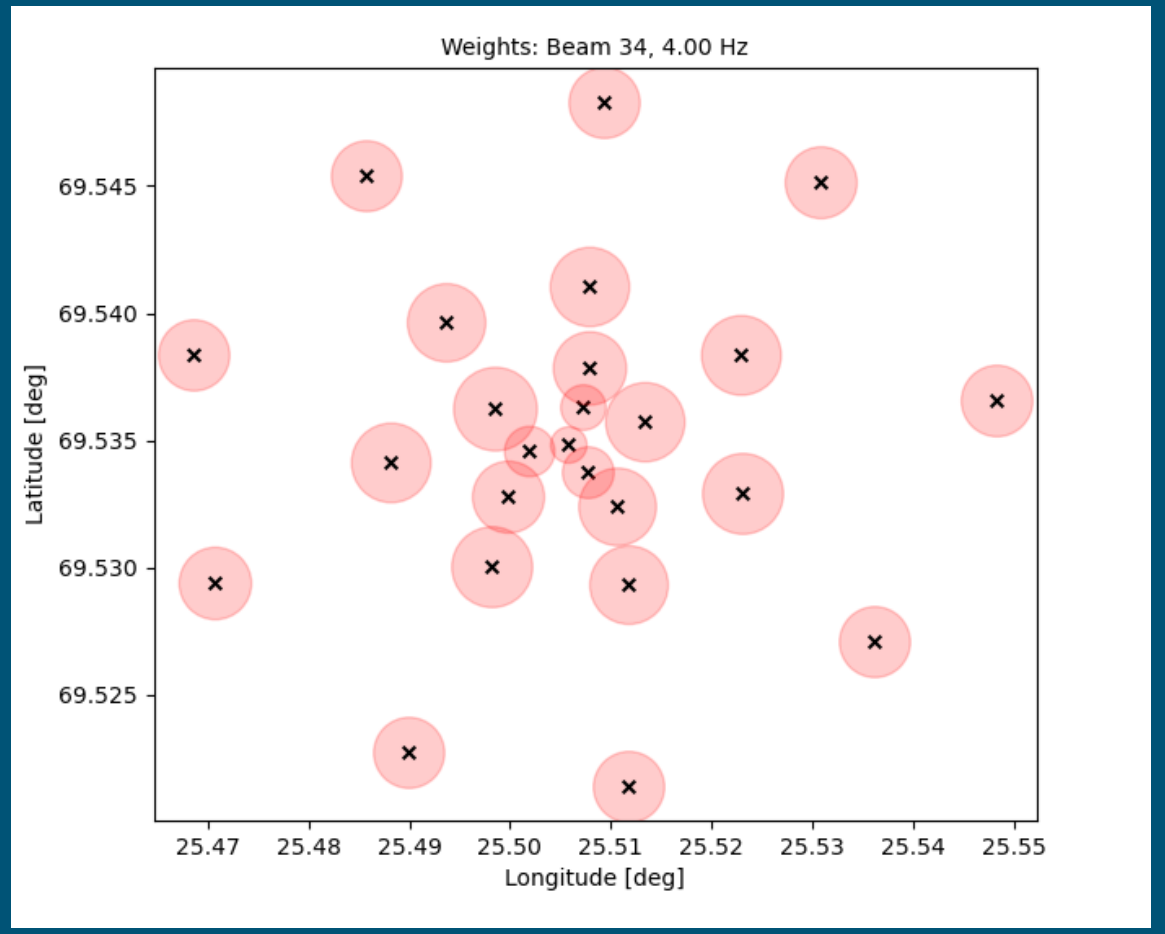
Gen-F Prototype Results: Minimum-Power Beam Weights

- Minimum-power detection beam weight per channel at frequencies 1.5 and 4.0 Hz.

0.50 Hz



4.50 Hz



Generalized F-detector Prototyping Summary

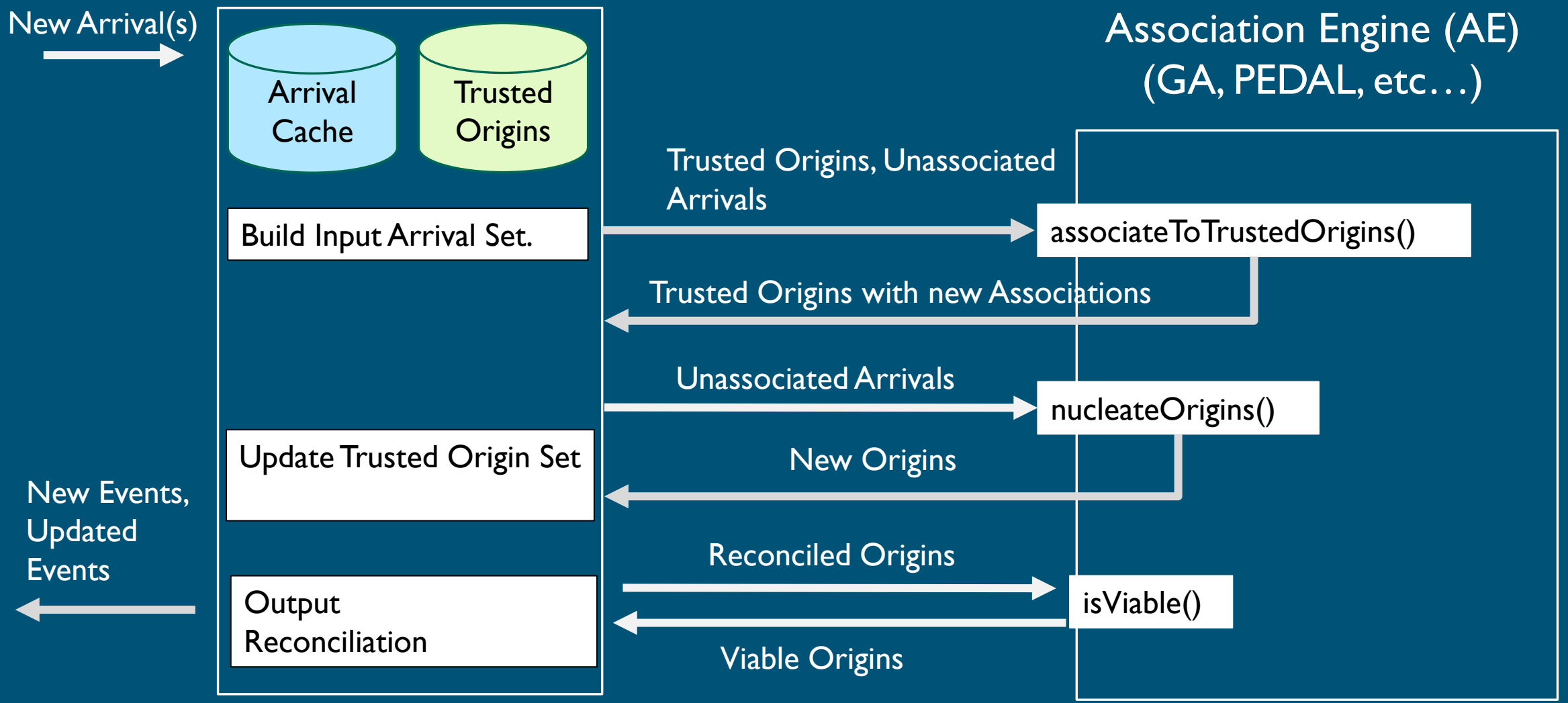
- We have successfully developed a pure Python Generalized F-detector prototype based on Selby's frequency-time-domain methodology (2013), initially implemented in Fortran.
- Prototype will later be used to guide operational (Java) code development and will be significantly easier to understand by developers.
- Gen-F 2.0 now processes waveform data from CSS database tables.
- Software packaging provides slick installation and offers easier operation through the command-line interface.
- Gen-F 2.0 and accompanying user guide was recently delivered to USNDC (AFTAC).
- Upcoming work:
  - Continue to process ARCES array and replicate key figures in Selby's papers.
  - Incorporate updated database reader codes and simplify associated configuration (par) files.
  - Expand user guide into a developer guide to include details on algorithm in terms of methodology and theoretical framework.
  - Incorporate feedback from USNDC.

Geophysical Monitoring System (GMS)
Global Associator (GA)

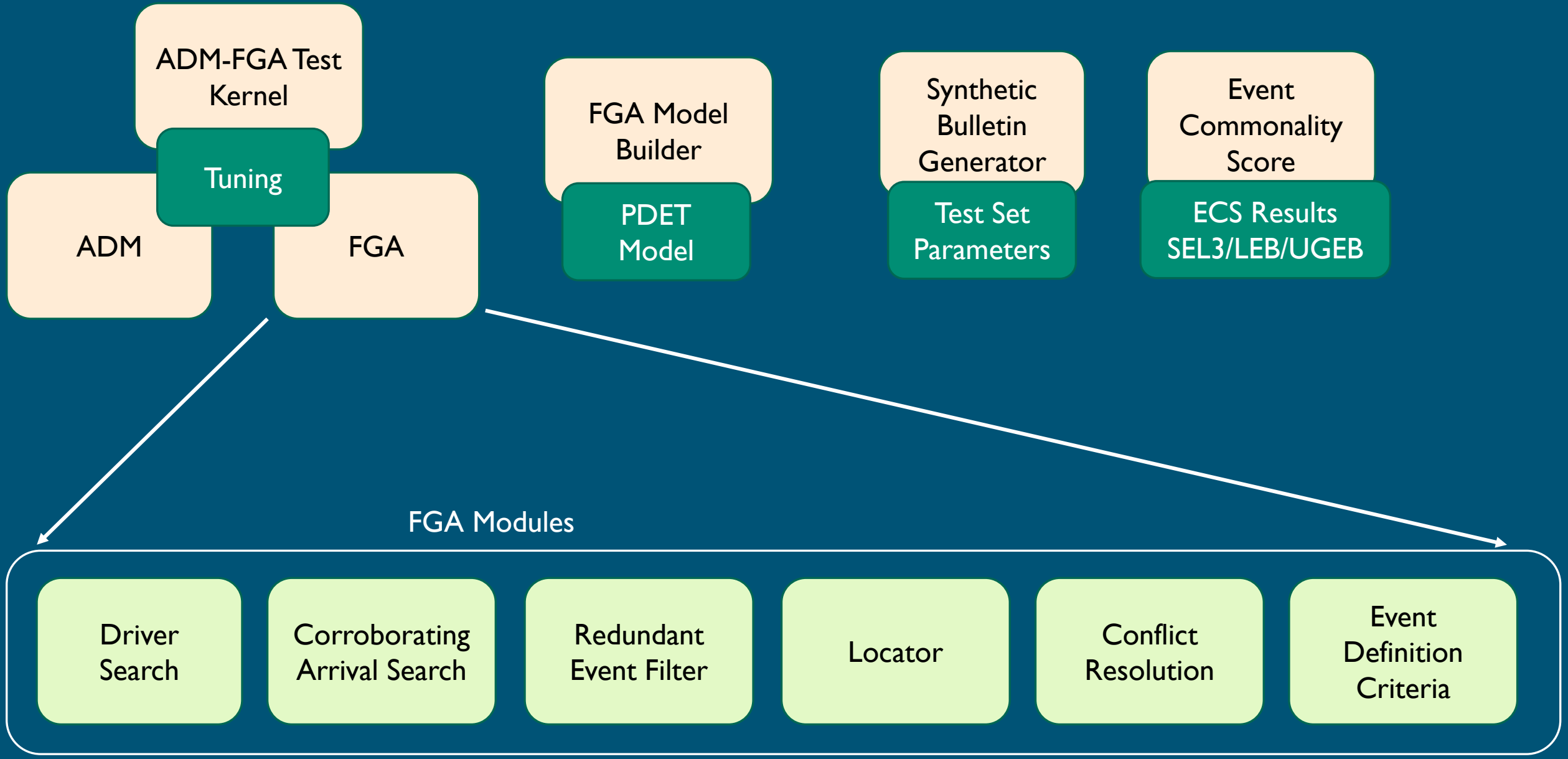
Stephen Heck

Association Prototype Core Architecture

Associator Data Management (ADM)



Prototyping Effort Java Software Libraries/Data Products



Global Associator Prototyping Accomplishments Jan 2021 - Present

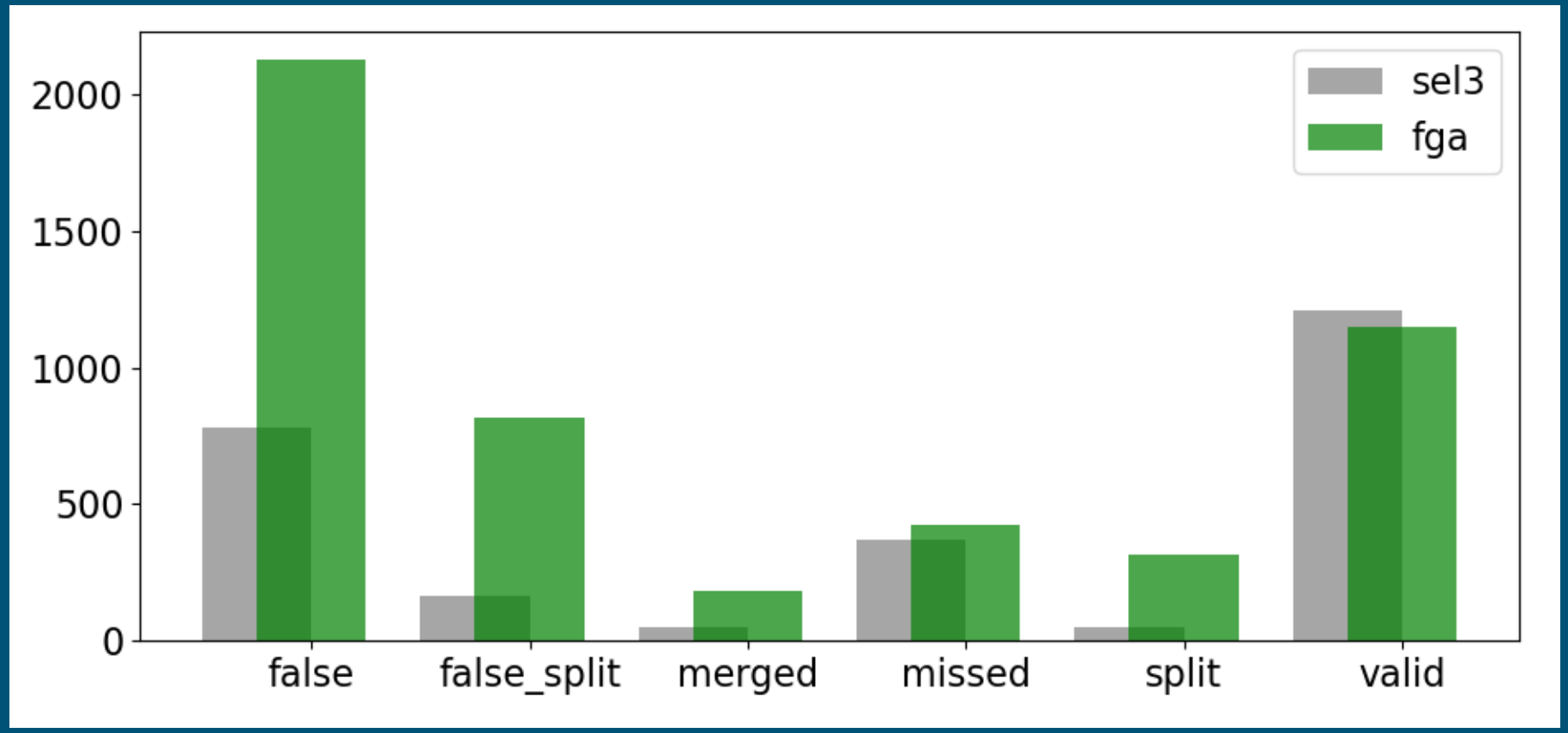
Code Updates

- Phase Translation
- **Large event processing**
- **Outlier analysis**
- **Network Probability of Detection**

ADM-FGA Testing

- Full signal environment problem (5/15/2010 – 5/29/2010)

ADM-FGA vs SEL3 ECS Results (May 15, 2010 – May 29, 2010)



ADM-FGA

n_true	n_false	n_ref	Precision	Recall	F(1)
1151	3344	1577	0.26	0.73	0.38

SEL3

n_true	n_false	n_ref	Precision	Recall	F(1)
1208	996	1647	0.55	0.73	0.63

Global Associator Prototyping Current Work

SEL3 Testing and Tuning (Subject Domain Concerns)

- Match or beat the SEL3 Bulletin

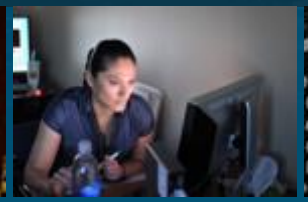
On the horizon (GMS Engineering Concerns)

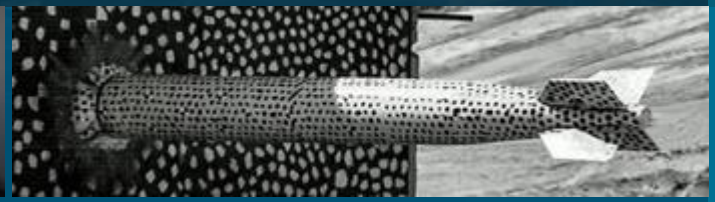
- Computational performance testing
- Design using GMS Architecture/Technologies/Data Model

Sandia National Laboratories

Geophysical Monitoring System (GMS)

Subject Matter Expert (SME) Testing Support





PRESENTED BY

Chris Young

U.S. DEPARTMENT OF ENERGY | NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2022-#####

4 Ways SMEs Test GMS or Support GMS Testing

1. Create test data sets for software developer use
2. Validate COI data access
3. Validate data processing algorithms
4. Alpha test the Analyst Interface

1 Creation of Test Data Sets for Software Development

The Need

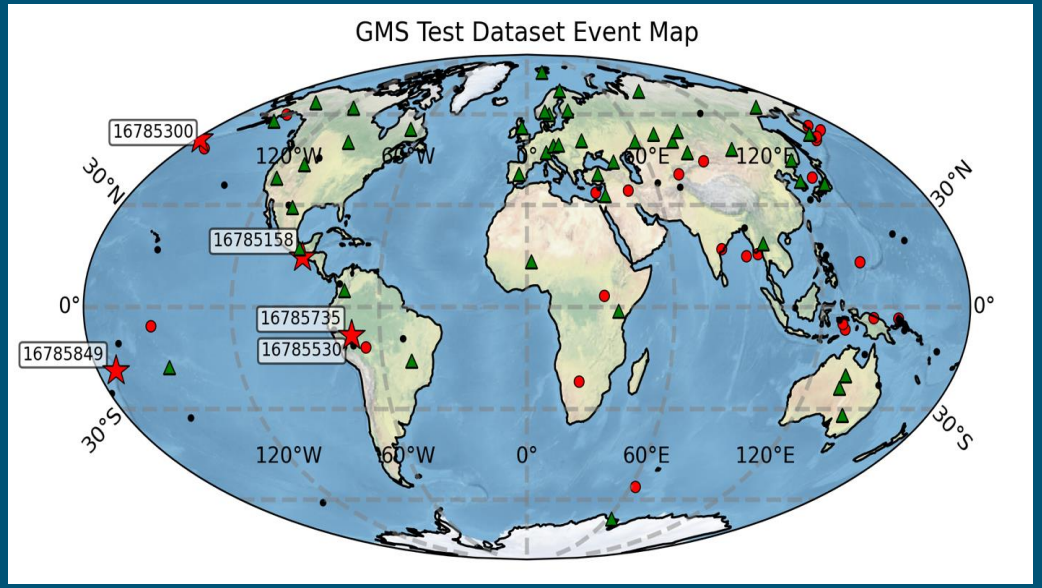
- Current focus of GMS is on development of Interactive Analyst Interface (IAN).
- IAN uses data stored in USNDC database, brought across the data bridge, and accessed via COI.
- Until very recently (April 2022), SNL has not had a real USNDC test data set (USNDC data is classified; GMS development is not).

SME Solution

- Use IDC data (IDCIDCX, SEL3, LEB).
- Augment with additional table content needed to represent USNDC schema.

The 2019 Jan 5 Data Set

- A “typical” interval of 2 hours from 2019, Jan 05 (with 2 hours before and after)
  - Several good sized events (NDEF 15-50) in different regions, but nothing huge
- Data editing/augmentation
  - Copy data from appropriate IDC accounts/tables
  - Get rid of any info for stations we don’t have waveform data for (subset of IMS, no infra, no hydro)
  - GLOBAL
    - 5 min Detection “beams” and event “beams” linked via WFTAG
  - DETPRO/SOCCPRO
    - Added AR_INFO table
    - Added ARRIVAL_DYN_PARS_INT, FILTER, FILTER_GROUP
  - AL1
    - Added AR_INFO table
    - Added EVENT_CONTROL table rows
    - Added ARRIVAL_DYN_PARS_INT, FILTER, FILTER_GROUP



ORID	LAT	LON	DEPTH	TIME	NASS	NDEF	MB
16785735	-8.127	-71.604	586	07:25:39 PM	79	39	5.6
16785530	-8.126	-71.660	0	07:26:44 PM	79	36	6.13
16785300	51.414	-178.221	0	06:47:07 PM	109	31	5.40
16785158	14.494	-93.021	45.6	06:52:59 PM	32	19	4.31
16785849	-18.505	-172.302	0	05:08:35 PM	41	18	4.60

2 Validation of COI Data Access

The Need

- Data is bridged from the current USNDC storage (db tables + files), translated into GMS objects, then accessed via the COI
- We need to verify both the bridging of information and the COI access

SME Solution

- Create Python code that accesses data directly from db tables and via the COI and compare
- Separate validation for each type of data (data processing intervals, station reference info, waveforms, signal detections, events)
- Top level code module is made as simple as possible; complexity hidden

Validation of COI Data Access: Interval Example

```
# Import parameters from "params_interval.py"
from params_interval import*

# Import functions for queries"
from queryFunctions import*


# Define time range
startTime_datetime = "2019-01-01T00:00:00Z"
endTime_datetime   = "2019-02-01T00:00:00Z"



# Query the database using the Workflow Manager Service >>>>>>>
wm_status, wm_stage, wm_startTime = callService(startTime_datetime, endTime_datetime, stageidAN)

# Query the database directly >>>>>>>
# Initialize cx_Oracle client if it cannot find the lib directory. Only execute once.
#cx_Oracle.init_oracle_client(lib_dir="/Users/astanci/Library/Oracle/instantclient_12_2")

jnb_status, jnb_stage, jnb_startDateTime, jnb_statusWM = callDB(startTime_datetime, endTime_datetime, stageidAN, tableID, INTclass)

# Compare the rresults and mark mismatches >>>>>>>
compareResults(wm_status, wm_stage, wm_startTime, jnb_status, jnb_stage, jnb_startDateTime, jnb_statusWM)
```

- DB config params (accounts, passwords, table names) hidden in params_interval.py
- Complicated formation of JSON queries hidden in queryFunctions.py

3 Validation of Data Processing

The Need

- Algorithms are usually coded from scratch by computer scientists (not geophysicists or electrical engineers)
- Need to verify that coded algorithms are producing the right results.

SME Solution

- Create Python code that sends the same data to the GMS version of the code and a trusted version and compare the results
- We had done several of these prior to the Pivot to focus on SOH and IAN (these were used for ATG)
- We are just getting to the point of IAN development where data processing is required (e.g. filtering, beaming, FK), so we are starting to work on these again
- Will leverage previous work
- A major difference is that in some cases GMS will use trusted third-party data processing libraries (via WASM) instead of writing everything from scratch

4 Alpha Testing of the Analyst Interface

The Need

- We want to deliver versions of IAN that will meet the expectations of AFTAC expert analyst users
- Our software developers are not geophysicists nor seismic analysts, not are our I&T team staff; they will not use the interface the way AFTAC will

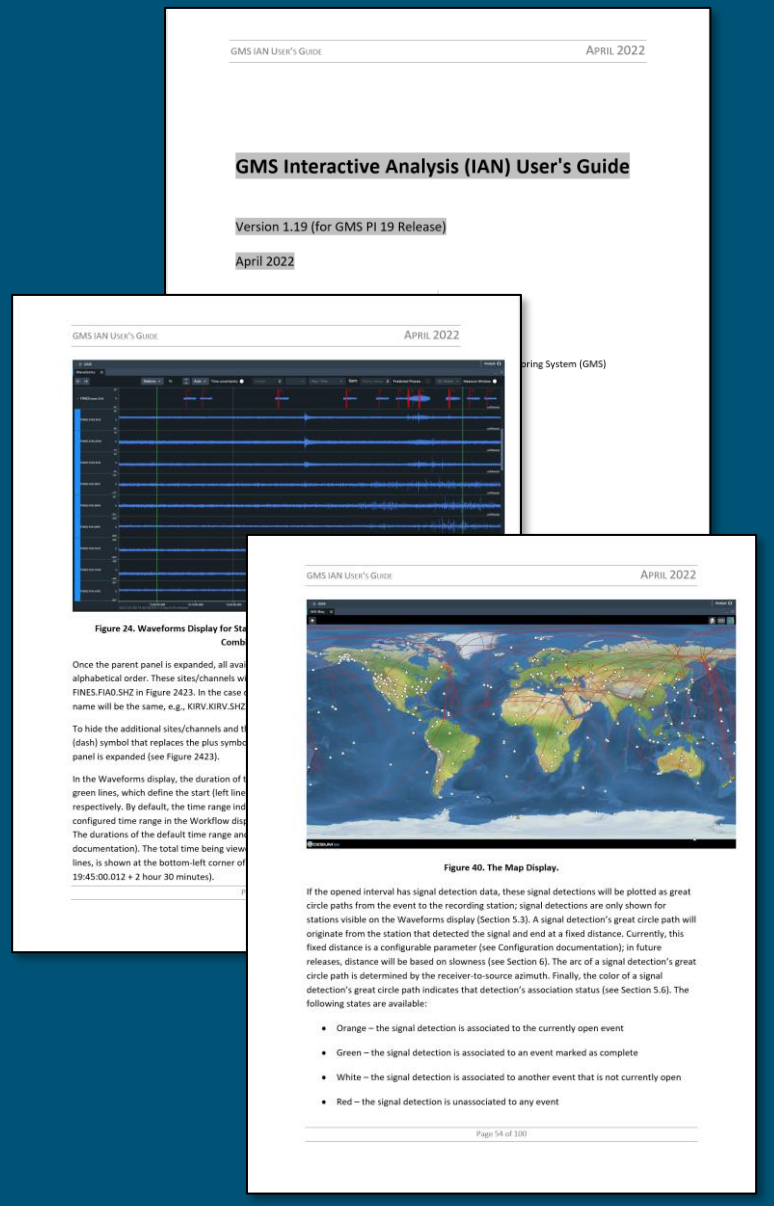
SME Solution

- SMEs will perform alpha testing:
  "Alpha Testing is a type of acceptance testing; performed to identify all possible issues and bugs before releasing the final product to the end users. Alpha testing is carried out by the testers who are internal employees of the organization. The main goal is to identify the tasks that a typical user might perform and test them."

  from https://www.guru99.com/alpha-beta-testing-demystified.html
- A first step is for the SMEs to write the User's Guide for IAN, which we have been doing since IAN development started.
- SME IAN testing has been minimal so far because SME functionality has been so basic (displaying waveforms, panning, zooming, etc.); the complexity of IAN functionality will increase dramatically in FY23 (filtering, beaming, FK, event location).

The IAN User's Guide

- Updated for each PI release to cover new displays and functionality
- Written by a SME (Andrea Conley), proof-read and tested by other SMEs
- Written from the perspective of someone who has done seismic analysis
- Not a tutorial, but shows all the displays and describes how to use them in detail
- Unlike the pre-Pivot User's Guide, this has only IMS stations so can be released to the IDC without modifications



Summary

- The SME Team tests or supports testing of GMS in a variety of ways
- SME efforts ensure that GMS releases are robust and correctly implement the required data processing and analyst functionality
- SME efforts also provide code and documentation that the IDC may want to make use of
- Please contact me if you need more information (cjyoung@sandia.gov)