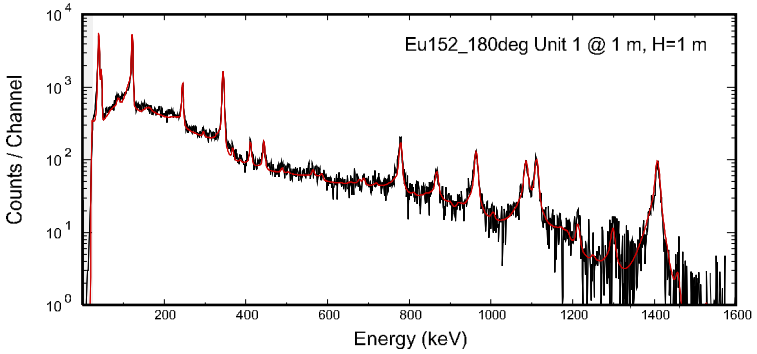
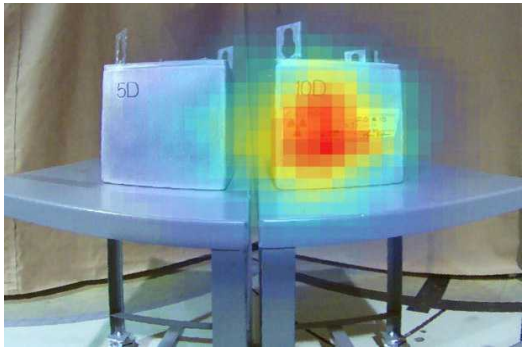
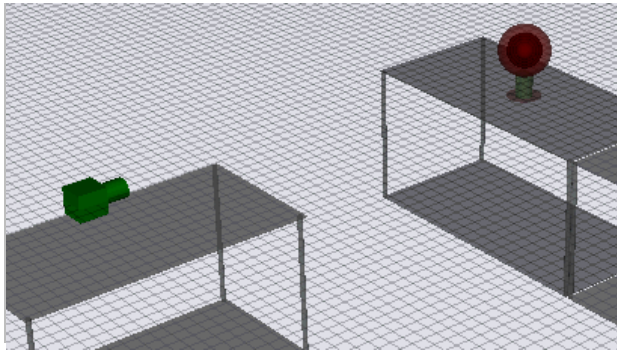
SAND2016-11953PE

Exceptional service in the national interest

Sandia National Laboratories







Inject Calculations with GADRAS

12/7/2016

Greg Thoreson, Lee Harding

U.S. DEPARTMENT OF ENERGY

National Nuclear Security Administration

Sandia National Laboratories is a multi-mission laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

GADRAS Capabilities for Inject

Sandia National Laboratories

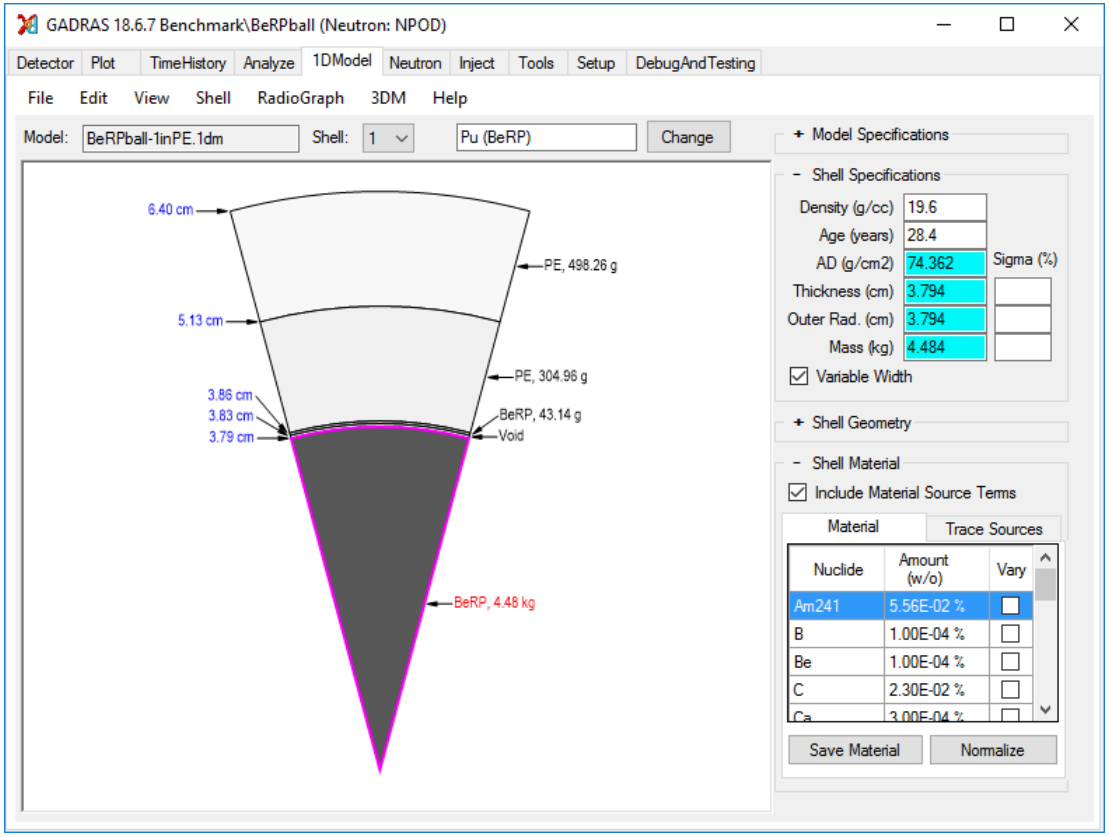
- Source term creation
  - Backgrounds or distributed sources (e.g. primordial radionuclides)
  - 1D models (e.g. shielded HEU)
  - 3D models (e.g. shielded HEU in a truck)
- Inject tool in GADRAS GUI enables synthesis of foreground and background components
  - Batch computation of inject setup (INJ) files
  - External Python/Excel applications exist to create 1000's of INJ files
- Spectrum file tools enable batch computation of directives
- Application Programming Interface (API)
  - This is used extensively by some users
  - C# API with examples distributed with each version
  - Linux API not distributed, but many projects using it

Sandia National Laboratories

SOURCE TERM CREATION

Spherical 1D Models

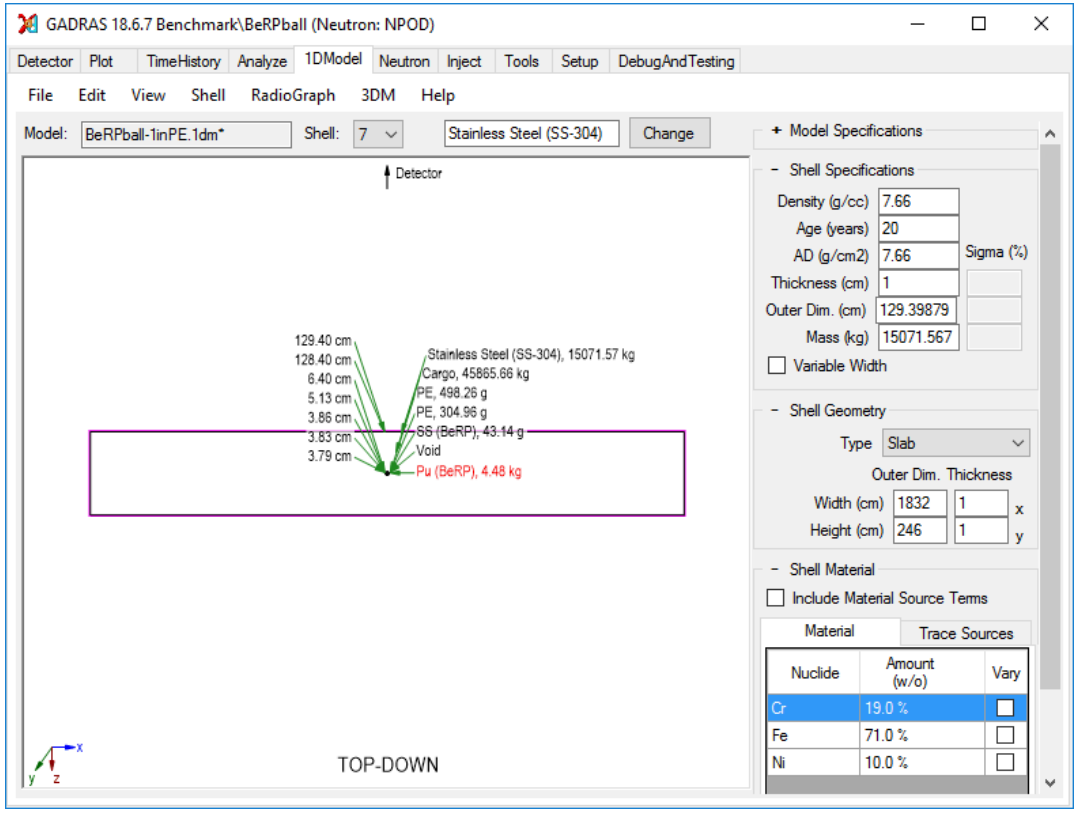
- Simple interface for quickly building nested spherical shells
- Computes gamma/neutron leakage into 4π in a few seconds
  - Also runs electron transport for bremsstrahlung source term
- Distributed material library is customizable
- Can also access this functionality from the C# API



4.5 kg WGPu BeRP Ball in 1” PE

Mixed Geometry Models

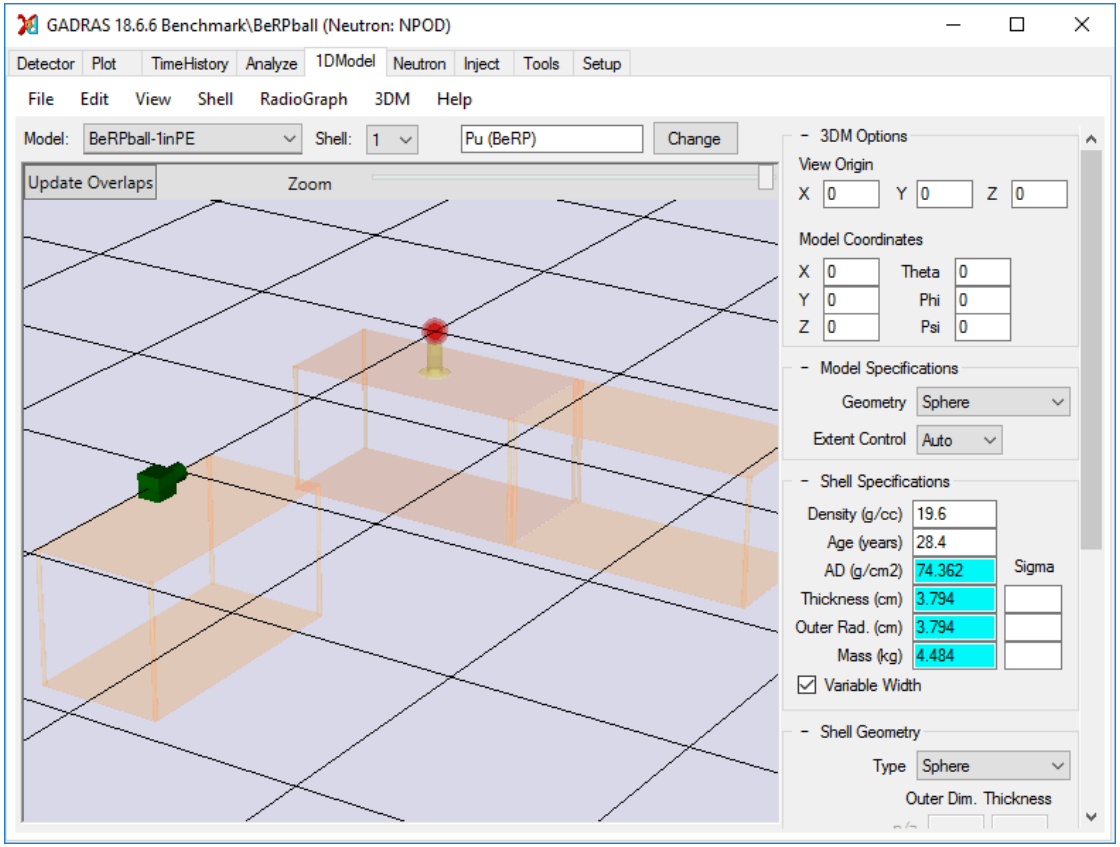
- Intermediary between 1D and full 3D capability
- Can nest spherical shells inside slab geometries
- Limited to having source in center of cargo



BeRP Ball inside 60’ Container Filled with Cargo

3D Models

- Allow full suite of shapes to be used with translation/rotations:
  - Spheres
  - Slabs
  - Cylinders
  - Cones
  - Spherical Caps
  - Spherical-capped Cylinders
- Shapes can be overlapped to create complex geometries



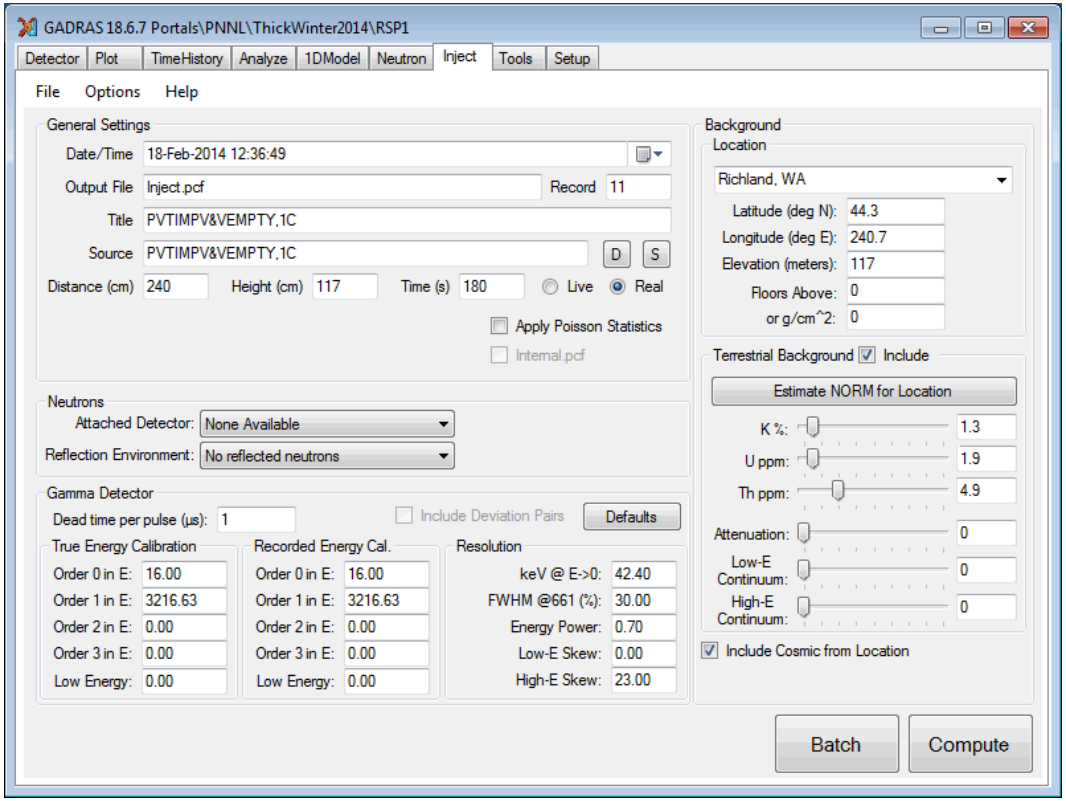
BeRP Ball on Steel Tables

Sandia National Laboratories

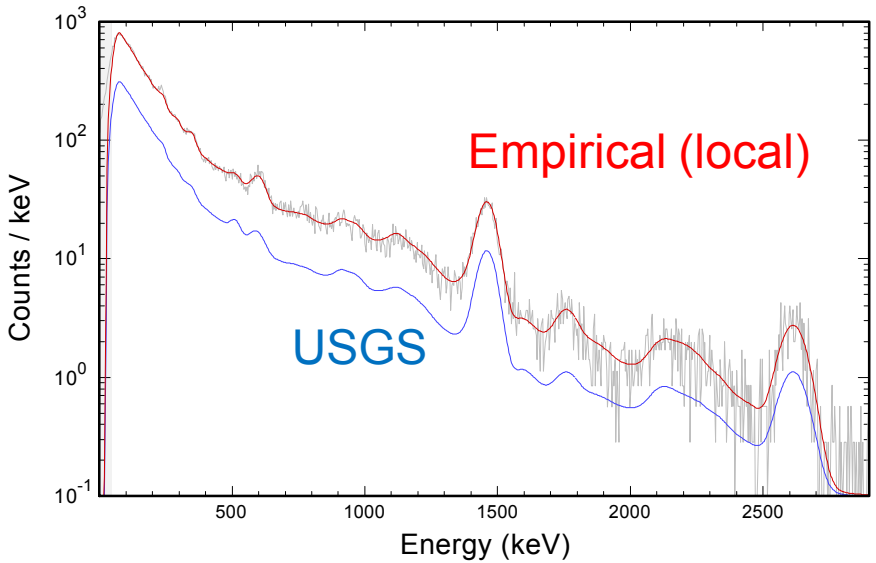
INJECT CREATION

Example Inject Form

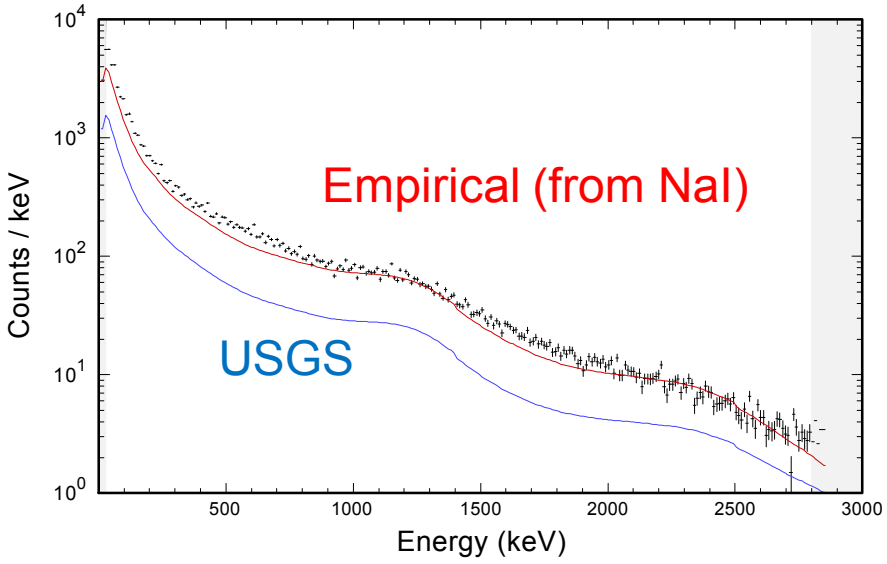
- Features:
  - Input any radionuclide, 1D/3D models
  - Variable source distance, height, translations (for moving sources)
  - With or without Poisson statistics
  - Cosmic and terrestrial background levels based on USGS (adjustable)
  - Co-located neutron detector to output neutron counts to file
  - Tweak spectral output for training (e.g. wrong energy calibration in file)
  - Output a variety of spectral formats (PCF, SPE, N42, etc)



NORM Background Calculations

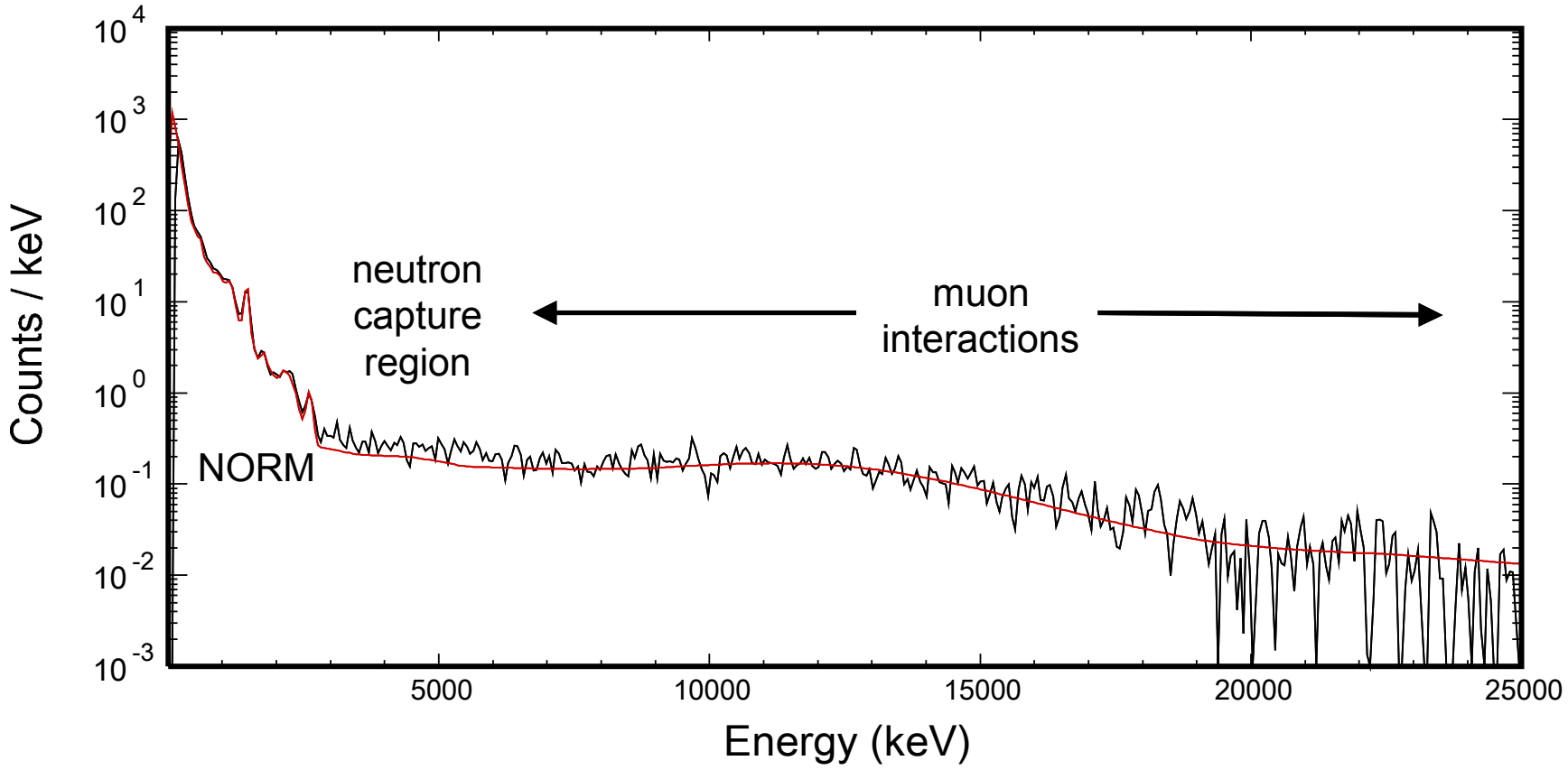


	K (%)	U (ppm)	Th (ppm)
USGS	0.55	0.42	1.86
Empirical	1.3	1.9	4.9



Cosmic Background Calculations

- Computed backgrounds include NORM plus neutron and muon interactions derived from cosmic radiation (1”x1” NaI background shown below)
- The cosmic term is derived from latitude, longitude, an overburden with no adjustable parameters



Background Suppression

Sandia National Laboratories

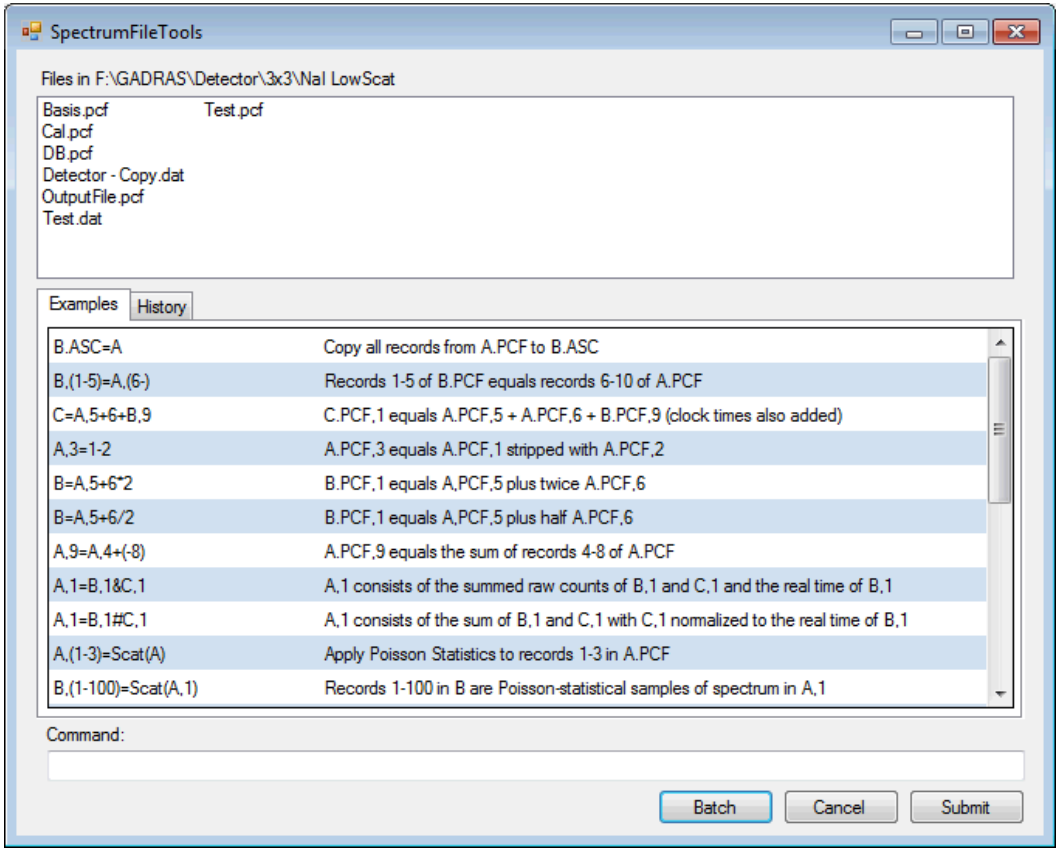
- Attenuating materials consistent with 1D model and dimensions are applied to attenuate and scatter background radiation

- Background radiation is assumed to be isotropic, and different shielding can be applied to the front, back, and sides of the detector
  - The relative background radiation is influenced by shielding

- Can compute suppressed background radiation if no radioactive materials are included in the source model

Spectrum File Tools

- Utility from GADRAS GUI to perform file conversions and manipulations of spectral data
- Can read batch commands from file (which can be written from an external program)
- Can use an infinite-statistic inject and quickly generate randomly sampled spectra for Poisson statistics



Sandia National Laboratories

API

GADRAS API

- Exposed GADRAS functionality written in C#
  - Installed by default to C:\GADRAS\Program\Documentation\APIUsageExamples
- Doxygen documented
  - C:\GADRAS\Program\Documentation\html\index.html
- Examples provided for various GADRAS tools
  - Detector Response
  - Spectra Data File Tools
  - Spectra Analysis Tools
  - Modeling
  - **Inject**

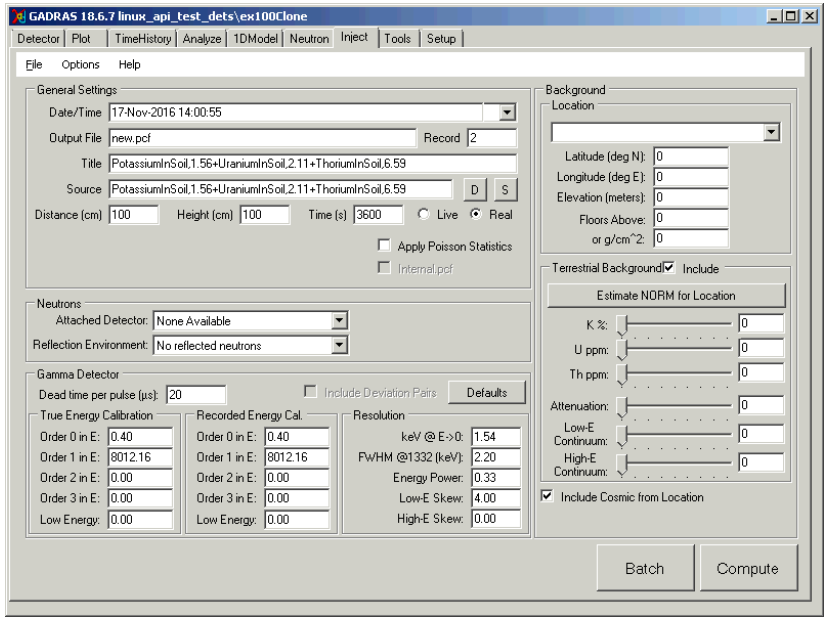
C# Inject API

Sandia National Laboratories

- Allows developers the ability to directly call into GADRAS InjectData function
- Example that is provided with GADRAS API shows how to run many .gam files
- Provides the ability to write a script to generate various inject scenarios
- Also includes API for parallel inject

API Structure

- Simple example provided in C# and Data is passed through a simple structure




```
injectSetup.setDefaults(m_gadrasAPI);

injectSetup.FileName = System.IO.Path.ChangeExtension(p_gamFile, ".pcf");          //PCF filename full path
injectSetup.Title = sourceName;
injectSetup.Record = 1;
injectSetup.Source = sourceName;                                                    //If blank, will generate background record.
injectSetup.ContainsInternalSource = false;                                         //Looks for internal.pcf to include as internal source (common in LaBr detectors)
injectSetup.DetectorDeadTimeUs = 12;                                                //Dead time of detector per event (microseconds)
injectSetup.DetectorHeightCm = 100;                                                 //Height of detector (cm)
injectSetup.DistanceToSourceCm = 100;                                               //Distance to source (cm)
injectSetup.DwellTimeIsLiveTime = false;                                            //If true, the DwellTimeSec is the live time, otherwise it's the real time
injectSetup.DwellTimeSec = 100;                                                     //Length of measurement
var eCal = new EnergyCalibration()
{
    Order0 = 0,
    Order1 = 3000,
    Order2 = 0,
    Order3 = 0,
    LowEnergy = 0
};
injectSetup.EnergyCalibration = eCal;                                               //Actual energy calibration of the recording
injectSetup.EnergyCalibrationFile = eCal;                                           //Energy calibration of the recording written to file (can simulate uncalibrated detector)
var eRes = new EnergyResolution()                                                   //
{                                                                                   //
    FWHM = 3,                                                                       //
    Offset = 0,                                                                     //
    Power = 0,                                                                      //
    LowEnergySkew = 0,                                                              //
    HighEnergySkew = 0,                                                             //
    SkewPower = 0,                                                                  //
    SkewExtent = 0                                                                  //
};                                                                                  //
injectSetup.EnergyResolution = eRes;                                                //Energy resolution of the recording
injectSetup.IncludePoissonVariations = true;                                        //Flag to simulate poisson variance in each channel (simulate measurement as opposed to perfect source)
var locInfo = new LocationInfo()                                                    //
{                                                                                   //
    Elevation = 0,                                                                  //
    Latitude = 65,                                                                  //
    Longitude = 120,                                                                //
    Overburden = 0                                                                  //
};                                                                                  //
injectSetup.LocationInfo = locInfo;                                                 //Location info for cosmic background (can be used to estimate terrestrial background)
injectSetup.IncludeCosmicBackground = true;                                         //Flag to put cosmic background in spectrum (calculated from location)
injectSetup.IncludeTerrestrialBackground = true;                                    //Flag to put terrestrial background inspectrum
injectSetup.NeuMeasEnv = NeutronMeasurementEnvironment.OUTSIDE_OR_LARGE_BAY;       //
var terrestrialBackground = new TerrestrialBackground()                             //
{                                                                                   //
    Attenuation = 0,                                                                //
    K40 = 0,                                                                        //
    Uranium = 0,                                                                    //
    Th232 = 0,                                                                      //
    LowEnergyContinuum = 0,                                                         //
    HighEnergyContinuum = 0                                                         //
};                                                                                  //
injectSetup.TerrestrialBackground = terrestrialBackground;                          //Terrestrial background contribution
injectSetup.TimeStamp = DateTime.Now;                                               //Time stamp to put on spectrum

injectSetups.Add(injectSetup);
```

Linux API

- Work in Progress
- Current API consists of
  - Shared object files (.so) that can be built for various architectures (which inludes Android systems)
  - Usage example written in C and C header file

```
// this is a c interoperable interface to Gadras' InjectData function
int32_t GenerateInjectData(char *injectFile, float *percentDone, char **spectralFile);

/**
 * @brief A minimal version of inject info, used to generate inject data for a given source
 */
struct DRInjectInfo {
    float distance;                 /**< distance from the detector face to the source */
    float height;                   /**< if you are using a precomputed gam file from an external transport code, and the transport code */
                                    /**< did the calculations for scattering, this parameter will not be used (i.e., Detector.dat scattering parameters */
                                    /**< for a detector using precomputed gam files from external transport codes will be set to zero)height of the */
                                    /**< detector above the ground */
    float measurementTime;          /**< either live or realtime for measurement */
    bool measurementTimeIsLiveTime; /**< true if the measurementtime is the livetime, false if it's the real time */
    char *sourceString;             /**< anything GADRAS can use on it's "Simulated or Calibration Source" line on the plot line */
                                    /**< i.e., either a source (e.g., 137Cs,10uCi) or a precomputed gam file */
    float energyCalibration[CALIBRATION_COEFFS];        /**< array of size 4 containing coefficients for a 3rd order polynomial energy calibration. */
    bool eCalIsFRF;                 /**< the provided ecal is a 3rd order Full Range Fraction polynomial. If false, it is a standard polynomial */
                                    /**< (and the user will most likely need to know the number of channels the detector has) */
    // this will most likely be enabled for gadras version >= 18.6.7
    //    bool useDeviationsPairs;          /**< specify whether to use deviation pairs in Deviation.gadras */
    bool usePoisson;                /**< Apply Poisson statistics for the spectrum */
};


/**
 * @brief This is more automated method for generating spectrum from inject data. this data
 *        simple sources (such as precomputed gam files and isotopes or combinations of isotopes
 * @param  injectInfo a struct of type DRInjectInfo.
 * @param  nchannelsForResponse this variable will be set to the numbers of channels that gadras used to calculate the response
 *                              it will most likely correspond to the number of channels set in Detector.dat
 * @param  energy  float array that contains the energy boundaries, should be of size nchannelsForResponse+1
 * @param  countsResponse  float array that contains the spectrum counts for the generated response, should be of size nchannelsForResponse
 **/
int32_t GenerateSpectrumFromInject(struct DRInjectInfo *injectInfo, int *nchannelsForResponse, float **energy, float **countsResponse);
```

Sandia National Laboratories

HOW TO GET GADRAS

GADRAS-DRF

Sandia National Laboratories

- Public version available on ORNL's RSICC site
- Contains all the features of full GADRAS except:
  - Most commercial detector responses
  - Advanced analyses (Automatic Isotope ID, SNM Analysis, 1D model fitting)
  - Radiation transport (1D or 3D models)
- Can utilize inject tool, API, and existing source files
- Can characterize new detectors

How to get GADRAS

- Free and readily available for DHS/DOE sites and contractors
  - Email: GADRAS-support@sandia.gov for more information
- For other government agencies (e.g. DoD) and their contractors, an email from one of our POCs will suffice for NTK
- For commercial entities and academia, recommend using the GADRAS-DRF version