

PNNL-28251

Python Performance Prediction Package for the PGET Instrument

User Guide

November 2021

Erin Miller
Nikhil S. Deshmukh
Richard S. Wittman
Vladimir V. Mozin

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor Battelle Memorial Institute, nor any of their employees, makes **any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.** Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or Battelle Memorial Institute. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from
the Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062

www.osti.gov
ph: (865) 576-8401
fox: (865) 576-5728
email: reports@osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
or (703) 605-6000
email: info@ntis.gov
Online ordering: <http://www.ntis.gov>

Python Performance Prediction Package for the PGET Instrument

User Guide

November 2021

Erin Miller
Nikhil S. Deshmukh
Richard S. Wittman
Vladimir V. Mozin

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Python Performance Prediction Package for the PGET Instrument

User Guide

Erin Miller, Nikhil Deshmukh, Vladimir Mozin, Rick Wittman

IR#_____

November 21, 2018

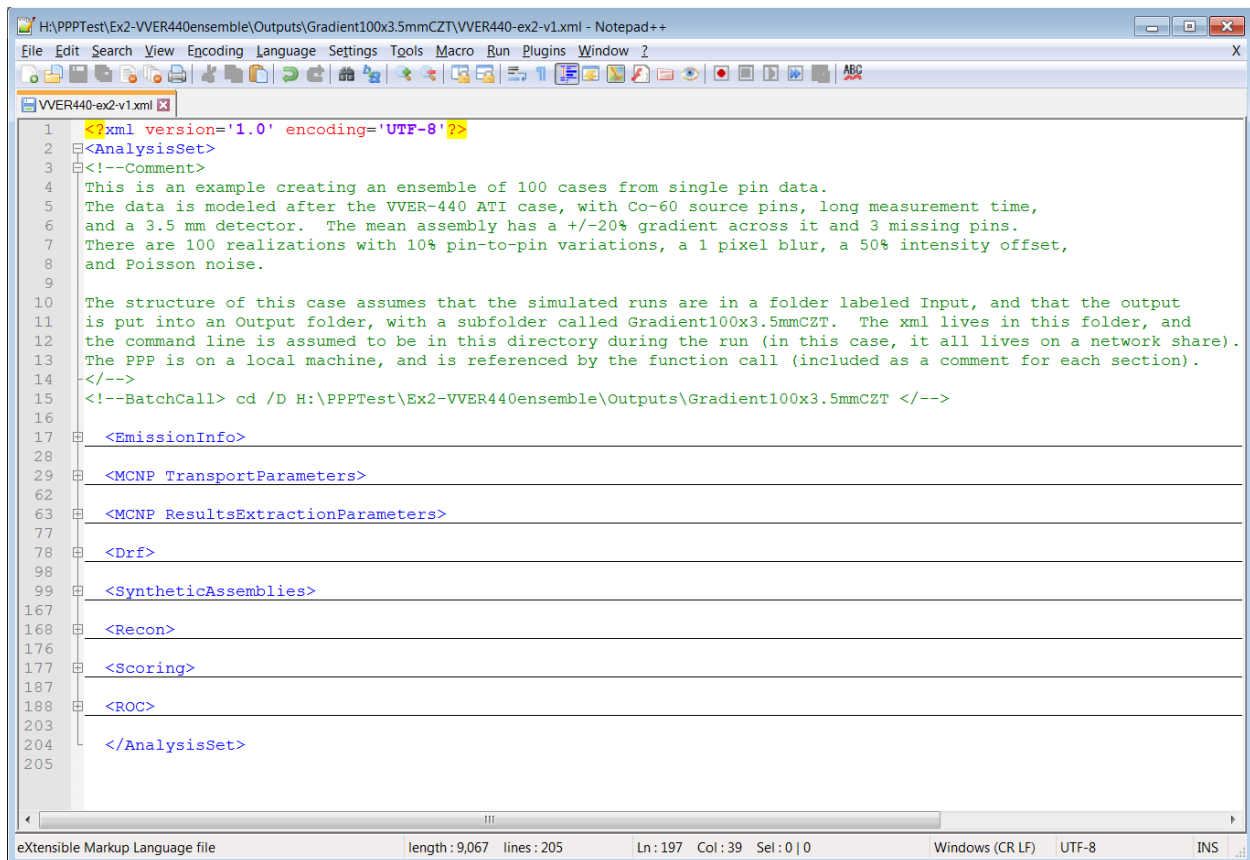
Introduction

The Performance Prediction Package (PPP) is a software suite for generating radiation transport results, or leveraging existing transport results, to produce synthetic emission tomography measurements of spent nuclear fuel assemblies. Such synthetic data can be used to mimic specific measurements, and to systematically investigate detection capabilities. Possible uses for the tool include:

- Instrument design
- Predictive evaluation of observables prior to an inspection visit
- Post-measurement assessment of measured data
- Evaluation of unusual cases – closed containers with anomalous objects
- Sensitivity studies for planning inspections
- Sensitivity studies for diversion scenarios
- Evaluation of analysis methods

A scenario is defined with an assembly type, burnup, and cooling time; as well as assembly rotation angle and position relative to the PGET instrument. Pin-to-pin activity variations or a burnup gradient may be present, but in an inspection scenario these are unlikely to be specified. The measurement will specify a measurement time and a particular detector for detector response, and an optional empirical correction can further adjust the model to better mimic the measurement. The goal is to detect missing pins, or burnup anomalies consistent with a pin substituted with a fresh pin after partial burnup.

The Performance Prediction Package is a software suite that walks through the assembly description, radiation transport modeling, detector response function, and producing synthetic data in absolute units. There are two modes: assembly and ensemble. Assembly mode builds a single synthetic measurement. Ensemble mode separates radiation transport results for each pin, then recombines them to create an arbitrary number of assemblies with statistical variations, suitable for ROC-type analysis. A GUI is included for selecting pin-level properties. The primary functions of the PPP are performed by Python scripts, with parameters passed using xml-formatted files which can be assembled separately or in a single script. Each section of the process can be run independently with a separate xml section and python script, although in several cases parameters may be copied by the user for consistency. Work is documented via the xml parameter file as well as several text files with pin-level properties. An example of the overall structure of the xml parameter file is shown in Figure 1.



```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <AnalysisSet>
3 <!--Comment>
4 This is an example creating an ensemble of 100 cases from single pin data.
5 The data is modeled after the VVER-440 ATI case, with Co-60 source pins, long measurement time,
6 and a 3.5 mm detector. The mean assembly has a +/-20% gradient across it and 3 missing pins.
7 There are 100 realizations with 10% pin-to-pin variations, a 1 pixel blur, a 50% intensity offset,
8 and Poisson noise.
9
10 The structure of this case assumes that the simulated runs are in a folder labeled Input, and that the output
11 is put into an Output folder, with a subfolder called Gradient100x3.5mmCZT. The xml lives in this folder, and
12 the command line is assumed to be in this directory during the run (in this case, it all lives on a network share).
13 The PPP is on a local machine, and is referenced by the function call (included as a comment for each section).
14 <!-->
15 <!--BatchCall> cd /D H:\PPPTTest\Ex2-VVER440ensemble\Outputs\Gradient100x3.5mmCZT <!-->
16
17 <EmissionInfo>
18
19 <MCNP TransportParameters>
20
21 <MCNP ResultsExtractionParameters>
22
23 <Drf>
24
25 <SyntheticAssemblies>
26
27 <Recon>
28
29 <Scoring>
30
31 <ROC>
32
33 </AnalysisSet>
```

extensible Markup Language file | length : 9,067 | lines : 205 | Ln : 197 | Col : 39 | Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

Figure 1. XML file structure used to step through the process of creating and evaluating synthetic PGET assembly data.

We recommend the use of an xml viewer, such as Notepad++ (<https://notepad-plus-plus.org/>), for increased clarity while working through the xml file steps.

Problem Definition

The initial problem definition includes the selection of an assembly type, identification of partial defects or activity variations, and calculating emission intensity.

Pin Picking Interface

A GUI has been developed for defining pin-by-pin properties. This includes both attenuation properties, such as partial defects and pin substitutions, as well as activity (gradients, pin-to-pin noise, etc.). These properties are output into a text file (called the Pins file) which is used for setting up the MCNP case and for normalization and ensemble creation. A screen shot of the GUI is shown in Figure 2.

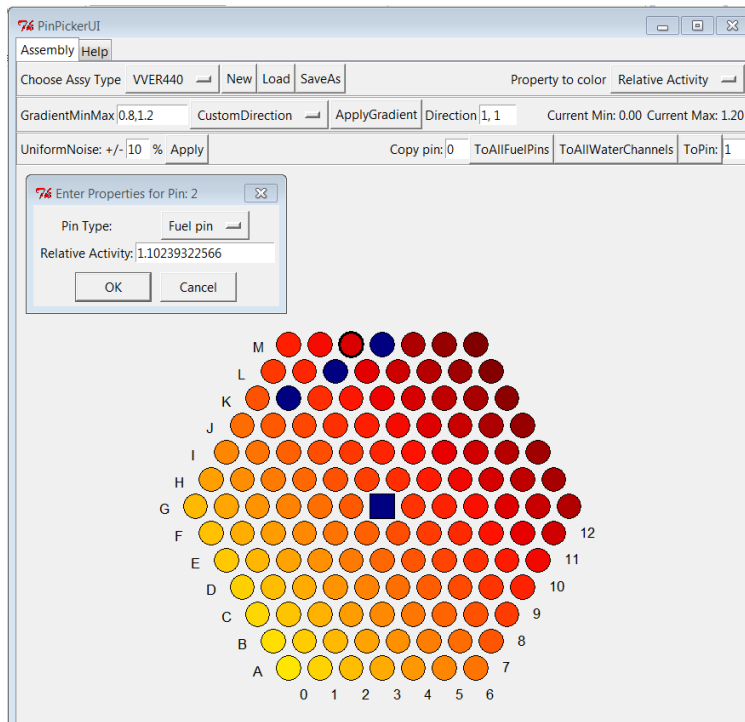


Figure 2. Pin Picker graphical user interface. The user selects the assembly type from a drop-down list, adjusts assembly-wide properties using the gradient and noise options, and then specifies individual pin properties using the pin interface.

The user first selects the assembly type from the drop-down list. Currently three choices are implemented: PWR, VVER-440, and BWR; this choice specifies the pin diameter and assembly geometry. The user can specify values by individual pin, and options are available copy the value on a given pin to all other pins, water channels, or to a specified pin. Once the assembly type has been selected, the user can load a previously-defined pin picker file, or simply begin editing properties.

The first property to edit is the overall activity gradient. The user can select linear or radial; for linear, an arbitrary direction can be defined with a vector entry. The intensity of the gradient is specified by a min/max (for example, 0.8, 1.2 for a +/- 20% relative intensity change).

Once the gradient is specified, the user can apply random noise if desired. This is particularly useful for adding variation to single assembly cases. For ensemble runs, where single pin sinograms will be later combined to create assemblies, the user should generally NOT apply noise here and instead use the realization-to-realization random pin variations in the Synthetic Assembly section.

Individual pin properties can be varied by clicking on the pin of interest. The user can specify an activity directly; typically these are relative activities, with 1 indicating that the pin emission will be set by the values specified during assembly generation. The user can also select pin type from a list of partial defects. Available partial defects include missing pin (water), air-filled cladding, water-filled cladding, water channel, or a steel pin.

The pin values across the assembly can be reset, if needed, by defining the desired properties (such as fuel pin with activity 1) on a single pin, then copying the properties of that pin to all pins in the

assembly. Once the user is satisfied with the definitions, the results can be saved. The resulting text file is shown in Figure 3.

PinIndex	AltCoords	MCNPCoords	Pin Type	Relative Activity
0	M6	1	Fuel pin	1.05358983849
1	M7	2	Fuel pin	1.07799153207
2	M8	3	Fuel pin	1.10239322566
3	M9	4	Missing Pin (water)	0.0
4	M10	5	Fuel pin	1.15119661283
5	M11	6	Fuel pin	1.17559830641
6	M12	7	Fuel pin	1.2
7	L5	8	Fuel pin	1.02025650515
8	L6	9	Fuel pin	1.04465819874
9	L7	10	Missing Pin (water)	0.0
10	L8	11	Fuel pin	1.09346158591
11	L9	12	Fuel pin	1.1178632795
12	L10	13	Fuel pin	1.14226497308
13	L11	14	Fuel pin	1.16666666667
14	L12	15	Fuel pin	1.19106836025
15	K4	16	Fuel pin	0.98692317182
16	K5	17	Missing Pin (water)	0.0
17	K6	18	Fuel pin	1.03572655899
18	K7	19	Fuel pin	1.06012825258
19	K8	20	Fuel pin	1.08452994616
20	K9	21	Fuel pin	1.10893163975
21	K10	22	Fuel pin	1.13333333333
22	K11	23	Fuel pin	1.15773502692
23	K12	24	Fuel pin	1.1821367205
24	J3	25	Fuel pin	0.953589838486

Figure 3. Pin Picker output file.

The pin picker output file is a comma-delimited text file. Several different pin numbering conventions are included, with a number assigned to each lattice position. The pin type is specified in the fourth column; the text labels shown here correspond to text labels used in the MCNP template. Additional defect types, if defined in the MCNP template can be added to the GUI by editing an xml file. The attenuation information is used in the setup of the MCNP deck. The activity information is used in the setup of the MCNP deck for a single assembly ONLY; for an ensemble run the single pins are modeled at equal intensity and the variations are applied during synthetic assembly generation. The values in this file are also used during normalization.

Note that in the case of a partial defect, MCNP will set the emissions to zero, so the user is recommended to set the activity to zero for the pin picker file as well. The user will be prompted to check this upon saving, if any non-zero activity partial defects are found.

Generally the numbers specified in the act file are intended as a variation about the bulk activity specified by burnup and cooling time, so numbers are typically around 1. However, if an absolute variation exists (for example, the mean value of the act pins is 1.2, or even absolute units such as emissions per second), it will be applied to the overall scaling. This allows for the use of alternate units

or absolute units, as long as the overall scale factors during synthetic assembly creation are modified accordingly.

Emission Definitions

Information about absolute emission rates can be specified a number of ways: by burnup and cooling time, by the activity of a particular isotope, or by the emission rate of individual lines. The PPP is intended to support any of these input choices. A specification of burnup (MWd/GTu) and cooling time (years), along with an assembly type, will enable a calculation of isotope disintegrations (Bq/pin-cm) for a set of common isotopes and the intensities of the associated emission lines (photons/pin-cm-s). The calculation can also start at the isotope level, where isotopic activity is specified and line emission rates are calculated. Finally, specification can be done at the level of individual lines, whether physical or not, in which case the script is not necessary. The user can then refer to the line intensities of choice in setting up the MCNP deck and in constructing the synthetic assemblies. An example of the emissions xml script is shown in Figure 4.

Input:

```

7 <EmissionInfo>
8   <AssemblyType>VVER440</AssemblyType>
9   <AssemblyHistory>
10     <AssemblyBU units="GWd"> 35 </AssemblyBU>
11     <AssemblyIE units="U235 wt-%"> 4 </AssemblyIE>
12     <AssemblyCT units="years"> 30 </AssemblyCT>
13   </AssemblyHistory>
14 </EmissionInfo>

```

Output:

```

7 <EmissionInfo>
8   <AssemblyHistory>
9     <AssemblyBU units="GWd"> 35 </AssemblyBU>
10     <AssemblyIE units="U235 wt-%"> 4 </AssemblyIE>
11     <AssemblyCT units="years"> 30 </AssemblyCT>
12   </AssemblyHistory>
13   <IsotopeDisintegrations units="Bq/cm-pin">
14     <Cs134>932493.955715</Cs134>
15     <Cs137>7509397006.75</Cs137>
16     <Eu154>75907725.2163</Eu154>
17     <Pr144>1.87052202504e-05</Pr144>
18   </IsotopeDisintegrations>
19   <EnergyLines units="photons/cm-pin-sec">
20     <E0 energy="0.6047" units="MeV">910297.802087</E0>
21     <E1 energy="0.7958" units="MeV">796906.537072</E1>
22     <E2 energy="0.6617" units="MeV">6390496852.74</E2>
23     <E3 energy="1.2744" units="MeV">26441469.2787</E3>
24     <E4 energy="0.7233" units="MeV">15230277.8029</E4>
25     <E5 energy="1.0048" units="MeV">13670222.2342</E5>
26     <E6 energy="0.8732" units="MeV">9169880.92931</E6>
27     <E7 energy="0.9963" units="MeV">7956192.31082</E7>
28     <E8 energy="2.1857" units="MeV">1.29776818097e-07</E8>
29   </EnergyLines>
30 </EmissionInfo>

```

Figure 4. Emissions xml script input and output. The emissions calculation is the only one that produces a modified xml file. The line intensities are used in the setup of the MCNP run and for normalization in the synthetic assembly construction. The user can choose to start with Assembly History, Isotope Disintegrations, or Energy Lines; the user can also select a subset of the lines to run as long as the normalization is made with the same selected subset.

Currently, the calculation spans a limited range of options. The burnup calculations are limited to BWR at 2% initial enrichment with burnup between 5.23 and 41.2 GWd/MTu, PWR at 4% initial enrichment and burnup between 4.43 and 39.9 GWd/MTu, and VVER-440 at 4% initial enrichment and burnup between 5.39 and 40.2 GWd/MTu. Isotopes are limited to ¹³⁴Cs, ¹³⁷Cs, ¹⁵⁴Eu, and ¹⁴⁴Pr from the burnup

calculation; ^{60}Co is additionally included for calculating line emissions based on isotopes. However, the user also has the option to specify any set of lines, physical or not, directly.

MCNP Interface

The MCNP interface includes a front-end portion, which is used to set up MCNP runs, and a back-end portion which parses the MCNP output scripts and assembles the results into sinograms in the floating-point tiff format. The MCNP calculation rests on a series of templates, which are specific to the assembly being modeled and include the model of the instrument. The parameters for both the front-end and back-end are specified in xml format and can be included as part of an overall script describing a particular case.

The MCNP interface is currently executed by running python scripts directly. The user will need to have Python installed, and the numpy library. The script structure currently assumes that the python scripts and the xml parameter file are in the same directory; MCNP input decks and extracted tiff files will be placed in automatically generated subfolders within the same directory structure.

For the front end, the user specifies the MCNP template to be used. This provides the basic assembly geometry. Options include AngularOrientations to vary the number of angles (over 360 degrees). AssemblyOffset shifts the position of the assembly and can provide a tilt between the pin direction and the rotation axis, but should be used with care to avoid collisions between the fuel and the instrument. The source term is specified in SourceSpecification using the lines and intensities calculated above in the EmissionInfo section. The user selects whether to include scattering (ScatterOn, N or Y) and whether to report results by single pin (SinglePinOn, N or Y) for an ensemble run or an assembly run. The output energies are defined by the EnergyThreshold fields, with the first entry indicating the minimum energy tracked (in MeV), and the region between each subsequent pair of energies defining a separate window for output. An example of the MCNP Transport Parameters section, along with the python script information, is shown in Figure 5.

```

<MCNP_TransportParameters>
  <AssemblyType>VVER440</AssemblyType> <!--Not yet used </-->
  <PinPickerFile> VVER440-3missing-gradient.txt </PinPickerFile><!--Not yet used?</-->
  <TemplateFileName> ../../Template files/PGET-Co60-VVER440-v6.2.template </TemplateFileName>

  <ScatterOn> N </ScatterOn>

  <SinglePinCalculation> y </SinglePinCalculation>

  <AngularOrientations> 360 </AngularOrientations>

  <AssemblyOffset>
    <OffsetX> 0 </OffsetX>
    <OffsetY> 0.0 </OffsetY>
    <OffsetZ> 0.0 </OffsetZ>
    <OffsetAngle> 0.0 </OffsetAngle>
  </AssemblyOffset>

  <EnergyLines units="relative intensity">
    <E0 energy="1.1732" isotope="Co60" units="MeV">9985000.0</E0>
    <E1 energy="1.3325" isotope="Co60" units="MeV">9998000.0</E1>
  </EnergyLines>

  <DetectorEnergyBin units="MeV"> 0.400 </DetectorEnergyBin>
  <DetectorEnergyBin units="MeV"> 0.600 </DetectorEnergyBin>
  <DetectorEnergyBin units="MeV"> 0.700 </DetectorEnergyBin>
  <DetectorEnergyBin units="MeV"> 1.173 </DetectorEnergyBin>
  <DetectorEnergyBin units="MeV"> 1.174 </DetectorEnergyBin>
  <DetectorEnergyBin units="MeV"> 1.332 </DetectorEnergyBin>
  <DetectorEnergyBin units="MeV"> 1.333 </DetectorEnergyBin>
  <DetectorEnergyBin units="MeV"> 1.500 </DetectorEnergyBin>

</MCNP_TransportParameters>

```

Figure 5. xml script describing MCNP transport parameters. This script is called by the ModelingToolMCNP python script, and references an MCNP template.

The MCNP script reads the template and makes modifications to the assembly based on the specified location parameters and on the pin picker file. Any partial defects are automatically assigned zero activity. In the single assembly case, the pin picker file is also used to adjust individual pin intensities; for the pin-by-pin case, this information is used later when the single pins are recombined. The script then writes out a series of MCNP decks, one for each angle.

```

63  <MCNP_ResultsExtractionParameters>
64
65      <Emin> 1.173 </Emin>
66      <Emax> 1.174 </Emax>
67
68      <ExtractFullAssemblyProjections> n </ExtractFullAssemblyProjections>
69      <InterlaceFullAssemblyProjections> n </InterlaceFullAssemblyProjections>
70      <CombineFullAssemblySinogram> n </CombineFullAssemblySinogram>
71
72      <ExtractIndividualPinProjections> y </ExtractIndividualPinProjections>
73      <InterlaceIndividualPinProjections> n </InterlaceIndividualPinProjections>
74      <CombineIndividualPinSinograms> n </CombineIndividualPinSinograms>
75
76  </MCNP_ResultsExtractionParameters>

```

Figure 6. xml script describing MCNP extraction parameters. This script is called by the ExtractionToolMCNP python script, and produces one or more sinograms in tiff format.

Once the MCNP runs are complete, a large amount of output information is available. Information is first extracted for each head of the assembly separately, then interleaved and recorded as a function of angle to form a sinogram. Sinograms are saved as floating-point tiff files. All sinograms are saved in a single directory. For a single assembly, the output will be a separate sinogram for each energy window selected. For a pin-by-pin run, the output will be a series of folders, one for each pin and numbered {Pin1, Pin2,PinN}, that each contain single pin sinograms at each energy window indicated. An example of the output is shown in Figure 7. In addition to the sinograms, an EnergyBinKeyFile will be created in the parent directory, which maps each filename to a particular energy region. This file is used as a reference during detector response calculation, to ensure that calculated flux at each energy is correctly mapped to energy deposition in the detector.

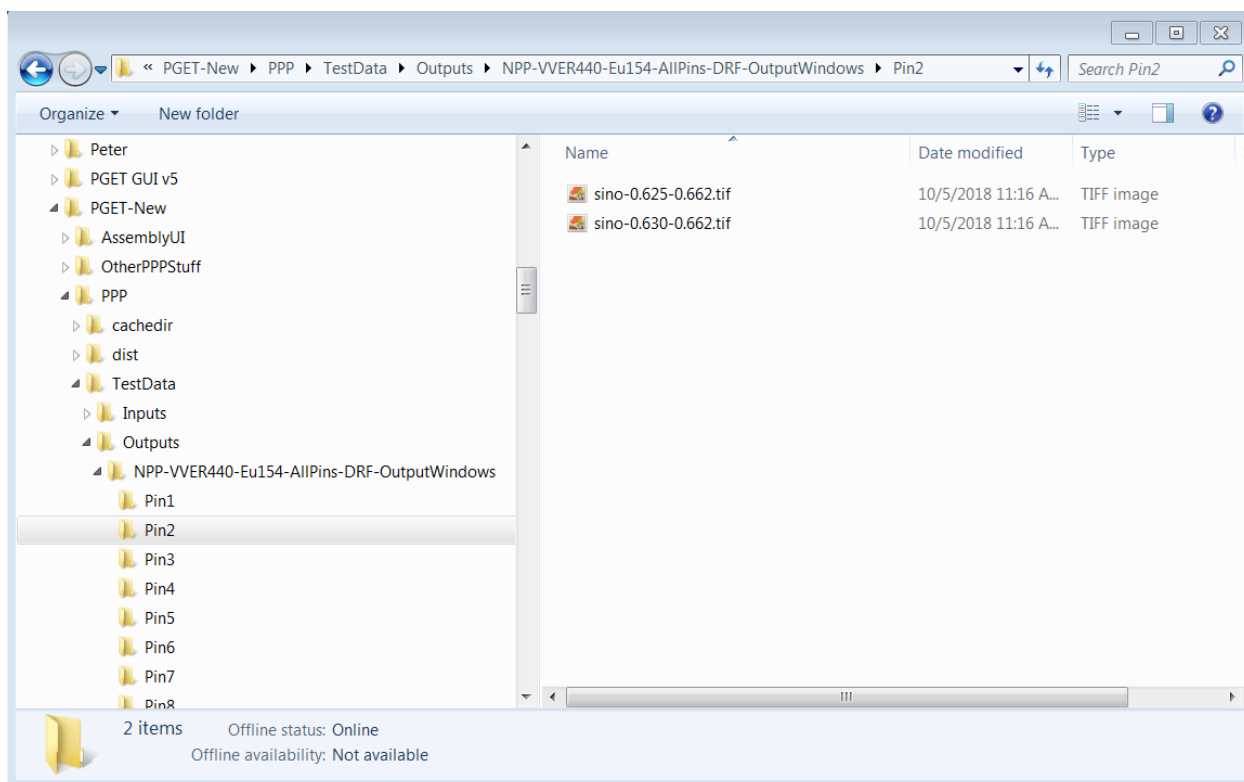


Figure 7. Example of output file structure in single pin mode. The overall directory indicates the case, and contains a separate folder for each pin. Within each folder is a floating-point tiff sinogram corresponding to each energy window in the output (in this case, a window from 625-662 keV and another from 630-662 keV).

Post processing

Post processing applies a detector response function, absolute scaling, and can create ensembles of assemblies for statistical studies. The approach described here is designed to allow the user flexibility in using or repurposing sinograms from radiation transport calculations: a single set of calculations can easily be evaluated using multiple detectors or measurement times. For single pin sinograms, assemblies can be created with arbitrary emission patterns, and ensembles of assemblies generated to allow for statistical exploration. Transport calculations which are completed for single isotopes can be recombined for arbitrary burnup and cooling time.

Tracking the units throughout the calculation is important for obtaining meaningful absolute synthetic data. The sinograms from the transport calculation are in units of flux per unit energy – photons per cm^2 per photon emitted per keV – at the detector face, but contain additional normalization factors that depend on the units used in the radiation transport calculation. For MCNP, the units are typically photons at the location of the tally per photon thrown. This includes the entire active volume during the calculation as well as all lines which are present. The detector response translates this into photons detected per cm^2 per second per keV. The synthetic assembly generator makes the final scaling into absolute units, taking into account the detector face area, the specification of the source in MCNP (number and length of active pins, number and relative intensity of source lines), the overall emission intensity, and the total measurement time. Finally, synthetic assemblies may be reconstructed and scored, and the resulting pin scores compared via ROC analysis.

Detector Response

The detector response takes the calculated scalar flux at the detector face, assumed to be incident perpendicular to the detector, and calculates the energy deposition within the active region. The detector response is a 3000 x 3000 matrix, which maps the incident energy of a photon to the pulse height tally spectrum indicating the amount of energy deposited. Each column in the matrix represents an independent MCNP calculation. Detector response functions are currently available for 5x5x2 mm CZT, 4.6x4.6x2 mm CZT, and 3.5x3.5x1.75 mm CZT, although the modular transport process allows for response calculated from any detector design. The detector response functions are typically stored as text files, with the detector dimensions visible on the first line, and the area facing the detector (limited by the collimator width of 1.5 mm) listed on the second line. Note that the choice of 1 keV bins for the calculation is beyond sufficient to capture the radiation transport and energy deposition properties; the purpose of the high resolution DRF matrix is to enable the user to choose energy windows for radiation transport and for DRF output arbitrarily. An example DRF, in text format and as an image, is shown in Figure 8.

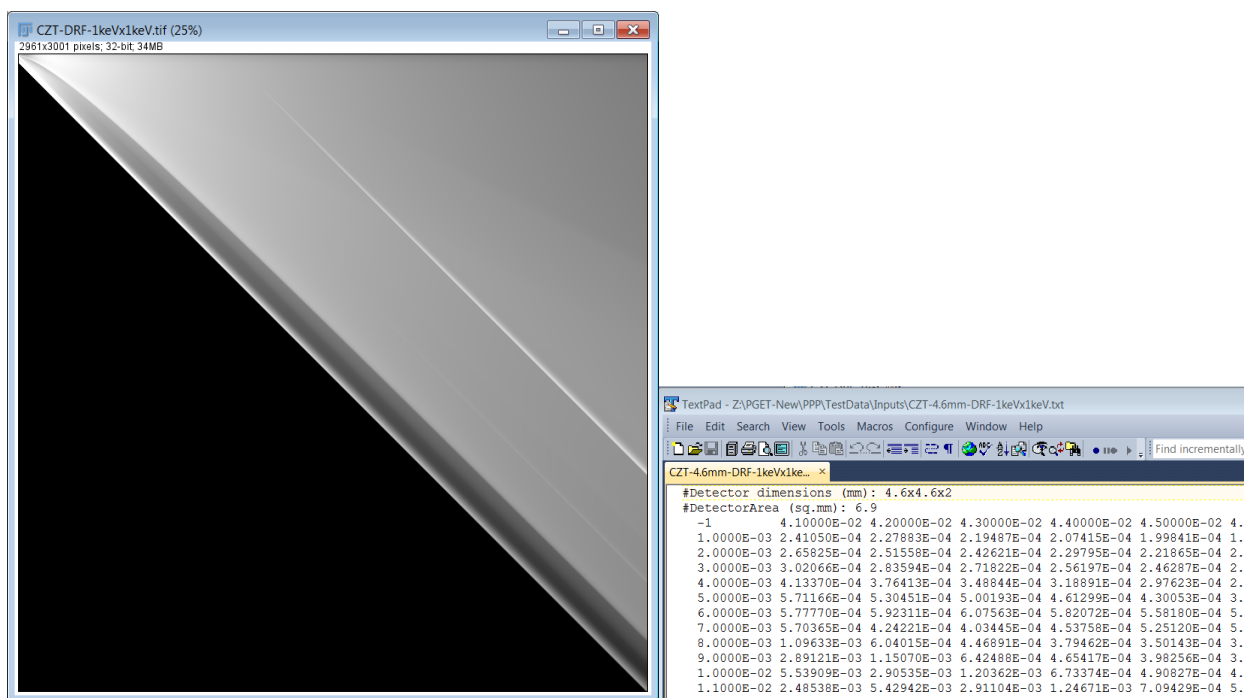


Figure 8. Left: detector response function, shown on a log scale, with each pixel corresponding to a 1 keV wide bin and ranging from 0 to 3 MeV. Data in each column indicates the detector response function for photons at a single energy. Right: screenshot of the text version of the detector response function, showing header information about the detector as well as numerical values.

The detector response section of the xml file includes a reference to the file containing the detector response matrix, a reference to the sinogram set to calculate detector response for, and an output file location. A list of output energy windows is also specified by the user; if this field is not defined, the output sinograms will match the input energy structure. Input energies are typically listed in the file names for user reference, but a text file known as the energy bin key file is referenced for the exact mapping of input flux sinograms to energy region. This also enables the user to choose to calculate detector response sinograms based only on a subset of the available flux sinograms (for example, if flux sinograms are available separately for a single line and for an energy bin containing downscattered radiation, the user could choose to calculate detector response either separately or in combination). An example of the xml section for calculating the detector response is shown in Figure 9.

```

78 <Drf>
79   <!--BatchCall> c:\PGETPPP\PPP\dist\ppp\ppp.exe drf VVER440-ex2-v1.xml </-->
80   <DrfFile> ../../../../CZT-4.6mm-DRF-1keVx1keV.txt </DrfFile>
81
82   <BaseSinogram>
83     <Path> ../../Inputs/PGET-Co60-VVER440-v6.2 </Path>
84   </BaseSinogram>
85
86   <!--> This can be produced alongside MCNP data </-->
87   <EnergyBinKeyFile> ../../Inputs/EnergyBinKeyCo60.txt </EnergyBinKeyFile>
88
89   <OutputEnergyWindows>
90     <OutputEnergyWindow units="MeV"> .400 .600</OutputEnergyWindow>
91     <OutputEnergyWindow units="MeV"> .600 .700</OutputEnergyWindow>
92     <OutputEnergyWindow units="MeV"> .700 1.50</OutputEnergyWindow>
93   </OutputEnergyWindows>
94
95   <BaseOutputFolder> VVER-Co60-Gradient-DRF/ </BaseOutputFolder>
96
97 </Drf>

```

Figure 9. Detector response section of the xml file.

Synthetic Assembly Generation

Synthetic assembly generation includes the steps necessary to turn radiation transport data into collections of synthetic sinograms. Generally speaking, this includes scaling to units of absolute counts, adding empirical corrections and Poisson noise if desired, and in some cases adding information from separate transport calculations (for example, if two isotopes were calculated independently). In the case of a full assembly calculation, this creates a synthetic sinogram (or possibly a set of realizations with different Poisson noise). In the case of a set of single pin calculations, information from single pins is recombined to form one or more assemblies, with intensity distributed according to the pin picker file and random variations if specified. An example of a Synthetic Assembly generation xml input for a single-pin case is shown in Figure 10.

```

<SyntheticAssemblies>
  <!--BatchCall> c:\PGETPPP\PPP\dist\ppp\ppp.exe synthassys VVER440-ex2-v1.xml </-->
  <AssemblyType>VVER440</AssemblyType>
  <SourceType> SinglePinSinograms </SourceType>

  <ScalingData>
    <DetectorArea units="mm^2"> 6.9 </DetectorArea>
    <ModeledPinLength units="cm"> 9 </ModeledPinLength>
    <TimePerProjection units="s"> 540 </TimePerProjection>
  </ScalingData>

  <BaseSinogramList>
    <BaseSinogram>
    </BaseSinogram>
    <BaseSinogram>
    </BaseSinogram>
    <BaseSinogram>
      <Path> VVER-Co60-Gradient-DRF/ </Path>
      <InputSinoFilename> ./sino-0.700-1.500.tif </InputSinoFilename>
      <ActivityFile> VVER440-3missing-gradient.txt </ActivityFile>
      <RelativeScalingFactor units="junk"> 1 </RelativeScalingFactor>
      <EnergyLines units="photons/cm-pin-sec">
        <E0 energy="1.1732" units="MeV">998500.0</E0>
        <E1 energy="1.3325" units="MeV">999800.0</E1>
      </EnergyLines>
    </BaseSinogram>
  </BaseSinogramList>

  <BaseOutputFolder> VVER-Co60-Gradient-EnsemblesV0/ </BaseOutputFolder>
  <OutputSinoFilename> sino-GC.tif </OutputSinoFilename>

  <SinglePinParams>
    <!-- The PercentActivityVariation applies a different pin to pin variation for each assembly.-->
    <!-- The PercentActivityVariation values will be output to RelativeActivity.txt file-->
    <NumActivePinsInMcnp> 123 </NumActivePinsInMcnp>
    <PercentActivityVariation> 10 </PercentActivityVariation>
  </SinglePinParams>

  <EmpiricalAdjustments>
    <GaussianBlurWidth> 1 </GaussianBlurWidth>
    <OffsetCountPercent> 50 </OffsetCountPercent>
  </EmpiricalAdjustments>

  <NoiseType> Poisson </NoiseType>
  <NumVariations> 100 </NumVariations>
  <RandomNumberSeed> 0 </RandomNumberSeed>
  <TestDescription> 3missing0.10gradient </TestDescription>

</SyntheticAssemblies>

```

Figure 10. Example synthetic assembly input for a single-pin case. In this case three base sinograms are referenced, corresponding to three different energy windows. Scaled assemblies, with matched pin-to-pin variations, are produced for each base sinogram and for the sum of all base sinograms.

The xml section for SyntheticAssemblies starts with AssemblyType, which should be consistent with prior declarations (VVER440, PWR, BWR) and SourceType (Assembly, or SinglePinSinograms).

ScalingData applies some basic normalizations: DetectorArea is the exposed area of the detector face (height x the smaller of either the detector crystal width, or the 1.5 mm collimator opening) in mm², which is needed to convert flux from MCNP into counts reaching the detector face. ModeledPinLength is the active pin length considered in the transport model (which should be accessible in the MCNP

template and log file), and the TimePerProjection is in seconds and is set to match the measurement time.

The BaseSinogramList collects one or more radiation transport results to include in synthetic assembly generation. For each BaseSinogram, the overall dataset is indicated in the path; for a full assembly the sinograms will be inside this directory, and for single-pin sinograms the directory will contain a separate folder for every pin. The InputSinoFilename selects the file of interest; with the default naming conventions this refers to a particular energy window at the detector. The EnergyLines provide a baseline for the absolute emissions, in photons/cm/pin/s. The lines included here must match those used in setting up the radiation transport run. The ActivityFile refers to the pin-to-pin activity variations defined in the Pin Picker at the beginning of a problem. The values in the activity indicate the intensity relative to the baseline for absolute emissions, with 1 corresponding to the intensity defined by the EnergyLines (note that the user can elect to set the sum of the EnergyLines to 1 and place absolute values into the activity file, if that better suits the problem at hand). The ActivityFile is used to normalize the overall intensity; in the case of single-pin sinograms it also provides the baseline for pin-to-pin intensities when the pins are recombined into a full assembly. Finally, a RelativeScalingFactor is included at the user's discretion. This can be set to 1 in many cases, but allows for adjustments in overall scaling or to account for non-standard units earlier in the process.

For the example shown, three sets of Base Sinograms are referenced. This allows the user an option to process multiple energy windows simultaneously, or to combine separate transport runs (such as for different isotopes), possibly with the relative ratio varied. Note that if more than one BaseSinogram section is specified, each should cover the same energy range and with the same detector used in order for the summed sinogram to be physically meaningful. For multiple BaseSinograms, each can have their own ActivityFile and RelativeScalingFactor, and the EnergyLines should be appropriate to those used in transport. A scaled sinogram will be constructed both for each BaseSinogram case (useful for processing multiple energy windows simultaneously), and for the sum of all included BaseSinogram cases (useful for combining multiple isotopes within a single energy window).

The BaseOutputFolder specifies the location for synthetic assemblies, and the OutputSinoFilename specifies the filename for the sinograms produced by summing BaseSinograms (the individual sinograms will retain the input label).

Two options are available specifically for synthetic assemblies created from single pin sinograms. For a full assembly run, the values in the Pin Picker file are used to set relative intensities in MCNP, and the reference to this file allows for proper normalization. For single pin sinograms, MCNP is run with equal intensity on all active pins, so the number of active pins must be specified to obtain proper normalization. Second, assemblies from single pin runs can be constructed with varying pin-to-pin intensities. Here, PercentActivityVariation specifies the width of a uniform distribution from which random pin-to-pin activity variations will be selected for each constructed assembly.

The user has the option to include noise due to counting statistics, either Poisson or None. NumVariations sets the number of realizations to create; for a full assembly sinogram these vary only in their Poisson noise realization, but for single pin sinograms this includes additional variation according to PercentActivityVariation. The user can specify a random seed and a label for the test, which is incorporated into the folder name.

Finally, this section includes the option for empirical corrections to better match data. The user can specify a Gaussian Blur Width, in pixels, and an overall offset for the projections, as a fraction of the mean sinogram value.

Reconstruction

This provides a simple filtered back-projection reconstruction, with an optional center offset. Other parameters include the base input folder, the input sinogram name, and the output reconstruction filename. In the case of an ensemble of results, the base input folder can be set to the parent folder containing all the realizations to reconstruct. Reconstructed data is saved as floating-point tiff files. This functionality is included for testing purposes. The synthetic sinograms with absolute units generated by the PPP and should be suitable for reconstruction and scoring in a similar manner to measured data.

Figure 11 shows an example of an intensity distribution selected in the Pin Picker, and the corresponding reconstructed sinogram.

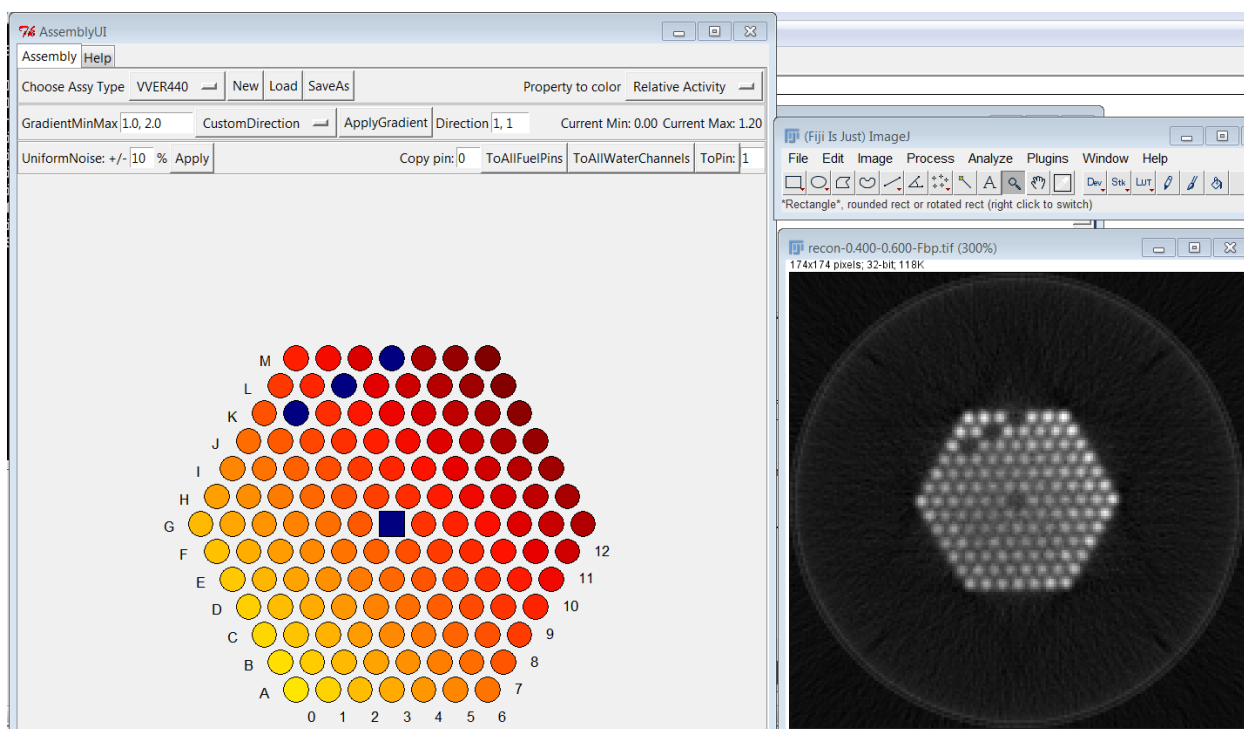


Figure 11. Selected intensity distribution (left) and resulting reconstructed synthetic sinogram (right). This sinogram is for the 400-600 keV energy window, with a Co-60 source and 4.6 mm detector. The realization includes a +/- 20% gradient, 10% random noise variations, Poisson noise, and 3 missing pins.

Pin scoring

This implements a simple region-based pin scoring algorithm. Inputs include the assembly type, which should be consistent with earlier parts of the process, and input and output file structure information. Fields are included for scoring technique and parameters, which serve as placeholders for future algorithm options. The module aligns the expected assembly shape with the assembly, places a grid of expected pin positions, and assigns a score based on the total intensity within each grid position. Pin scores are output into a PinScores.txt file in the directory for each realization, with a pin index and a score; scores for all pins are output into allPinScores.txt in the directory containing all the realizations;

each row corresponds to a pin and each column to a realization. This functionality is included for testing purposes. An example of the allPinScores file is shown in Figure 12.

allPinScores.txt

#Each row has pin scores for one pin

0008.4447	0007.6246	0008.6543	0008.3409	0007.4979	0008.2038	0007.6671	0007.6077
0008.2221	0008.7718	0007.9695	0008.5462	0007.2034	0007.4698	0007.6077	0007.6077
0007.9287	0007.2483	0008.0157	0007.9764	0008.3760	0007.6555	0007.5339	0007.5339
0007.2555	0008.0690	0007.7720	0006.7337	0008.1518	0007.3893	0007.0740	0007.0740
0007.5098	0006.3020	0007.2641	0007.4790	0006.4274	0006.5647	0006.4291	0006.4291
0006.7834	0007.2551	0007.1273	0006.0218	0007.0751	0006.9436	0006.2050	0006.2050
0006.8471	0006.9860	0006.9756	0007.2171	0006.5249	0006.4911	0006.1434	0006.1434
0008.2327	0008.4649	0007.9580	0007.8037	0008.3789	0007.5948	0008.4213	0008.4213
0007.9137	0007.3296	0007.6449	0007.9984	0007.0373	0007.3992	0007.7279	0007.7279
0007.3533	0007.2585	0006.2820	0007.5499	0006.2186	0007.7133	0007.0591	0007.0591
0006.4547	0006.0836	0006.1642	0007.0804	0007.1380	0007.1330	0007.2670	0007.2670
0005.8221	0006.0928	0007.1664	0006.2826	0006.0898	0005.7965	0006.0206	0006.0206
0006.5807	0007.3089	0006.9030	0006.3757	0006.8829	0006.3925	0006.4794	0006.4794
0006.1570	0006.0839	0006.8999	0006.7206	0006.1496	0006.4932	0006.5363	0006.5363
0007.3674	0007.0991	0006.2732	0006.0882	0006.3381	0006.4498	0007.5029	0007.5029
0007.7410	0007.1561	0008.1399	0008.1873	0006.9134	0008.7617	0008.1917	0008.1917
0007.8755	0008.0083	0007.0778	0007.2018	0007.2092	0007.8005	0007.7333	0007.7333
0006.8032	0006.3131	0007.1829	0006.9090	0007.0672	0007.0094	0007.3063	0007.3063
0006.6077	0006.7548	0006.2220	0006.4271	0005.9396	0006.0690	0006.5407	0006.5407
0005.6109	0006.2465	0006.5635	0006.3291	0006.4279	0006.4095	0005.7232	0005.7232
0006.0364	0005.6470	0005.9347	0005.4409	0005.6765	0005.9424	0005.7553	0005.7553

Figure 12. allPinScores.txt file, containing the scores for each pin (rows) and each assembly realization (columns).

ROC analysis

To understand the detection capabilities of a given measurement, the PPP includes functionality to calculate a receiver-operator curve (ROC). Given a distribution of scores from a pin which is absent or anomalous, specified in the first section, and a distribution of scores from a pin which is present and not anomalous, specified in the second section, the ROC calculates the probability of correctly identifying the anomalous pin (probability of detection) and the probability of incorrectly flagging the normal pin (false alarm rate) as the threshold value between the two distributions is varied in a 1000 steps. Figure 13 shows the ROC section of the xml file, the ROC-Data-9.txt file generated, and an example ROC curve. In this case (that shown in Figure 11), the ability to distinguish a missing pin at position 9 compared with a present pin at position 110 is essentially perfect. Note that the comparison can be made between different pins in the same assembly, or between pins from different synthetic datasets with and without anomalous pins.

```

90 <ROC>
91 <!--BatchCall> c:\PGETPPP\PPP\dist\ppp\ppp.exe roc VVER440-ex2-3.5mm-v1.xml </-->
92 <EnsembleScoresTrue>
93 <path> VVER-Co60-Gradient-EnsemblesV1\SynthAssys100-3missing0.10gradient-BUVar10-Poisson/allPinScores.txt </path>
94 <pinNumber> 9 </pinNumber>
95 </EnsembleScoresTrue>
96
97 <EnsembleScoresFalse>
98 <path> VVER-Co60-Gradient-EnsemblesV1\SynthAssys100-3missing0.10gradient-BUVar10-Poisson/allPinScores.txt </path>
99 <pinNumber> 110 </pinNumber>
00 </EnsembleScoresFalse>
01
02 <OutputFolder> VVER-Co60-Gradient-EnsemblesV1\SynthAssys100-3missing0.10gradient-BUVar10-Poisson </OutputFolder>
03 <OutputFilename> ROC-Data-9 </OutputFilename>
04 </ROC>

```

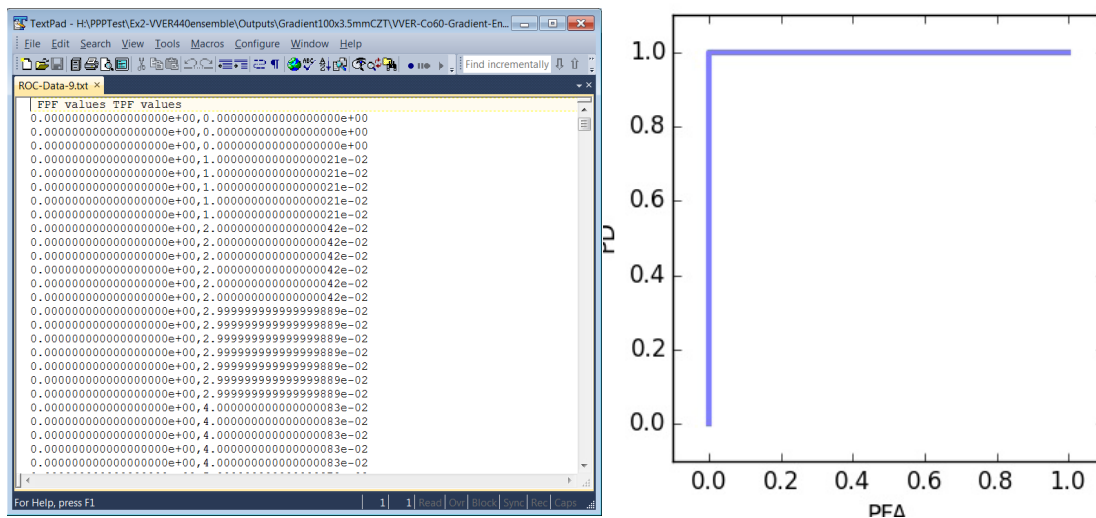


Figure 13. Top: XML file entries specifying anomalous data or pins and normal data or pins. Bottom left: ROC-Data.txt file. Bottom right: ROC curve plotted for Pin 9.

Appendix O. Installation and directory structure

The user has considerable flexibility in how and where to copy/keep the Performance Prediction Package software, as well as in the structure of the data throughout the process. However, a typical case involves large amounts of data and many files, so some care should go into setting up a reasonable directory structure.

We typically set up a single folder on a local drive, containing the Pin Picker, the MCNP setup and extraction code, and the post processing code in three different folders. A separate folder is used to contain the individual PPP run being generated.

For running a case, the Pin Picker is launched from the PinPickerUI-run.bat file inside the PinPickerUI folder. The output text file called the pins file can be saved to a directory specific to the case of interest. The MCNP setup and extraction scripts require Python and the numpy library. As presently set up, the scripts should be placed in the folder for the PPP run, and the MCNP decks and results will be placed into local subfolders. The post-processing functions are governed by an xml file, typically placed in the case directory. The examples below demonstrate how paths are specified within the case directory; it is useful to have the script create separate subdirectories for the DRF and for the synthetic assemblies. Each piece of the processing script is run based on a command line call to the ppp.exe file, along with a keyword which specifies which function to execute and the xml file specifying the parameters. The script can be run from either the directory with the ppp.exe, or the case directory.

An example of a possible directory structure is shown in Figure 14. Here, Ex2-VVER440ensemble is the case folder. This is divided into Inputs (holding MCNP simulation results which are not to be altered during post-processing) and Outputs. In this case, the Outputs directory contains a folder for results with a 3.5mm CZT detector and with a 4.6 mm CZT detector. Within the 3.5mmCZT folder, there is a Pin Picker file describing the intensity distribution, and three different xml files which specify parameters for three different post-processing options (in this case, varying measurement time and pixel-to-pixel

blurring). Each of the Ensembles cases contains a directory for Synthetic Assemblies, which hold the sinograms, reconstructions, and pin scores calculated.

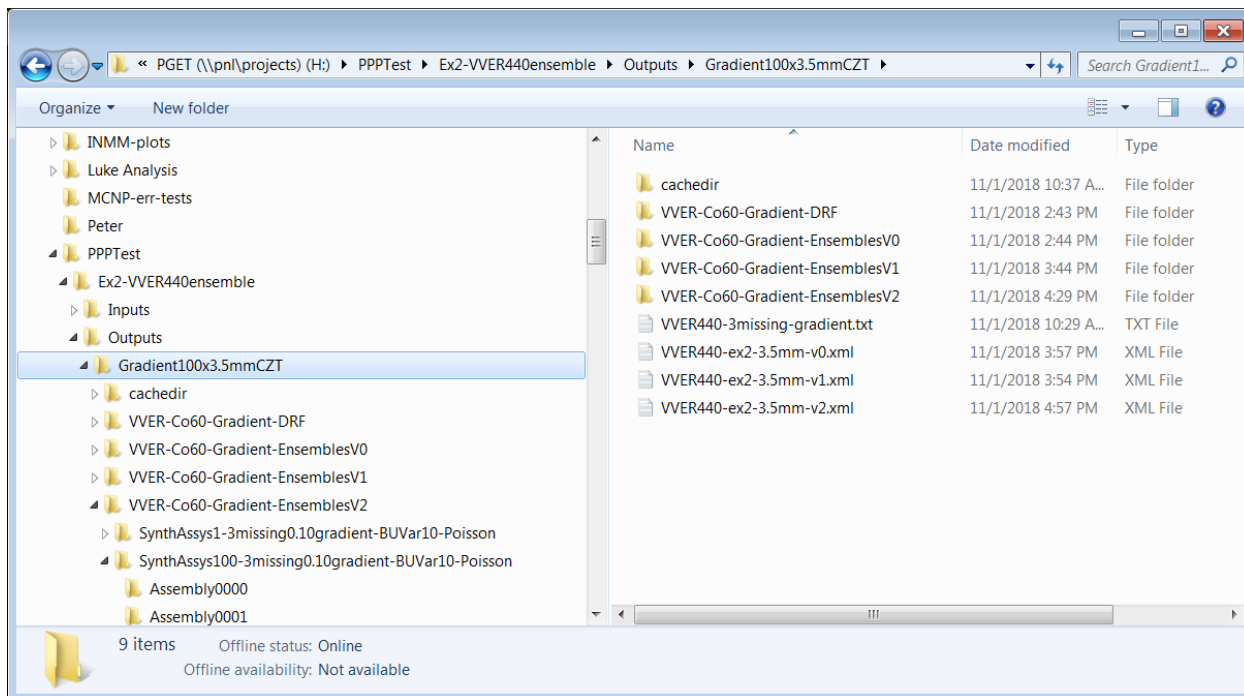


Figure 14. Example of a possible directory structure for results.

Appendix 1. Single Assembly Example

One important use case for the PPP is generation of a single dataset as rapidly as possible to mimic the data which might be seen during an inspection. Here, we set up a single assembly, primary only model of PWR assembly similar to the data collected at Ringhals by the IAEA PGET team. The following XML script describes creating a BWR assembly with 10% intensity variations, measured with a 4.6 mm CZT detector, with 0.8 seconds per projection measurement time. The transport calculation accounts for multiple emission lines, and the resulting data is collected into energy windows of 400-600 keV, 600-700 keV, and 700-1100 keV.

Appendix 2. Ensemble example

Another important case for the PPP is generation of an ensemble of assemblies in order to evaluate performance under a given scenario, or as a function of processing parameters. The following XML script describes creation of synthetic data modeled after the ATI VVER-440 mock assembly. The emitting rods are Co-60, the detectors are 3.5 mm CZT, and a measurement time of 540 seconds per projection is used. Transport calculates detector response on a pin-by-pin basis with three missing pins, tracking primary only radiation for both Co-60 lines. An ensemble of 100 assemblies with a +/-20% intensity gradient, 10% pin-to-pin variations, and Poisson noise, is created. A 1-pixel blur and a 50% intensity offset are included.

Appendix 3. BU variations example

Transport calculations are often the most time-consuming part of performance prediction, so the ability to repurpose existing calculations can increase usability. Here, we use a set of single pin sinograms calculated separately for Cs-137 and Eu-154 to create assemblies at two different burnups and cooling times.

Pacific Northwest National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99354

1-888-375-PNNL (7665)

www.pnnl.gov