ORNL/TM-2004/299

RECENT DEVELOPMENTS IN THE MCNP-POLIMI POSTPROCESSING CODE

Sara A. Pozzi

November 2004

Prepared by
OAK RIDGE NATIONAL LABORATORY
P. O. Box 2008
Oak Ridge, Tennessee 37831
managed by
UT-Battelle, LLC
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

ABSTRACT

The design and analysis of measurements performed with organic scintillators rely on the use of Monte Carlo codes to simulate the interaction of neutrons and photons, originating from fission and other reactions, with the materials present in the system and the radiation detectors. MCNP-PoliMi is a modification of the MCNP-4c code that models the physics of secondary particle emission from fission and other processes realistically. This characteristic allows for the simulation of the higher moments of the distribution of the number of neutrons and photons in a multiplying system. The present report describes the recent additions to the MCNP-PoliMi post-processing code. These include the simulation of detector dead time, multiplicity, and third order statistics.

CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

1. INTRODUCTION

This report describes recent enhancements that have been made to the MCNP-PoliMi Matlab™ post-processing code. The code is used to simulate detection events in liquid or plastic scintillators by post-processing the collision data output from the MCNP-PoliMi simulation. Previous versions of the code were used to simulate time-dependent covariance measurements of the signals from the source and a detector, and from two detectors. The current version of the code includes the simulation of third-order statistics (bicovariance functions) and multiplicity. Detector dead time is also included in the new post-processing code.

The main features of the MCNP-PoliMi code [1] can be summarized as follows. The simulation of spontaneous and induced fission includes the correct multiplicity of prompt neutrons and gamma rays from individual fission events. In addition, the algorithm generating secondary gamma rays from neutron interactions has been changed substantially from the standard MCNP algorithm. Neutron interaction and secondary gamma ray production are now linked to provide a better representation of the physics of individual particle interactions. A number of spontaneous fission and alpha/n neutron and gamma ray sources have been added. The output format of MCNP-PoliMi is in the form of a collision output table, which allows for the simulation of detector response with more flexibility. Further details on the modifications are given in reference 1.

This report is organized as follows. Section 2 describes the MCNP-PoliMi output file and the simulation of organic scintillator detector response. The simulation of the covariance and bicovariance functions is discussed in Section 3, and Section 4 describes the implementation of detector dead time. The simulation of multiplets is discussed in Section 5. A summary of the post-processing options is provided in Section 6, and some conclusions are drawn in Section 7. Appendix A contains a list of the variables used in the program and their meaning, and Appendix B lists the entire program.

2. SIMULATION OF DETECTOR RESPONSE

Organic scintillators are good candidates for the detection of fast neutrons and photons from fission, and have been applied in a number of measurement systems for the detection of nuclear materials [2]. The output format of the MCNP-PoliMi simulations allows for flexibility in simulating the response of this and other detector types [3-4].

An example of the collision output table from MCNP-PoliMi ver. 1.1 is given in Table 1. Each neutron and photon interaction occurring in user-specified cells is listed in the table, and information about each interaction is reported. Interaction type, target nucleus, energy deposited in the collision, time at which the collision occurred, and position of the collision are among the data reported in the output table. The last column of the table gives the energy of the incident particle before the collision. For example, the first row of the table refers to a 1.6 MeV neutron that interacts via elastic scattering with a carbon nucleus, depositing 339 keV in the collision. The interaction occurred 9.4 ns after the originating source event, in cell number 9.

Table 1. Excerpt from MCNP-PoliMi output file.

History number	Particle number	Projectile type*	Interaction type$^{\Delta}$	Target nucleus•	Cell number of collision event	Energy deposited in collision (MeV)	Time (shakes)	Collision position (x, y, z)			Generation number	Number scatterings	Code	Energy (MeV)
...														
151	4	1	-99	6000	9	0.33893	0.944	-16.71	-1.73	3.92	0	0	0	1.61E+00
151	4	1	-99	1001	9	0.71283	1.158	-15.55	-4.66	2.84	0	1	0	1.27E+00
151	20	1	-99	1001	8	0.10313	1.464	13.4	3.95	-4.48	3	0	0	2.61E+00
151	20	1	-99	1001	8	1.20735	1.488	13.91	3.96	-4.51	3	1	0	2.50E+00
151	20	1	-99	1001	8	0.05711	1.493	13.98	3.9	-4.49	3	2	0	1.30E+00
151	20	1	-99	1001	8	0.6134	1.64	15.9	2.76	-4.28	3	3	0	1.24E+00
151	20	1	-99	1001	8	0.42279	1.756	16.49	2.15	-3.34	3	4	0	6.24E-01
151	20	1	-99	1001	8	0.07376	1.974	17.29	2.74	-2.42	3	5	0	2.01E-01
151	20	1	-99	1001	8	0.08552	2.297	17.55	2.64	-0.89	3	7	0	1.14E-01
164	23	1	-99	1001	9	0.22836	4.21	-18.07	-4.43	-3.21	1	3	10	2.84E-01
164	4	2	1	1	8	0.11409	0.193	14.87	4.73	-1.2	0	1	0	6.49E-01

* 1 = neutron; 2 = photon
Δ -99 = elastic scattering; 1 = Compton scattering
• 1001 and 1 = hydrogen; 6000 = carbon

The simulation of the detector pulse requires that the energy deposited by neutrons and photons in the detectors be converted into light output by using measured detector response functions. Photons are detected primarily by Compton scattering and the pulse height to energy deposited response is very close to linear

$$L = E_{\gamma} \text{ ,} \tag{1}$$

where E_{γ} is the energy deposited by the photon (MeV) and L is the measured light output (MeVee).

Neutrons are detected primarily by elastic scattering on hydrogen, and the response is non-linear. The neutron and photon pulse-height to energy response functions have been measured [5] for liquid (BC 501) and plastic (BC 420) scintillators. The measured response functions can be fit to a quadratic function as follows:

$$L = 0.0364 \cdot E_n^2 + 0.125 E_n \text{ ,} \tag{2}$$

for the plastic scintillator, and

$$L = 0.0350 \cdot E_n^2 + 0.1410 E_n, \tag{3}$$

for the liquid scintillator where E_n is the energy deposited by the neutron on hydrogen (MeV) and L is the measured light output (MeVee). The results of Eq. (1) - (3) are shown in Fig.1. Neutron interactions with carbon are assumed to give a small light output, equal to

$$L = 0.02 E_n, \tag{4}$$

where E_n is the energy deposited by the neutron on carbon (MeV) and L is the corresponding light output (MeVee).

The detector pulse is generated by the post-processing code by transforming the energy deposited in the individual scattering events into light output using Eq (1), Eq. (4), and Eq. (2) or (3) depending on the scintillator type. The light outputs that occur in an adjustable time window are then added together and compared to a light output threshold. This time window is referred to as "pulse generation time." A typical setting for the pulse generation time is 10 ns for the scintillators used in the present applications.

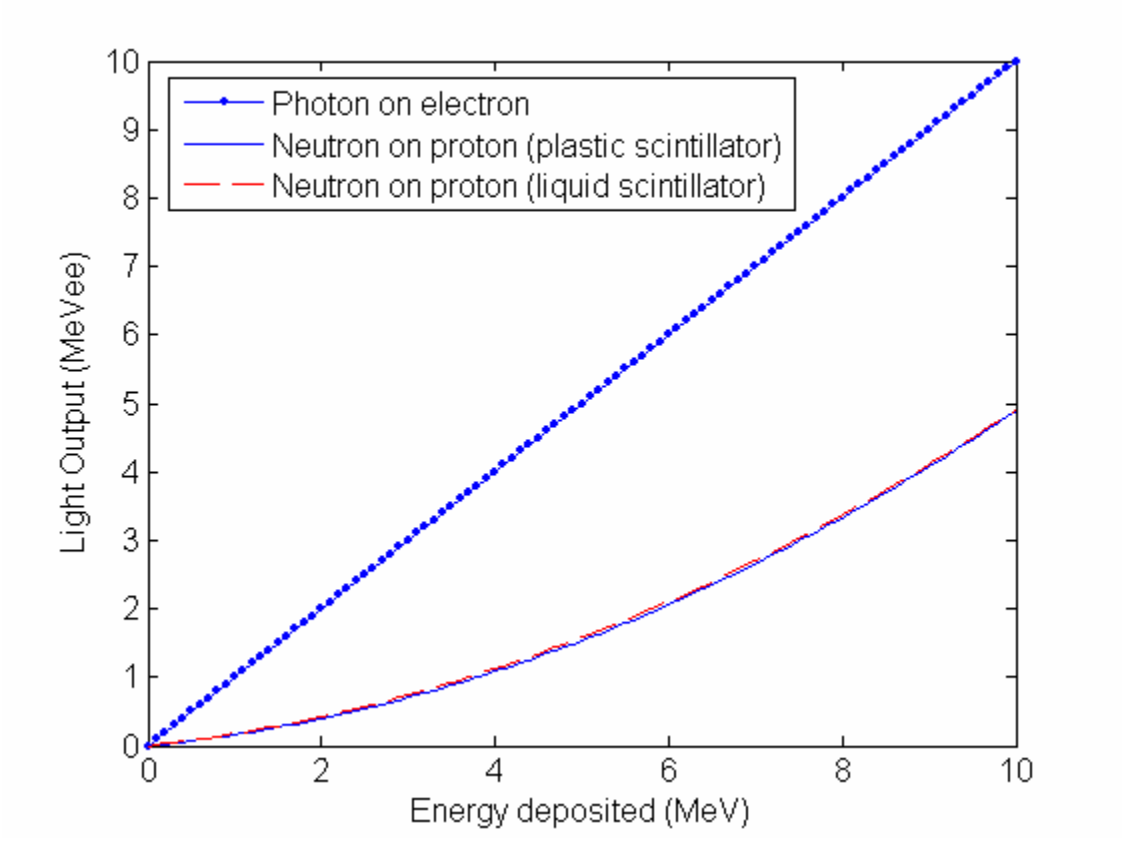


Figure 1. Measured light output functions for plastic (BC 420) and liquid (BC 501) scintillators.

3. SIMULATION OF COVARIANCE AND BICOVARIANCE FUNCTIONS

The post-processing options for the simulation of covariance and bicovariance functions among sequences of events registered by organic scintillators are discussed.

3.1 Simulation of covariance functions

The detector response model discussed in Section 2 is used to simulate the time sequences of neutron and photon detection by the scintillators. The time of detection is determined following a source event, which always occurs at time *t=0*. The distribution of two-way coincidence between pairs of sequences over the time delay between counts is given by

$$R_{xy}(\tau_{xy}) = E\left[x(t_x)y(t_y)\right] - \bar{x}\cdot\bar{y} \quad , \tag{5}$$

where $\tau_{xy} = t_y - t_x$. R_{xy} is known as the covariance between the two sequences of events $x(t)$ and $y(t)$. The product of the means, $\bar{x}\cdot\bar{y}$, yields the rate of accidental coincidence between the two channels. In the simulation, the rate of accidental coincidences is equal to zero because each history evolves independently of all other histories, so the covariance is given, according to Eq. (5), by the expected value of the product of the sequences of events $x(t)$ and $y(t)$. In the simulation, the sequence $x(t)$ (or *start* sequence) can be given by the source events or by a sequence of detector events. These two options correspond to the measurement of the source-detector covariance or the detector-detector covariance. In the post-processing program, one of these two options is evoked by answering the question 'Do you have a start detector?' If the answer to this question is yes, the detector-detector covariances are computed; otherwise the source-detector covariance is computed.

The post-processing code also performs an analysis of the covariance functions according to particle type (neutron or photon) and generation number (source or induced fission particles). This information is present in the output file in column 3 and 12 of Table 1, respectively. The variable names corresponding to these covariance functions are listed in Appendix A.

Figure 2 shows the comparison of an MCNP-PoliMi simulation with a measurement performed with a Cf-252 source and a plastic scintillator, having dimensions 7 by 7 by 10 cm, approximately. In the experiment, the Cf-252 source was placed at a distance of 1 m from the plastic scintillator, and the source-detector covariance was measured. The acquired signature consists of two peaks; the first, sharp peak is given by the gamma rays and the second, broad peak is given by the neutrons.

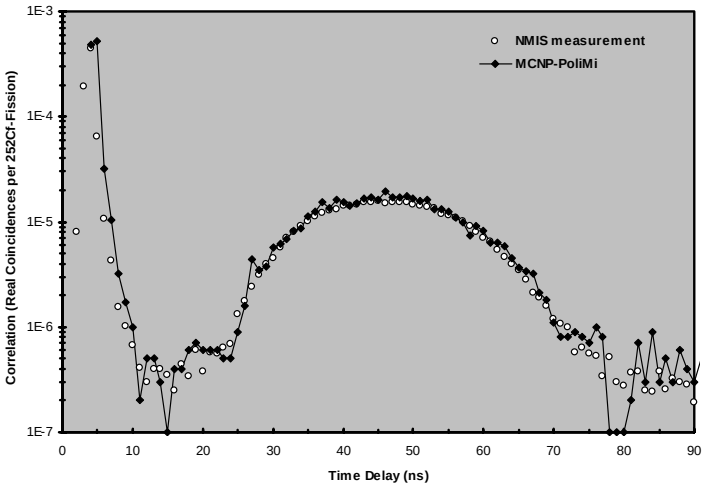


Figure 2. Comparison of Monte Carlo simulation and measurement for a Cf-252 source and detector covariance function.

Inspection of Figure 2 shows that there is very good agreement between the simulation and the measurement.

3.2 Simulation of bi-covariance functions

The extension of the covariance function to three sequences of events yields the bicovariance function, or three-way coincidence among three sequences of events:

$$R_{xyz}(\tau_{xy},\tau_{xz}) = E[x(t_x)y(t_y)z(t_z)] - \bar{x}\cdot R_{yz}(\tau_{xz}-\tau_{xy}) - \bar{y}\cdot R_{xz}(\tau_{xz}) - \bar{z}\cdot R_{xy}(\tau_{xy}) - \bar{x}\cdot\bar{y}\cdot\bar{z}\ . \qquad (6)$$

This distribution is a function of two delays, $\tau_{xy} = t_y - t_x$ and $\tau_{xz} = t_z - t_x$. In our application, the bicovariance function can be computed among the sequences of events in a source and two detectors, or among three detectors. In the first case, a contribution to the bi-covariance is made when detections occur in the two detectors as a result of the same source event. In the second case, a contribution is made when all three detectors register a count. The current version of the post-processing code can calculate the bicovariance functions in both these cases.

Figure 3 shows the sketch of an experiment performed on a plutonium metal shell having mass 4.0 kg, outer radius 6.0 cm and inner radius 5.35 cm. Further details on the experiments and their simulation are given elsewhere [6-7].

Figure 4 shows the comparison of the measured source-detectors bi-covariance for a plutonium metal shell (98 wt% Pu-239). Comparison of Figure 4 (a) and (b) shows that the four regions of the bi-covariance are well represented in the simulated signature. Inspection of Figure 4 (a) reveals that there was a 1 ns time offset in the timing alignment of the two detectors used in the measurement.

Contributions to the bi-covariance can be due to neutrons or photons and to particles from the source or particles from induced fission in the fissile material. The information on what type of particle was detected is present in the collision output file. This capability is illustrated in Figures 5 and 6. Figure 5 shows the result of Figure 4 (b) separated into four components according to the particle type that was detected. Experimentally, a split of the bi-covariance according to particle types can be achieved by using liquid scintillators, which have pulse shape discrimination properties.

Figure 6 shows a split of Figure 4 (b) performed according to generation of the particles pairs detected by the scintillators. In our nomenclature, generation zero particles are source particles than reach the detector; later generation particles originate from fission induced in the fissile material.

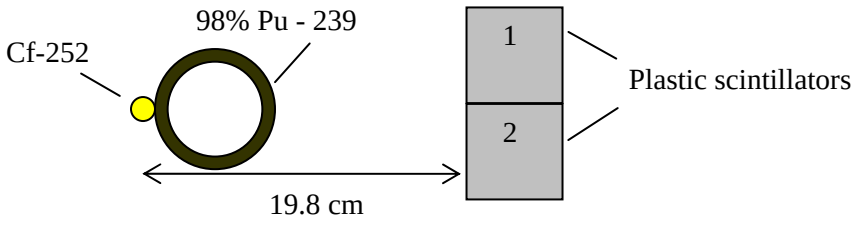
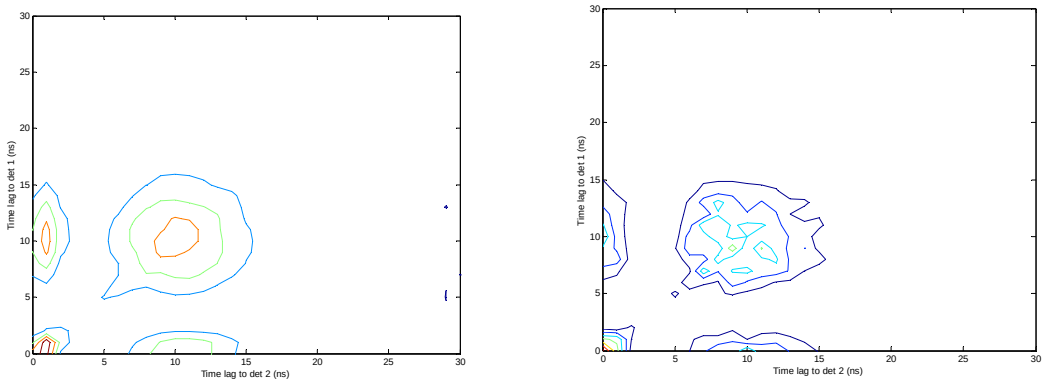


Figure 3. Sketch of the configuration for the simulation of the source-detectors bicovariance function with plutonium sample (not to scale).



(a) (b)

Figure 4. Source-detectors bi-covariance with plutonium sample: (a) Experimental result and (b) MCNP-PoliMi simulation.

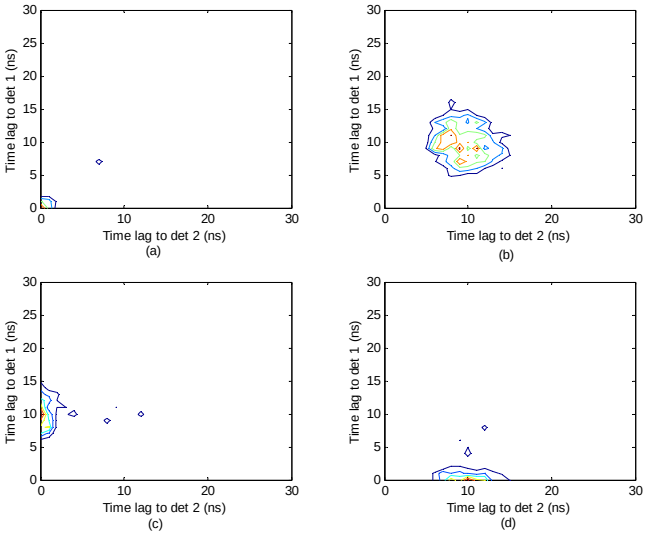


Figure 5. Simulation of source-detectors bi-covariance with plutonium sample: (a) photon – photon pairs, (b) neutron – neutron pairs, (c) neutron – photon pairs, and (d) photon – neutron pairs.

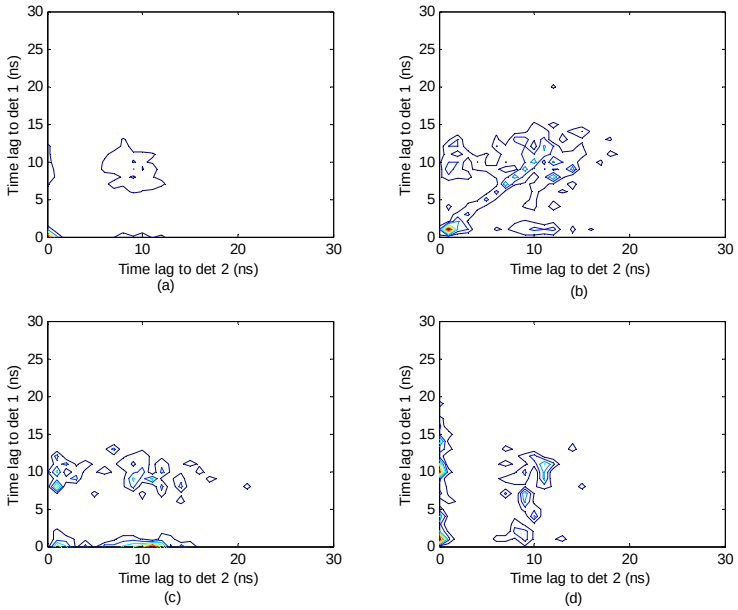


Figure 6. Simulation of source-detectors bi-covariance with plutonium sample: (a) generation zero – generation zero pairs, (b) induced fission – induced fission pairs, (c) generation zero – induced fission pairs, and (d) induced fission – generation zero pairs.

4. DETECTOR DEAD TIME

Previous versions of the post-processing code allowed only one detection per detector per history. The current version allows for up to three detections per detector per history, on average[1]. A parameter called "detector dead time" has been included to determine if successive pulses, pertaining to a given history and exceeding the detector threshold, are accepted in the detection sequence registered by the detector. In the experiment, this parameter corresponds to the constant fraction discriminator output pulse time-width. Typically, in our experiments this value lies between 50 and 100 ns.

Figure 7 shows the effect of varying the detector dead time in the simulation of an active measurement on a 3.3 kg plutonium metal shell (98 wt% Pu-239). The curve with the dots (top curve in Fig. 7) shows the result for a dead time of 0 ns: i.e. all pulses exceeding the threshold are accepted. The bottom curves show the result for a dead time of 5, 20, and 100 ns. There is very little difference in the bottom curves, showing that in this example the occurrence of successive pulses in the detector is rare after a few ns from the first pulse.

[1] This parameter can be easily increased as necessary.

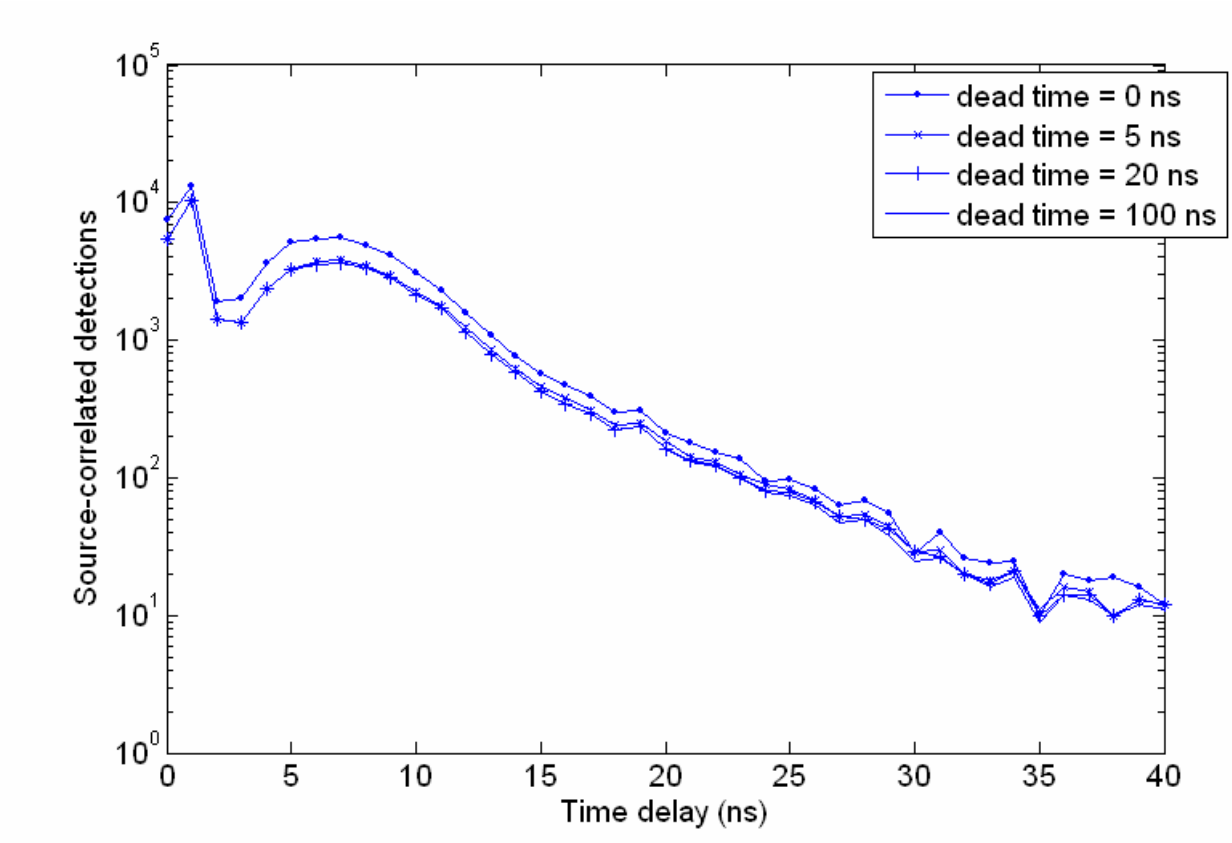


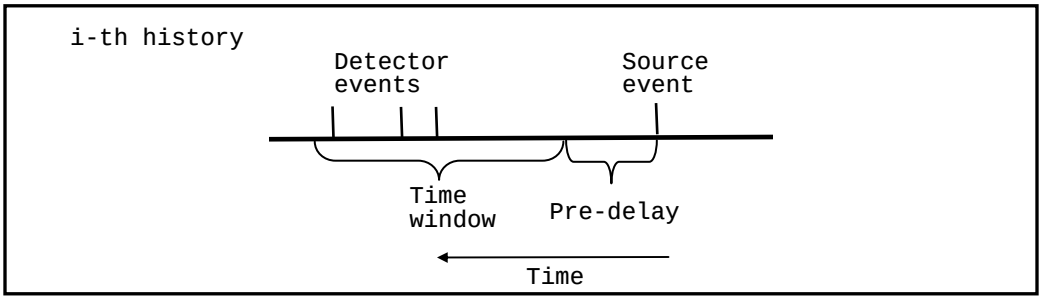
Figure 7. Source-detectors correlation function for 3.3 kg Pu metal sample and varying detector dead times.

5. SIMULATION OF DETECTOR MULTIPLETS

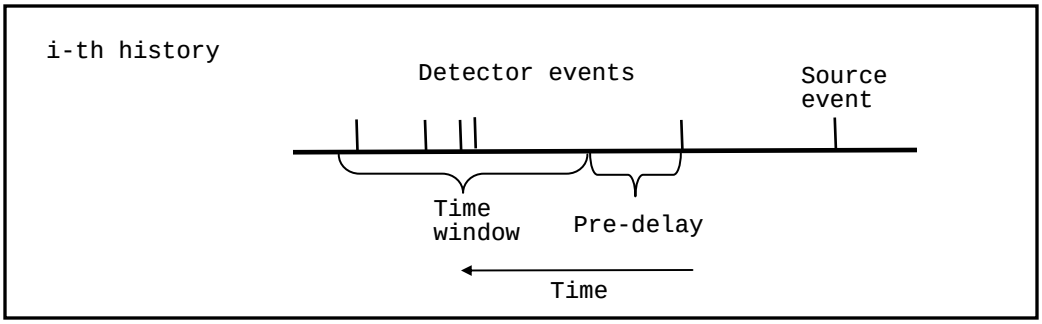
The code simulates the occurrence of multiple pulses in N detectors within a given time window. The time window can be started by a source event or by a detector event. The time window can be preceded by a pre-delay, time during which the multiplicity counting is dead.

The simulation does not take into account accidentals because each history is initiated by a single source event and evolves independently of any other history, until all the particles pertaining to it are tracked. Figure 8 illustrates the simulation of a source-triggered multiplet (Fig. 8a) and a detector-triggered multiplet (Fig. 8b) registered by N detectors following a source event. The detection events contribute to the multiplet if they occur in the time window, which begins after the pre-delay. The detections that contribute to the multiplets

can be given by neutrons or photons[2]. The post-processing code distinguishes between the two particle types.



(a)



(b)

Figure 8. Illustration of pulse train for multiplet simulation. (a) source-triggered multiplets (b) detector triggered multiplets.

Figure 9 shows the distribution of multiplets registered by two detectors in the active simulation of a 3.3 kg plutonium shell. The time window was set at 100 ns, and the results are shown for two detector dead times.

[2] In some cases, a pulse will result from the combined energy deposition of neutron(s) and photon(s). The occurrence of this event is rare when compared to the occurrence of a pulse generated by only one neutron or photon, interacting multiple times in the detector. In the current classification scheme a pulse is classified on the basis of the particle that contributes the first interaction to the pulse.

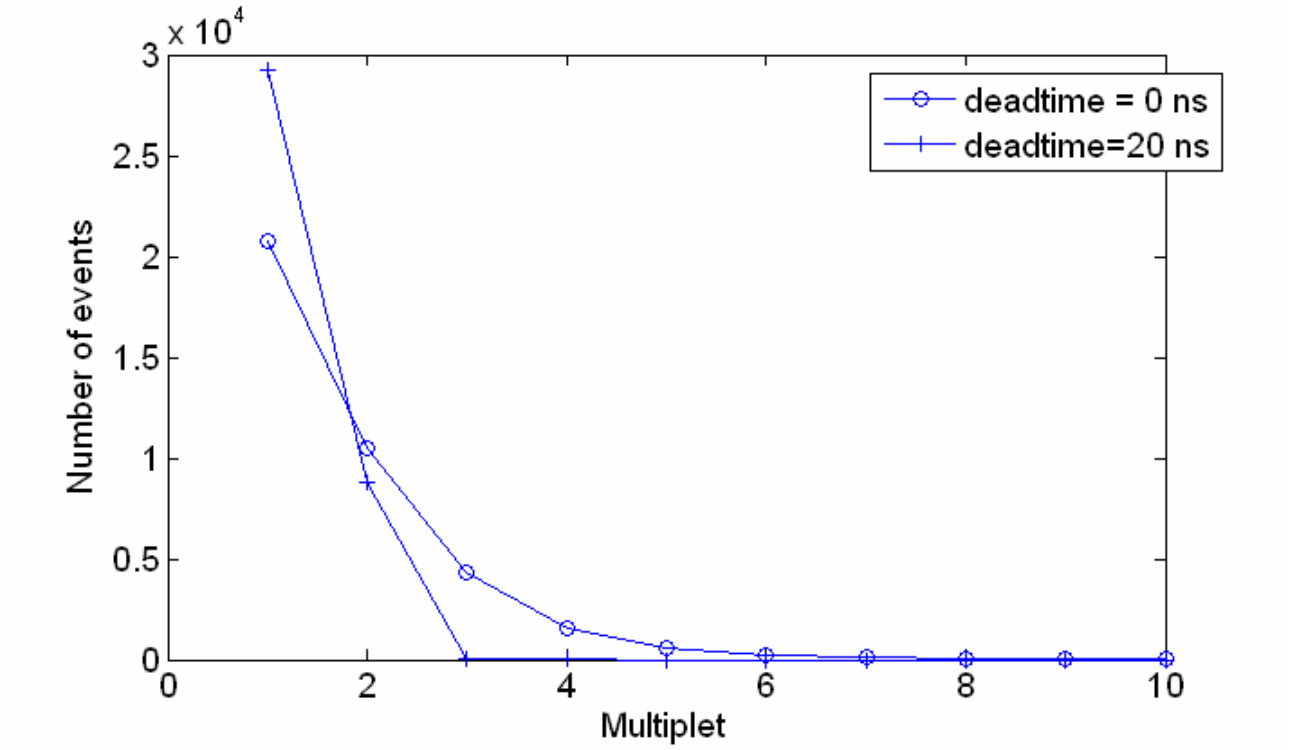


Figure 9. Source-triggered multiplets for 3.3 kg Pu metal sample and varying detector dead times. The time window was set to 100 ns.

6. SUMMARY OF POST-PROCESSING OPTIONS

Table 2 shows the adjustable parameters in the post-processing code, and indicates where these parameters are set, whether in the post-processing code itself or in the Matlab™ window. Care should be taken in setting the detector threshold, which can be determined for a given experiment by performing a simple calibration with a gamma source.

Table 2. Main input parameters in post-processing code.

Adjustable parameters	Possible values	Setting
Light output curve	1 plastic 2 liquid	In code
Pulse threshold	Threshold in MeVee	In code
Pulse generation time	Time in shakes (1 shake = 10 ns)	In code
Detector dead time	Time in shakes (1 shake = 10 ns)	In code
Detector numbers	MCNP-PoliMi cell numbers of detectors in input file	In Matlab™ window
Time for correlation window	Time in ns	In Matlab™ window
Higher order statistics (HOS) option	1 yes 0 no	In Matlab™ window
Multiplicity option	1 source triggered 2 detector triggered	In Matlab™ window

7. CONCLUSIONS

This report described the recent additions to the post-processing code used to analyze the MCNP-PoliMi data output file, and serves as a user's manual for the code itself. The new features include the addition of detector dead time and multiplet and bicovariance simulation. A list of the variables used in the post-processor is given in Appendix A. A complete listing of the code itself is given in Appendix B.

REFERENCES

1. S.A. Pozzi, E. Padovani, and M. Marseguerra, "MCNP-PoliMi: A Monte Carlo Code for Correlation Measurements," Nuclear Instruments and Methods in Physics Research A, 513/3 pp. 550-558, 2003.

2. J. T. Mihalczo, J. A. Mullens, J. K. Mattingly and T. E. Valentine, "Physical description of nuclear materials identification system (NMIS) signatures," Nuclear Instruments and Methods in Physics Research A, 450/2-3, pp. 531-555, 2000.

3. S.A. Pozzi, J.S. Neal, R.B. Oberer, and J.T. Mihalczo, “Monte Carlo Analysis of Neutron Detection with a BaF2 Scintillation Detector,” IEEE Transactions on Nuclear Science, volume 51, number 3, June 2004.

4. S.A. Pozzi, R.B. Oberer, and J.S. Neal, “Analysis of the Response of Capture-Gated Organic Scintillators,” 2004 IEEE Nuclear Science Symposium (NSS), October 16-22, 2004, Rome, Italy.

5. S.A. Pozzi, J.A. Mullens, and J.T. Mihalczo, “Analysis of Neutron and Photon Detection Position for the Calibration of Plastic (BC-420) and Liquid (BC-501) Scintillators,” Nuclear Instruments and Methods in Physics Research Section A, 524/1-3 pp. 92-101, 2004.

6. J. K. Mattingly, J. T. Mihalczo, L. G. Chiang, and J. S. Neal, “Preliminary Analysis of Joint RFNC-VNIIEF/ORNL Measurements Performed In Year 2000,” Y/LB-16,097, Y-12 National Security Complex, September 2001.

7. S. A. Pozzi, J. K. Mattingly, J. T. Mihalczo, and E. Padovani, “Validation of the MCNP-PoliMi Code for the Simulation of Nuclear Safeguards Experiments on Uranium and Plutonium Metal,” Nuclear Mathematical and Computational Sciences: A Century in Review, a Century Anew, Gatlinburg, Tennessee, April 6-11, 2003, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2003).

APPENDIX A - VARIABLES

Table I. Postprocessor variables and their meaning.

Variable name	Variable meaning
'codes'	Code of reaction generating particle
'deadtime'	Detector dead time
'gennums'	Particle generation number
'idet'	Cell numbers of detectors
'idn'	Number of detectors
'ilo'	Light output choice
'lonc'	Light output neutron on carbon
'lonh'	Light output neutron on hydrogen
'lop'	Light output photon
'multopt'	Multiplicity option
'n'	Covariance function (one column per detector)
'nc'	Covariance function cross talk events
'ncal'	Neutron light output calibration
'nch'	Number of collisions in history
'ng'	Covariance function induced fission events
'nn'	Covariance function neutron events

'nnc'	Covariance function no cross talk events
'np'	Covariance function neutron-photon events
'npulses'	Number of detections per detector per history
'ns'	Covariance function source events
'parnums'	Particle number
'partyps'	Particle type
'pcal'	Photon calibration
'pgt'	Pulse generation time
'pn'	Covariance function photon-neutron events
'pp'	Covariance function photon-photon events
'ptimes'	Sequence of detection times
'sdis'	Number of source events
'thresh'	Light output threshold
'tim'	Time binning
'tmax'	Covariance max time
'tmin'	Covariance min time
'zzz'	PoliMi data output

APPENDIX B – MATLAB™ CODE

The following is a listing of Matlab program **postmain.m,** and functions **pulsemultip.m** and **binit.m.**

postmain.m

```
% Loads and analyzes MCNP-POLIMI output file

% 16 columns of ascii output file have the following information:
% # of Start Event, Part #, Part Type, Reaction type (Ntyn), ZAID collision nucleus,
detector cell #,
% energy rel (MeV), time (shakes), x, y, z, wgt, generation #, # scatterings, mtp or
code, energy of
% particle prior to present collision

% note: in the variable names p=photon; n=neutron
% For further information contact Sara Pozzi (pozzisa@ornl.gov)

global pgt lonc ncal pcal thresh deadtime zzz index nch lindex ilo

%       ******************   ADJUSTABLE PARAMETERS   ******************
%ilo=input('Enter your choice for light output: (1) constant (2) lin fit ');
ilo=2;              % recommended value: ilo=2 for plastic scint. and ilo=3 for liquid
scint.
isep=0;             % if separate detector responses isep=1
pgt=0.5;            % pulse generation time in shakes (1 shake = 10 ns)
lonh=0.16;          % light output for n on hydrogen (not used if ilo=2)
lonc=0.02;          % light output for n on carbon
lop=1;              % light output for gamma on electron
ncal=[lonh  0];     % parameters for line in calib: neutrons
pcal=[lop   0];     % parameters for line in calib: photons
if ilo==2
    ncal=[0.0364   0.125 0];  % parameters for line in calib: neutrons
end
if ilo==3
    ncal=[0.035 0.141 0];     % parameters for line in calib: neutrons
end
thresh=0.187;     % selectable threshold for pulse acceptance MeVee units if lop=1
deadtime=50;       % detector dead time in shakes (1 shake = 10 ns)
% note: detector dead time is measured starting from the first
% interaction in the pulse

% multiplicity options

timewindow=51.2;  % time window for multiplicity (shakes; 1 shake = 10 ns)
predelay=0;  % predelay for detection triggered multiplicity


%       ****************************************************************

filen=input('Please enter the file name ','s');  %file name
ign=input('Generation analysis ? (0) no (1) yes  ');
isc=input('Separate neutron and photon contributions ? (0) no (1) yes ');
idn=input('Enter number of detectors  ');
istar=0;
if idn>1
    istar=input('Do you have a START detector ? (0) no (1) yes ');
end
istop=idn-istar;
if istop<1
    error('Must have at least one STOP detector')
end
idet=zeros(1,idn);
if istar==1
    disp('   Insert the cell numbers for detectors.')
    disp('   The START detector must be the first one entered.')
elseif istar==0
    disp('   Insert the cell numbers for STOP detectors. ')
else
    error('Invalid istar entry')
end

for j=1:idn
    cell=input(['Enter cell number for detector ',num2str(j),'  ']);
```

```
    idet(j)=cell;                % array with detector cell numbers
end
isep=0;
if (idn>2 | (idn==2 & istar==0))
    isep=input('Do you wish to treat STOP detectors separately? (0) no (1) yes ');
end
ict=0;
if (idn>=2 & istar==1)
    ict=input('Cross talk analysis ? (0) no (1) yes ');
end
ihos=0;
if (idn==2 & istar==0 | idn==3 & istar==1)
    ihos=input('Higher order statistics analysis ? (0) no (1) yes ');
end
tmin=input('Enter time frame for correlation: minimum time lag (ns) ');
tmax=input('Enter time frame for correlation: maximum time lag (ns) ');
sdis=input('Enter number of source disintegrations ? ');
if sdis==0
    error('Number of source disintegrations must be > 0')
end

% multiplicity analysis/options
imt=input('Multiplicity analysis ? (0) no (1) yes ');
multopt=0;
if imt==1
    disp('Select multiplicity options')
    multopt=input('(1) source triggered (2) detector triggered (3) help ');
    if multopt==3
        disp('source triggered = multiplicity window triggered by external source event')
        disp('detector triggered = multiplicity window triggered by the first detection
in the history')
        disp('In all cases, so far no accidental coincidences are modeled')
        disp('Select multiplicity options')
        multopt=input('(1) source triggered (2) detector triggered ');
    end
end

% begin postprocessing
format compact
zzz=load(filen);  % load data file
zzz=zzz(:,1:15);
[nrow,ncol]=size(zzz);
disp(' ')
disp([' ************************************** '])
disp(' ')
disp([' Number of rows in file = ',int2str(nrow)])
inde1=find(diff(zzz(:,1))>0)+1;   % index where the source history changes
index1=[1 inde1'];                % add index of first history
lindex1=length(index1);
index1=[index1 nrow+1];
disp(' ')
disp([' Number of source histories = ',int2str(lindex1)])
disp(' ')
disp([' Threshold (MeVee) = ', num2str(thresh)])
disp(' ')
disp([' Deadtime (ns) = ', num2str(deadtime*10)])



index=zeros(1,lindex1); % index of hist. involving both START and at least one STOP det
if istar=1
% else, index of histories involving at least one STOP detector
nch=zeros(1,lindex1);    % number of collisions in histories beginning in index
kk=0;
if istar==1  % there is a START detector in cell idet(1) STOP dets in cells idet(2:idn)
    for i=1:lindex1
        if(any(zzz(index1(i):index1(i+1)-1,6)==idet(1)))
            for j=2:idn % check if history involves both the START and at least one STOP
det
                if(any(zzz(index1(i):index1(i+1)-1,6)==idet(j)))
                    kk=kk+1;
                    index(kk)=index1(i);               % if so, index of history in
'index'
                    nch(kk)=index1(i+1)-index1(i);      % number of collisions in history
                    break
                end
            end
        end
    end
elseif istar==0   % only STOP detectors in cells idet(1:idn)
```

```
    for i=1:lindex1
        for j=1:idn % check if history involves at least one STOP det
            if(any(zzz(index1(i):index1(i+1)-1,6)==idet(j)))
                kk=kk+1;
                index(kk)=index1(i);               % if so, index of history in 'index'
                nch(kk)=index1(i+1)-index1(i);      % number of collisions in history
                break
            end
        end
    end
else
    error('Invalid istar entry')
end

index=index(1:kk);
nch=nch(1:kk);
lindex=length(index);

% find sequence of accepted pulse times, particle types, generation numbers, particle
numbers,
% codes, and number of pulses per history for each detector -
% it is assumed that, on average, there will be at most
% 3 detections per detector per source history.  So far this assumption has proven to be
valid
% If the actual detections were to exceed this number, an error message is
% given by 'pulsemultip' and new definitions must be given here and in
% 'pulsemultip'

ptimes=zeros(lindex*3,idn);
partyps=zeros(lindex*3,idn);
gennums=zeros(lindex*3,idn);
parnums=zeros(lindex*3,idn);
codes=-999*ones(lindex*3,1);
npulses=zeros(lindex,idn); % records number of detections per detector per history
% CALL function pulsemultip
j=0;

for i=1:idn
    j=j+1;

[ptimes(:,j),partyps(:,j),gennums(:,j),parnums(:,j),codes(:,j),npulses(:,j)]=pulsemultip(
idet(i));
end

tim=tmin/10:0.1:tmax/10;          % delay binning (shakes)
n=zeros(istop,length(tim));

if istar==1    % START detector exists
    nn=zeros(istop,length(tim));
    pp=zeros(istop,length(tim));
    np=zeros(istop,length(tim));
    pn=zeros(istop,length(tim));
    ns=zeros(istop,length(tim));
    ng=zeros(istop,length(tim));
    nc=zeros(istop,length(tim));
    for i=2:idn
        indptime=find(ptimes(:,1)~=-999 & ptimes(:,i)~=-999 ); % index of histories with
detections in START and one STOP detector
        difftime=ptimes(indptime,i)-ptimes(indptime,1);
        parst=partyps(indptime,1);     % particle type for START detector
        partyp=partyps(indptime,i);    % particle type for STOP detectors
        inn=find(partyp==1 & parst==1);
        ipp=find(partyp==2 & parst==2);
        inp=find(partyp==2 & parst==1);
        ipn=find(partyp==1 & parst==2);
        nni=hist(difftime(inn),tim);
        ppi=hist(difftime(ipp),tim);
        npi=hist(difftime(inp),tim);
        pni=hist(difftime(ipn),tim);
        nn(i-1,:)=nni;
        pp(i-1,:)=ppi;
        np(i-1,:)=npi;
        pn(i-1,:)=pni;
        is=find(gennums(indptime,i-1)==0);     % index of particles from the source
        ig=find(gennums(indptime,i-1)>0);      % index of particles with generation
number >0
        nsi=hist(difftime(is),tim);
        ns(i-1,:)=nsi;
        ngi=hist(difftime(ig),tim);
        ng(i-1,:)=ngi;
```

```
        tot=hist(difftime,tim);
        n(i-1,:)=tot;
        ic=find(parnums(indptime,1)==parnums(indptime,i));     % index of cross talk
events
        icn=find(parnums(indptime,1)~=parnums(indptime,i));    % index of non-cross talk
events
        nci=hist(difftime(ic),tim);
        nc(i-1,:)=nci;
        nnci=hist(difftime(icn),tim);
        nnc(i-1,:)=nnci;
    end
    check1=n-nn-pp-np-pn;
    check2=n-ng-ns;
    check3=n-nc-nnc;
    if any(check1)
        error('Total time difference does not equal sum of n and p contributions')
    elseif any(check2)
        error('Total time difference does not equal sum of generations')
    elseif any(check3)
        error('Total time difference does not equal sum of cross talk and non-cross talk
events')
    end

elseif istar==0 % No START detector (starts at time = 0)
    nn=zeros(idn,length(tim));
    pp=zeros(idn,length(tim));
    ns=zeros(idn,length(tim));
    ng=zeros(idn,length(tim));
    for i=1:idn
        indptime=find(ptimes(:,i)~=-999); % index of histories with detections
        in=find(partyps(:,i)==1);
        ip=find(partyps(:,i)==2);
        nni=hist(ptimes(in,i),tim);
        nn(i,:)=nni;
        ppi=hist(ptimes(ip,i),tim);
        pp(i,:)=ppi;
        tot=hist(ptimes(indptime,i),tim);
        n(i,:)=tot;
        is=find(gennums(:,i)==0);     % index of particles from the source
        ig=find(gennums(:,i)>0);      % index of particles with generation number >0
        nsi=hist(ptimes(is,i),tim);
        ns(i,:)=nsi;
        ngi=hist(ptimes(ig,i),tim);
        ng(i,:)=ngi;
    end
    check1=n-nn-pp;
    check2=n-ng-ns;
    if any(check1)
        error('Total time difference does not equal sum of n and p contributions')
    elseif any(check2)
        error('Total time difference does not equal sum of generations')
    end

else
    error('Invalid istar parameter')
end

% Plot correlation functions
for i=1:istop
    n(i,:)=n(i,:)/sdis;
    nn(i,:)=nn(i,:)/sdis;
    pp(i,:)=pp(i,:)/sdis;
    ns(i,:)=ns(i,:)/sdis;
    ng(i,:)=ng(i,:)/sdis;
    if istar==1
        np(i,:)=np(i,:)/sdis;
        pn(i,:)=pn(i,:)/sdis;
    end
end
if isep==1
    for i=1:istop
        figure
        semilogy(tim*10,n(i,:))
        xlabel('Time delay (ns)');
    end
elseif isep==0   % lump all detections in different detectors together
    n=sum(n,1);
    figure
    semilogy(tim*10,n)
    hold on
```

```
    xlabel('Time delay (ns)');
else
    error('Invalid isep entry')
end

% Plot separate neutron and photon contributions

if isc==1
    if isep==1
        for i=1:istop
            figure
            semilogy(tim*10,n(i,:))
            hold on
            semilogy(tim*10,nn(i,:),'.')
            semilogy(tim*10,pp(i,:),'x')
            xlabel('Time delay (ns)');
            if istar==1
                semilogy(tim*10,np(i,:),'--')
                semilogy(tim*10,pn(i,:),':')
                legend('total','n n pairs','p p pairs','n p pairs','p n pairs')
            else
                legend('total','neutrons','photons')
            end
        end
    elseif isep==0   % lump all detections in different detectors together
        n=sum(n,1);
        nn=sum(nn,1);
        pp=sum(pp,1);
        figure
        semilogy(tim*10,n)
        hold on
        semilogy(tim*10,nn,'.')
        semilogy(tim*10,pp,'x')
        xlabel('Time delay (ns)');
        if istar==1
            np=sum(np,1);
            pn=sum(pn,1);
            semilogy(tim*10,np,'--')
            semilogy(tim*10,pn,':')
            legend('total','n n pairs','p p pairs','n p pairs','p n pairs')
        else
            legend('total','neutrons','photons')
        end
    else
        error('Invalid isep entry')
    end
end

% Plot separate generations

if ign==1
    if isep==1
        for i=1:istop
            figure
            semilogy(tim*10,n(i,:))
            hold on
            semilogy(tim*10,ns(i,:),'.r')
            semilogy(tim*10,ng(i,:),'xb')
            xlabel('Time delay (ns)');
            legend('Total','Generation zero particles','Induced fission particles')
        end
    elseif isep==0   % lump all detections in different detectors together
        n=sum(n,1);
        ng=sum(ng,1);
        ns=sum(ns,1);
        figure
        semilogy(tim*10,n)
        hold on
        semilogy(tim*10,ns,'.r')
        semilogy(tim*10,ng,'xb')
        xlabel('Time delay (ns)');
        legend('total','Generation zero particles','Induced fission particles')
    else
        error('Invalid isep entry')
    end
%     figure
%     perc=(ng./n)*100;
%     plot(tim*10,perc,'.') % plot percentage of signature that is given by induced
fission
%     xlabel('Percentage of signature that is given by induced fission');
```

```
end

% Plot cross talk analysis

if ict==1
    nc=nc/sdis;
    nnc=nnc/sdis;

    for i=1:istop
        figure
        semilogy(tim*10,n(i,:))
        hold on
        semilogy(tim*10,nc(i,:),'.r')
        semilogy(tim*10,nnc(i,:),'xb')
        xlabel('Time delay (ns)');
        legend('Total','Cross talk events','Different particle events')
    end
end

if (multopt==0 & ihos==0)
    return
end

% Multiplicity

maxmultip=20;
multip=zeros(maxmultip,1);
multipn=zeros(maxmultip,1);

for j=1:lindex
    pulsetrain=[];
    ptyps=[];
    for i=1:idn              % index of detectors,
        if(npulses(j,i)>0)  % at least one detection in jth history, ith detector
            a=sum(npulses((1:j),i));  % previous pulses in detector i
            indini=a;                      % index for beginning of time train in jth
history
            indfinal=a+npulses(j,i)-1;  %  "         end          "             "
            time=ptimes(indini:indfinal,i); % times for the ith detector and the jth
history
            ptyp=partyps(indini:indfinal,i); % particle type of the ith detector, jth
history
            pulsetrain=[pulsetrain; time];  % stores pulsetrain times for the jth
history, all detectors
            ptyps=[ptyps; ptyp]; % store particle types jth history, all detectors
        end
    end
    if ~isempty(pulsetrain)
        ind=find(pulsetrain>0);
        pulsetrain=sort(pulsetrain(ind)); %order from earliest to latest detection
        if any(find(pulsetrain<=0));error('Negative times in multiplicity
calculation');end
        % tally multiplicity
        if multopt==1  % source triggered
            indm=find(pulsetrain<=timewindow);
            mm=length(indm);
            indmn=find(ptyps(indm)==1);
            mmn=length(indmn);
            if mm<maxmultip
                multip(mm+1)=multip(mm+1)+1;  % multip(1)=multiplet 0; multip(2)=
multiplet 1; multip(3)=multiplet 2 and so on....
                multipn(mmn+1)=multipn(mmn+1)+1; % multiplicity given by neutrons only
            else
                disp('Warning: increase parameter "maxmultip" in "postmain": multiplicity
processing continues')
            end
        elseif multopt==2 % triggered by a detection event
            for jj=1:length(pulsetrain)
                mm=length(find(pulsetrain-pulsetrain(jj)<=timewindow+pulsetrain(jj) &
pulsetrain-pulsetrain(jj)>=predelay ));
                if mm>0 & mm<maxmultip
                    multip(mm+1)=multip(mm+1)+1;  % multip(1)=multiplet 0; multip(2)=
multiplet 1; multip(3)=multiplet 2 and so on....
                elseif mm==0
                    multip(1)=multip(1)+1;
                elseif mm>=maxmultip
                    disp('Warning: increase parameter "maxmultip" in "postmain":
multiplicity processing continues')
                end
            end
```

```
        end
    end
end



multip(1)=sdis-sum(multip); % set multiplicity of 0's to number of source events - sum of
any multiplet

multiplets=0:19;
figure
semilogy(multiplets,multip(1:maxmultip)/sdis,'o-')
hold on
semilogy(multiplets,multipn(1:maxmultip)/sdis,'x-')
xlabel('Multiplet')
ylabel('Number of events per source emission')
legend('Total multiplicity','Neutron only multiplicity')
totcounts=sum(multip);
disp(['Total counts in multiplicity time window = ',num2str(totcounts)])
disp(['Time window = ',num2str(timewindow*10),' ns'])

% Higher order statistics - bicovariance plots

    if ihos==1
        indptime=find(ptimes(:,1)~=-999 & ptimes(:,2)~=-999); % index of histories with
detections in both dets
        indnn=find(partyps(:,1)==1 & partyps(:,2)==1);  % index of particles by particle
type: nn,pp,pn,np
        indpp=find(partyps(:,1)==2 & partyps(:,2)==2);
        indpn=find(partyps(:,1)==2 & partyps(:,2)==1);
        indnp=find(partyps(:,1)==1 & partyps(:,2)==2);
        indss=find(gennums(:,1)==0 & gennums(:,2)==0); % index of particles by particle
type: ss,gg,sg,gs
        indgg=find(gennums(:,1)>0 & gennums(:,2)>0);
        indsg=find(gennums(:,1)==0 & gennums(:,2)>0);
        indgs=find(gennums(:,1)>0 & gennums(:,2)==0);

        deltat=tmax-tmin+1;

        % call function binit to bin HOS data

        mathos=binit(ptimes,indptime,deltat);
        mathosnn=binit(ptimes,indnn,deltat);
        mathospp=binit(ptimes,indpp,deltat);
        mathosnp=binit(ptimes,indnp,deltat);
        mathospn=binit(ptimes,indpn,deltat);
        mathosss=binit(ptimes,indss,deltat);
        mathosgg=binit(ptimes,indgg,deltat);
        mathossg=binit(ptimes,indsg,deltat);
        mathosgs=binit(ptimes,indgs,deltat);

        check1=mathos-mathosnn-mathospp-mathosnp-mathospn;
        check2=mathos-mathosss-mathosgg-mathossg-mathosgs;
        if any(check1)
            error('Total HOS data does not equal sum of n and p contributions')
        elseif any(check2)
            error('Total HOS data does not equal sum of source and generation
contributions')
        end

        mathos=mathos/sdis;   % perform normalization to number of source events
        mathosnn=mathosnn/sdis;
        mathospp=mathospp/sdis;
        mathosnp=mathosnp/sdis;
        mathospn=mathospn/sdis;

        mathosss=mathosss/sdis;
        mathosgg=mathosgg/sdis;
        mathossg=mathossg/sdis;
        mathosgs=mathosgs/sdis;

        det1ax=tmin:tmax;
        det2ax=tmin:tmax;

        figure % plot total signature

surf(det1ax,det2ax,mathos,'FaceColor','interp','EdgeColor','none','FaceLighting','phong')
        title('Total signature')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')
```

```
        figure % plot separate photon and neutron contributions

SUBPLOT(2,1,1),surf(det1ax,det2ax,mathospp,'FaceColor','interp','EdgeColor','none','FaceL
ighting','phong')
        title('Photon-photon')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')

SUBPLOT(2,1,2),surf(det1ax,det2ax,mathosnn,'FaceColor','interp','EdgeColor','none','FaceL
ighting','phong')
        title('Neutron - neutron')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')

        figure

SUBPLOT(2,1,1),surf(det1ax,det2ax,mathosnp,'FaceColor','interp','EdgeColor','none','FaceL
ighting','phong')
        title('Neutron - photon')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')

SUBPLOT(2,1,2),surf(det1ax,det2ax,mathospn,'FaceColor','interp','EdgeColor','none','FaceL
ighting','phong')
        title('Photon - neutron')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')

        figure % plot separate generation zero and induced fission contributions

surf(det1ax,det2ax,mathosss,'FaceColor','interp','EdgeColor','none','FaceLighting','phong
')
        title('Gen. zero - gen. zero')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')

        figure

surf(det1ax,det2ax,mathosgg,'FaceColor','interp','EdgeColor','none','FaceLighting','phong
')
        title('Ind. fission - ind. fission')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')

        figure

surf(det1ax,det2ax,mathossg,'FaceColor','interp','EdgeColor','none','FaceLighting','phong
')
        title('Gen. zero - ind. fission')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')

        figure

surf(det1ax,det2ax,mathosgs,'FaceColor','interp','EdgeColor','none','FaceLighting','phong
')
        title('Ind. fission - gen. zero')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')

        % contour plots
        figure % plot total signature
        contour(det1ax,det2ax,mathos)
        title('Total signature')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')
        axis equal
        figure % plot separate photon and neutron contributions
        SUBPLOT(2,2,1),contour(det1ax,det2ax,mathospp)
        set(gca,'XLim',[tmin,tmax],'YLim',[tmin,tmax])
        title('Photon-photon')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')

        SUBPLOT(2,2,2),contour(det1ax,det2ax,mathosnn)
        set(gca,'XLim',[tmin,tmax],'YLim',[tmin,tmax])
        title('Neutron - neutron')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')
```

```
        SUBPLOT(2,2,3),contour(det1ax,det2ax,mathosnp)
        set(gca,'XLim',[tmin,tmax],'YLim',[tmin,tmax])
        title('Neutron - photon')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')

        SUBPLOT(2,2,4),contour(det1ax,det2ax,mathospn)
        set(gca,'XLim',[tmin,tmax],'YLim',[tmin,tmax])
        title('Photon - neutron')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')
        axis equal

        figure % plot separate generation zero and induced fission contributions
        SUBPLOT(2,2,1),contour(det1ax,det2ax,mathosss)
        set(gca,'XLim',[tmin,tmax],'YLim',[tmin,tmax])
        title('Gen. zero - gen. zero')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')
        SUBPLOT(2,2,2),contour(det1ax,det2ax,mathosgg)
        set(gca,'XLim',[tmin,tmax],'YLim',[tmin,tmax])
        title('Ind. fission - ind. fission')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')
        SUBPLOT(2,2,3),contour(det1ax,det2ax,mathossg)
        set(gca,'XLim',[tmin,tmax],'YLim',[tmin,tmax])
        title('Gen. zero - ind. fission')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')
        SUBPLOT(2,2,4),contour(det1ax,det2ax,mathosgs)
        set(gca,'XLim',[tmin,tmax],'YLim',[tmin,tmax])
        title('Ind. fission - gen. zero')
        xlabel('Time lag to det 2 (ns)')
        ylabel('Time lag to det 1 (ns)')

    end

return
```

pulsemultip.m

```
function [time,partyp,gennum,parnum,code,npulse]=pulsemultip(ndet)
% function pulsemultip(ndet) returns pulse time vector for detector cell number = ndet
% includes detector deadtime - set in postmain

global pgt lonc ncal pcal thresh deadtime zzz index nch lindex ilo distance idi itm
time=-999*ones(lindex*3,1);
partyp=-999*ones(lindex*3,1);
gennum=-999*ones(lindex*3,1);
parnum=-999*ones(lindex*3,1);
code=-999*ones(lindex*3,1);
npulse=zeros(lindex,1);

ind=1;
maxmultip=20;

for i=1:lindex
stride=find(zzz(index(i):index(i)+nch(i)-1,6)==ndet); % find interactions involving ndet
pilaux=zzz(index(i)-1+stride,1:15);          % submatrix involving ndet in ith history
if any(diff(pilaux(:,8))<0)
   pilaux=sortrows(pilaux,8);        % sort in terms of increasing times
end

lout=zeros(length(stride),1);
% convert energy depositions to light (lout) for each (stride) interaction
   for j=1:length(stride)
      if pilaux(j,3)==1          % particle = neutron
         if pilaux(j,5)==1001      % nucleus  = hydrogen
            lout(j)=polyval(ncal,pilaux(j,7));  % light output for energy dep.
         elseif floor(pilaux(j,5)/1000)==6     % nucleus = carbon
            lout(j)=pilaux(j,7)*lonc;  % light output = energy deposited * light
         else disp(['! error, the struck nucleus is unknown  ', int2str(pilaux(j,5))])
         end
      elseif pilaux(j,3)==2         % particle = photon
         lout(j)=polyval(pcal,pilaux(j,7));     % light output for energy dep.
      else disp(['! error, the incident particle is unknown  ', int2str(pilaux(j,3))])
      end
   end
   lightpil=zeros(length(stride),1);

   for j=1:length(stride)
      timediff=pilaux(:,8)-pilaux(j,8);  % difference in times of energy depositions
      indtime=find(timediff>=0 & timediff<pgt); % time difference within pulse generation
time
      lightpil(j)=sum(lout(indtime));    % pileup of light pulses
   end
   indtim=find(lightpil>=thresh);        % indices of light pulses exceeding the
threshold (MeVee)
   lindtim=length(indtim);               % number of light pulses exceeding the threshold
(MeVee)
   indtime2=zeros(1,length(indtim));

   if lindtim>=1  % at least one pulse above the threshold
      indtime2(1)=indtim(1);  % register index for time of first pulse
      for k=1:lindtim-1
         timediffpulse=pilaux(indtim,8)-pilaux(indtime2(k),8);
         ind2=find(timediffpulse>=deadtime); % index of pulses where time between pulses
exceeds system dead time
         if ~isempty(ind2)
             indtime2(k+1)=indtim(ind2(1));

         end
         if indtime2(k+1)==0
            break
         end
      end
      indtm=find(indtime2);
      if indtm>maxmultip
          disp('Warning: increase maxmultip parameter in function "pulsemultip" ')
      end
      tmultip=length(indtm); % multiplicity for this history and detector (includes
deadtime)
      time(ind:ind+tmultip-1)=pilaux(indtime2(indtm),8);     % vector of pulse times
      partyp(ind:ind+tmultip-1)=pilaux(indtime2(indtm),3);   % vector of particle types
      gennum(ind:ind+tmultip-1)=pilaux(indtime2(indtm),13);  % vector of generation
numbers
```

```
      parnum(ind:ind+tmultip-1)=pilaux(indtime2(indtm),2);   % vector of particle numbers
      code(ind:ind+tmultip-1)=pilaux(indtime2(indtm),15);
      npulse(i)=tmultip; % number of pulses per history in this detector
      ind=ind+tmultip;
      if (ind>lindex*3)
          error('Pulsemultip: increase vector assignment lengths in "pulsemultip" and
"postmain" !')
      end
   end
end
```

binit.m

```
function [mat]=binit(ptimes1,index1,deltat1)
% function binit(ptimes1, index1, deltat1) returns HOS matrix
% called by postmain when HOS option is chosen
% times 0 - 0.9999 ns = bin 1
%      1.0- 1.9999 ns = bin 2
%      ..


mat=zeros(deltat1);

for a=1:length(index1)
   i=floor(ptimes1(index1(a),1)*10)+1; j=floor(ptimes1(index1(a),2)*10)+1; % bins time of
detection
   if(1<=i & i<=deltat1 & 1<=j & j<=deltat1)
      mat(i,j)=mat(i,j)+1;
   end
end
```