Functions List

[ Par ] = **SamplingParameters**( Par )

*SamplingParameters* takes the geometrical parameters saved in struct variable *Par* and computes the position of the bins centers (Par.x\_knot and Par.y\_knot) which are the position where the cost function of the statistical method is going to be evaluated.

[ Frame ] = **CalibrationDatasetLoad**( file\_name )

*CalibrationDatasetLoad* loads the data matrix (N events x M channels) *Frame* saved in the file (.mat) with the name extension *file\_name*.

[ Energy ] = **EnergySpectrum**( Frame , depict )

*EnergySpectrum* computes the signal energy from the data matrix *Frame*, as sum of signal from all channels and plots the signal spectrum if *depict* is equal to 1. Set *depict* to 0 to silence the spectrum plot.

[ x\_rec , y\_rec , Counts , energy\_window , number\_det\_ch ] = **CentroidReconstruction**( Frame , Par , Filt , Energy , depict )

*CentroidReconstruction* applies the Centroid Method to raw dataset *Frame* to compute the event reconstructed coordinates (*x\_rec*, *y\_rec*). The method requires also geometrical information contained in *Par* and applies filter passed to the function through *Filt*. The *Energy* input is redundant, but required. When *depict* is set to **1**, the function plots the event reconstructed image. *Counts* is a 2D matrix representing the reconstructed image: each cell in the matrix represents a pixel of dimension Par.pixel\_image; the value of each cell represents the number of events associated to that 2D coordinate.  
*energy\_window* is a logic vector with the same dimension of *x\_rec* and *y\_rec* and defines with 1 all the event indexes which respect the filter boundaries and with 0 the events outside the filter windows.  
*number\_det\_ch* is a vector with the same dimension of *x\_rec* and *y\_rec* and, for each event, defines the number of detectors that received signal, after baseline subtraction.

[ LRF ] = **LRF\_Load**( Num\_channels , LRFs , Par )

*LRF\_Load* takes as input the parameters of the optical models saved in *LRFs* (which are originally saved in the LRF file), the geometrical data in *Par* and the number of detection channels *Num\_channels*. The output *LRF* is a numerical sampling of the analytical LRFs, for all the detection channels. *LRF* is a 3D matrix ( Num\_channel x BinY x BinX), where BinY and BinX are respectively the number of bin in the two coordinate directions, calculated as length of the crystal in one direction divided by the Par.sempling parameter (BinX, BinY and other key parameters for map sampling are computed in *SamplingParameters* function).

[ x\_rec , y\_rec , energy , error , Filt ] =

**StatisticalMethod**( Frame , Par , Tune , Filt )

StaticalMethod takes as input the raw dataset *Frame*, the geometrical information *Par*, the tuning parameters in *Tune* and the filter in *Filt*. The function recalls the proper statistical method for reconstruction defined in *Filt.Recon\_Method* and gives as output the reconstructed coordinates (*x\_rec,y\_rec*), the reconstructed energy, the root mean square error on the reconstruction *error*.

The methods that the function can recall are:

* ML\_reconstruction
* LS\_reconstruction
* WSE\_reconstruction

[Counts] = **DisplayReconstruction**( output , Par , Filt , Tune , isLRF , Unif\_correction )

DisplayReconstruction takes the reconstruction coordinates calculated with a statistical method and returns *Counts*, a 2D histogram of the number of counts per pixel (pixel dimension is defined through Par.pixel). The reconstructed image can be depicted by command *imagesc(Counts)*

[ index , data ] = **FcnMovingRegionCalculation**( coord, search\_dim, Par )

This function generates the index within a general research matrix of a local reaserch window with dimension *search\_dim*, centered in *coord.*

[ DY,DX] = **fitting\_max\_ML**( val\_fit )

This function takes a 3 x 3 region of pixel around the maximum (*val\_fit*) and finds the real maximum position by interpolating the data and looking for the zero position of the derivative term.

[weights] = **FcnWLSweightCalc**(Frame,Par,Filt)

This function calculates the weights required by the Weighted Least Square Error method for reconstruction

[LRFs] = **OpticalModelEngine**( REC,Par,Tune, noise\_on\_data, fit\_method)

This function estimates the analytical Light Response Functions (LRFs), given the input parameters to the iterative method by Solovov/Morozov.

[options]=**OptionFitDistribution**(fit\_method)

This function recalls some fundamental parameters for LRFs fitting like initial research parameters and boundaries of the research

[fitresult, gof] = **GaussianFit2D**(X\_REC, Y\_REC, F\_REC, opzioni, x0, y0)

This functions fits the input data with a 2D Gaussian bell and returns the fitting parameters. Please be aware of the fact that the boundaries for the (x,y) position of the Gaussian peak are fixed inside this function and this could represent a limit while trying to implement the iterative method with camera’s geometries different from the INSERT module one.

[ LRF\_par\_spline ] = **Bspline\_Fitting**(Par,n\_div,X\_REC,Y\_REC,F\_REC,immagine,Tune)

This functions fits the input data with a 2D Bsplinesl and returns the fitting parameters.