GLAST/MPE/Mayer-Hasselwander/00/08/31

Data Structures representing the GLAST- PSD, EDP, SAR

(Pointspread Distribution, Energy Dispersion, Sensitive Area)

Draft 4

Data Structures proposed for storage and transfer of intermediate results (no event tuples involved) between the following processing steps:

Analysis and raw data histogram creation form Beam or Simulation runs
Smoothing / Fitting/ Functional representation of the raw histograms
Display of results from runs for quality assurance
Construction of the ultimate 'response' datasets

Processing:

Step 1:

- create raw 'run' datasets (essentially parameters+histograms) from a series of MC or beamtest runs and store them sequentially into a disc file; this process includes application of cuts and of beam information.

Step 2:

- Read 'raw' datasets from step 1 and process its tuples
- Write the processed tuples, extending the input data, into a new disc dataset
- Write 'compressed' result tuples to a second new disc dataset (small)

Step 3:

- Read output from step 2 for display (IDL) and for input to analysis programs
- Read output from step 2 for construction of response calibration datasets.

R&D phase:
The tuples and disc datasets are defined keeping in mind arguments concerning disc storage, software accessibility andprocessing sequence.
The datasets are suggested to be individually self explaining and readable in an editor (ASCII, even on the expense of some storage inefficiency).

Fixed length tuples, allowing for fixed length data records on disc, are adopted in order to avoid any possible difficulty in disc writing/reading on the various platforms, operating systems and languages.

The tuples as described below can be comfortably combined into output records for storage in disc datasets as needed. Currently the following combinations are envisaged:

The 'run' oriented basic processing (cuts, beam analysis) jobs produces one output dataset containing records which each consist of the tuples:

Record in file ‘Run_Rawdat’:

‘run_param’ + 'run_psd_binning’ + ‘run_psd_rawdat’ + ‘run_edp_rawdat’

The processing step which analyzes and processes further the PSD, EDP, DEF, SAR typically produces two output files, one file containing the complete information and one file containig the compact response representations:

Record in file ‘Run_Long’:

'run_param' + 'run_psd_binning' +
'run_psd_smooth’ + ‘run_edp_smooth’ +
‘run_psd_param’ + 'run_edp_param'

Record in file ‘Run_Short’:

‘run_param’ + ‘run_psd_param’ + 'run_edp_param’

Operational requirements:

Interpretation of records according to key
Sequential processing of many runs
Appending run-records (continue processing)
Editing (e.g. adding files using editor, eliminating records, changing values)
Selective processing (adding, deleting, replacing run-records)

Content of tuples:

1. ‘run_param’ tuple

record key	e.g. 'Run_short'	string	Char
origin_string	Origin designator	string	Char
cut_types	Cut combinations	strings	CharArray
cut_range_low	Cut low range limit	numbers	FloatArray
cut_range_high	Cut high range limit	numbers	FloatArray
energy_mode	Mono-E or E-Range	string	Char
incid_energy	MC or Beam Energy	MeV	Int
spect_index	Spectral index	number	Float
incid_E_low	low limit measured E	MeV	Int
incid_E_high	high limit measured E	MeV	Int
incid_inclin	Incidence Inclination	deg	Float
incid_azimut	Incidence Azimut	deg	Float
incid_3vec	Incidence vector	numbers	FloatArray
incid_type	single dir. or average	string	Char
incid_pos_x	Incidence X-position	mm	Float
incid_pos_y	Incidence Y-position	mm	Float
incid_pos_z	Incidence Z-position	mm	Float
incid_flux_area	area covered by input flux	cm^2	Float
incid_photons	number of incid photons	number	Int
detected_events	number of detected evts	number	Int
def_value	detection efficiency	number	Float
sar_value	sensitive area	number	Float

2. ‘run_psd_param’ tuple:

psd_hwhm	Halfwidth-Halfmaximum	deg	Float
psd_68	68% containement angl	deg	Float
psd_90	90% containment angl	deg	Float
psd_g_ncomp	number of gauss comp.	number	Int
psd_g_width	width of gauss compon.	deg	FloatArray
psd_g_amp	amplitude of compon.	prob/rad	FloatArray

3. ‘run_edp_param’ tuple:

edp_peak	Peak of distribution	number (E_P/E_T)	Float
edp_hwhm	Halfwidth-Halfmax	% of E_{Peak}	Float
edp_68	68% containement	% of E_{Peak}	Float
edp_90	90% containment	% of E_{Peak}	Float
edp_g_ncomp	number of components	number	Int
edp_g_offset	offset of gauss compon.	number	FloatArray
edp_g_width	width of gauss compon.	number	FloatArray
edp_g_ampl	amplitude of components	prob/rad	FloatArray

4. ‘run_psd_binning’ tuple:

psd_bin_number	number of bins	number	Int
psd_bincent_array	bin center locations	deg	FloatArray
psd_binsize_array	binsizes	radian	FloatArray

5. ‘run_psd_rawdat’ tuple:

psd_overflow_c	Overflow bin events	counts	Int
psd_counts_array	PSD distribution	events/bin	IntArray

6. ‘run_psd_smooth’ tuple:

psd_overflow_p	Overflow bin content	probability	Float
psd_prob_array	PSD distribution	probability/rad	FloatArray

7. ‘run_edp_rawdat’ tuple:

edp_overflow_c	Overflow bin content	counts	Int
edp_counts_array	EDP distribution	counts / (E_M/E_T)bin	IntArray

8. ‘run_edp_smooth’ tuple:

edp_overflow_p	Overflow bin content	probability	Float
edp_prob_array	EDP distribution	probab / (E_M/E_T)bin	FloatArray

9. ‘response_calc_const’ tuple:

The following parameters are assigned as constants in the programs:

psd_array_size	psd array dimension	number	IntConst
edp_array_size	number of bins = arr dim	number	IntCons
edp_highend	highest (E_M/E_T)	number	FloatConst
edp_bin_size	binsize	number	FloatConst

psd_comp_max	max n. of gauss comp.	number
IntConst		
edp_comp_max	max n. of gauss comp.	number
IntConst		