QSAS Array Handling

Steve Schwartz
Queen Mary, University of London

Document Status Sheet			
1. Document Title: **QSAS Array Handling**			
2. Document Reference Number: **CU-QMW-TN-0015**			
3. Issue	4. Revision	5. Date	6. Reason for Change
0	0	24 May 2001	First draft; to accompany CEF format doc

Contents

1 Background and Introduction

Now that QDOS2 can formally hold arrays, and with PEACE and other data in mind, we need to examine what functionality QSAS needs to provide. This includes metadata and arithmetic or other manipulations.

Note that we deal here with array entities in a manipulative sense. That is, for example, matrices which obey matrix algebra. Thus for a time series of matrices, the timetag (or record) is NOT an array index, but a sequence indicator. This corresponds to their treatment in QDOS2. A time series of vector velocity is thus a sequence of 3×1 vectors.

In the following, I denote these sequence indicators by subscript "s".

2 Examples

I give here a few real examples of array data. Later sections comment on the metadata and treatment aspects. Appendix A illustrates a possible but incomplete implementation of most of these arrays and their metadata.

Physical Vectors These are already supported by QSAS, but it is perhaps instructive to include them here to see how they fit into the more general framework. Physical vector time series are 1-D arrays of the form

$$a(i)(t_s)$$

Here i indexes the vector component ((x,y,z) or (r,θ,ϕ) or whatever) of a quantity, such as magnetic field or flow velocity, measured at time t_s.

Energy-time or frequency-time These are examples of sequences of 1-D arrays with the following structure

$$a(i)(t_s)$$

For typical particle instruments, a is counts or phase space density or differential energy flux or whatever measured at various scalar energies E_i and times t_s. For wave experiments, a could be the spectral power measured at frequencies ν_i (at times t_s).

Pitch angle distributions A 2-D array of the form

$$a(i,j)$$

containing counts, phase space density, ... a measured at energies E_i and pitch angles θ_j.

Phase space density A 3-D array of the form

$$a(i,j,k)$$

containing phase space density or counts or ... measured at velocities vx_i, vy_j, vz_k at time t_s OR at energies E_i, polar angles θ_j, and azimuthal angles ϕ_k .

Rotation Matrices A 2-D array of the form

$$a(i,j)$$

of cosines e_i, e_j. We may also often deal with a single matrix. Note that a_{ij} is orthonormal.

Pressure Tensors A 3-D array of the form

$$a(i,j)$$

of the pressure tensor P_{ij}. Note that i and j index base vectors in a cartesian frame, and that P_{ij} is symmetric, has therefore only 6 independent values, is diagonalisable, and can be transformed from one frame to another by (suitable) double application of a rotation matrix.

3 Metadata and Related Matters

The correct interpretation, and therefore manipulation, of the above arrays requires additional information in the form of associated data and descriptors for the values of *a* and also for each of the dimenstions *i, j, k* . . .

3.1 Universal Metadata

Some metadata is the same for all array elements. This is similar to metadata for a scalar quantity, such as the Name, Frame, Units, and SI_conversion for the values of *a*. It is typically descriptive strings (with some embedded numerical values in some cases).

3.2 Timetags

For completeness, let us begin with some brief comments on the timetags. As noted in the Introduction, these play the role of sequence indicators rather than an array dimension. Nonetheless, many considerations which apply to array dimensions also apply to timetags.

Time series of arrays will have timetags (as in the examples given above). Although the Cluster convention is time-centred tags, it is quite common for data to be tagged with accumulation start time. PEACE high resoloution data is delivered naturally in this way and ingestion routines could (probably) shift these to conform to QSAS conventions. Alternatively, since we will probably write some of these PEACE IDFS routines, we could force them to return time-centred tags. The same may not be true of (all) foreign data. **To shift to centres on ingestion may need some extention to QIE. For ASCII files, using a header with FreeTimeFormat timetags would enable this to be done easily if the shift is constant and known.**

At present, I recommend that effort not be invested into this problem, and shift it to either the data provider or to a plug-in. The codification used in Appendix A for energy and angle bins would also work for time.

3.3 Metadata for Array Dimensions

Conceptually, the same issues arise for array dimensions as do for timetags. Manipulation may require more than one dimension to be treated together.

3.3.1 Basic Concepts

Each dimension *i* corresponds to a 1-D array of length Size_i. I assume here that all arrays *a* are conceptually rectangular; that is, the values assigned to one dimension apply for all positions down all other dimensions, with the possible exception of the time (see below). It is possible that some dimensions are closely related, such as v_{x_i}, v_{y_j}, v_{z_k} in the phase space density time series $f(v_{x_i}, v_{y_j}, v_{z_k}) \equiv f(\vec{v}_{ijk})$. This association could prove useful since $\vec{v}$ has the properties of a vector, can be rotated or (be) expressed in polars, etc.

It is possible that a dimension corresponds itself to an array of arrays, rather than a 1-D array of scalars; one use for such a notion (which in any case is straightforward to implement) is in bin boundaries. Note however, that the array *a* cannot index into these arrays, so using them requires information about their dimensionality to be visible somewhere.

Thus each dimension requires specification of:

Size_i The number of entries in this dimension. It is not necessary to have an explicit metadata item for this if it can be deduced by counting the number of entries.

FIELDNAM$_i$ The name of this dimension, e.g., “Energy” or “Frequency” or “polar angle.

UNITS$_i$ A text string describing the units, e.g., “keV”

Frame Usually “scalar>na” but perhaps we should extend this to include indexing information and/or relationships to other dimensions, such as “index>gse_xyz” if the dimension corresponds to the cartesion components of a physical vector or “x>gse_xyz” if the dimension hold the *x*-component values of a set of vectors. **This to be revisited after CEF implementation is finalised.**

SI_conversion$_i$ To convert the values to base SI units, e.g., “1.0e3>Hz”

Values$_i$ The actual values for the Size$_i$ elements of the array. Although this looks like metadata for the array *a*, these Values are just another array (object) and so, like Size, may be inferred by the fact that it is associated with *a*. However, it is necessary to ensure that the association of the value arrays is ordered to associate to the correct dimension of *a*. In QDOS-speak, it is not sufficient that there be a set of cross-references to, say, EnergyLevelObject and PitchAngleObject; EnergyLevelObject needs to be associated with the first dimension of *a* and PitchAngleObject the second if the array is held as $a(E_i, \theta_j)$.

Note that these values may be independent of all the other dimensions, e.g., the same set of energy level values for all time “records” or there may be different values for different records. An example of the latter case is a particle instrument which alternates between two or more bin ranges or continually changes its energy range from spin to spin, or from one instrument mode to another. Infrequent changes in instrument mode are easiest to accommodate by splitting into separate objects. Alternating modes could be accommodated by a flag for each record. Truly wandering bins require more metadata. Cluster CIS (and PEACE) have some alternating mode high resolution data products.

3.3.2 Bin Information

For interpolation/joining and plotting arrays, it is often necessary to know more detailed information about the bin (linear, area, volume, ...) in which each measurement lies.

Bin Boundaries Something which indicates whether the values are at the centres of each bin or somewhere else, together with information on the bin width. Two models are prevalent, though other variants are also found:

1. Bin start or centres with given bin width or touching bins. If the bins are centred and touching, are they linearly or logarithmically (or arbitrarily) spaced? It is easy to generalise to bin values located a fraction f (linearly or logarithmically as appropriate) from the start, e.g., $f = 0$ are bin starts, $f = 0.5$ are centred bins, $f = 1.0$ indicates that the values correspond to the end of the bin. For readability, Appendix A uses “centred” as synonomous with 0.5. Logarithmic widths are $\log(end/start)$
2. Separate arrays, or an $n \times 2$ array, for bin start and bin end values to cope with bins of varying sizes, etc.

Note that arrays are often plotted as spectrograms, so each array element is mapped to a finite region on the plot, requiring start and end bin values for all plottable dimensions.

Note also that in much of the discussion which follows I assume explicitly that the bin boundaries are such that the end values are always arithmetically larger than the start values. Some bins are often returned in the opposite sense, such as decaying energy sweeps or azimuths from a spacecraft with spin axis anti-aligned with the coordinate system cylindrical axis.

Table 1: Bin "Volume" Factors

	Dimension		
System	1-D	2-D	3-D
Cartesian	$x_2 - x_1$	$(x_2 - x_1)(y_2 - y_1)$	$(x_2 - x_1)(y_2 - y_1)(z_2 - z_1)$
Cylindrical	$\rho_2 - \rho_1$ $z_2 - z_1$ $\phi_2 - \phi_1$	$\rho(\phi_2 - \phi_1)(z_2 - z_1)$ $\frac{1}{2}(\rho_2^2 - \rho_1^2)(\phi_2 - \phi_1)$ $(\rho_2 - \rho_1)(z_2 - z_1)$	$\frac{1}{2}(\rho_2^2 - \rho_1^2)(\phi_2 - \phi_1)(z_2 - z_1)$
Spherical	$r_2 - r_1$ $\theta_2 - \theta_1$ $\phi_2 - \phi_1$	$r^2(\cos\theta_1 - \cos\theta_2)(\phi_2 - \phi_1)$ $\frac{1}{2}(r_2^2 - r_1^2)\sin\theta(\phi_2 - \phi_1)$ $\frac{1}{2}(r_2^2 - r_1^2)(\theta_2 - \theta_1)$	$\frac{1}{3}(r_2^3 - r_1^3)(\cos\theta_1 - \cos\theta_2)(\phi_2 - \phi_1)$

Bin "Volumes" The geometry of the area or volume of a bin in 2-D or 3-D arrays needs to be understood in order to join or interpolate them. For example, bin (ij) of a 2-D (plus time) array $a(i,j)$ measured in cartesian coordinates has an area

$$A_{ij} = (x_i^{end} - x_i^{start})(y_j^{end} - y_j^{end})$$

whereas a 2-D array $a(i,j)$ of Energy vs. pitch angle data has a bin "area"

$$A_{ij} = \frac{1}{2}\left(E_i^{end\,2} - E_i^{start\,2}\right)\left(\theta_j^{end} - \theta_j^{start}\right)$$

in the $E - \theta$ plane but a bin area

$$A_{ij} = \frac{1}{m_\alpha}\left(E_i^{end} - E_i^{start}\right)\left(\theta_j^{end} - \theta_j^{start}\right)$$

in velocity space. Spherical coordinates introduce further complications. In the pitch angle case, for example, the volume of each bin in velocity space would be

$$V_{ij} = \frac{1}{3}\sqrt{\frac{8}{m_\alpha^3}}\left(E_i^{end\,3/2} - E_i^{start\,3/2}\right)\left(\cos\theta_j^{start} - \cos\theta_j^{end}\right)\left(\phi_k^{end} - \phi_k^{start}\right)$$

Such rules need to be associated with the metadata somehow. Note that these concepts require more than one dimension of the original array a to be considered simultaneously. The simplest multi-dimensional interpolation algorithms merely treat each dimension in turn as a sequence of 1-D interpolations.

Table 1 illustrates the range of possible factors for cartesian, cylindrical, and spherical systems. It does not consider the simultaneous transformation from, e.g., energy to velocity space shown above. Note the complication of some 2-D areas which require a value for the third coordinate to specify a real area. This suggests that the bin "volume" information needs to be kept with the parent array rather than inside the separate array dimension variables.

3.3.3 Periodic Dimensions

Some dimensions, in particular azimuthal angle bins, are cyclic. This information needs to be kept and considered as it affects, for example, interpolation and averaging techniques.

3.4 Related Dimensions

Some dimensions may have some logical relationship between one another. For example, phase space density is measured in bins in velocity space, so three dimensions correspond to the three components of a velocity vector, with accompanying frame attributes. There is probably considerable utility in keeping these three dimensions coupled, to enable frame transformations, rotations, etc.

4 Generalised Joining

Before two arrays can be combined arithmetically they need to be "joined" onto a common discretised set of dimensions. There is no conceptual difference between joining onto a common time axis and joining onto a common energy axis. However, in practice, there are two differences with which the user is often faced:

1. The energy dimension (and others such as polar bin/solid angle, ...) are often highly nonlinear, so that simple (boxcar) averaging or linear interpolation are often inappropriate.

2. The values associated with the array *a* can be either integrated over a sampling bin, such as count rates or partial densities, or differential measures, such as phase space density.

Interpolating count rates without taking account of bin size will lead to bizarre results. The only quantities which can be meaningfully interpolated are densities over the corresponding space. That is, to join a count rate measured in energy bins E_i onto energy bins E_j, we should first convert the count rate to a density (by dividing by the size, area, or volume of the associated bin), interpolate this density to the new bin centres E_j, and then multiply by the corresponding new bin size, area, or volume to calculate the interpolated count rate.

The straightforward mechanism to do all this consists of the following steps;

1. Convert all quantities to density in the appropriate space. As a concrete example, let us take the (realistic) case of phase space density measured at energies E_i and polar angles θ_j; we shall assume that the sampling in azimuthal angle ϕ is uniform (i.e., linear) and can be treated as simple linear interpolation at a later stage. Then the original array is $a(E_i, \theta_j)$. There is an associated polar bin area[1] given by

$$A_{ij} \propto (E_i^{end} - E_i^{start})(\cos\theta_j^{start} - \cos\theta_j^{end})$$

   If *a* is phase space density, do nothing. If *a* is counts (i.e., integrated over the bin), form $f_{ij} \equiv a_{ij}/A_{ij}$.

2. Let the new bins have area A_{IJ}.

3. For each new bin, find all the old bins which overlap it. Calculate:

$$F_{IJ}^{new} = \sum_{\text{overlapping bins}} f_{ij} \times \text{overlap area}$$

$$w_{IJ} = \sum_{\text{overlapping bins}} \text{overlap area}$$

[1] Interestingly, this seems an obvious choice, but it does NOT, in fact, correspond to the 2-D $E-\theta$ factors presented in Table 1, nor to part of the 3-D volume given there. For many applications, it may be most appropriate to use the 3-D volume factors even when interpolating over only two dimensions. For example, a pitch angle distribution is integrated in ϕ so that each velocity-angle bin has a weight $\propto v^3$ NOT v^2.

4. Calculate the interpolated density
$$f_{IJ} = F_{IJ}/w_{IJ}$$

5. If the desired quantity is an integral measure (e.g., counts),
$$a_{IJ} = f_{IJ} \times A_{IJ}$$

Joining onto a cyclic dimension, such as azimuthal angle ϕ is accomplished by duplicating bins at either end. Once two quantities have been "joined" onto the same grid, algebraic manipulation can proceed element by element.

5 Reducing Dimensionality

For plotting and other purposes, it will be necessary to reduce the dimensionality of the array *a* down to that of the plot or other target space. This is accomplished by one of two means:

5.1 Slicing

Take a slice of the data. The simplest slices correspond to fixing all indices apart from those in the target. For example, given an array of phase space densities $a(i,j,k)$ at some time, take a slice in which energy E_i varies but angles θ_j and ϕ_k are fixed at user-specified values θ_J and ϕ_K, to generate an array $a'(i)_{JK}$. Such simple slices do not involve manipulations other than selecting values out of the existing array. More complicated slices invoke some relation $I(i,j,k)$ to deliver an array a_I and may require some interpolation as discussed above in the section on generalised joins.

5.2 Integration

Collapse the data onto a sub-dimension by integration. For example, find the energy spectrum of a phase-space distribution $a(i,j,k)$ by integration over angles θ_j and ϕ_k. This requires summation weighted by areas A_{jk} which in this case are given by $A_{jk} = (\cos\theta_j^{start} - \cos\theta_j^{end}) \times (\phi_k^{end} - \phi_k^{start})$.

More complicated integrations require integration onto a surface $S(i,j,k)$. One example is to construct the pitch angle distribution $P(v_I, \alpha_J)$ from the 3-D phase space density $f(v_i, \theta_j, \phi_k)$ given the field direction $\vec{B}$ and bulk flow velocity vector $\vec{V}$. Formally this is

$$P(v', \theta') = \int f(\vec{v} - \vec{V})\, d\phi'$$

where the prime'd frame has zero bulk flow and its *z*-axis aligned with $\vec{B}$. Thus this requires a translation and rotation into the field-aligned, bulk flow frame, followed by an azimuthal integration. It is possible to do this as a single 3-D interpolation following the methodology described in the generalised joining section. Alternatively, it may be more efficient (and accurate) to devise tailor-made algorithms to calculate such pitch angle distributions.

6 Implementation Challenges

I give here a few concrete examples of array manipulations. They use particle phase space density $f(E,\theta,\phi,t)$ and power spectra $P(\nu,t)$.

1. Calculate the different in power between two spectra P_1 and P_2 which are measured at frequencies ν_i. These frequencies are the centres of equally linearly spaced frequency bins.

2. As in the preceding example, but for equally logarithmically space frequency bins.

3. Calculate the difference in power between two spectra P_1 and P_2 measured in frequency bands $\nu_i^{low} \rightarrow \nu_i^{hi}$ and $\nu_I^{low} \rightarrow \nu_I^{hi}$ respectively.

4. Calculate the energy-time spectrogram of f given bin limits $E_i^{low} \rightarrow E_i^{hi}$, $\theta_j^{low} \rightarrow \theta_j^{hi}$, $\phi_k^{low} \rightarrow \phi_k^{hi}$.

5. Calculate the pitch angle distribution of f in the previous example given $\vec{V}$ and $\vec{B}$. [Hard]

A Example Arrays and Their Metadata

This appendix provides some incomplete examples of arrays and their metadata. Outstanding problems include:

1. Codification of information about how related dimensions are related. Perhaps it would be better to codify the nature of the coordinate system (cartesian, spherical polars, etc.) and the matching of dimensions to their place within this system as shown for the Phase Space Density example.

2. Specific handling of cyclic dimensions (codifying as an azimuthal angle would imply cyclic behaviour).

3. Special requirements, e.g., that energies and polar angles must be positive, to ensure that interpolation algorithms don't generate a negative value.

4. Automatic transformation to velocity from energy or vice versa. It is not at all obvious whether joining of, say particle distribution functions, should be done in a spherical system with energy as the radial dimension (so the volume $\propto E^3$) or speed (so the volume is $v^3 \propto E^{3/2}$).

A.1 Physical Vectors

Consider a time series of N_{pts} measurements of magnetic field vectors $\vec{B}(t)$. These are held as a sequence of 1-D arrays of the form $B(comp_i)(t_s)$ with the following structure and metadata:

$B(comp_i)(t_s)$

Object	B	$comp_i$	t_s
Name	Magnetic Field	Cartesian Index	Epoch
Frame	vector>gse_xyz	index>gse_xyz	n/a
UNITS	nT	()	msecs
SI_conversion	1.0e-9>T	1>()	1.0e-3>s
Size	N_{pts}	3	N_{pts}
Values	B_{ij}	(0,1,2)	t_s
Bin_Boundaries	n/a	n/a	centred>touching>linear
Bin_Volume	n/a	n/a	end - start

A.2 Energy-Time Spectrograms

Consider a time series of N_{pts} measurements of omni-directional particle flux $F(t)$. These are held as a sequence of 1-D arrays of the form $F(E_i)(t_s)$, where the energy bins E_i are given by their starting values and spaced logarithmically.

$F(E_i)(t_s)$

Object	F	E_i	t_j
Name	Omni-flux	Energy	Epoch
Frame	scalar>na	scalar>na	scalar>na
UNITS	# cmˆ-2 sˆ-1	keV	msec
SI_conversion	1.0e4> ()mˆ-2 sˆ-1	1.234e5>J	1.0e-3>s
Size	N_{pts}	N_{lev}	N_{pts}
Values	F_{ij}	E_i	t_s
Bin_Boundaries	n/a	0.0>touching>log	centred>touching>linear
Bin_Volume	n/a	end - start	end - start

A.3 Pitch-Angle Distributions

Consider a time series of N_{pts} measurements of phase space pitch angle distributions $p(E,\theta,t)$. These are held as a sequence of 2-D array of the form $p(E_i,\theta_j)(t_s)$. The energies are of fixed logarithmic width around their centre values, whereas the θ bins have start and end values supplied.

$p(E_i,\theta_j)(t_s)$

Object	p	E_i	θ_j	t_s
Name	PAngle Dist	Energy	Pitch Angle	Epoch
Frame	scalar>na	scalar>na	scalar>na	scalar>na
UNITS	# kmˆ-6 sˆ3	keV	deg	msec
SI_conversion	1.0e-9>()mˆ6 sˆ3	1.234e5>J	1.0>deg	1.0e-3>s
Size	N_{pts}	N_{lev}	N_{pa}	N_{pts}
Values	p_{ijk}	E_i	$(\theta_j^{start},\theta_j^{end})$	t_s
Bin_Boundaries	n/a	0.5>0.3>log	explicit	c>t>lin
Bin_Volume	n/a	1.1e30(end - start)	cos(start)-cos(end)	end - start

A.4 Phase Space Density

Consider a time series of N_{pts} measurements of a phase space distribution $f(\vec{v})(t)$. These are held as a sequence of 3-D arrays of the form $f(v_i,\theta_j,\phi_k)(t_s)$. The speeds v_i are bins of varying width with start and stop values supplied, the polar angles θ_j and azimuthal angles ϕ_k are both linearly spaced touching bins given by their start values.

$f(v_i,\theta_j,\phi_k)(t_s)$

Object	f	v_i	θ_j	ϕ_k	t_s
Name	PSD	Speed	Polar Angle	Az. Angle	Epoch
Frame	scalar>na	scalar>na	scalar>na	scalar>na	scalar>na
UNITS	# kmˆ-6 sˆ3	km/s	deg	deg	msec
SI_conversion	1.0e-9>()mˆ6 sˆ3	1.0e3>msˆ-1	1.0>deg	1.0>deg	1.0e-3>s
Size	N_{pts}	N_{lev}	N_{pol}	N_{az}	N_{pts}
Values	f_{ijk}	(v_i^{start},v_i^{end})	θ_j	ϕ_k	t_s
Bin_Boundaries	n/a	explicit	0.0>t>lin	0.0>t>lin	c>t>lin
Bin_Volume	n/a	r>sphere	pol>sphere	az>sphere	end - start

A.5 Rotation Matrix

Consider a rotation matrix from GSE to GSM coordinates. This is held as a 2-D array of the form a_{ij}. In fact, the rows of a are the GSM-unit vectors expressed in the GSE system. Missing from the example

is the fact that it rotates vectors into the GSM frame, i.e., that it is a rotation matrix in addition to being a tensor in the GSE frame.

$a(comp_i, comp_j)$

Object	a	$comp_i$	$comp_j$
Name	Rot. Matrix	Row	Column
Frame	tensor>gse_xyz	index>gse_xyz	index>gse_xyz
UNITS	()	()	()
SI_conversion	1.0>()	1.0>()	1.0>()
Size	1	3	3
Values	a_{ij}	$(1,2,3)$	$(1,2,3)$
Bin_Boundaries	n/a	n/a	n/a
Bin_Volume	n/a	n/a	n/a