Manipulation of Time Series Data Objects

Steven J. Schwartz
Astronomy Unit
Queen Mary & Westfield College
Mile End Road
London E1 4NS
United Kingdom
e-mail: S.J.Schwartz@qmw.ac.uk

Document Status Sheet			
1. Document Title: **Manipulation of Time Series Data Objects**			
2. Document Reference Number: **CU-QMW-TN-0007**			
3. Issue	4. Revision	5. Date	6. Reason for Change
Draft	0	22 June 95	First Draft
Draft	1	11 August 95	Changed title to make generic time series document. Added first section on Time series Data
Draft	2	17 August 95	Added figs, removed join tables. Numerous implementation additions.
1	0	15 September 95	Changed "Half Interval" to "Half Sample Period" to avoid confusion with CSDS/ISTP Half-interval.
1	1	13 February 96	Added Sample Spacing algorithm

Contents

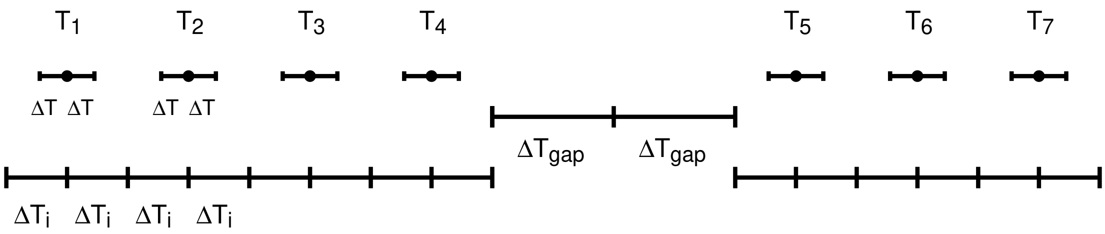

Figure 1: Illustration of various time tag-related parameters

1 Background

Cluster data will almost entirely be in the form of time series, and the many operations required to analyse the data will involve manipulations of such time series. Accordingly, we set out in this document the basic terminology and functions which will be required. This document will grow as new functions are required.

2 Time Series Data

2.1 Nomenclature

A time series data object is a set of values for a quantity X associated with time T, i.e., the set (T_i, X_i). The data object will also include metadata relating to both X and T, such as the units for X, the sample interval for T, etc. While these general notions are commonplace, we begin here by recording a variety of definitions and terminology. Figure 1 illustrates most of the time-related concepts and definitions.

Time Tags T_i The times T_i are known as time tags. Each T_i tags a particular value X_i with a single time to which that value is associated. The time tag could be the start of the time interval over which the measurement resulting in the value X_i was made, the middle time, or some other time. As the Cluster Prime and Summary Parameters all use centred time tags, we shall assume that the time tag is the centre of the time interval over which the value is valid. The format and units of the T_i must be established.

Half-Spacing ΔT_i The half-spacing is simply half the *nominal* interval between successive time tags. It is nominal in the sense that a single value is assumed to apply across the entire time interval of the time series. This half-spacing is required in order to distinguish genuine data gaps (see below) from regular but sparsely sampled data. For Cluster Prime and Summary Parameters, this is held in the variable Half-Interval.

Half Sample Period ΔT The half sample period ΔT gives the time period on either side of the time tag T_i over which the value of X_i is valid; i.e., X_i applies to the time interval $T_i - \Delta T \leq t \leq T_i + \Delta T$. The half sample period may be related to either the half-spacing ΔT_i of the time tags or an averaging/integration interval used in determining/measuring X_i. Note in particular that ΔT can be less than ΔT_i as illustrated in Figure 1, e.g., if the quantity is not continuously sampled, or greater than ΔT_i, e.g., when a sliding boxcar average is performed.

Data Gaps A data gap is a time interval within which a time-tag and datum is expected, based on the surrounding time tags T_i, knowledge of the half-spacing ΔT_i, and a tolerance threshold, but for which there is no entry in the time series. The width of a gap is $2\Delta T_{gap} = T_{i+1} - T_i - 2\Delta T_i$. It is current QSAS strategy NOT to include fill values or gap markers in the time series, so analysis and plotting functions which require knowledge of data gaps will need to identify them internally, based on the above definition and their own tolerance specification.

Datum X_i The X_i's represent values of a particular data quantity, which could be a scalar, vector, array or other form/collection of values. In this document, we shall represent all such quantities by a single X_i, rather than any explicit naming of components thereof. Some operations, e.g., frame rotation, require specific types of data object. Each datum has associated with it metadata, including units, conversion to base SI values, etc.

2.2 Determining Half-Spacing ΔT_i

The identification and treatment of data gaps is an important part of any time series analysis. CSDS variables have an associated variable called "Half Interval" which can be intepreted as both the Half-Spacing ΔT_i *and* the Half Sample Period ΔT. For the purposes of time series analysis, data gaps are times for which a datum X_i is expected but for which none is present in the file. Thus identifying data gaps requires knowledge of the Half-Spacing ΔT_i. Although the CSDS CDF files include a Global Attribute called Data Intervals, which indicates regions of continuous coverage, it would be unwise to rely on this information. Additionally, it is record based, and would thus require careful renumbering of data records when a subset of a data file is read. Moreover, QSAS will need to import other data for which the Half-Spacing is completely unknown.

It is possible to construct an algorithm to determine the Half-Spacing in a set of time series data provided that there are sufficient consecutive elements within the data set. The simplest approach, assuming the data are completely regular (i.e., they have identical spacings) is to determine the *minimum* spacing between successive values. In practice, some allowance must be made for (near) duplicate entries (which would give an extremely small value) and for nearly regular data (i.e., similar but not identical spacings).

The algorithm used within QSAS therefore requires two quantities:

$\Delta T_{idiscard}$ Half-Spacings less than this value are rejected.

Tol Multiplicative tolerance (> 1) within which two spacings in the data will be regarded as a match.

Of course, at least 3 data points are required to determine ΔT_i. Provided this is the case, the algorithm finds the minimum half-spacing which exceeds $\Delta T_{idiscard}$, and counts the number of half-spacings which are within a factor of Tol of the minimum half-spacing found. If less than an acceptable fraction (e.g., 1%) of the half-spacings are included in this count, the algorithm is repeated, setting $\Delta T_{idiscard}$ to the minimum half-spacing. The algorithm repeats until a suitable minimum half-spacing is found, or until the number of iterations exceeds a preset number. Thus the algorithm is:

1. If number of data values < 3, Return Error.
2. Let $\Delta T_i = \min$ (Half-Spacings which are $> \Delta T_{idiscard}$)
3. If number of half-spacings in range $(\Delta T_i/Tol, Tol \times \Delta T_i)$ is more than 1% of the total number of values, Return ΔT_i.
4. Else set $\Delta T_{idiscard} = \Delta T_i$ and Repeat
5. If number of iterations exceeds maximum, Return Error.

Typical values are $\Delta T_{idiscard} = 0$, $Tol \ = 1.1 - 1.5$, maximum number of iterations $= 5$. These values cope adequately with typical satellite data for the purposes of locating real data gaps and for determining the Half-Spacing of such data. For other purposes, it may be necessary to increase the number of iterations and decrease the tolerance. Note in particular that the above algorithm as stated always returns the lower bound on the Half-Spacing.

3 Joining Time Series Data

The term "Join" comes from the database notion of generating from two tables a single table by associating entries from a column or columns of each table for records with the same value in one of their common "key-field" columns. In the case where the "key-field" is a floating point number (e.g., time), joining such data usually requires an algorithm to interpolate or otherwise generate pairs in the new table (see BJ Read & MA Hapgood, 'Approximate Joins in Scientific Databases in Practice,' RAL 91347PRO0003 presented at the Sixth Interantional Working Conference on Scientific and Statistical Database Management, Switzerland, 9-11 June 1992). An "outer" join simply matches identical time tagged records and interleaves records with non-identical times from each table, with "missing values" inserted in the data columns as necessary, and will not be pursued further here.

In terms of the QMW Science Analysis System for Cluster, the purpose of joining two or more time series data objects is to generate a new set of data objects with identical time tags for subsequent analysis or plotting, e.g., algebraic combinations of data products or $x - y$ plots. It will be important to preserve the data object properties (scalar/vector/etc., metadata, ...) suitably modified as appropriate. Moreover, the result will not be a single simple table, but a collection of joined data objects. In other respects, the database concepts of join apply.

The join process can be broken down into three steps (although it need not be implemented in this manner):

1. Define the time tags t_j for the resulting data objects. These could come from one of the input time series, modified if necessary, another time series, or generated independently.

2. For each input time series (T_i, X_i) assign values χ_j to the times t_j. This amounts to interpolation of some sort.

3. Formally join the resulting data objects. This step is needed to ensure that the resulting data objects have the same subset of time tags t_j. Subsetting may be forced by the treatment of data gaps or missing values.

Each of these steps will be discussed below.

3.1 The Time Tags t_j

The time tags t_j are generated by one of:

1. taking them from a time series data object (T_i, X_i), typically but not necessarily one of the input time series, for either the whole range of the time series or a selected sub-interval.

2. as above but filling in data gaps less than some maximal value ΔT_{max}, with, e.g., uniformly space tags of separation $\sim \Delta T_i$ adjusted to produce commensurate intervals at either end of the gap.

3. generating uniformly spaced tags specified by a start time t_{start}, end time t_{end}, and interval $2\Delta t$ (Δt is the "half spacing"). Tags start precisely at t_{start}, use increments of $2\Delta t$, and end at the last tag generated with $t_i < t_{end}$.

Other possibilities, e.g., the union of tags from two or more time series, non-uniformly space tags (e.g., logarithmic), etc., could be imagined but are probably not required at present. Figures 2 and 3 illustrate the inputs and options required to generate the time tags.

3.2 Interpolation

3.2.1 Concepts

Formally, we interpolate a time series (T_i, X_i) onto a set of time tags (t_j) to produce a set of values for X at times t_j via one of several algorithms. Each algorithm may in turn have additional inputs or options. The default algorithm may depend on the half-spacings ΔT_i and Δt of the two sets of time tags, and also possibly on the the half-sample period ΔT

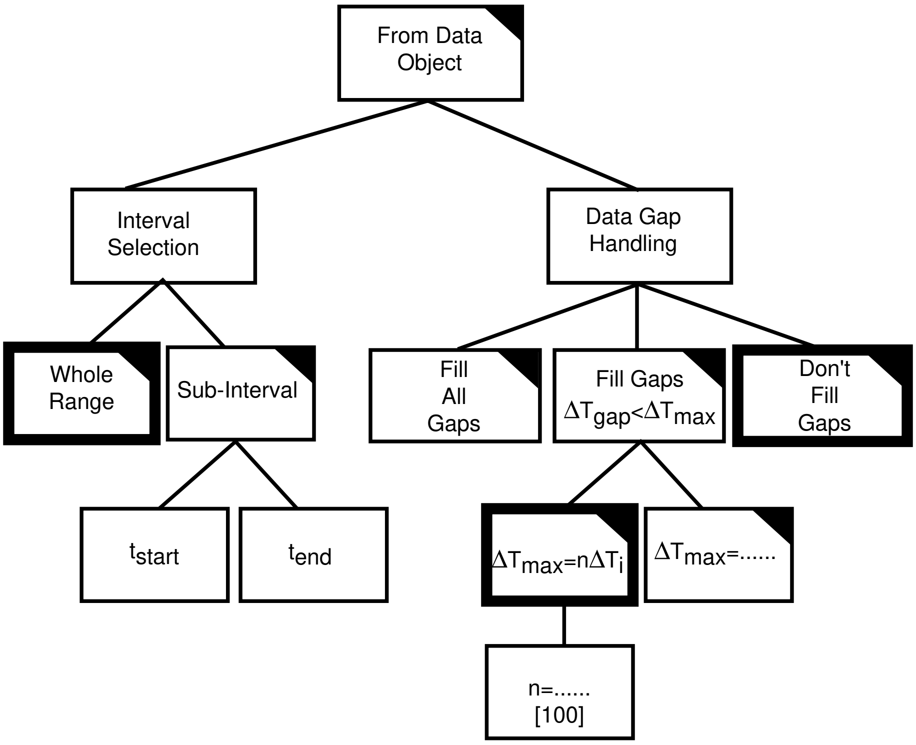

Figure 2: Inputs and options to generate time tags from a time series.

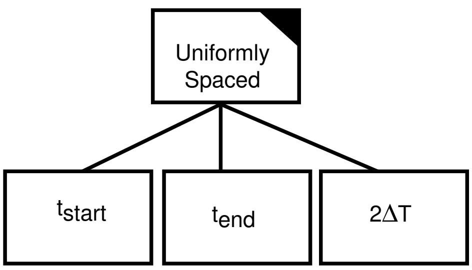

Figure 3: Inputs and options to generate a uniform set of time tags.

of the interval for which the measurement is valid if this differs significantly from ΔT_i. Since new objects are being produced, their measurement validity interval is probably uncertain and thus Δt is simply the half-spacing of the t_j's. The basic notion is

$$\left. \begin{array}{c} (t_j) \\ (T_i, X_i) \end{array} \right\} \rightarrow (t_j, \chi_j) \tag{1}$$

where the χ_j's are the interpolated values of the X_i's. There are at least four possible interpolation algorithms, as described below. Structure diagrams of inputs and options, incorporating implementation decisions, are given in Section 3.4.

3.2.2 Algorithms

1. **Nearest Neighbour**

   These are often described as "join" algorithms, but it is useful to put them alongside other more traditional interpolation algorithms. See Figures 5 and 8 for inputs and options required to implement these algorithms.

   (a) Simple
   Pick the time T_i which is nearest to t_j and use the value X_i, i.e.,

   $$\chi_j = X_i \tag{2}$$

   where i minimizes $|T_i - t_j|$.

   (b) Fuzzy
   As in "simple" but only accept if $|T_i - t_j| < \Delta T_{tol}$ where ΔT_{tol} is the tolerance allowed. Typically $\Delta T_{tol} = \Delta T + \Delta t$, i.e., the interval over which the measurement is valid must overlap the output time tag spacing. That is $T_i \pm \Delta T$ must overlap $t_j \pm \Delta t$. Suitable recovery needs to be provided if this criterion is not satisfied, probably by treating it as a data gap (see below). Note that if the original time series measurements are valid for only a portion of the original time tag spacing, i.e., $2\Delta T < T_{i+1} - T_i$, this fuzzy algorithm can fail to find a valid χ_j even if there is not data gap in the original time series. Another logical choice for ΔT_{tol} is therefore $\Delta T_{tol} = \Delta T_i + \Delta t$, which fails only at genuine data gaps.

2. **Linear Interpolation**

   Find i such that $T_i \leq t_j \leq T_{i+1}$. Then

   $$\chi_j = X_i + \left(\frac{t_j - T_i}{T_{i+1} - T_i} \right) (X_{i+1} - X_i) \tag{3}$$

   See Figure 10 for the inputs and options required to implement this algorithm.

3. **(Cubic) Spline**

   As above but use cubic spline algorithm, presumably with "natural" end conditions, in place of Equation 3. See Figure 12 for the inputs and options required to implement this algorithm.

4. **Boxcar**

   Define a "boxcar" half-width ΔT_{box}. Then

   $$\chi_j =< X_i >_{box} \tag{4}$$

   where $< \ldots >_{box}$ denotes an average over all i's satisfying

   $$t_j - \Delta T_{box} \leq T_i \leq t_j + \Delta T_{box}$$

   See Figure 14 for the inputs and options required to implement this algorithm. This fails if $\Delta T_{box} < \Delta T$ but can be useful if $\Delta T \ll \Delta t$, i.e., if the original time series is at higher time resolution than the resulting time tags. Again, failure can be treated as for a data gap (see below). Small data gaps, i.e., those not resulting in failure, can either be ignored and Equation 4 applied, or the missing T_i's can be calculated, the corresponding X_i's calculated by another algorithm, and then Equation 4 applied. See Figure 15 for the inputs and options required in this case.

3.2.3 Data Gaps/Missing Values

The interpolation process will need to trap and deal with data gaps or missing values (if any). There may be a question/option of defining a data gap. The width of a gap is $2\Delta T_{gap} = T_{i+1} - T_i - 2\Delta T_i$. Once a gap is encountered, there are several options for the interpolation algorithm to deal with it:

1. Apply the chosen interpolation to fill the gap (i.e., ignore the fact that it is a gap). This will not help for fuzzy interpolation which is outside ΔT_{tol}. Similarly it will not help for boxcar interpolation if the gap is too wide, i.e., $T_i + \Delta T_i + \Delta T_{box} < t_j$ and $T_{i+1} - \Delta T_i - \Delta T_{box} > t_j$. This can happen for gap widths $2\Delta T_{gap}$ as small as the boxcar width $2\Delta T_{box}$ and will certainly occur if the the gap is more than twice this wide. In these cases, the algorithm has already failed, or will fail, to find a suitable value.

2. Apply a *different* algorithm to fill in the gap, e.g., linearly interpolate across the gap where in contiguous data a spline is used.

3. Remove (or fill with missing values) the entries (t_j, χ_j) for t_j lying within a gap in the T_i's.

4. Apply the above differently for small and large gaps by setting a threshold $\Delta T_{max-gap}$. For example, linearly interpolate across small gaps and remove entries corresponding to large ones. Typically $\Delta T_{max-gap}$ might be related to one of the time tag

half-spacing ΔT_i, the measurement half-sample period ΔT (if different), or the output time tag spacing Δt.

In specifying data gap handling, there is the possibility of looping (i.e., choosing an alternative algorithm which fails, then another, etc.). In the implementations shown in Section 3.4 this is eliminated by restricting the first alternative algorithm to fill the gap itself or leave a gap.

3.3 The Join

Once the individual time series are interpolated to the same time tags t_j, the only remaining step is to ensure that all resulting data objects have exactly the same tags. Differences at this stage would arise if entries corresponding to a time tag t_j were removed from one data object, e.g, due to a data gap. Entries with this time tag must be removed from all the resulting time series (t_j, χ_j) which are part of the joined set of data objects. Whether this requires a real operation depends on the implementation of the interpolation. Note, however, that it would be perfectly reasonable for different interpolation algorithms to be applied to different input time series. For example, a boxcar interpolation might be applied to a high(ish) resolution product while a spline would be used on a lower resolution one. Other factors dictating the choice of algorithm include the the noise/smoothness of the data in a particular time series, the manner in which the joined data will subsequently be manipulated, etc.

3.4 Summary of Inputs, Options, and Outputs

The structure diagrams in Figures 4-17 below summarise the inputs and options for joining time series, and include implicit implementation decisions. The inputs are specified by following boxes down a particular branch level by level. Adjacent boxes marked by a black triangle are exclusive choices. Default choices are indicated by a heavy border. Default numerical values are enclosed in square brackets.

3.5 Other Issues

A variety of questions arise out of this subject, many concerned with the implementation aspects. Amongst these are:

1. In many cases, some of the input time series will share time tags T_i with each other (e.g., n_e, T_e). It would be more efficient to interpolate these objects at the same time, thereby avoiding duplication of searches to find the T_i nearest t_j, the calculation of the fractional time factor in Equation 3, etc. Such efficiency needs to be weighed against the complexity of identifying such cases, and generalizing the interpolation algorithms to cope with arbitrary numbers of data objects simultaneously.

2. Vector (and higher dimensional object) properties *must* be preserved, either by applying any algorithms simultaneously to all components or else by decomposing and then re-composing the separate components.

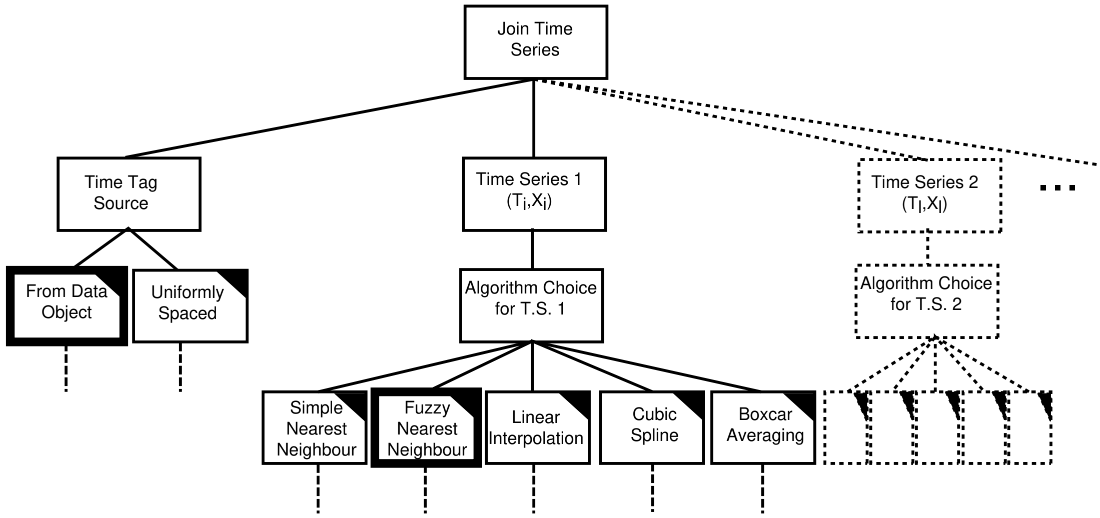

Figure 4: Overview of inputs and options for joining time series.

3. When a single vector cartesian component is required, it can be treated as a scalar. However, when an angular coordinate is required (e.g., θ), the end user should derive it from interpolated cartesian components rather than interpolating the angle coordinate as a scalar. [Note that $\theta(\vec{B}) = \cos^{-1} B_z/B$ so that $< \theta(\vec{B}) > \neq \theta(< \vec{B} >)$.] Nonetheless, there may be good reason in some instances for the user to interpolate θ as a scalar. QSAS should not, however, force the choice, but rather treat a vector as a vector. It is up to the user to first decompose it before passing a derived quantity to the interpolation.

4. Current plans for the QMW Science Analysis System systematically remove missing values, so that all such values appear as data gaps in the data objects. Reference here to missing values is therefore only for completeness. It *might* be worth filling entries for t_j's with an internal missing value at the interpolation stage, so that the final join step could then step in sequence through all the data objects with the same index, looking for entries to be removed from all data objects. This is left to the implementation phase.

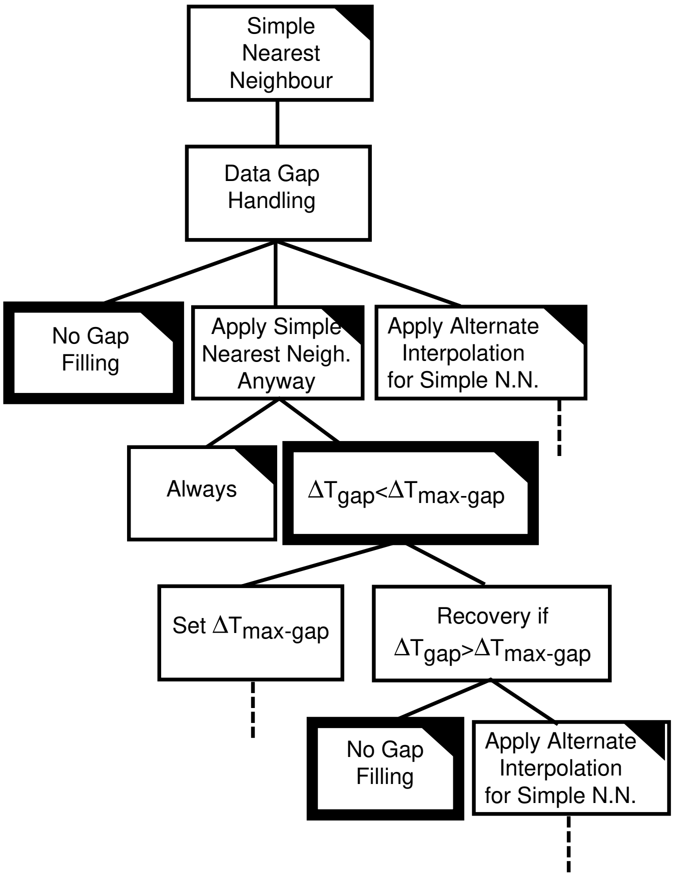

Figure 5: Inputs and options required for a Simple Nearest Neighbour Interpolation. The alternative algorithm referred here and required in the case of data gaps is shown in Figure 7. See also Figure 6 concerning the setting of a maximum data gap.

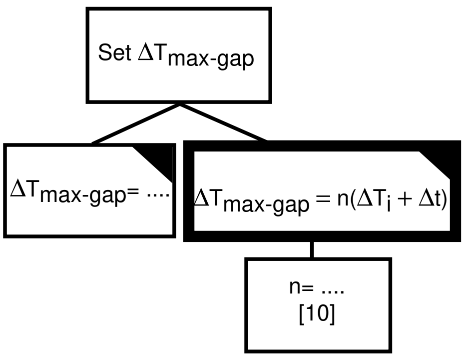

Figure 6: Inputs and options to set a maximum data gap half-width.

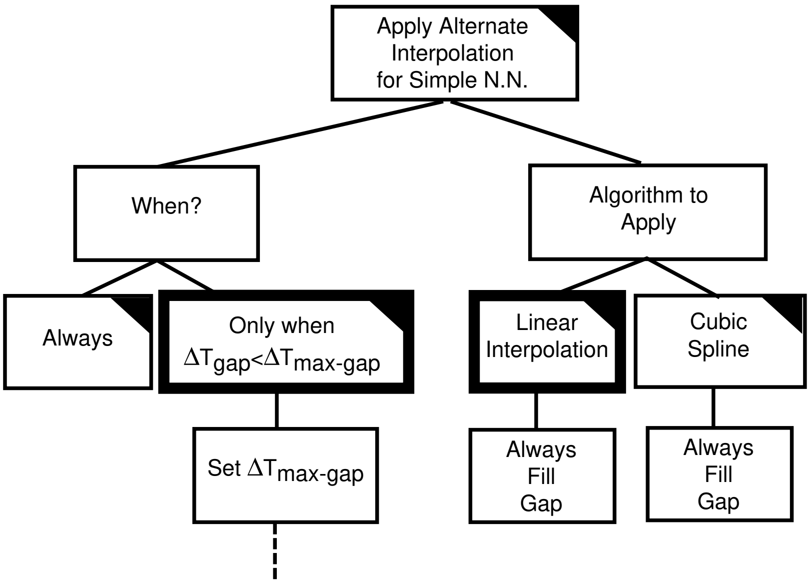

Figure 7: Alternate Interpolation for the Simple Nearest Neighbour algorithm.

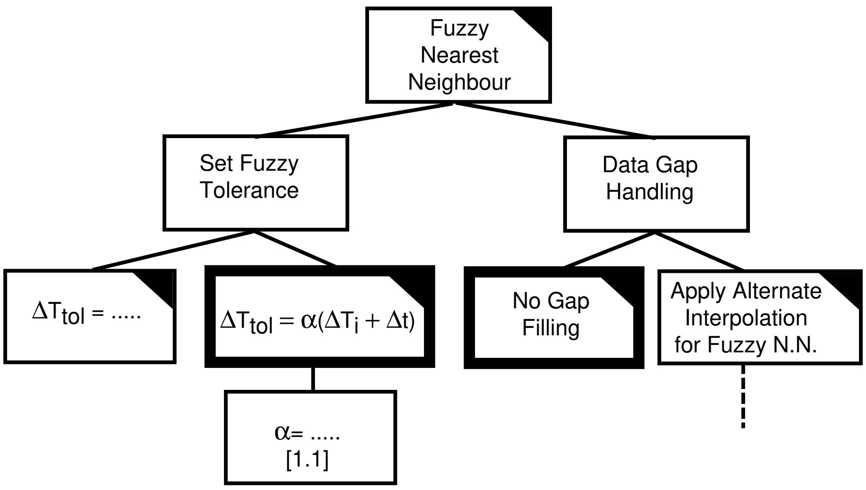

Figure 8: Inputs and options for the Fuzzy Nearest Neighbour Algorithm. See Figure 9 forthe alternate interpolation.

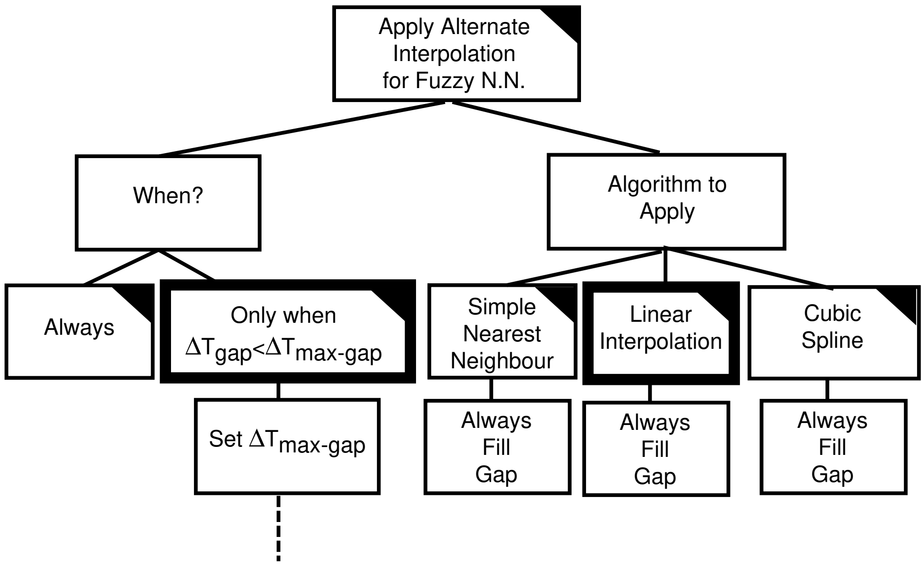

Figure 9: Alternate Interpolation for the Fuzzy Nearest Neighbour algorithm.

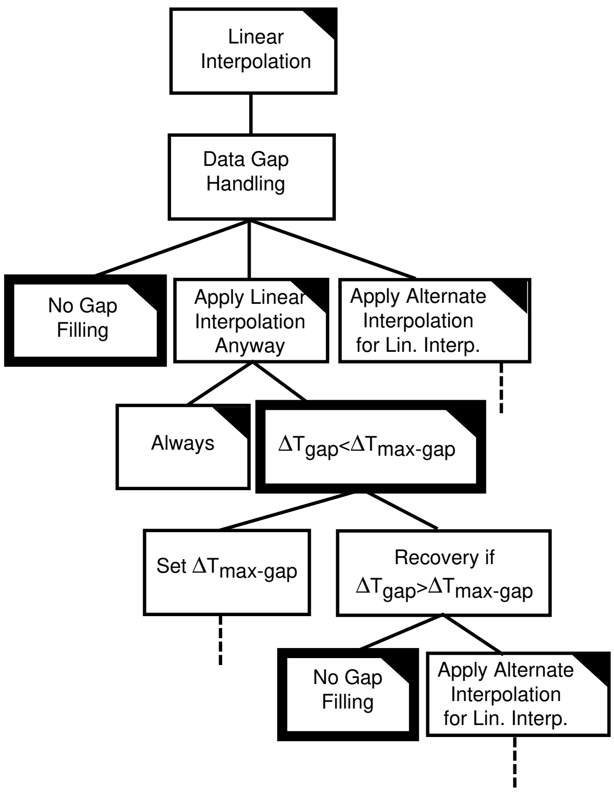

Figure 10: Inputs and options for the Linear Interpolation Algorithm. See Figure 11 for the alternate interpolation.

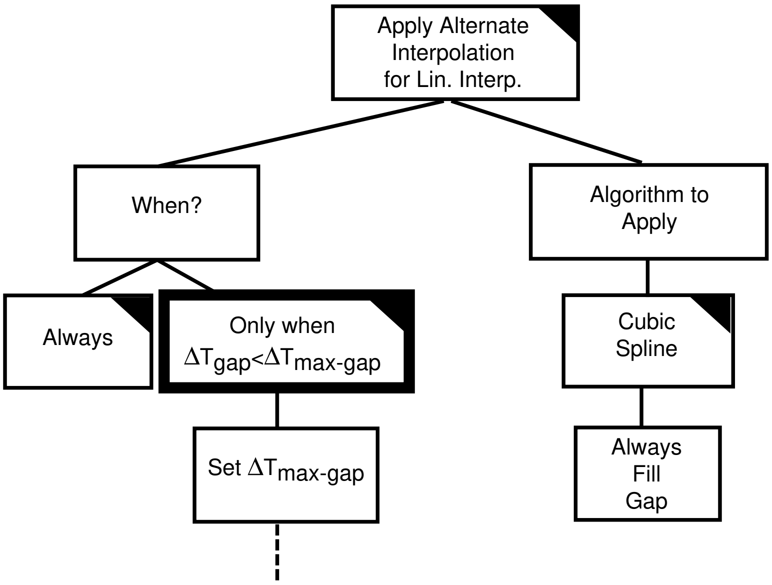

Figure 11: Alternate Interpolation for the Linear Interpolation Algorithm.

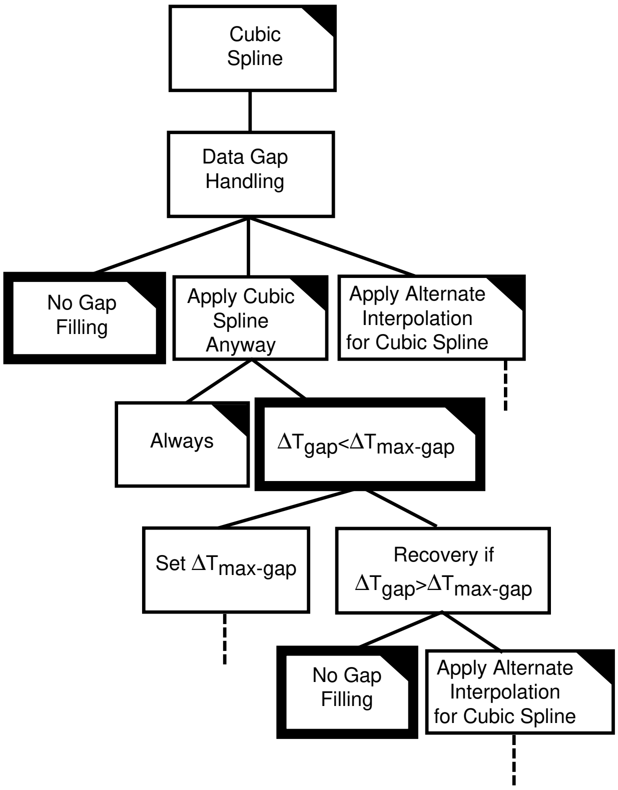

Figure 12: Inputs and options for the Cubic Spline Interpolation Algorithm. See Figure 13 for the alternate interpolation.

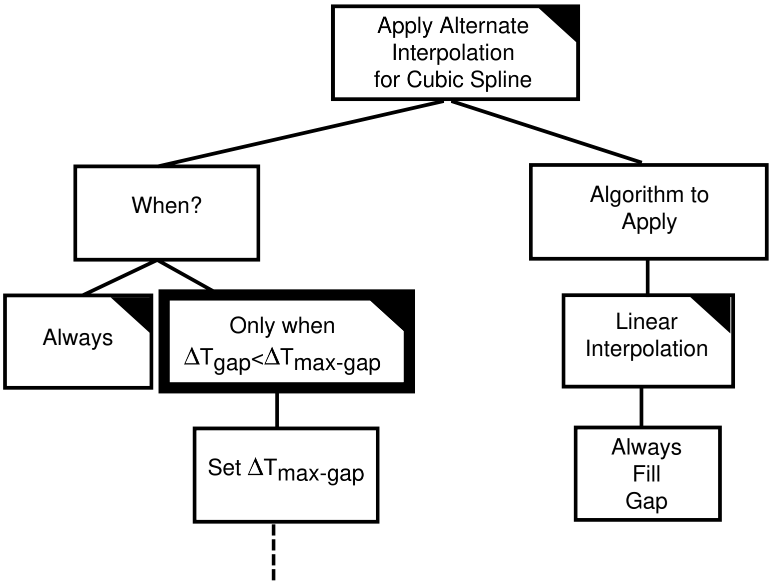

Figure 13: Alternate Interpolation for the Cubic Spline Interpolation Algorithm.

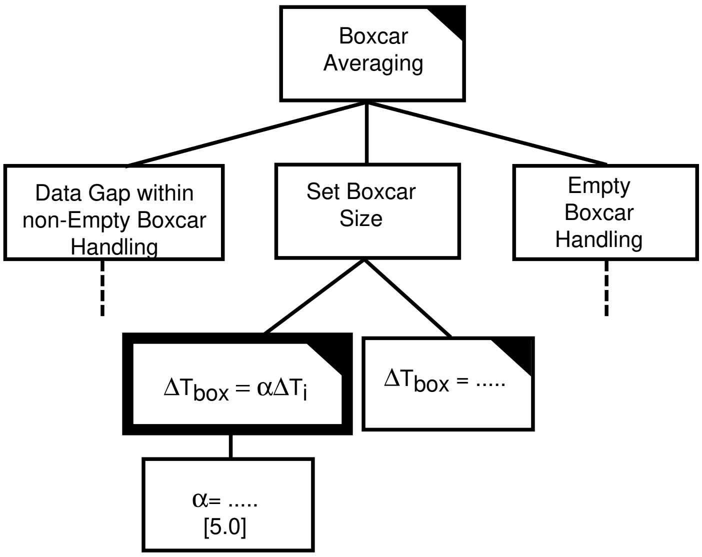

Figure 14: Inputs and options for the Boxcar Averaging Algorithm. See Figures 15, 16, 17, and for continuation.

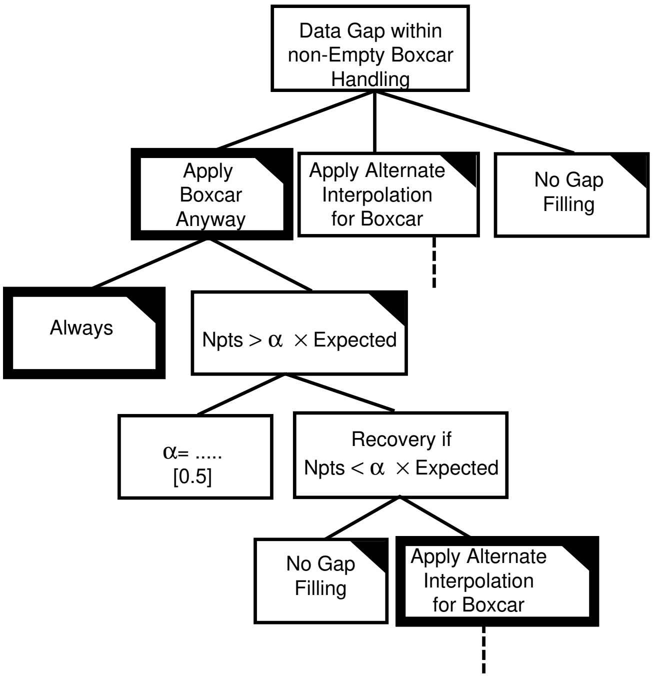

Figure 15: Data Gap Handling for the Boxcar Averaging Algorithm in the case of a Non-empty boxcar.

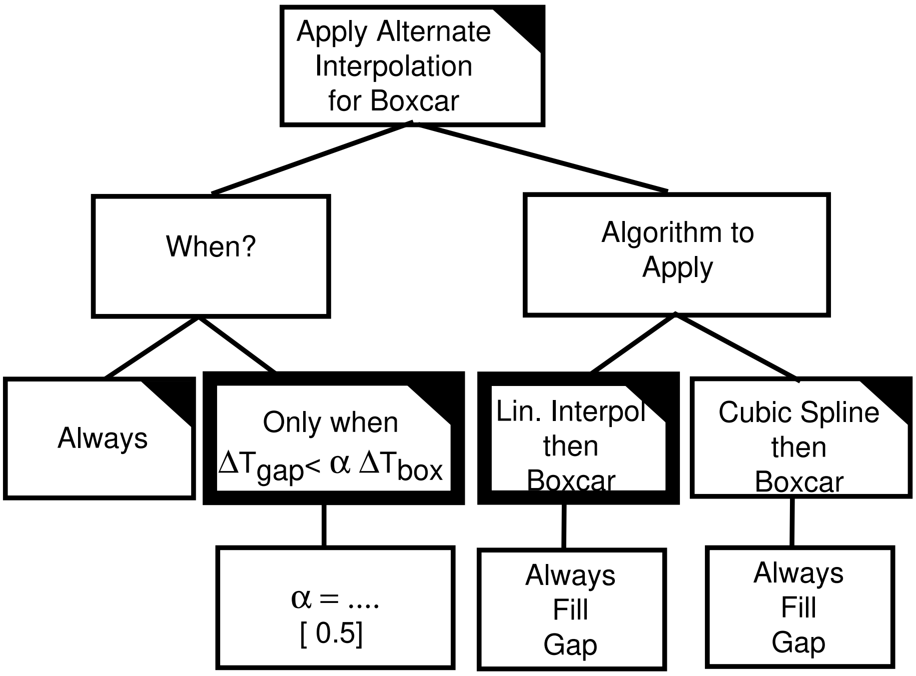

Figure 16: Alternate algorithms to be used in data gap handling for non-empty boxcars.

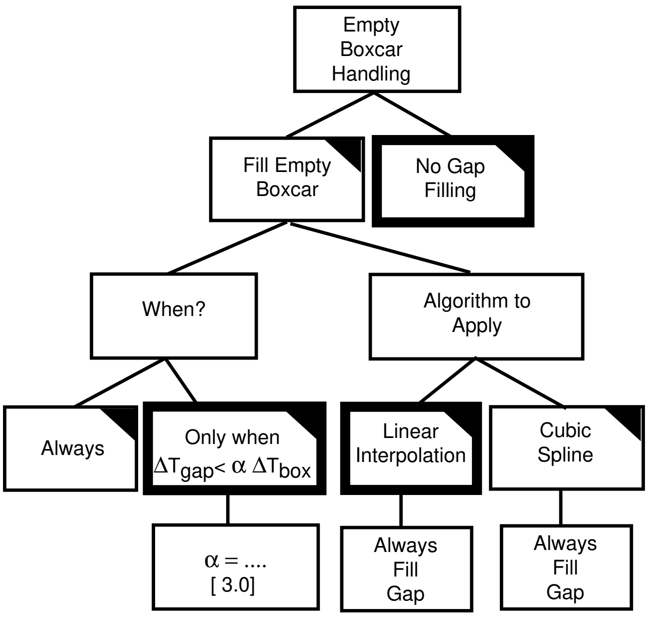

Figure 17: Data Gap Handling for the Boxcar Averaging Algorithm in the case of an Empty boxcar.