

THE KNOWLEDGBASE INTERFACE FOR PARAMETRIC GRID INFORMATION

James R. Hipp, Chris J. Young, Randall W. Simons, Sandia National Laboratories

Sponsored by U.S. Department of Energy
Office of Nonproliferation and National Security
Office of Research and Development
Contract No. DE-AC04-94AL85000

ABSTRACT

The parametric grid capability of the Knowledge Base (KBase) provides an efficient, robust way to store and access interpolatable information that is needed to monitor the Comprehensive Nuclear Test Ban Treaty. To meet both the accuracy and performance requirements of operational monitoring systems, we use an approach which combines the error estimation of kriging with the speed and robustness of Natural Neighbor Interpolation. The method involves three basic steps: data preparation, data storage, and data access. In past presentations we have discussed in detail the first step. In this paper we focus on the latter two, describing in detail the type of information which must be stored and the interface used to retrieve parametric grid data from the Knowledge Base.

Once data have been properly prepared, the information (tessellation and associated value surfaces) needed to support the interface functionality, can be entered into the KBase. The primary types of parametric grid data that must be stored include: (1) generic header information; (2) base model, station, and phase names and associated ID's used to construct surface identifiers; (3) surface accounting information; (4) tessellation accounting information; (5) mesh data for each tessellation; (6) correction data defined for each surface at each node of the surfaces owning tessellation; (7) mesh refinement calculation set-up and flag information; and (8) kriging calculation set-up and flag information. The eight data components not only represent the results of the data preparation process but also include all required input information for several population tools that would enable the complete regeneration of the data results if that should be necessary.

Data retrieval and subsequent client processing are handled by a library of C++ KBase interface routines known as libKBI_grid (library Knowledge Base Interface for gridded information). Specific function calls can be inserted into client source code to make requests from the libKBI_grid API during execution. The libKBI_grid functionality can be divided into six specific tasks including: (1) database connection, (2) surface fetch initialization, (3) data cluster fetch size initialization, (4) surface association, (5) location initialization, and (6) surface interpolation.

Key Words: Knowledge Base interface, parametric grid, tessellation, surface

OBJECTIVE

To improve the U.S. National Data Center (USNDC) capability to monitor a Comprehensive Test Ban Treaty (CTBT), researchers at the U.S. Department of Energy (DOE) national laboratories are collaborating to develop a collection of data and associated access tools collectively known as the Knowledge Base. The Knowledge Base (hereafter KBase) will consist of many types of information which will improve the performance of the four principle monitoring technologies -- seismic, hydroacoustic, infrasonic, and radionucleic -- at various regions of interest around the world. The number of different data sets in the KBase is expected to be huge, but viewed from an information representation perspective, there are only four basic types of data: reference event information, parametric grid information, contextual information, and supporting information (Shepherd et al., 1998). In this report parametric grid information is the primary focus.

Parametric grid information is interpolatable data associated with particular geographic locations (i.e. a grid) on the earth. The interpolation part of the definition is important: not all data sets which have geographic coordinates should be interpolated. Examples of proper, interpolatable parametric grid data sets are travel-time corrections for any of the sensor technologies. The grid of locations for which information has been specified may be regular or irregular, depending on the method of collection or generation of data. Regardless, the KBase information derived from these grids must meet certain requirements, i.e. the interpolated values must: 1) include a robust error estimate; 2) be first and second order continuous; and 3) be delivered to applications "quickly" (the precise speed requirement depends on the application).

In previous papers we discussed methods of preparing parametric grid data including the mesh refinement methodology (Hipp 98), for obtaining rigorous representations of the data, and methods for interpolating the data quickly and accurately (Hipp 97). In this paper we shall concentrate on the type of information that must be stored, and more importantly, on the description of the interface used to retrieve parametric grid data from the Knowledge Base.

RESEARCH ACCOMPLISHED

Stored Parametric Grid Information

After the parametric grid has been determined, which accurately represents the data value and modeling error surfaces, the information (tessellation and associated value surfaces) can be entered into the KBase. The primary types of parametric grid data that must be stored in a database regardless of format include:

1. **Generic header information including the date of creation, the version numbers of the kriging and mesh refinement software used to generate the data, and the maximum tessellation, basemodel, station and phase ID's used to construct the surface identifier hashing function;**
2. **Basemodel, Station, and Phase names and associated ID's used to construct data surface identifiers;**
3. **Surface accounting information including the number of surfaces, and for each surface, the surface id, the basemodel, station, and phase names from which the surface was formed, the back-ground modeling error used by the surface, and the tessellation id of the tessellation upon which the surface is represented;**
4. **Tessellation accounting information including the number of tessellations, and for each tessellation, the tessellation id, the number of nodes, triangles, and represented surfaces, and the tessellation's modified-gradient parameter flag;**
5. **Mesh data for each tessellation including a description of all nodes that comprise the tessellation and the connectivity of the tessellation in the form of**

triangle information. Each node is defined by an ID, a location that includes the longitude, latitude, and depth of the node, and a type flag that indicates specific node type information. Each triangle is defined by an ID, a set of three node ID's that are the triangle's vertices, and a type flag that gives additional triangle specific information;

6. **Correction data defined for each surface at each node of the surface's owning tessellation. This data includes a correction value, a longitudinal and latitudinal derivative, and a modeling error value (this is travel-time specific... other data types, such as hydro-acoustic or infrasound, may have different data storage requirements).**
7. **Mesh refinement calculation setup and flag information which is necessary to reliably regenerate the mesh data that is contained within the database; and**
8. **Kriging calculation setup and flag information which is necessary to completely regenerate the kriged surfaces from which the refined mesh was evaluated.**

In the list above the concept of a surface is introduced which refers to the actual stored data defined at all nodes of a specific tessellation for a unique identifying basemodel, station, and phase set. Each surface is owned by a single tessellation. That is, each surface only uses one tessellation's node-set to tie its values to. Surfaces are not multiply defined across many tessellations. However, it is possible to include more than one tessellation in the data storage. Each tessellation has its own node and connectivity (triangles) definition and owns some subset of the surfaces defined in the database.

In the next section we shall discuss how the application interface software (libKBI_grid) uses stored data to provide requesting client applications with interpolated results.

Data Access Methodology: Application Interface

Once data have been processed and stored, they are ready to be accessed and used by client applications. The access and use of the data consists of three primary steps: data retrieval from the KBase, containing triangle search via the walking triangle method, and the Natural-Neighbor Interpolation (NNI) calculation to return results. Each of these functional steps are executed by the libKBI_grid software which serves as the implementor of the Data Access process.

This section will begin by discussing briefly the functional interface of libKBI_grid. Subsequent sections will describe each of the three data access steps including data retrieval, which defines the methodology and format for extracting data from the KBase; the containing triangle search which refers to the searching method used by libKBI_grid to find the client requested interpolation point within the tessellation(s) connectivity structure; and, the NNI including the gradient-modification used to ensure that the NNI is differentiable at all node points.

libKBI Functionality

Parametric grid data retrieval and subsequent processing is handled by a library of C++ KBase interface routines known as libKBI_grid (library Knowledge Base Interface for Gridded Information). Specific function calls can be inserted into client source code to make requests from the libKBI_grid API during execution. The libKBI_grid functionality can be divided into six specific tasks including:

1. **database connection,**
2. **surface fetch initialization (SFI),**
3. **data cluster fetch size initialization,**
4. **surface association,**

5. **location initialization, and**
6. **surface interpolation**

The first task, database connection, is simply a call to establish a connection to the KBase through a database manager such as Oracle, SDE, or even a flat-file representation. This call is a standard request providing the names of the server, database, password, and other pertinent information required to make a proper connection.

The second task, surface fetch initialization (SFI), refers to the manner in which DA is restricted, or enveloped, to include only a subset of the data available in the KBase. DA is restricted to optimize data retrieval performance by minimizing the amount of data that must be retrieved from the KBase and stored in-core. This is generally accomplished by eliminating data (surfaces) from retrieval consideration if they are not necessary for the current problem defined by the client application accessing libKBI_grid. As discussed above, the concept of a surface, used here and throughout the remainder of this paper, refers to the data that comprise travel-times, magnitudes, amplitudes, etc., and their derivative and modeling error values for a specific basemodel, station and phase assignment. The concept of a surface, as well as the methodologies and software abstractions used to enable the data access restrictions are discussed in the next section on data retrieval.

The third task, data cluster fetch size initialization, defines another restriction on the amount of data from particular surfaces that can be loaded at one time. The data cluster is best viewed as a spatial cookie-cutter that only retrieves data from requested surfaces whose nodes lie within the boundaries of a pre-defined polygon (the cookie-cutter). This concept is also described in detail in the next section.

The fourth task, surface association, allows restriction of which surfaces are actually interpolated to a subset of those that were loaded based on the SFI prescription. Like the previous two tasks, this one is also described in more detail in the next section.

The fifth task, location initialization, simply sets the longitude and latitude of an interpolation request from the client application. Once libKBI_grid receives this information it is able to perform a containing triangle search which is prerequisite to calculating the interpolated values at the provided location.

Finally, the sixth task, surface interpolation, refers to the actual NNI calculation at the current location specified by the client.

As noted above, past reports have been written that document the formation and population of parametric grid data (Hipp 1996, and Hipp 1998) and provide examples of the use of the entire method for performing quality event location (Young 1998). In those reports the methods and techniques for performing the containing triangle search and natural-neighbor interpolation on a Delaunay tessellation (Delaunay, 1934) were discussed. The interested reader should consult Lawson (1977) or Sambridge (1995) for a discussion of the triangle-walk location algorithm or Watson (1992) or Sambridge (1995) for more information concerning the natural-neighbor interpolation technique. Those topics will not be discussed further in this paper. Here we will concentrate on data retrieval tasks only (tasks 2 through 4 above).

Data Retrieval

We have just seen that libKBI_grid handles all issues associated with retrieving data from the KBase. Specifically, it accomplishes the data retrieval by restricting the amount of information that is loaded in-core and by further restricting interpolation calculations on a surface-specific basis. The amount of data retrieved is restricted in two different ways. First, the available surfaces in the KBase are culled to a predefined subset as defined by the client application. Second, a spatial data cluster (cookie cutter) is used to further restrict the amount of data loaded to that which lies within the boundary of the data cluster. We shall describe each of these methods of restricting the accessed and processed data in the paragraphs below. But

first we need to define the abstract representation of a tessellation object to fully understand how libKBI_grid accesses the data and makes it available for use by a client application.

Tessellation And Surface Object Abstraction

We discovered above that the KBase, regardless of implementation, must store surface, node, and triangle information within the database. We also saw that the KBase was capable of storing many tessellations and that each tessellation, in turn, could have many surfaces assigned to it. Figure 1 illustrates this concept abstractly with an imaginary KBase that includes 4 tessellations. Each tessellation contains the nodes and triangles that define spatial location and connectivity information for that tessellation. In addition, each tessellation contains one or more surfaces that utilize the spatial information of the tessellation (the nodes) as locations to define its values on. For example, assume that “surface 1” is travel-time information which consists of a single travel-time value for each and every node defined within the tessellation. The assembly of all such values over their respective nodes forms a discrete surface. Using an interpolation function one can define a continuous interpolated surface over the entire range of the tessellation.

In our imaginary KBase of Figure 1 we see that “tessellation 1” has 4 surfaces, “tessellation 2” has 5 surfaces, “tessellation 3” has 2 surfaces, and “tessellation 4” has 3 surfaces. Notice that the index number of each surface is unique regardless of which tessellation owns the surface. This is an intended data requirement of the KBase. The KBase does not allow a surface to be represented by more than one tessellation. One can imagine the trouble that would undoubtedly occur if libKBI_grid tried to access the surface assigned to say basemodel ‘AK135’, station ‘CMAR’, and phase ‘P’ only to find that the surface index existed in two different tessellations. Which one would be used?

The relationship between a tessellation and the surfaces that it owns can be made clearer by examining Figure 2. Figure 2 shows constituent parts of “tessellation 4” of the imaginary KBase. The parts of a tessellation consist of geometry information (nodes and triangles) and value information (surfaces). Notice that all geometric spatial location information is provided by the nodes of the tessellation. A node is, quite simply, the latitude, longitude, and depth of some point on the earth. In and of itself it has no other attributes assigned to it. The triangles, on the other hand, are a means of tracking the mesh connectivity of a tessellation. A triangle is simply the indices of each of the three nodes that comprise the triangle.

The node indices are always ordered in a counter-clockwise fashion relative to an outward pointing normal on the surface of the Earth. Triangle connectivity provides a means for determining node neighbors and is vital for performing the containing triangle search and the node neighbor calculation both of which are required to perform NNI.

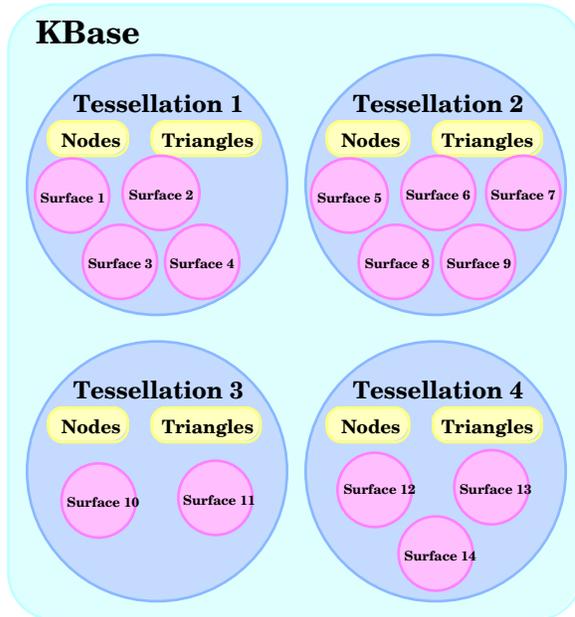


FIGURE 1. Abstraction of KBase storage for multiple tessellations

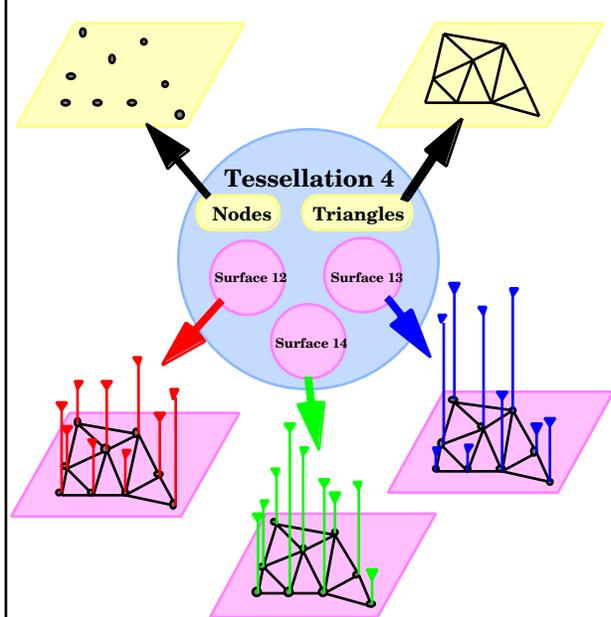


FIGURE 2. Abstraction of KBase tessellation components

The actual data values that are to be interpolated are contained within the surface object representations. In Figure 2 three surfaces are shown; 12, 13, and 14. Notice that all three use the tessellations’ nodes as the spatial location upon which their values are defined. The only difference between one surface and the next is the magnitude of the value at each of the tessellation’s fixed nodes and not its location.

Surface Fetch Initialization (SFI)

Now that we have examined the abstract nature of the tessellation object relative to how it is represented in libKBI_grid we are ready to investigate the three primary data retrieval tasks of libKBI_grid. Recall from above that the first data retrieval task following database connection was surface fetch initialization (SFI). The purpose of SFI is to define a specific set of surfaces that are defined as necessary and required by the calling client application to properly perform its intended function. By restricting the number of surfaces accessed by libKBI_grid to a subset of those available in the KBase a substantial performance gain may be noticed. Also, computer resources (memory) may be in short supply and minimizing the amount of in-core data can help to alleviate this problem.

Figure 3 depicts a surface fetch subset from the imaginary KBase. In the example presented in Figure 3 surfaces 1, 2, and 3 of “tessellation 1”, surfaces 5, 7, and 9 of “tessellation 2”, surface 10 of “tessellation 3”, and surface 14 of “tessellation 4” are designated for access while surfaces 4, 6, 8, 11, 12, and 13 of their respective tessellations are restricted from access. In this example we have reduced the number of surfaces required for access by almost 50%. In reality the number of surfaces may be in the thousands while the fetch

subset is generally in the single or double digits. Real problems can see savings of a factor of 10 to 100 in access performance.

As an example of the use of SFI, consider the location of a seismic event with a small subset of the station/phase combinations available in the KBase, e.g. only P phases for the stations at regional distances. Once the event is ready to be located, the relevant station/phase combinations are known, so SFI can be used to limit the scope of the queries to the KBase in the data access step and thereby improve the speed of the KBase access.

Data Cluster Access

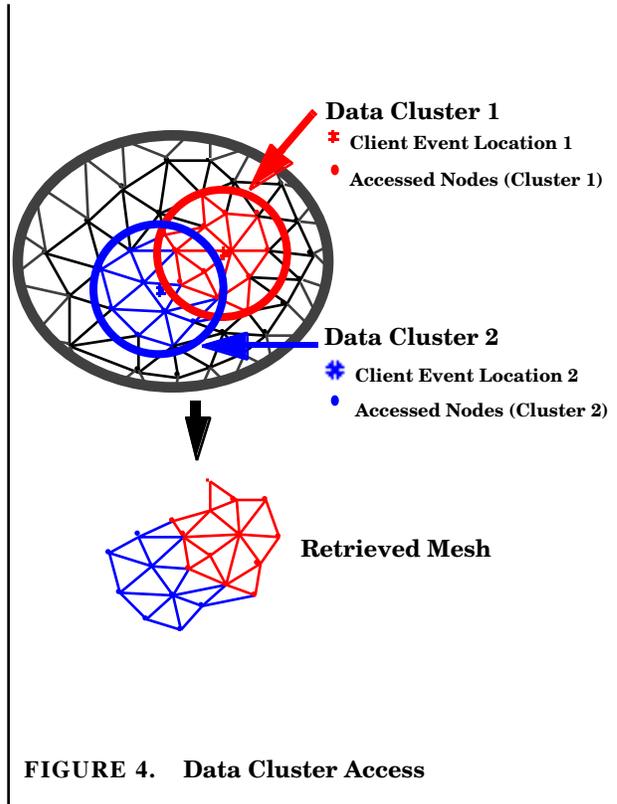
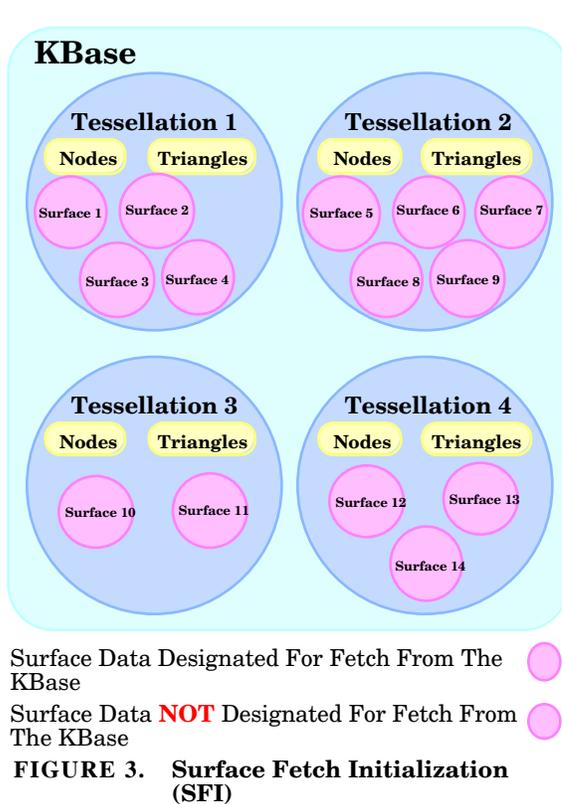
The next data retrieval task performed by the libKBI_grid API is data cluster fetch size initialization. Defining a data cluster fetch size is also a method of minimizing the amount of data that must be acquired and stored in-core. Recall that the data cluster was best viewed as a spatial cookie-cutter that only retrieves the requested surface data from nodes that lie within the confines of a pre-defined polygon (the cookie-cutter). The need for this type of functionality may provide the best reason to use a sophisticated commercial product for DS rather than a simple flat-file: for example, the ESRI product SDE, which operates on top of an ORACLE relational database, directly supports this type of retrieval.

Figure 4 depicts this concept where the data cluster is an imaginary circle centered over a region of interest for the client application. The figure shows two separate data clusters and the resulting nodes and triangles accessed by both clusters. Notice that the data cluster method is intelligent and only loads data that is the set difference between the current accessing cluster and all other clusters that were defined before it. For the example given in Figure 4, cluster 1 is used to access some data from a surface. At a later time cluster 2 is defined to access additional data. The data returned by cluster 2 is the set difference of the data between cluster 2 and cluster 1 (cluster 2 - cluster 1). This process is repeated every time a new cluster is designated, initialized, and its data are loaded. In this way only the difference in data between the current cluster and all previously loaded clusters are retrieved from the database which can substantially reduce the data throughput from the KBase to the libKBI_grid software. Additionally, minimizing the radius, or spatial range, of the data cluster to a size that is reasonable for the current problem can also significantly reduce the total amount of data transferred from the KBase to libKBI_grid.

When data are fetched using the data cluster, all surfaces defined in the SFI are accessed. Figure 5 depicts this process and shows (abstractly) how three subsequent data cluster accesses are made from all surfaces designated for access by the SFI task of the libKBI_grid software. By way of the imaginary example the total data retrieval from the KBase following the three data cluster accesses is the sum of the data contained in the union of the three data clusters as retrieved from each accessible surface.

Surface Association

The last task to be discussed concerning data retrieval is surface association. Surface association is a means of restricting which surfaces are actually interpolated from those that were loaded based on the SFI prescription. In many cases, the list of surfaces would be the same, but not always. For instance, in a batch mode event location run, it may be known up front that all events were detected by a given network, hence the SFI call can limit access to that network. However, for a given event, the particular subset of network stations that actually recorded may well be different. This means that libKBI_grid would have to load additional data (surfaces) beyond the initial SFI prescription.



The purpose of surface association is to define a specific set of surfaces for which results are required for the current event location being processed by the calling client application. This set is generally a subset of those defined in the SFI task but may include all surfaces defined by the SFI, and in some cases, even surfaces that were designated not to be accessed by the SFI. This could happen if a user simply forgot or left out important surfaces from their SFI specification. Under those circumstances libKBI_grid simply loads the new surface into its SFI specification and loads all necessary data for the new surface as determined by the union of all accessed data clusters.

Figure 6 shows an example of surface association using our imaginary KBase where surfaces 3, 7, 9, and 14 are designated as surfaces associated with the current event being processed by the client application. Given this specification only surfaces 3, 7, 9, and 14 will be processed for performing interpolation calculations at client specified locations.

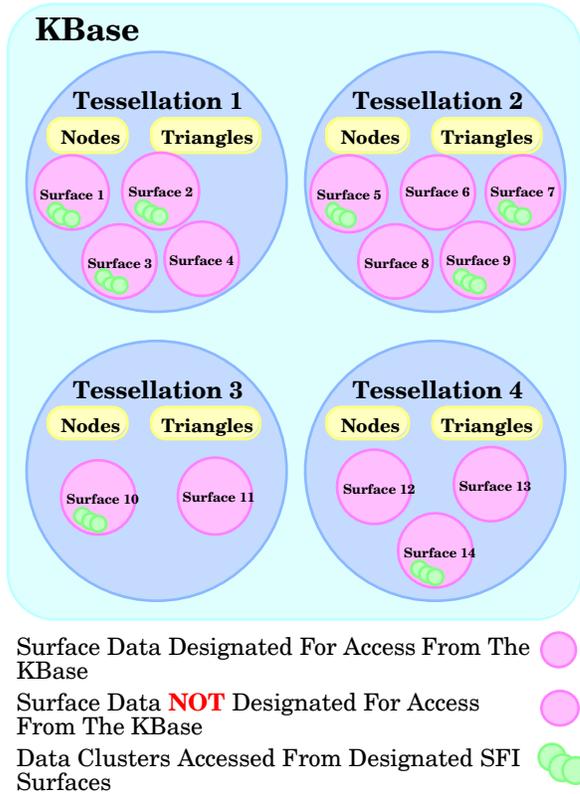


FIGURE 5. SFI Surface Data Cluster Access

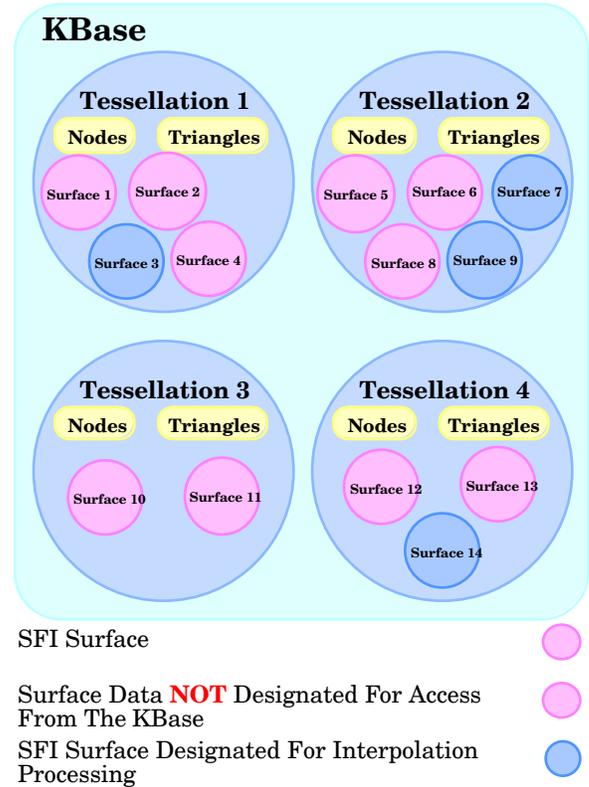


FIGURE 6. Surface Association

CONCLUSIONS

In this paper we have presented the Knowledge Base interface for parametric grid data (libKBI_grid), focusing on the storage and access information. The storage information includes eight data components that represent the results of the data preparation process and also include all required input data for several population tools. This enables complete reprocessing of the storage data whenever necessary.

When information is requested by a client through libKBI_grid, data from the KBase is extracted using a surface fetch initialization (SFI) prescription to restrict the number of accessible surfaces and a representative data cluster size to restrict the amount of data accessed per SFI surface. This set of data is further restricted from interpolation processing on a per event basis by the client through a surface association prescription. The remaining surfaces are processed to find the containing triangles of all locations as prescribed by the client applications. Then a NNI calculation is performed and all interpolated values are returned to the client for each of the final surfaces defined in the event’s surface association set.

REFERENCES

- Delaunay, B.N. (1934). *Sur la sphere vide*, Bull. Acad. Science USSR VII: Class. Sci. Math., 793-800.
- Hipp, J. R., Keyser, R. G., Young, C. J, Shepherd, E., Chael, E. P., (1996). ***Knowledge Base Interpolation of Path- Dependent Data Using Irregularly Spaced Natural Neighbors***, 18th Annual Seismic Research Symposium On Monitoring a CTBT, PL-TR-96-2153, pp 1014-1025.
- Hipp, J. R., Young, C. J, Shepherd, E., Moore, S. G., (1998). ***The DOE Knowledge Base Methodology for the Creation of an Optimal Spatial Tessellation***, 20th Annual Seismic Research Symposium On Monitoring a CTBT, pp 717-726.
- Lawson C. L., 1977, ***Software for CI surface interpolation***, Mathematical Software III, ed. Rice. J., Academic Press, New York.
- Sambridge, M., J. Braun, and H. McQueen (1995). ***Geophysical Parameterization and Interpolation of Irregular Data Using Natural Neighbors***, Geophys. J. Int.
- Watson, D. F., (1992). ***Contouring: A Guide to the Analysis and Display of Spatial Data***, Pergamon, ISBN 0-08- 040286-0, 321 pp.
- Young, C. J., Hipp, J. R., Shepherd, E., Moore, S., G., (1998). ***The DOE Model For Improving Seismic Event Locations Using Travel Time Corrections: Description and Demonstration***, 20th Annual Seismic Research Symposium On Monitoring a CTBT, PL-TR-96-2153, pp 782-790.