ICSV13 - Vienna
The Thirteenth International Congress on Sound and Vibration
Vienna, Austria, July 2-6, 2006

ADVANCED BRANCH AND BOUND METHOD TO INTERPOLATE ACOUSTIC DATA ON STRUCTURAL FINITE ELEMENT MESHES

Michael Rose*[1] and Björn Nagel[1]

[1]Institute of Composite Structures and Adaptive Systems, German Aerospace Center
Lilienthalplatz 7, D-38108, Germany
michael.rose@dlr.de

INTRODUCTION

Multiple field calculations of acoustical and structural coupled problems need algorithms to interpolate data on different meshes, which are especially designed for the underlying physical theories. Particularly in three dimensional structural models with complicated vibrating surfaces, sophisticated algorithms have to be used for the propagation of e.g. pressure and velocity data between the different meshes. This paper describes a new algorithm, based on a branch and bound strategy, which interpolates arbitrary data onto a triangle mesh in logarithmic time complexity for each data point. It is implemented in MATLAB and successfully tested. Due to the coding in C++ and to the enhanced algorithm, it outperforms the available MATLAB tools with respect to precision and time complexity.

The algorithm is used in an operational application for automatic Finite Element modelling, which also will be presented in the paper. Based on a parametric description of the structure and surface meshes like used in aerodynamic or acoustic simulations, direct modelling of aircraft structures is performed. This enables efficient high fidelity simulations in MDO applications with best possible match between the different disciplines' models.

CHAPTER 1: THE ALGORITHM

The algorithm is coded in C++ and compiled as a dynamic link library, which can be called like any other command within MATLAB. Its name is *gridapprox* and to describe its facilities it is convenient to first describe the input and output parameters. A call from a MATLAB-script has the following syntax with five to nine input

parameters and optionally further options to control the algorithm:

[ret,dd] = gridapprox(x,y,z,xx,yy,zz,tri,f,oi,<options>);

x, y, z are vectors, which contain the coordinates of the points $P_i, 1 \le i \le n$ from the source net with known function values f_i and the components of the points $Q_j, 1 \le j \le N$, on which interpolation should be performed, are contained in the vectors xx, yy, zz. If only two dimensional problems are considered, *z=[]* or *zz=[]* indicate zero z-coordinates. If the points P_i are connected by a triangle mesh, it can be specified by the $m \times 3$ matrix *tri*, which contains the point indices $[i_1, i_2, i_3]$ of each triangle in the rows. This is the standard way in MATLAB to specify triangle meshes. Otherwise *tri=[]* can be set and point objects P_i instead of triangle objects $T_k, 1 \le k \le m$ are considered. If *f=[]* then only the indices of the points with smallest distance to Q_j are returned in *ret*. They can be used in another call to *gridapprox* as parameter *oi*, if the search algorithm should be skipped and only the interpolation should be done. Otherwise *f* contains the function values f_i and the interpolated function values g_j are returned in *ret*. Optionally in *dd* the distances from the neighbouring objects to Q_j are returned. The additional options specify the following properties:

- The best $n_b \in \{1, 2, ...\}$ objects should be returned for each point Q_j.
- If $n_d > 1$ is given, then the nearest triangle and its n_d neighbouring layers of triangles are returned as best objects.
- Interpolation: The function value(s) of the nearest point(s) are returned, or the linear interpolated value(s) on the best triangle(s) are returned.
- Combination: If $n_b > 1$ or $n_d > 0$ then different strategies how the interpolated values are combined to one single value can be choosen. Besides the value of the nearest object, the arithmetic mean value can be selected or alternatively a weighted arithmetic mean value with powers of the distances as weights.
- Returned indices can be sorted for each Q_j.

The heart of the implemented procedure is a bounding box algorithm, which is well known in raytracing software for photorealistic picture generation. Before the search for the nearest objects (points or triangles) is done, a bounding box tree is created internally. Each bounding box contains the coordinates of an axis-parallel cuboid which contains all childs of the bounding box. Childs can be other sub boxes or triangles or points. The tree generation is done with time complexity $O(m \log m)$. If no triangle mesh is given and points are considered, *m* has to be replaced by *n* in the following text. The search algorithm uses for each point Q_j this bounding box tree

with the time complexity $O(\log m)$. Therefore the time complexity of *gridapprox* is given by $O((N+m)\log m)$. Note that this estimation is only valid for non degenerated bounding box trees. But for non pathological cases this is heuristically always the case. The memory complexity of $O(N+m)$ for the internal data structures (in addition to the input and output parameters) is not critical. In the following table the mean time complexity of three different approaches is compared:

	sequential search	MATLAB *griddata*	*gridapprox*
mean time complexity	$O(Nm^2)$	$O(N\sqrt{m})$	$O((N+m)\log m)$

If *N* and *m* are of the same order or if $N > m$, then *gridapprox* outperforms the other strategies. If only a few points N are requested for interpolation, *griddata* is faster then *gridapprox*, because the time for the generation of the bounding box tree is not needed. The original MATLAB 6.5 command *griddata* searches the nearest triangles by stepping through the planar triangle mesh until no further reduction in distance is achievable. In meshes, which are not star shaped or with holes, this search strategy can fail, terminating at a local minimum. This can principally not happen with the bounding box strategy of *gridapprox*. In these cases *griddata* returns unsatisfactory NaN (not a number) values as interpolation result. If the points Q_j are outside of the triangle mesh, NaN values are returned too (see figure 1).

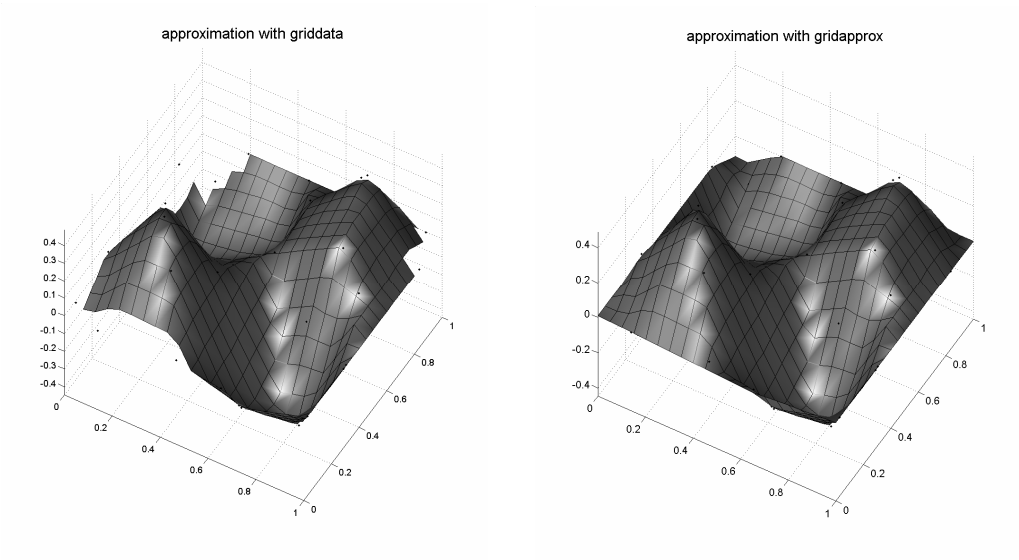

Figure 2 - missing data at boundary.

This is very disadvantageous for example in the case of irregular data points which are contained in a rectangle and the interpolation should be performed on a regular rectangular mesh. Many interpolation values on the rectangular boundary are NaN values in this case, if *griddata* is used. With *gridapprox* an extrapolation is done for

outer points, avoiding NaN values at all.

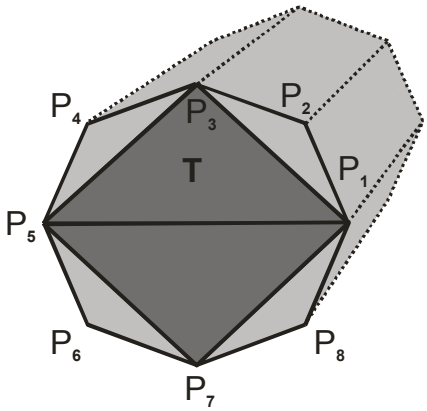

Figure 2 - Bad tetrahedronal volume mesh for a cylindrical surface.

Interpolation on triangular meshes in three space dimensions is not implemented by the standard MATLAB commands. There is a three dimensional command *griddata3,* but this command uses tetrahedronal volume meshes internally to do the interpolation. In most cases this gives no satisfactory results, if interpolation on surface meshes is required. And the internal creation of the Delaunay tetrahedronal mesh is extreme costly with respect to time if the number of data points n is big. In figure 2 a cross section of a cylindrical surface with eight circumferential nodes is displayed. An automatically created Delaunay tetrahedronal mesh of the cylinder fills its interior and the diametrical points P_1 and P_5 are connected by the inner tetrahedron T. This means that the function value f_1 erroneously influences the interpolation of inner points near P_5. This unwanted effect is also possible for outer points if the surface is not convex.

CHAPTER 2: MOTIVATION FOR MESH BASED MODELLING

Multidisciplinary analyses can either base on models merging the constitutive equations of all physical disciplines concerned or on the linking of multiple mono discipline codes to one coupled simulation environment. Nowadays, the second approach has become very popular since this so called *weak coupling* permits the utilisation of proven high sophisticated tools. A prominent example showing this trend is aeroelastics: Classically the interaction of aerodynamic forces and elastic deflections of lifting bodies is described by one system of equations comprising both aerodynamic and structural properties. The resulting problem is compact and can be handled analytically to some extend. But the effort in modelling increases rapidly if complex structures or higher order aerodynamic effects may not be disregarded.

High sophisticated tools like Finite Element Analysis (FEA) for structures and Finite Volume Analysis (FVA) for aerodynamics exploit today available simulation capacities in the single disciplines. Coupling can be established by an outer control algorithm, which calls the specialised codes sequentially and feeds the results from one code as input to the other. Hence, the aerodynamic pressure distribution for the initial wing shape is conventionally calculated and subsequently mapped onto the surface of the structural model. Following the deflections are calculated and then are mapped back on the aerodynamic model, which has to be updated to the new geometry. This equilibrium loop is run until convergence. Grid deformation as well as single discipline analysis is widely state-of-the-art. But coupling still constitutes a challenge. The physics of the disciplines requires considerable different discretisations and following state variables can not directly be transferred as shared degrees of freedom. Especially bi-directional coupling requires appropriate interpolation techniques and might be sensitive to numerical loss e.g. due to non-conservative coupling algorithms. While systematic discrepancies caused by mesh topology can be considered in interpolation strategies, geometric mismatch between the interface areas of the two models can disturb even elaborated coupling strategies like volume spline techniques significantly.

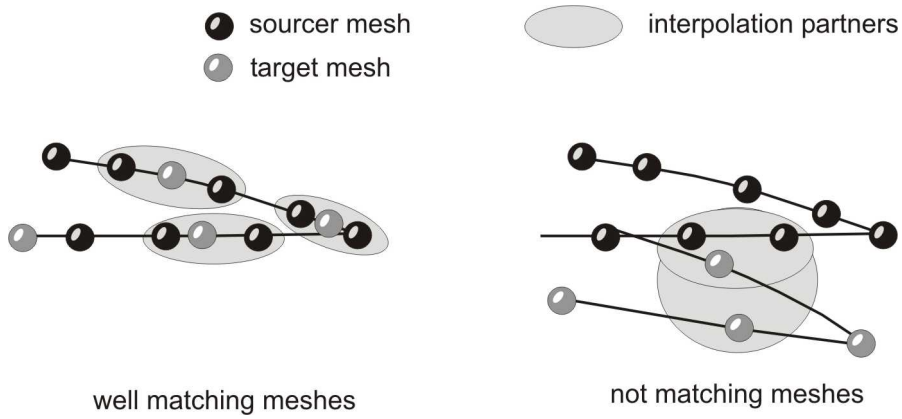

Figure 3 - Interpolation problems due to trailing edge mismatch.

If all modelling base on only one high quality CAD description, there is an eligible aspiration for receiving well matching models. But the experience in real life projects has shown that this precondition can only be realised exceptionally. Confusions about CAD model versions are as typical as blocking of information e.g. due to uncertainties regarding the rights of third parties on the data. To enable modelling without sensible CAD data and to avoid potential trouble with interpolation the modelling tool *PARA_MAM* was created based on the *gridapprox* routine. *PARA_MAM* is a suite of MATLAB functions which create structural Finite Element wing models based on a versatile parametric description of the inner structure and the aerodynamic surface mesh as shape reference. *PARA_MAM* abbreviates PARAmetric simple and fast Mesh based Aircraft Modelling tool. Hence, an important characteristic is higher simplicity and speed compared with CAD based modelling approaches.

CHAPTER 3: PARAMETRIC MODELING WITH PARA_MAM

More than a century of evolution in human flight has led to shell based light weight wings which are very similar among most types of aircraft. Besides the load carrying surface spanwise oriented spars and airflow directed ribs constitute the primary structure. The number and arrangement of spars and ribs is variable. They can be straight or kinked several times. *PARA_MAM* uses a parametric description of spar and rib geometry in the XY projection: Spars are defined in a matrix where each line corresponds to one spar or a set of similar spars. In the rows the information about the path of the spar from the aircraft's symmetry plane to the wing tip. The minimum input is a line vector with four rows

1) Chord position of the set's first spar at the symmetry plane
2) Chord position of the set's first spar at the symmetry plane
3) Number of spars in the set
4) Status: real spar or virtual spar (will be explained later)

The programme distributes the specified number of ribs linearly between the first and the last chord position at the symmetry plane. If no more rows are appended to the matrix, each spar will end at the same chord position it has at the symmetry plane. If not straight but kinked spars are desired, spar definitions points may be added by four more rows per point. The first point information is the spanwise position followed by the two chord information and the status. The spars' chord position at the tip is aligned with the last definition point. This strategy permits to define an arbitrary number of spars with arbitrary definition points each. Rib definition is set up corresponding to spanwise point definitions along the path from leading edge to trailing edge. All geometric data are normalised to wing span and local chord, what enables easy adaption of the structure to new aerodynamic shapes.

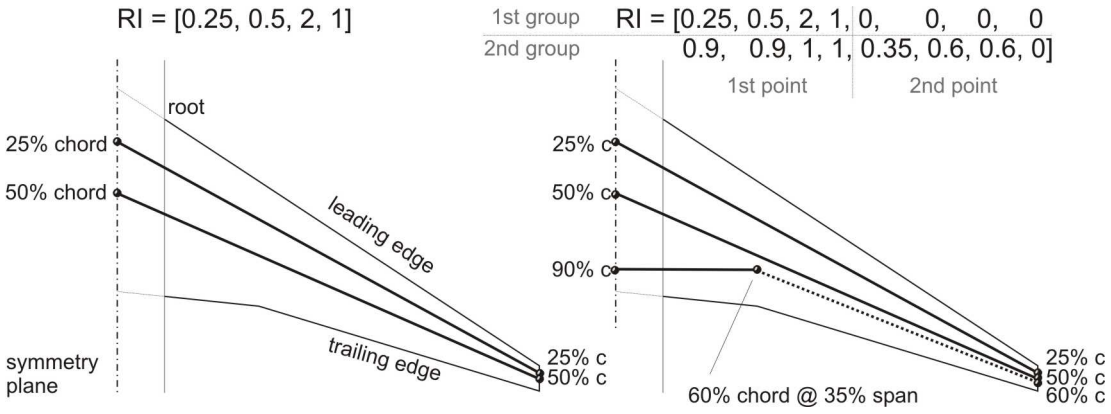

Figure 4 - Parametric description of spars.

In *PARA_MAM* the structural geometry is based on the intersection points between ribs and spars which are connected by straight lines. If this idealisation does not deliver sufficient accuracy e.g. due to kinks in the aerodynamic shape between rib positions, additional ribs and spars can be introduced. They are designated *'virtual'* since they are handled like conventional ribs but will not be exported as structural

elements to the FEA code. Like touched at the spar definition, ribs and spars can change their status between definition points. This enables designs like intermediate spars ranging only to the wing's kink. Anyway, ribs and spars constitute a regular grid hence geometric keypoints in the XY projection can be calculated easily using elementary trigonometry.

Leading and trailing edges introduce dimensions into modelling. They are captured from the aerodynamic mesh which can be either structured or unstructured. Lines in airflow direction connecting leading edge with trailing edge form the chord area. *Gridapprox* is used in 2D interpolation mode to calculate the z coordinates on the chord area for the keypoints' already known x and y coordinates. The flat representation of the wing in 3D space serves as starting point to calculate the keypoints of the volumetric wing. They are the intersection points of spars, ribs and aerodynamic surface and thus lay pair-wise on the intersection lines between spars and ribs. Since the points on the chord area also are located at the straight intersection lines and rip and spar rotation angles are specified in the *PARA_MAM* input, the orientation of the vector from the known points is defined while the length is to be determined. A rudiment of *PARA_MAM* is the handling of aerodynamic hull to be complexly shaped and following the calculation of the intersections with a ray is not a simple task.

The chosen solution bases on an iterative procedure based on *gridapprox* in 3D mode: The vector length are guessed and for the resulting positions the x,y and z components from the aerodynamic mesh are interpolated. To avoid latency due to repeated *gridapprox* calls with repeated calculation of the bounding box structure, initially several guesses are evaluated. The dependency of the distance *D* between the guessed points from the hull and the vector length *R* is obtained. Following zero points are approximated and resulting new vector length are evaluated to widen the database for zero point search. In first applications the procedure has led to satisfactory convergence within few iteration loops. Special cases like multiple intersection points and vector staring points outside the hull are considered. The treatment of aerodynamic hull to be complex permits easy morphing of the structure to changed aerodynamic mesh like performed in multidisciplinary optimisation processes.

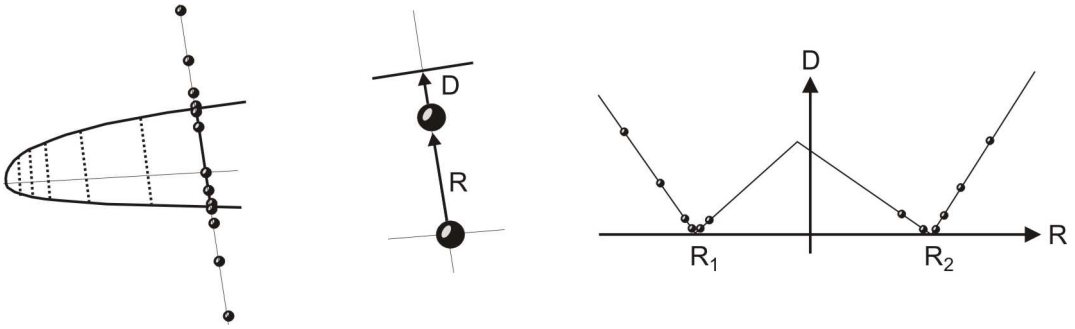

Figure 5 - Keypoint search based on gridapprox.

The structured parameterisation facilitates an efficient passing of the data to FEA pre-processors using *.ascii files. Based on very simple operations the wing can be build up from scratch using elementary commands only. Critical modelling operations like interceptions of free areas are not necessary. Beyond geometry, controlled meshing, clamping and material definition can be performed. Since the complete wing representation is available in *PARA_MAM*, export can be extended to various pre-processors with batch capacities easily. Based on the presented structured model generation also advanced structures can be realised like intercepting ribs, explicit stringers or attachment parts like engines, nacelles and pylons. Referring rib and spar definition not to the projection but directly to the chord area permits extreme shapes like O-wing configurations.

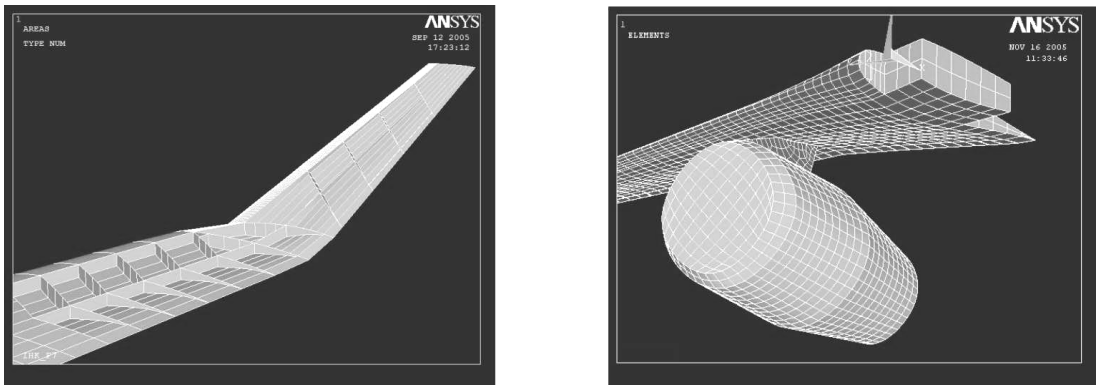

Figure 6 - FE wing examples (ANSYS).

CHAPTER 4: CONCLUSION

Gridapprox is a C++ coded MATLAB routine for 3D interpolation. The implemented bounding box algorithm makes it extremely fast and robust especially with large and complex topologies. Interpolation problems due to geometric mismatch of interface areas can be avoided by mesh based modelling like realised in the *PARA_MAM* MATLAB function. Based on a parametric description of the structural geometry the 3D keypoints of aircraft wings are fitted to aerodynamic meshes. PARA_MAM can take key positions in MDO processes; not only for wing structures. Efficiency and feasibility of the approach base on robust interpolation with *gridapprox*.

REFERENCES

[1] Yunhong, Z., Subhash, S.: “Analysis of a Bounding Box Heuristic for Object Intersection”, Journal of the ACM, Vol. 46, No. 6, p. 25 (1999).
[2] Edelsbrunner, H, Geometry and Topology for Mesh Generation, Cambridge University Press, 2001.
[3] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*, Sec. Ed., MIT Press and McGraw-Hill, Chapter 19: Binomial Heaps, 455-475, 2001.