NAG C Library Chapter Introduction

d06 – Mesh Generation

Contents

1 Scope of the Chapter

This chapter is concerned with automatic mesh generation

with line segments, over the boundary of a closed two-dimensional connected polygonal domain;

with triangles, over a given two-dimensional region using only its boundary mesh.

2 Background to the Problems

An important area of scientific computing in engineering is the solution of partial differential equations of various type (for solid mechanics, fluid mechanics, thermal modelling, ...) by means of the finite element method. In essence, the finite element method is a numerical technique which solves the governing equations of a complicated system through a discretisation process. The user may wish to consult Cheung *et al.* (1996) to see an application of the finite element method to solid mechanics and field problems.

A key requirement of the Finite Element method is a mesh, which subdivides the region on which the partial differential equations are defined. Note that such meshes are also essential to other discretisation processes, such as the Finite Volume method. However, for the purpose this description we focus (without loss of generality) on the Finite Element method. Thus, meshing algorithms are of crucial importance in every numerical simulation based on the finite element method. In particular, the accuracy and even the validity of a solution is strongly tied to the properties of the underlying mesh of the domain under consideration.

In this chapter, the Delaunay constrained $2D$ triangulation (see George and Borouchaki (1998) or Chapter 7 of Cheung *et al.* (1996)) is considered and routines are provided to triangulate a closed polygonal domain of $\mathbb{R}^2$, given a mesh of its boundary (in a later Mark of the Library, software for the $3D$ case will be available). A domain in $\mathbb{R}^2$ is given via a discretisation of its boundary. The boundary is described as a list of segments, with given end-point coordinates. Then an incremental method is used to generate the set of interior vertices.

Let Ω be a closed bounded domain in $\mathbb{R}^2$ or $\mathbb{R}^3$. The question is how to construct a triangulation (mesh) of this domain suitable for a finite element framework. Following the definition in George and Borouchaki (1998):

$\sqcup$ is a mesh of Ω if

$\Omega = \bigcup_{k \in \sqcup} k$.

Every element k in $\sqcup$ is non-empty.

The intersection of the interior of any two elements is empty.

The intersection of any two elements in $\sqcup$ is either,

the empty set,

a vertex,

an edge,

a face (in $\mathbb{R}^3$).

In the finite element method, the meshes are in general denoted $\sqcup$ or $\sqcup_h$, where the index h refers to a measure of the diameter (length of the longest edge) of the elements in the mesh. A triangulation is a set of entities described in a suitable manner by picking an adequate data structure. The algorithm for triangulation construction creates a table of elements in the triangulation as well as the neighbourhood relationships between the elements. Those elements are meant to satisfy the so-called 'empty sphere criterion' which means that the open ball associated with the element (the circumcircle of the triangle in $2D$, and the circumsphere of the tetrahedron in $3D$) does not contain any vertices (while the closed ball contains the vertices of the element in consideration only). This criterion is a characterisation of the Delaunay triangulation.

Given $\sqcup_i$ the Delaunay triangulation of the convex hull of the first i points, the purpose of **the incremental method** (which is the main method to generate nodes and elements inside the domain) is to obtain $\sqcup_{i+1}$ the Delaunay triangulation which includes an $(i+1)$th point p as an element vertex. To this end, one can

introduce a procedure referred to as the 'Delaunay kernel' construction. This kernel is

$$\sqcup_{i+1} = \sqcup_i - \rfloor_p + \lfloor_p,$$

where $\lfloor_p$ is the ball associated with p and $\rfloor_p$ is the associated cavity. The ball associated with a given point p is the set of elements in the triangulation including p as a vertex, while the cavity is the set of elements whose circumcircles or circumballs enclose the point p. One can prove that, given $\sqcup_i$ a Delaunay triangulation of a convex hull of the first i points, then $\sqcup_{i+1}$ is a Delaunay triangulation of the hull that includes p as the $(i+1)$th vertex. The completion of a Delaunay triangulation relies on applying the Delaunay kernel procedure to every point.

The problems here are

to choose the input data $\sqcup_0$ of the incremental method, and

to generate at each iteration this $(i+1)$th point, such that $\sqcup_{i+1}$ is still a Delaunay triangulation of the convex hull of the $(i+1)$ points.

For a finite element application, it is required to construct a mesh of the domain Ω whose elements are as close to equilateral as possible.

The mesh generation methods include an initial creation stage resulting in a mesh $\sqcup_0$, without internal points, except for any specified interior points (see George and Borouchaki (1998) for more details). Such a mesh is referred to as the 'empty mesh'. This mesh consists of a box which includes the whole geometry plus some vertices on the edge of that box. From here the methods differ in how the required internal points are created.

The general principle of interior mesh generation is to either create a point and insert it immediately by means of the Delaunay method (the so-called Delaunay kernel), repeating the process as long as points can be created, or to generate a series of points, insert this series and iterate the process as long as a non-empty series is created. At this stage it is quite useful to define the notion of a control space to govern the internal point creation. The 'ideal' control is the input of a function defined analytically at any point of $\mathbb{R}^2$ and which specifies the size and the direction features that must be conformed to anywhere in the space.

To construct such a function, one can consider several approaches. For our purpose in this chapter, this control function computes, from data, the local step sizes (the desired distance between two points) related to the given points. A generalised interpolation then enables us to obtain the function everywhere. This process is purely geometric in the sense that it relies only on the geometric data properties: boundary edge lengths, and so on. The user is advised to consult George and Borouchaki (1998) for more details about this strategy, especially about the other approaches which can be considered to construct the control function.

3 Recommendations on Choice and Use of Available Functions

3.1 Boundary Mesh Generation

The first step to mesh any domain of $\mathbb{R}^2$ or $\mathbb{R}^3$ is to generate a mesh of the domain boundary. In this chapter, since only the $2D$ case is considered, the relevant routine is nag_mesh2d_bound (d06bac). This routine meshes with segments a boundary of a closed connected polygonal domain of $\mathbb{R}^2$, given a set of characteristic points and characteristic lines which define the shape of the frontier. The boundary has to be partitioned into geometrically simple lines. Each line segment may be a straight line, a curve defined by an equation of the type $f(x,y) = 0$, or simply a polygonal curve, delimited by characteristic points (end-points of the lines). Then, the user can assemble those lines into connected components of the domain boundary.

3.2 Interior Mesh Generation

In this chapter three routines are provided to mesh a domain given a discretisation of its boundary with optionally specified interior points.

nag_mesh2d_delaunay (d06abc) uses an internal point construction method along the internal edges. Using the control function, a small number of points are generated along each edge.

nag_mesh2d_front (d06acc) uses a point creation method based on an advancing front point placement strategy, starting from the 'empty mesh'.

nag_mesh2d_inc (d06aac) uses a simple incremental method based on a control function given analytically via the argument **power**.

Any point construction method results in a set of points. These points are then inserted by means of the Delaunay kernel.

The point insertion process is completed by successive waves. The first wave results from the empty mesh edge analysis (edge method) or from the empty mesh front analysis (advancing front method). Subsequent waves correspond to the analysis of the edges of the previous mesh. For the advancing front strategy, the waves follow the analysis of the front associated with the current mesh.

One can propose a general scheme for a mesh generation method. Seven steps can be identified as follows.

Preparation step.

Data input: point coordinates, boundary edges and internal edges (if any),

construction of the bounding box,

meshing of this box by means of a few triangles.

Construction of the box mesh.

Insertion of the given points in the box mesh using the Delaunay kernel.

Construction of the empty mesh.

Search for the missing specified edges,

enforcement of these edges,

definition of the connected components of the domain.

Internal point creation and point insertion.

Control space definition,

(1) internal edge analysis, point creation along these edges,

point insertion via the Delaunay kernel and return to (1).

Domain definition.

Removal of the elements exterior to the domain,

classification of the elements with respect to the connected components.

Optimisation.

edge swapping,

point relocation, ...

File output.

When using the advancing front approach described earlier, one has to replace the step denoted by (1) of the general scheme. The analysis of the edges of the current mesh is then replaced by the front analysis.

Due to the fact that the particular mesh generated by nag_mesh2d_inc (d06aac), nag_mesh2d_delaunay (d06abc) and nag_mesh2d_front (d06acc) may be sensitive to the platform being used; there may be differences between generated nodal coordinates and connectivities. However all meshes generated should be expected to satisfy the 'empty sphere criterion'.

3.3 Mesh Management and Utility Routines

In addition to meshing routines, management and utility routines are also available in this chapter.

A mesh smoother routine nag_mesh2d_smooth (d06cac), is provided to improve mesh triangle quality.

Since the Finite Element framework includes a requirement to solve matrices based on meshes, the routine nag_mesh2d_sparse (d06cbc) generates the sparsity pattern of such a matrix. Due to the fact that the numbering of unknowns in a linear system could be crucial in term of storage and performance issues, a vertex renumbering routine nag_mesh2d_renum (d06ccc) is provided. This routine also returns the new sparsity pattern based on the renumbered mesh.

To mesh a complicated geometry, it is sometimes better to partition the whole geometry into a set of geometrically simpler ones. Some geometry could also be deducted from another geometry by an affine transformation and nag_mesh2d_trans (d06dac) could be used for that purpose. nag_mesh2d_join (d06dbc) is provided to join all the simple geometry meshes. This routine can also handle the joining of two adjacent as well as overlapping meshes, which may be useful in a domain decomposition framework.

4 Example of Use in the Solution of a Partial Differential Equation

The use of d06 mesh generation routines, together with sparse solver routines from Chapter f11 to solve partial differential equations with the finite element method is described in a NAG Technical Report (see Bouhamou (2001)). This report, and accompanying source code, is available from the NAG web site, or by contacting one of the NAG Response Centres.

5 Index

6 Functions Withdrawn or Scheduled for Withdrawal

None.

7 References

Bouhamou N (2001) The use of NAG mesh generation and sparse solver routines for solving partial differential equations *NAG Technical Report TR 1/01* NAG Ltd, Oxford

Cheung Y K, Lo S H and Leung A Y T (1996) *Finite Element Implementation* Blackwell Science

George P L and Borouchaki H (1998) *Delaunay Triangulation and Meshing: Application to Finite Elements* Editions HERMES, Paris

Quarteroni A and Valli A (1997) Numerical Approximation of Partial Differential Equations *Comp. Maths.* **23**