The use of NAG mesh generation and sparse solver routines for solving partial differential equations

Nadir Bouhamou, NAG Ltd

Abstract

This report demonstrates the use of routines from the NAG Fortran Library D06 Chapter, combined with sparse solver routines from the F11 Chapter to solve a partial differential equation using the finite element method.

Keywords: Finite Element Method, Numerical Simulations, Triangular Mesh, Linear Approximation.

1 Introduction

An important area of scientific computing in engineering is the solution of partial differential equations (PDEs) of various type (for example in solid mechanics, fluid mechanics or 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.

Key requirements in the solution of a PDE using the finite element method are (among others):

- a mesh, which subdivides the region on which the PDE is defined,
- a discretisation of the PDE resulting in a system of algebraic equations,
- efficient matrix solvers to compute the numerical solution of the PDE.

The D06 Chapter of the NAG Fortran Library [1] provides meshing routines based on algorithms described in [2]. Such meshing algorithms are of crucial importance in any 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.

This report shows how to combine the use of D06 Chapter routines to mesh a 2-dimensional domain, and F11 routines to solve the linear system arising from the discretisation of a

simple PDE. The building of the sparse matrix and right-hand side based on the finite element mesh is also described. The source code, data file and results for this example are also included in this report.

Necessary mathematical background is provided, but no attempt is made to give a full treatment of the finite element method. The user may wish to consult [3] to see an application of the finite element method to solid mechanics and field problems. For more information on this subject, consult [4].

2 The Model Problem

Let Ω be a closed bounded domain in $\mathbb{R}^2$ and $\Gamma = \partial\Omega$ its boundary. Let Γ_D and Γ_N be two parts of Γ such that

$$\Gamma = \Gamma_D \cup \Gamma_N \text{ and } \Gamma_D \cap \Gamma_N = \begin{cases} \emptyset \quad \text{or,} \\ \text{a finite set of points.} \end{cases}$$

For the purpose of this report, a $2D$ Helmholtz equation is considered:

$$\begin{cases} -\alpha\nabla^2 u + \beta u &= f \text{ in } \Omega; \\ u &= g_D \text{ on } \Gamma_D; \\ \dfrac{\partial u}{\partial n} &= g_N \text{ on } \Gamma_N. \end{cases} \tag{1}$$

where $\nabla^2 = \dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2}$, and $\alpha > 0$, $\beta \geq 0$. f and u are defined in Ω. g_D (the Dirichlet boundary condition) is defined on Γ_D while g_N (the Neumann boundary condition) is defined on Γ_N.

3 Spatial Integration

It can be proven that if u is a solution of (1), then u is also a solution of the variational (weak) formulation,

$$\begin{cases} \text{Find } u \in V \text{ such that} \\ \displaystyle\int_\Omega (\alpha\nabla u.\nabla v + \beta uv)dxdy = \int_\Omega fvdxdy + \int_{\Gamma_N} \alpha g_N v d\sigma \quad \text{for all} \quad v \in V. \end{cases} \tag{2}$$

where

$$V = \{v; v \text{ is continuous on } \Omega, \frac{\partial v}{\partial x} \text{ and } \frac{\partial v}{\partial y} \text{ are piecewise continuous functions and } v = g_D \text{ on } \Gamma_D\},$$

and $\nabla = \begin{pmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \end{pmatrix}$.

The variational formulation of the problem is the starting point of the finite element method. This method consists of finding an approximate solution in a finite dimensional subspace. Suppose Ω is a polygonal open subset of $\mathbb{R}^2$ discretised into a set of nt triangular elements K_k, for $k = 1, \ldots, nt$, which forms a regular triangulation $\mathcal{T}_h$ with nv vertices x_i, for $i = 1, \ldots, nv$, and nf edges E_l, for $l = 1, \ldots, nf$, which form the partition of the boundary Γ:

$$\Omega = \bigcup_{K_k \in \mathcal{T}_h} K_k = \bigcup_{k=1}^{nt} K_k, \quad \Gamma = \bigcup_{l=1}^{nf} E_l.$$

The subscript h in $\mathcal{T}_h$ represent a measure of the size of the mesh: $h = \max_{K_k \in \mathcal{T}_h} \mathrm{diam}(K_k)$ where $\mathrm{diam}(K_k)$ is the length of the longest edge of the triangle K_k.

In order to solve the variational problem (2), one should pose the problem in a finite dimensional (discrete) subspace:

$$\begin{cases} \text{Find } u_h \in V_h \\ \displaystyle\int_\Omega (\alpha \nabla u_h . \nabla v_h + \beta u_h v_h) dxdy = \int_\Omega f v_h dxdy + \int_{\Gamma_N} \alpha g_N v_h d\sigma \quad \text{for all} \quad v_h \in V_h. \end{cases} \tag{3}$$

where

$$V_h = \{v; v \text{ is continuous on } \Omega, v|_K \in P^1(K) \text{ and } v = g_D \text{ on } \Gamma_D\}.$$

$P^1(K)$ is the space of polynomials in K of degree less than or equal to 1; which is also the space of linear functions in the two space variables x and y in K. One can prove of course that V_h is a finite dimensional space. Let us consider the space

$$\tilde{V_h} = \{v; v \text{ is continuous on } \Omega, v|_K \in P^1(K)\} \supseteq V_h,$$

one can prove that $\dim(\tilde{V_h}) = nv$ and that $\{\phi_i\}_{i=1}^{nv}$ defined by

$$\phi_i \in P^1(K) \text{ and } \phi_i(x_j) = \delta_{ij} \text{ for all } i, j = 1, \ldots, nv$$

is a basis of $\tilde{V_h}$. One can then write for all $v \in \tilde{V_h}$

$$v(x) = \sum_{j=1}^{nv} \eta_j \phi_j(x), \text{ where } \eta_j = v(x_j) \text{ for } x \in \Omega.$$

Note that for all $i = 1, \ldots, nv$, the support of function ϕ_i is then reduced to the set of triangles of which x_i is a vertex (the support of a function is roughly the set of points where the function is different from 0). One can note also that $\sum_{j=1}^{nv} \phi_j(x) = 1$ for all $x \in \Omega$.

So, the variational problem posed in a finite dimensional space (3) is equivalent to the linear system:

$$A\xi = b \tag{4}$$

where $A = (a_{ij})_{1 \le i,j \le nv}$, $\xi = (\xi_i)_{1 \le i \le nv}$ and $b = (b_i)_{1 \le i \le nv}$ with

$$\begin{aligned} a_{ij} &= \int_\Omega (\alpha \nabla \phi_i . \nabla \phi_j + \beta \phi_i \phi_j) dxdy; \\ b_i &= \int_\Omega f \phi_i dxdy + \int_{\Gamma_N} \alpha g_N \phi_i d\sigma; \\ \xi_i &= u_h(x_i), \end{aligned}$$

for all i and j such that x_i, $x_j \in \Omega$ and $\notin \Gamma_D$. For the rows i, such that $x_i \in \Gamma_D$, then

$$\begin{aligned} &a_{ii} = 1.0; \\ &a_{ij} = 0.0, \text{ for all } j \neq i; \\ &b_i = g_D(x_i). \end{aligned}$$

Let $M = (m_{ij})_{1 \leq i,j \leq nv}$ and $R = (r_{ij})_{1 \leq i,j \leq nv}$ be respectively the mass matrix and the stiffness matrix

$$\begin{aligned} m_{ij} &= \int_\Omega \phi_i \phi_j dxdy = \sum_{k=1}^{nt} \int_{K_k} \phi_i \phi_j dxdy = \sum_{k=1}^{nt} m_{ij}^k; \\ r_{ij} &= \int_\Omega \nabla\phi_i . \nabla\phi_j dxdy = \sum_{k=1}^{nt} \int_{K_k} \nabla\phi_i . \nabla\phi_j dxdy = \sum_{k=1}^{nt} r_{ij}^k. \end{aligned}$$

So $A = \alpha R + \beta M$. Notice that for all $i = 1, \ldots, nv$

$$\sum_{j=1}^{nv} m_{ij} = \int_\Omega \phi_i dxdy; \tag{5}$$

$$\sum_{j=1}^{nv} r_{ij} = 0. \tag{6}$$

So

$$\begin{aligned} m_{ij}^k &= \int_{K_k} \phi_i \phi_j dxdy, \\ r_{ij}^k &= \int_{K_k} \nabla\phi_i . \nabla\phi_j dxdy; \end{aligned}$$

for all $i, j = 1, \ldots, nv$ and $k = 1, \ldots, nt$; are the element contributions to the mass and the stiffness matrices respectively.

Then

$$\begin{aligned} b_i &= \sum_{k=1}^{nt} \int_{K_k} f\phi_i dxdy + \sum_{l \in L_N} \int_{E_l} \alpha g_N \phi_i d\sigma \\ &= \sum_{k=1}^{nt} b_i^k + \sum_{l \in L_N} b_{edge_i}^l, \end{aligned}$$

for all $i = 1, \ldots, nv$; where

$$L_N = \{l; 1 \leq l \leq nf \text{ and } E_l \text{ is a triangle edge lying on } \Gamma_N\}.$$

Thus

$$\begin{aligned} b_i^k &= \int_{K_k} f\phi_i dxdy, \\ b_{edge_i}^l &= \int_{E_l} \alpha g_N \phi_i d\sigma \end{aligned}$$

for all $i = 1, \ldots, nv$, $k = 1, \ldots, nt$ and $l \in L_N$; are the element and edge contributions to the right-hand side of the linear system (4) respectively.

4 The 3-node Triangle, Linear Approximation

Let $K = \{(x_i, y_i)\}_{i=1,\ldots,3}$ be an element of $\mathcal{T}_h$, which is a triangulation of Ω. Also let $\widehat{K} = \{(\hat{x}_i, \hat{y}_i)\}_{i=1,\ldots,3} = \{(0,0); (1,0); (0,1)\}$ be the reference element and F_K the P^1 transformation such that $F_K(\hat{x}_i, \hat{y}_i) = (x_i, y_i)$, for all $i = 1, \ldots, 3$.

Thus

$$\begin{array}{rcl} F_K : \widehat{K} & \longrightarrow & K \\ (\hat{x}, \hat{y}) & \longmapsto & (x, y) = F_K(\hat{x}, \hat{y}), \end{array}$$

where

$$F_K(\hat{x}, \hat{y}) = \begin{pmatrix} F_K^1(\hat{x}, \hat{y}) \\ \\ F_K^2(\hat{x}, \hat{y}) \end{pmatrix} = \sum_{i=1}^{3} \lambda_i(\hat{x}, \hat{y}) \begin{pmatrix} x_i \\ y_i \end{pmatrix},$$

and λ_i are the elements of P^1 such that $\lambda_i(\hat{x}_j, \hat{y}_j) = \delta_{ij} \quad \forall i, j = 1, \ldots, 3$

$$\begin{array}{l} \lambda_1(\hat{x}, \hat{y}) = 1 - \hat{x} - \hat{y}, \\ \lambda_2(\hat{x}, \hat{y}) = \hat{x}, \\ \lambda_3(\hat{x}, \hat{y}) = \hat{y}. \end{array}$$

The transformation F_K is a bijective function if no three points of the set $\{(x_i, y_i),\ i = 1, \ldots, 3\}$ are aligned. The Jacobian of this transformation is

$$\begin{array}{rcl} JF_K(\hat{x}, \hat{y}) & = & det[DF_K(\hat{x}, \hat{y})], \text{ where} \\ [DF_K(\hat{x}, \hat{y})]_{i,j} & = & \partial_j F_K^i(\hat{x}, \hat{y}),\ \forall i, j = 1, 2 \text{ and one has} \\ \{DF_K(\hat{x}, \hat{y})\}^{-1} & = & \dfrac{1}{JF_K(\hat{x}, \hat{y})} {A_K}^T(\hat{x}, \hat{y}), \\ \text{where } A_K(\hat{x}, \hat{y}) & = & \{Cof[DF_K(\hat{x}, \hat{y})]\} \quad \forall (\hat{x}, \hat{y}) \in \hat{K}, \quad \text{is the co-matrix.} \end{array}$$

For a function f defined on K, one defines $\hat{f} = f \circ F_K$ on $\hat{K}$. Note that f is a $\mathcal{C}^1$ function on K, if and only if $\hat{f}$ is a $\mathcal{C}^1$ function on $\hat{K}$ and the following equality holds,

$$\nabla f \circ F_K(\hat{x}, \hat{y}) = \frac{1}{JF_K(\hat{x}, \hat{y})} A_K(\hat{x}, \hat{y}).\hat{\nabla} \hat{f}(\hat{x}, \hat{y});$$

$$\text{with } \hat{\nabla} = \begin{pmatrix} \dfrac{\partial}{\partial \hat{x}} \\ \\ \dfrac{\partial}{\partial \hat{y}} \end{pmatrix}.$$

The computation of the element contribution to the mass and stiffness matrices can be done isoparametrically using the formulae:

$$\int_K \gamma(x, y) f(x, y) g(x, y) dx dy = \int_{\hat{K}} \hat{\gamma}(\hat{x}, \hat{y}) \hat{f}(\hat{x}, \hat{y}) \hat{g}(\hat{x}, \hat{y}) JF_K(\hat{x}, \hat{y}) d\hat{x} d\hat{y}, \tag{7}$$

$$\int_K \mu(x, y) \nabla f(x, y) \nabla g(x, y) dx dy =$$

$$\int_{\hat{K}} \frac{1}{JF_K(\hat{x}, \hat{y})} \hat{\mu}(\hat{x}, \hat{y}) \hat{\nabla} \hat{f}(\hat{x}, \hat{y}) A_K(\hat{x}, \hat{y})^T A_K(\hat{x}, \hat{y}) \hat{\nabla} \hat{g}(\hat{x}, \hat{y}) d\hat{x} d\hat{y}, \tag{8}$$

where γ and μ are, for example, $\mathcal{C}^{(1)}$ functions defined on the element K. Finally to compute integrals on $\hat{K}$ one can use an integration formula on $\hat{K}$.

5 Computation of Element Contributions

For a given triangle $K_k = \{(x_i^k, y_i^k)\}_{i=1,\ldots,3}$ of the triangulation, $k = 1, \ldots, nt$:

$$JF_{K_k}(\hat{x}, \hat{y}) = JF_{K_k} = (x_2^k - x_1^k)(y_3^k - y_1^k) - (x_3^k - x_1^k)(y_2^k - y_1^k);$$

$$A_{K_k}(\hat{x}, \hat{y}) = A_{K_k} = \begin{pmatrix} (y_3^k - y_1^k)(y_1^k - y_2^k) \\ \\ (x_1^k - x_3^k)(x_2^k - x_1^k) \end{pmatrix}.$$

Thus the element and edge contributions to both sides of the linear system (4) are: for all $i, j = 1, \ldots, nv$; $k = 1, \ldots, nt$;

$$\begin{aligned}
m_{ij}^k &= JF_{K_k} \int_{\hat{K}} \hat{\phi}_i \hat{\phi}_j d\hat{x} d\hat{y}; \\
r_{ij}^k &= \frac{1}{JF_{K_k}} \int_{\hat{K}} \hat{\nabla}\hat{\phi}_i A_{K_k}^T A_{K_k} \hat{\nabla}\hat{\phi}_j d\hat{x} d\hat{y} \\
&= \frac{1}{JF_{K_k}} \{\alpha_k \int_{\hat{K}} \frac{\partial\hat{\phi}_i}{\partial\hat{x}} \frac{\partial\hat{\phi}_j}{\partial\hat{x}} d\hat{x} d\hat{y} + \gamma_k \int_{\hat{K}} \{\frac{\partial\hat{\phi}_i}{\partial\hat{x}} \frac{\partial\hat{\phi}_j}{\partial\hat{y}} + \frac{\partial\hat{\phi}_i}{\partial\hat{y}} \frac{\partial\hat{\phi}_j}{\partial\hat{x}}\} d\hat{x} d\hat{y} \\
&\quad + \beta_k \int_{\hat{K}} \frac{\partial\hat{\phi}_i}{\partial\hat{y}} \frac{\partial\hat{\phi}_j}{\partial\hat{y}} d\hat{x} d\hat{y}\}; \\
b_i^k &= JF_{K_k} \int_{\hat{K}} \hat{f}\hat{\phi}_i dx dy;
\end{aligned}$$

and for $l = 1, \ldots, nf$ such that the edge E_l is on Γ_N

$$b_{edge_i}^l = \int_{E_l} \alpha g_N \phi_i d\sigma,$$

where

$$\begin{aligned}
\alpha_k &= (y_3^k - y_1^k)^2 + (x_1^k - x_3^k)^2, \\
\beta_k &= (y_1^k - y_2^k)^2 + (x_2^k - x_1^k)^2, \\
\gamma_k &= (y_3^k - y_1^k)(y_1^k - y_2^k) + (x_1^k - x_3^k)(x_2^k - x_1^k), \\
JF_{K_k} &= (x_2^k - x_1^k)(y_3^k - y_1^k) - (x_3^k - x_1^k)(y_2^k - y_1^k).
\end{aligned}$$

Due to the fact that the basis $\{\phi_i\}_{i=1}^{nv}$ is such that for all $i = 1, \ldots, nv$, the support of ϕ_i is reduced only to the set of triangles for which x_i is a vertex; all those contributions are 0 if x_i or x_j is not a vertex of the triangle K_k or the edge E_l.

Assuming that i, j, k are such that x_i and x_j are vertices of the triangle K_k; one can compute exactly the element contribution involving a derivative of ϕ_i:

$$\int_{\hat{K}} \frac{\partial\hat{\phi}_i}{\partial\hat{x}} \frac{\partial\hat{\phi}_j}{\partial\hat{x}} d\hat{x} d\hat{y} = \begin{cases} \frac{1}{2}, & \text{if } i \text{ and } j \text{ are both either the first or the second node of } K_k; \\ \\ -\frac{1}{2}, & \text{if } i \text{ is the first and } j \text{ is the second; or if } i \text{ is the second and} \\ & j \text{ is the first node of } K_k; \\ \\ 0 & \text{otherwise.} \end{cases}$$

$$\int_{\hat{K}} \frac{\partial \hat{\phi}_i}{\partial \hat{y}} \frac{\partial \hat{\phi}_j}{\partial \hat{y}} d\hat{x}d\hat{y} = \begin{cases} \frac{1}{2}, & \text{if } i \text{ and } j \text{ are both either the first or the third node of } K_k; \\ -\frac{1}{2}, & \text{if } i \text{ is the first and } j \text{ is the third; or if } i \text{ is the third and } j \text{ is the first node of } K_k; \\ 0 & \text{otherwise.} \end{cases}$$

$$\int_{\hat{K}} \frac{\partial \hat{\phi}_i}{\partial \hat{x}} \frac{\partial \hat{\phi}_j}{\partial \hat{y}} d\hat{x}d\hat{y} = \begin{cases} \frac{1}{2}, & \text{if } i \text{ and } j \text{ are both the first or if } i \text{ is the second and } j \text{ is the third node of } K_k; \\ -\frac{1}{2}, & \text{if } i \text{ is the first and } j \text{ is the third or if } i \text{ is the second and } j \text{ is the first node of } K_k; \\ 0 & \text{otherwise.} \end{cases}$$

This is due to the fact that the vertex x_i could be the first, the second or the third node of the triangle from which it is a vertex, for $i = 1, \ldots, nv$.

For those involving simply ϕ_i, one can use the $2D$ trapezoidal quadrature formula:

$$\int_{\hat{K}} \psi d\hat{x}d\hat{y} \approx \frac{1}{3}\text{area}(\hat{K}) \sum_{i=1}^{3} \psi(\hat{\mu}_i) = \frac{1}{6} \sum_{i=1}^{3} \psi(\hat{\mu}_i), \tag{9}$$

where $\hat{\mu}_i$ is the middle of the triangle $\hat{K}$ edge i; $\hat{\mu}_1 = (0.5, 0.5)$, $\hat{\mu}_2 = (0, 0.5)$, $\hat{\mu}_3 = (0.5, 0)$ and area$(\hat{K})$ is the area of the triangle $\hat{K}$. This integration formula is actually exact on P^2 (the space of polynomials of degree less than or equal to 2 in the two variables). Then

$$\int_{\hat{K}} \hat{\phi}_i \hat{\phi}_j d\hat{x}d\hat{y} \approx \begin{cases} \frac{1}{24}, & i \neq j, \\ \frac{1}{12}, & i = j; \end{cases}$$

$$\int_{\hat{K}} \hat{f}\hat{\phi}_i dxdy \approx \begin{cases} \frac{1}{12} f(x_1^k, y_1^k) + \frac{1}{24} f(x_2^k, y_2^k) + \frac{1}{24} f(x_3^k, y_3^k), \text{if } i \text{ is the first node of } K_k; \\ \frac{1}{24} f(x_1^k, y_1^k) + \frac{1}{12} f(x_2^k, y_2^k) + \frac{1}{24} f(x_3^k, y_3^k), \text{if } i \text{ is the second node of } K_k; \\ \frac{1}{24} f(x_1^k, y_1^k) + \frac{1}{24} f(x_2^k, y_2^k) + \frac{1}{12} f(x_3^k, y_3^k), \text{if } i \text{ is the third node of } K_k, \end{cases}$$

because of the fact that if $f \in \tilde{V}_h$ then for all $(x, y) \in K_k$;

$$f(x, y) = \hat{f}(\hat{x}, \hat{y}) = \sum_{l=1}^{3} f(x_l^k, y_l^k) \lambda_l(\hat{x}, \hat{y}).$$

Finally, assuming that for an i such that x_i is an extremity of the edge E_l on Γ_N; one can use the $1D$ trapezoidal quadrature formula

$$\int_a^b \psi(t)dt \approx \frac{(b-a)}{2}(\psi(b) + \psi(a)),$$

to compute the contribution on the boundary:

$$b^l_{edge_i} \approx \frac{\text{length}(E_l)}{2} \alpha g_N(x_i),$$

where length(E_l) is the length of the edge E_l.

6 Numerical Simulations

This example meshes the geometry shown in Figure 1 and solves the Helmholtz equation on the resulting mesh. The geometry is partitioned into three subdomains which are meshed separately and restitched using two calls to D06DBF. The boundary of the first two geometries are generated using D06BAF. The first geometry is then meshed by the Delaunay-Voronoi process (using D06ABF), the second one is meshed by an advancing front algorithm (using D06ACF), while the third one is the result of a rotation (by $-\pi/2$) of the second one (using D06DAF). The resulting geometry is then smoothed using D06CAF; a vertex renumbering is carried out and the sparsity structure of the matrix associated with the renumbered mesh is generated using D06CCF. The resulting geometry has 2718 vertices

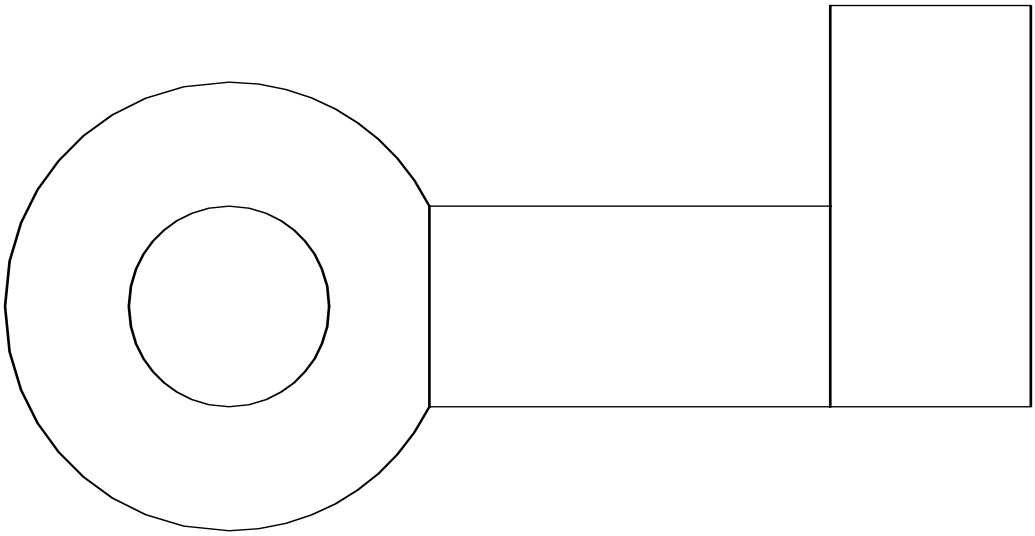

Figure 1: the boundary and the interior interfaces of the geometry.

and 5113 triangles, while the boundary and the internal interfaces have 361 edges. Figure 2 shows a coarser mesh with only 643 vertices, 1133 triangles and 171 edges on the boundary and the internal interfaces.

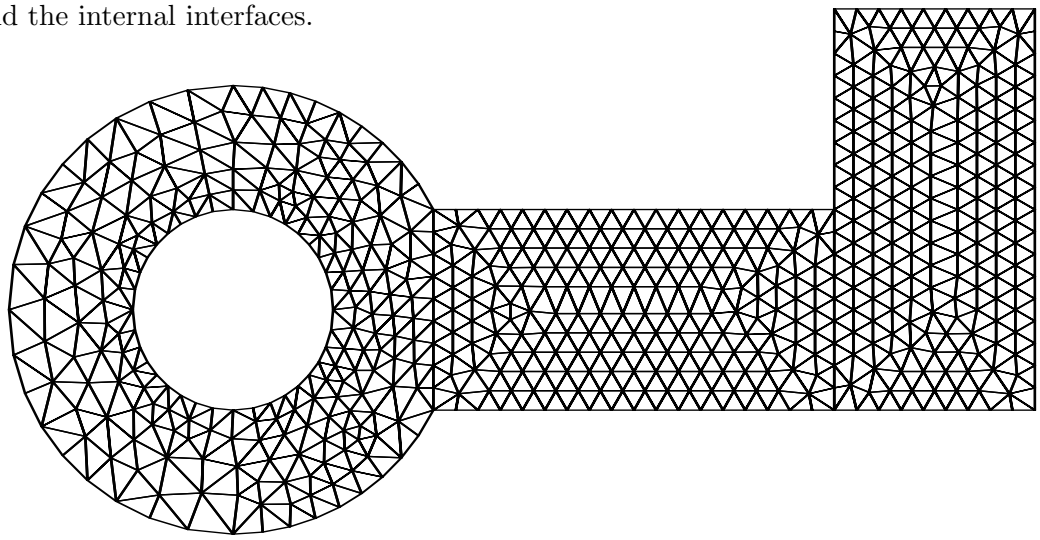

Figure 2: the interior mesh of the geometry.

The entire outer boundary is considered as a Dirichlet boundary (its edges are tagged via EDGE$(3,:)$ by 1, 2 or 3), while the inner boundary is a Neumann boundary (its edges are tagged by 5).

To check the validity of the matrix A, formulae (5) and (6) are verified in the program below, associated with the integration formula (9). This test is not numerically efficient in some cases but it can give an idea of the validity of the matrix. A better check can be carried out with any polynomial of degree less than or equal to 1, for which the result should be exact (in the data file choose TYPE = 'P1').

A typical value of both α and β might be 1, and for this test $u(x) = x^2 + y^2$ is considered (TYPE = 'P2'). The user is free to choose the method of solution of the sparse linear system:

- SOLUT = 'D' for a direct method. An LU desomposition of the matrix is performed using F11DAF; then the system is solved using that decomposition via a call to F11DBF.
- SOLUT = 'I' for an iterative method. An incomplete LU decomposition of the matrix is performed using F11DAF; then the system is solved by a call to F11DCF using either
  - a restarted generalised minimal residual method (METHOD = 'RGMRES');
  - a conjugate gradient squared method (METHOD = 'CGS');
  - a stabilised bi-conjugate gradient method (METHOD = 'BICGSTAB');
  - or a transpose-free quasi-minimal residual method (METHOD = 'TFQMR');

with incomplete LU preconditioning. The user is advised to consult [1] for the documentation of all the D06 and the F11 routines used in this program.

At the end of the program the norm of the error between the true and the computed solution, as well as its gradient are computed (using formulae (7), (8) and (9)). It can be proven (consult [4]) that the relative error is of order h^2 while the relative error on the gradient of the solution is of order h.

6.1 Program Text

```
*     Tutorial Example Program Text
*     Mark 20 Release. NAG Copyright 2001
C     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NEDMX, NVMAX, NELTMX, NLINEX, NUS, NCOMPX, NVIMX,
     +                 MAXCAN, NNZMAX, NVFXMX, LRWORK, LIWORK
      PARAMETER        (NEDMX=1000,NVMAX=6000,NELTMX=2*NVMAX+5,
     +                 NLINEX=50,NUS=100,NCOMPX=5,NVIMX=20,MAXCAN=10000,
     +                 NNZMAX=50*NVMAX,NVFXMX=20,
```

```
     +                 LRWORK=12*NVMAX+3*MAXCAN+15,
     +                 LIWORK=8*NEDMX+55*NVMAX+MAXCAN+78)
      DOUBLE PRECISION C12, C24, HALF, MHALF, ZERO
      PARAMETER        (C12=1.D0/12.D0,C24=1.D0/24.D0,HALF=0.5D0,
     +                 MHALF=-HALF,ZERO=0.D0)
C     .. Local Arrays ..
      DOUBLE PRECISION AMAT(NNZMAX), CHKMAT(NVMAX), COOR(2,NVMAX),
     +                 COOR1(2,NVMAX), COOR2(2,NVMAX), COOR3(2,NVMAX),
     +                 COOR4(2,NVMAX), COORCH(2,NLINEX), COORUS(2,NUS),
     +                 DXDX(3,3), DXDY(3,3), DYDX(3,3), DYDY(3,3),
     +                 F(NVMAX), PHI(3,3), RATE(NLINEX), RHS(NVMAX),
     +                 RUSER(1), RWORK(LRWORK), TRANS(6,1), UDIR(NVMAX),
     +                 UNEU(NVMAX), USOL(NVMAX), UTRUE(NVMAX),
     +                 WEIGHT(NVIMX)
      INTEGER          CONN(3,NELTMX), CONN1(3,NELTMX), CONN2(3,NELTMX),
     +                 CONN3(3,NELTMX), CONN4(3,NELTMX), EDGE(3,NEDMX),
     +                 EDGE1(3,NEDMX), EDGE2(3,NEDMX), EDGE3(3,NEDMX),
     +                 EDGE4(3,NEDMX), ICOL(NNZMAX), IROW(NNZMAX),
     +                 IROW1(NVMAX+1), ITYPE(1), IUSER(1),
     +                 IWORK(LIWORK), LCOMP(NLINEX), LINE(4,NLINEX),
     +                 NLCOMP(NCOMPX), NUMFIX(NVFXMX), REFT(NELTMX),
     +                 REFT1(NELTMX), REFT2(NELTMX), REFT3(NELTMX),
     +                 REFT4(NELTMX)
C     .. Local Scalars ..
      DOUBLE PRECISION ALPHA, ALPHAK, BETA, BETAK, DL12, DTOL, EPS, FK1,
     +                 FK2, FK3, GAMMAK, H1, H1NRM, H1NRM0, H2, H3,
     +                 HSIZE, JFK, L2NRM, L2NRM0, RNORM, TOL, XK1, XK2,
     +                 XK3, XL1, XL2, YK1, YK2, YK3, YL1, YL2
      INTEGER          I, IFAIL, IJ, IK, ITN, ITRACE, J, JK, K, K1, K2,
     +                 K3, L, L1, L2, LFILL, LL1, LL2, M, MAXITN, NCOMP,
     +                 NEDGE, NEDGE1, NEDGE2, NEDGE3, NEDGE4, NELT,
     +                 NELT1, NELT2, NELT3, NELT4, NLINES, NNZ, NNZC,
     +                 NPIVM, NPROPA, NQINT, NTRANS, NV, NV1, NV2, NV3,
     +                 NV4, NVB1, NVB2, NVFIX, NVINT
      CHARACTER        CHECK, MILU, PSTRAT, SOLUT, TRANS1
      CHARACTER*2      TYPE
      CHARACTER*8      METHOD
C     .. External Subroutines ..
      EXTERNAL         D06ABF, D06ACF, D06BAF, D06CAF, D06CCF, D06DAF,
     +                 D06DBF, F06DBF, F11DAF, F11DBF, F11DCF
C     .. External Functions ..
      DOUBLE PRECISION FBND, X02AJF
      EXTERNAL         FBND, X02AJF
C     .. Intrinsic Functions ..
      INTRINSIC        ABS, DABS, MAX, SQRT
C     .. Data statements ..
      DATA             PHI/C12, C24, C24, C24, C12, C24, C24, C24, C12/
      DATA             DXDX/HALF, MHALF, ZERO, MHALF, HALF, ZERO, ZERO,
     +                 ZERO, ZERO/
```

```
      DATA              DYDY/HALF, ZERO, MHALF, ZERO, ZERO, ZERO, MHALF,
     +                  ZERO, HALF/
      DATA              DXDY/HALF, MHALF, ZERO, ZERO, ZERO, ZERO, MHALF,
     +                  HALF, ZERO/
      DATA              DYDX/HALF, ZERO, MHALF, MHALF, ZERO, HALF, ZERO,
     +                  ZERO, ZERO/
C     .. Executable Statements ..
      WRITE (NOUT,FMT='(A)') 'Tutorial Example Results '
*
*     Build the mesh of the 1st domain
*
*     Initialise boundary mesh inputs:
*     the number of lines and characteristic points on
*     the boundary mesh
*
      READ (NIN,FMT=*)
*
      READ (NIN,FMT=*) NLINES
*
*     Characteristic points of the boundary geometry
*
      READ (NIN,FMT=*) (COORCH(1,J),J=1,NLINES)
      READ (NIN,FMT=*) (COORCH(2,J),J=1,NLINES)
*
*     The lines of the boundary mesh
*
      READ (NIN,FMT=*) ((LINE(I,J),I=1,4),RATE(J),J=1,NLINES)
*
*     The number of connected components on the boundary
*     and their specification
*
      READ (NIN,FMT=*) NCOMP
      J = 1
      DO 20 I = 1, NCOMP
         READ (NIN,FMT=*) NLCOMP(I)
*
         READ (NIN,FMT=*) (LCOMP(K),K=J,J+ABS(NLCOMP(I))-1)
         J = J + ABS(NLCOMP(I))
   20 CONTINUE
*
*     Call to the 2D boundary mesh generator
*
      ITRACE = 0
      IFAIL = 0
*
      CALL D06BAF(NLINES,COORCH,LINE,FBND,COORUS,NUS,RATE,NCOMP,NLCOMP,
     +            LCOMP,NVMAX,NEDMX,NVB1,COOR1,NEDGE1,EDGE1,ITRACE,
     +            RUSER,IUSER,RWORK,LRWORK,IWORK,LIWORK,IFAIL)
*
```

```
*     Mesh using Delaunay-Voronoi method
*
*     Initialise mesh control parameters
*
      ITRACE = 0
      NPROPA = 1
      NVINT = 0
      IFAIL = 0
*
*     Call to the 2D Delaunay-Voronoi mesh generator
*
      CALL D06ABF(NVB1,NVINT,NVMAX,NEDGE1,EDGE1,NV1,NELT1,COOR1,CONN1,
     +            WEIGHT,NPROPA,ITRACE,RWORK,LRWORK,IWORK,LIWORK,IFAIL)
*
*     Build the mesh of the 2nd domain
*
*     Initialise boundary mesh inputs:
*     the number of lines and characteristic points on
*     the boundary mesh
*
      READ (NIN,FMT=*) NLINES
*
*     Characteristic points of the boundary geometry
*
      READ (NIN,FMT=*) (COORCH(1,J),J=1,NLINES)
      READ (NIN,FMT=*) (COORCH(2,J),J=1,NLINES)
*
*     The lines of the boundary mesh
*
      READ (NIN,FMT=*) ((LINE(I,J),I=1,4),RATE(J),J=1,NLINES)
*
*     The number of connected components on the boundary
*     and their specification
*
      READ (NIN,FMT=*) NCOMP
      J = 1
      DO 40 I = 1, NCOMP
         READ (NIN,FMT=*) NLCOMP(I)
*
         READ (NIN,FMT=*) (LCOMP(K),K=J,J+ABS(NLCOMP(I))-1)
         J = J + ABS(NLCOMP(I))
   40 CONTINUE
*
*     Solution method 'D' direct method or 'I' iterative method
*
      READ (NIN,FMT=*) SOLUT
*
*     The iterative method to be used RGMRES, CGS, BICGSTAB or TFQMR
*
```

```
      READ (NIN,FMT=*) METHOD
*
*     Type of problem which will be solved P1 or P2
*
      READ (NIN,FMT=*) TYPE
*
      ITRACE = 0
*
*     Call to the 2D boundary mesh generator
*
      IFAIL = 0
*
      CALL D06BAF(NLINES,COORCH,LINE,FBND,COORUS,NUS,RATE,NCOMP,NLCOMP,
     +            LCOMP,NVMAX,NEDMX,NVB2,COOR2,NEDGE2,EDGE2,ITRACE,
     +            RUSER,IUSER,RWORK,LRWORK,IWORK,LIWORK,IFAIL)
*
*     Mesh using the advancing front method
*
*     Initialise mesh control parameters
*
      ITRACE = 0
      NVINT = 0
      IFAIL = 0
*
*     Call to the 2D advancing front mesh generator
*
      CALL D06ACF(NVB2,NVINT,NVMAX,NEDGE2,EDGE2,NV2,NELT2,COOR2,CONN2,
     +            WEIGHT,ITRACE,RWORK,LRWORK,IWORK,LIWORK,IFAIL)
*
*     Rotation of the 2nd domain mesh to produce
*     the 3rd mesh domain
*
      NTRANS = 1
      ITYPE(1) = 3
      TRANS(1,1) = 6.D0
      TRANS(2,1) = -1.D0
      TRANS(3,1) = -90.D0
      ITRACE = 0
      IFAIL = 0
*
      CALL D06DAF(NV2,NEDGE2,NELT2,NTRANS,ITYPE,TRANS,COOR2,EDGE2,CONN2,
     +            COOR3,EDGE3,CONN3,ITRACE,RWORK,LRWORK,IFAIL)
*
      NV3 = NV2
      NELT3 = NELT2
      NEDGE3 = NEDGE2
*
*     Tag triangles and boundary edges for all 3 domains
*
```

```
      DO 60 K = 1, NELT1
         REFT1(K) = 1
   60 CONTINUE
*
      DO 80 K = 1, NELT2
         REFT2(K) = 2
   80 CONTINUE
*
      DO 100 K = 1, NELT3
         REFT3(K) = 3
  100 CONTINUE
*
      DO 120 K = 1, NEDGE1
         EDGE1(3,K) = 1
  120 CONTINUE
*
      DO 140 K = 1, NEDGE2
         EDGE2(3,K) = 2
  140 CONTINUE
*
      DO 160 K = 1, NEDGE3
         EDGE3(3,K) = 3
  160 CONTINUE
*
*     Restitching of meshes 1 and 2 to form mesh 4
*
      EPS = 1.D-3
      ITRACE = 0
      IFAIL = 0
*
      CALL D06DBF(EPS,NV1,NELT1,NEDGE1,COOR1,EDGE1,CONN1,REFT1,NV2,
     +            NELT2,NEDGE2,COOR2,EDGE2,CONN2,REFT2,NV4,NELT4,NEDGE4,
     +            COOR4,EDGE4,CONN4,REFT4,ITRACE,IWORK,LIWORK,IFAIL)
*
*     Restitching of meshes 3 and 4 to form the final mesh
*
      ITRACE = 0
      IFAIL = 0
*
      CALL D06DBF(EPS,NV4,NELT4,NEDGE4,COOR4,EDGE4,CONN4,REFT4,NV3,
     +            NELT3,NEDGE3,COOR3,EDGE3,CONN3,REFT3,NV,NELT,NEDGE,
     +            COOR,EDGE,CONN,REFT,ITRACE,IWORK,LIWORK,IFAIL)
*
*     Call the smoothing routine
*
      NVFIX = 0
      NQINT = 10
*
      CALL D06CAF(NV,NELT,NEDGE,COOR,EDGE,CONN,NVFIX,NUMFIX,ITRACE,
```

```
     +            NQINT,IWORK,LIWORK,RWORK,LRWORK,IFAIL)
*
*     Call the renumbering routine and get the new sparsity
*
      IFAIL = 0
      ITRACE = 0
      CALL D06CCF(NV,NELT,NEDGE,NNZMAX,NNZ,COOR,EDGE,CONN,IROW,ICOL,
     +            ITRACE,IWORK,LIWORK,RWORK,LRWORK,IFAIL)
*
*     Special tag for the Neumann boundary
*
      EPS = 1.D-5
      DO 180 K = 1, NEDGE
         IF ((DABS(COOR(1,EDGE(1,K))**2+COOR(2,EDGE(1,K))**2-1.D0)
     +       .LT.EPS) .AND. (DABS(COOR(1,EDGE(2,K))**2+COOR(2,EDGE(2,K))
     +       **2-1.D0).LT.EPS)) THEN
            EDGE(3,K) = 5
         END IF
  180 CONTINUE
*
*     Find number of elements in each row.
*
      CALL F06DBF(NV,0,IWORK,1)
*
      DO 200 I = 1, NNZ
         IWORK(IROW(I)) = IWORK(IROW(I)) + 1
         AMAT(I) = 0.D0
  200 CONTINUE
*
*     Set up pointer to start of each row and
*     set data for the Helmholtz problem
*
      ALPHA = 1.D0
      BETA = 1.D0
      IROW1(1) = 1
*
      DO 220 I = 1, NV
         IROW1(I+1) = IROW1(I) + IWORK(I)
         RHS(I) = 0.D0
         CHKMAT(I) = 0.D0
*
         IF (TYPE.EQ.'P2') THEN
            UTRUE(I) = COOR(1,I)**2 + COOR(2,I)**2
            F(I) = BETA*UTRUE(I) - 4.D0*ALPHA
         ELSE IF (TYPE.EQ.'P1') THEN
            UTRUE(I) = 1.D0 + COOR(1,I) + COOR(2,I)
            F(I) = BETA*UTRUE(I)
         END IF
*
```

```
         UDIR(I) = 0.D0
         UNEU(I) = 0.D0
         USOL(I) = 0.D0
  220 CONTINUE
*
*     Boundary conditions
*
      DO 260 L = 1, NEDGE
         L1 = EDGE(1,L)
         L2 = EDGE(2,L)
*
         IF (EDGE(3,L).EQ.1 .OR. EDGE(3,L).EQ.2 .OR. EDGE(3,L).EQ.3)
     +       THEN
            UDIR(L1) = UTRUE(L1)
            UDIR(L2) = UTRUE(L2)
*
            USOL(L1) = UDIR(L1)
            USOL(L2) = UDIR(L2)
         ELSE IF (EDGE(3,L).EQ.5) THEN
            IF (TYPE.EQ.'P2') THEN
               UNEU(L1) = -2.D0
               UNEU(L2) = -2.D0
            ELSE IF (TYPE.EQ.'P1') THEN
               UNEU(L1) = -COOR(1,L1) - COOR(2,L1)
               UNEU(L2) = -COOR(1,L2) - COOR(2,L2)
            END IF
         END IF
  260 CONTINUE
*
*     Compute the FE matrix and the right-hand side of the linear system
*
      HSIZE = -1.D0
      DO 340 K = 1, NELT
         K1 = CONN(1,K)
         K2 = CONN(2,K)
         K3 = CONN(3,K)
*
         XK1 = COOR(1,K1)
         YK1 = COOR(2,K1)
*
         XK2 = COOR(1,K2)
         YK2 = COOR(2,K2)
*
         XK3 = COOR(1,K3)
         YK3 = COOR(2,K3)
*
         H1 = SQRT((XK2-XK3)**2+(YK2-YK3)**2)
         H2 = SQRT((XK1-XK3)**2+(YK1-YK3)**2)
         H3 = SQRT((XK2-XK1)**2+(YK2-YK1)**2)
```

```
*
         HSIZE = MAX(H1,HSIZE)
         HSIZE = MAX(H2,HSIZE)
         HSIZE = MAX(H3,HSIZE)
*
         FK1 = F(K1)
         FK2 = F(K2)
         FK3 = F(K3)
*
         JFK = (XK2-XK1)*(YK3-YK1) - (XK3-XK1)*(YK2-YK1)
         ALPHAK = (YK3-YK1)**2 + (XK3-XK1)**2
         BETAK = (YK2-YK1)**2 + (XK2-XK1)**2
         GAMMAK = (YK3-YK1)*(YK1-YK2) + (XK1-XK3)*(XK2-XK1)
*
         DO 320 I = 1, 3
            IK = CONN(I,K)
            CHKMAT(IK) = CHKMAT(IK) + BETA*JFK/6.D0
*
*     Element contribution to the FE matrix
*
            DO 300 J = 1, 3
               JK = CONN(J,K)
               IFAIL = 0
*
               L2NRM = 1.D0/JFK*(ALPHAK*DXDX(I,J)+GAMMAK*(DXDY(I,J)
     +                  +DYDX(I,J))+BETAK*DYDY(I,J))
               L2NRM0 = JFK*PHI(I,J)
*
               DO 280 IJ = IROW1(IK), IROW1(IK+1) - 1
                  IF (ICOL(IJ).EQ.JK) THEN
                     AMAT(IJ) = AMAT(IJ) + ALPHA*L2NRM + BETA*L2NRM0
                  END IF
  280          CONTINUE
  300       CONTINUE
*
*     Right-hand side contributions
*
            RHS(IK) = RHS(IK) + JFK*(FK1*PHI(I,1)+FK2*PHI(I,2)
     +                +FK3*PHI(I,3))
  320    CONTINUE
  340 CONTINUE
*
*     Check of the FE matrix
*
      EPS = 1.D+1*X02AJF()
*
      DO 380 I = 1, NV
         L2NRM = 0.D0
         DO 360 J = IROW1(I), IROW1(I+1) - 1
```

```
            L2NRM = L2NRM + AMAT(J)
  360    CONTINUE
*
         IF (ABS(CHKMAT(I)-L2NRM).GT.EPS) THEN
            WRITE (*,FMT=*) ' Prob with FE Matrix ', I, IROW1(I),
     +        IROW1(I+1) - 1, L2NRM, CHKMAT(I)
         END IF
  380 CONTINUE
*
*     Set the boundary conditions
*
      IFAIL = 0
      DO 440 L = 1, NEDGE
         L1 = EDGE(1,L)
         L2 = EDGE(2,L)
*
         XL1 = COOR(1,L1)
         YL1 = COOR(2,L1)
*
         XL2 = COOR(1,L2)
         YL2 = COOR(2,L2)
*
         DL12 = SQRT((XL2-XL1)**2+(YL2-YL1)**2)
*
*     Dirichlet
*
         IF (EDGE(3,L).EQ.1 .OR. EDGE(3,L).EQ.2 .OR. EDGE(3,L).EQ.3)
     +       THEN
            RHS(L1) = UDIR(L1)
            RHS(L2) = UDIR(L2)

            DO 400 LL1 = IROW1(L1), IROW1(L1+1) - 1
               AMAT(LL1) = 0.D0
*
               IF (ICOL(LL1).EQ.L1) THEN
                  AMAT(LL1) = 1.D0
               END IF
  400       CONTINUE
*
            DO 420 LL2 = IROW1(L2), IROW1(L2+1) - 1
               AMAT(LL2) = 0.D0
*
               IF (ICOL(LL2).EQ.L2) THEN
                  AMAT(LL2) = 1.D0
               END IF
  420       CONTINUE
*
*     Neumann
*
```

```
         ELSE IF (EDGE(3,L).EQ.5) THEN
            RHS(L1) = RHS(L1) + 0.5D0*ALPHA*DL12*UNEU(L1)
            RHS(L2) = RHS(L2) + 0.5D0*ALPHA*DL12*UNEU(L2)
         END IF
  440 CONTINUE
*
      IF (SOLUT.EQ.'D') THEN
*
*     Direct solution
*     Compute the LU factorization of the FE matrix
*
         LFILL = -1
         DTOL = 0.D0
         PSTRAT = 'C'
         MILU = 'N'
         IFAIL = 0
*
         CALL F11DAF(NV,NNZ,AMAT,NNZMAX,IROW,ICOL,LFILL,DTOL,PSTRAT,
     +               MILU,IWORK,IWORK(NV+1),IROW1,IWORK(2*NV+1),NNZC,
     +               NPIVM,IWORK(3*NV+2),(LIWORK-3*NV-1),IFAIL)
*
*     Solve the system AMAT*USOL = RHS
*
         TRANS1 = 'N'
         CHECK = 'C'
         IFAIL = 0
*
         CALL F11DBF(TRANS1,NV,AMAT,NNZMAX,IROW,ICOL,IWORK,IWORK(NV+1),
     +               IROW1,IWORK(2*NV+1),CHECK,RHS,USOL,IFAIL)
*
         WRITE (NOUT,FMT='(A,I10)') 'Dimension of the system = ', NV
         WRITE (NOUT,FMT='(A,A,I10)') 'Number of triangles of the ',
     +     'mesh = ', NELT
         WRITE (NOUT,FMT='(A,I10)') 'Number of edges = ', NEDGE
      ELSE IF (SOLUT.EQ.'I') THEN
*
*     Iterative solution
*     Compute the incomplete LU factorisation
*
         LFILL = 1
         DTOL = 0.D0
         PSTRAT = 'C'
         MILU = 'N'
         IFAIL = 0
*
         CALL F11DAF(NV,NNZ,AMAT,NNZMAX,IROW,ICOL,LFILL,DTOL,PSTRAT,
     +               MILU,IWORK(1),IWORK(NV+1),IROW1,IWORK(2*NV+1),NNZC,
     +               NPIVM,IWORK(3*NV+2),(LIWORK-3*NV-1),IFAIL)
*
```

```
*     Solve the system AMAT*USOL = RHS
*
         TOL = 1.D-12
         MAXITN = NV
         M = 10
         IFAIL = -1
*
         CALL F11DCF(METHOD,NV,NNZ,AMAT,NNZMAX,IROW,ICOL,IWORK(1),
     +               IWORK(NV+1),IROW1,IWORK(2*NV+1),RHS,M,TOL,MAXITN,
     +               USOL,RNORM,ITN,RWORK,LRWORK,IFAIL)
*
         WRITE (NOUT,FMT='(A,I10)') 'Dimension of the system = ', NV
         WRITE (NOUT,FMT='(A,A,I10)') 'Number of triangles of the ',
     +     'mesh = ', NELT
         WRITE (NOUT,FMT='(A,I10)') 'Number of edges = ', NEDGE
         WRITE (NOUT,FMT='(A,I10,A)') 'Converged in', ITN, ' iterations'
         WRITE (NOUT,FMT='(A,1P,D16.3)') 'Final residual norm =', RNORM
         WRITE (NOUT,FMT=*)
      END IF
*
*     L2 and H1 Error estimations
*
      L2NRM = 0.D0
      L2NRM0 = 0.D0
      H1NRM = 0.D0
      H1NRM0 = 0.D0

      DO 460 K = 1, NELT
         K1 = CONN(1,K)
         K2 = CONN(2,K)
         K3 = CONN(3,K)
*
         XK1 = COOR(1,K1)
         YK1 = COOR(2,K1)
*
         XK2 = COOR(1,K2)
         YK2 = COOR(2,K2)
*
         XK3 = COOR(1,K3)
         YK3 = COOR(2,K3)
*
         JFK = (XK2-XK1)*(YK3-YK1) - (XK3-XK1)*(YK2-YK1)
         ALPHAK = (YK3-YK1)**2 + (XK3-XK1)**2
         BETAK = (YK2-YK1)**2 + (XK2-XK1)**2
         GAMMAK = (YK3-YK1)*(YK1-YK2) + (XK1-XK3)*(XK2-XK1)
*
         FK1 = USOL(K1)
         FK2 = USOL(K2)
         FK3 = USOL(K3)
```

```
*
         XK1 = UTRUE(K1)
         XK2 = UTRUE(K2)
         XK3 = UTRUE(K3)
*
         L2NRM0 = L2NRM0 + JFK*(XK1*XK1+XK2*XK2+XK3*XK3)/6.D0
         H1NRM0 = H1NRM0 + 1.D0/(2.D0*JFK)*(ALPHAK*(XK2-XK1)*(XK2-XK1)
     +            +2.D0*GAMMAK*(XK2-XK1)*(XK3-XK1)+BETAK*(XK3-XK1)
     +            *(XK3-XK1))
*
         XK1 = XK1 - FK1
         XK2 = XK2 - FK2
         XK3 = XK3 - FK3
*
         L2NRM = L2NRM + JFK*(XK1*XK1+XK2*XK2+XK3*XK3)/6.D0
         H1NRM = H1NRM + 1.D0/(2.D0*JFK)*(ALPHAK*(XK2-XK1)*(XK2-XK1)
     +           +2.D0*GAMMAK*(XK2-XK1)*(XK3-XK1)+BETAK*(XK3-XK1)
     +           *(XK3-XK1))
  460 CONTINUE
*
      L2NRM = SQRT(L2NRM)
      L2NRM0 = SQRT(L2NRM0)
      H1NRM = SQRT(H1NRM)
      H1NRM0 = SQRT(H1NRM0)
      ALPHAK = L2NRM/L2NRM0
      BETAK = H1NRM/H1NRM0
*
      WRITE (NOUT,FMT=*) 'L2 Error = ', L2NRM
      WRITE (NOUT,FMT=*) 'Relative L2 Error = ', ALPHAK
      WRITE (NOUT,FMT=*) 'H1 Error = ', H1NRM
      WRITE (NOUT,FMT=*) 'Relative H1 Error = ', BETAK
      WRITE (NOUT,FMT=*) '         Mesh step = ', HSIZE
*
      STOP
      END
*
      DOUBLE PRECISION FUNCTION FBND(I,X,Y,RUSER,IUSER)
C     .. Scalar Arguments ..
      DOUBLE PRECISION               X, Y
      INTEGER                        I
C     .. Array Arguments ..
      DOUBLE PRECISION               RUSER(*)
      INTEGER                        IUSER(*)
C     .. Local Scalars ..
      DOUBLE PRECISION               R2, X0, Y0
C     .. Executable Statements ..
      FBND = 0.D0
      IF (I.EQ.1) THEN
*
```

```
*     inner circle
*
         X0 = 0.D0
         Y0 = 0.D0
         R2 = 1.D0
         FBND = (X-X0)**2 + (Y-Y0)**2 - R2
      ELSE IF (I.EQ.2) THEN
*
*     outer circle
*
         X0 = 0.D0
         Y0 = 0.D0
         R2 = 5.D0
         FBND = (X-X0)**2 + (Y-Y0)**2 - R2
      END IF
*
      RETURN
      END
```

6.2 Program Data

```
Tutorial Example program data
 9                                  1st geometry NLINES (m)
  2.0000   2.0000   1.0000
 -1.0000  -2.2361   0.0000
  0.0000   0.0000   0.0000                (COORCH(1,1:m))
 -1.0000   1.0000   0.0000
  0.0000   0.0000  -2.2361
 -1.0000   1.0000   2.2361                (COORCH(2,1:m))
20  1  2  0   1.0000 20  2  9  2   1.0000
20  9  5  2   1.0000 20  5  6  2   1.0000
20  6  1  2   1.0000 20  3  8  1   1.0000
20  8  4  1   1.0000 20  4  7  1   1.0000
20  7  3  1   1.0000            (LINE(:,j),RATE(j),j=1,m)
 2                              NCOMP (n, number of contours)
 5                               number of lines in contour 1
 1 2 3 4 5                                lines of contour 1
-4                               number of lines in contour 2
 9 8 7 6                                 lines of contour 2
 4                                    2nd geometry NLINES (m)
  2.0000   6.0000   6.0000   2.0000       (COORCH(1,1:m))
 -1.0000  -1.0000   1.0000   1.0000       (COORCH(2,1:m))
39  1  2  0   1.0000 20  2  3  0   1.0000
39  3  4  0   1.0000 20  4  1  0   1.0000 (LINE(:,j),RATE(j),j=1,m)
 1                              NCOMP (n, number of contours)
 4                               number of lines in contour 1
 1 2 3 4                                  lines of contour 1
```

```
'I'      SOLUT, Solution method: 'D' Direct or 'I' Iterative method
'CGS'                            METHOD:Iterative method to be used
'P2'                             TYPE:type of problem to be solved
```

6.3 Program Results

```
Tutorial Example Results
Dimension of the system =       2718
Number of triangles of the mesh =       5113
Number of edges =        361
Converged in        21 iterations
Final residual norm =       2.857D-10

 L2 Error =    4.6909766199424103E-03
 Relative L2 Error =    2.9518468203940402E-05
 H1 Error =    5.6430901688359333E-02
 Relative H1 Error =    1.1607758257109531E-03
          Mesh step =    0.2861529929873701
```

7 Conclusion

This report has demonstrated the use of a combination of NAG Fortran Library routines (from the D06 and the F11 Chapters) in the solution of a simple PDE using the finite element method. In a forthcoming release of the Library more routines dedicated to finite element problems will be provided.

I would like to thank the MODULEF group from INRIA Rocquencourt for the use of their software as the basis of routines in the D06 Chapter, and Dr G.J. Shaw for his helpful comments.

References

[1] *The NAG Fortran Library, Mark 20* (2001) NAG Ltd, Wilkinson House, Jordan Hill Road, Oxford, UK.

[2] P. L. George and H. Borouchakis (1998) *Delaunay Triangulation and Meshing: Application to Finite Elements* Editions HERMES, Paris.

[3] Y. K. Cheug, S. H. Lo and A. Y. T. Leung (1996) *Finite Element Implementation* Blackwell Science.

[4] A. Quateroni and A. Valli (1997) *Numerical Approximation of Partial Differential Equations* Comp. Maths **23**.