Technical University of Chemnitz–Zwickau
Department of Mathematics

Gerhard Globisch

On an automatically parallel generation technique for tetrahedral meshes

Summary :

In order to prepare modern finite element analysis a program for the efficient parallel generation of tetrahedral meshes in a wide class of three dimensional domains having a generalized cylindric shape is presented. The applied mesh generation strategy is based on the decomposition of some 2D-reference domain into single connected subdomains by means of its triangulations the tetrahedral layers are built up in parallel. Adaptive grid controlling as well as nodal renumbering algorithms are involved. In the paper several examples are incorporated to demonstrate both program's capabilities and the handling with.

Preprint Reihe der Chemnitzer DFG-Forschergruppe
"Scientific Parallel Computing"

SPC 94_6 **April, 1994**

1 Introduction

The state of the art in research and development more and more adresses the problem of generation a mesh of nodes and elements to discretize a solid object really in three dimensions. Such a mesh must be of the appropriate structure and quality to form the basis of a mathematical model the modern techniques of the numerical analysis can be applied to. A good survey of the topic of mesh generation up to now can be found e.g. in [10, 19]. In recent time the parallel computing of large scale field problems e.g. arisen in solid mechanics, electrical engineering and computational fluid dynamics based upon domain decomposition is very much under discussion, (see e.g. [2, 7, 8, 9]). There is no doubt that the discretization of the underlying partial differential equation and the applied parallel solution method still require the main effort of the computational time during problem's simulation. However the numerical ingredients of the main processing capitalize from the mesh data structure generated adaptively in parallel. Moreover the produced mesh must reflect the geometry of the domain with sufficient resolution to model accurately the effects of geometric detail to a degree especially appropriated for the adaptive numerical analysis, cf. [9, 13, 14, 15, 11]. In [4] a parallel mesh generator for the efficient parallel generation of coarse or finer triangulations in arbitrary bounded plane domains was presented. Special advantages of the advancing front technique of the underlying algorithm (see [4, 20]) were the ability to vary the mesh density within the domain as well as to adapt the sizes of the triangles to geometric peculiarities e.g. such as point singularities. Furthermore the coupling of this mesh generation technique with further hierarchical mesh refinement procedures (see [4]) using the produced edge related element-data structure was proven to be very effective.

The aim of this paper is the extension of our parallel triangular mesh generator into the third dimension conserving both its effectiveness, adaptivity, compatibility and versatility now based on the edge related data structure for tetrahedral elements. For distinct opportunities to perform load balanced triangular mesh generation the efficiency of our 3D-mesh generator is essentially determined by we refer back to [4, 18]. Provided that this adequate handling with the program is guaranteed our parallel mesh generator will be much more efficient than every conventional one, whereas often the latter tools are capable of sequential mesh generating in 3D-regions of near arbitrary shape, cf. e.g. [13, 14]. But the introduced mesh generation strategy can be generalized by the description of more variable curvilinear boundary-faces to become more robust in this sense.

In section 2 we describe the specific mesh generation method for producing regularly connected tetrahedral layers based on its 2D-reference triangulations. In addition to we explain the structure of the input-data file the geometry of the class of meshable 3D-domains is reflected by, where the specification of the corresponding boundary conditions is incorporated into. In section 3 the background for the parallelization of the mesh generation is given, where the other program's capabilities e.g. such as parallel grid smoothing and (inner) nodal renumbering are overviewed. Section 4 gives the output-data structure of the tetrahedral mesh. Here we are able to complete the edge-related data structure of triangular meshes introduced in [4] appropriately. Finally in section 5 several numerical examples are involved in order to demonstrate both user's operating with the program and its efficiency.

2 The specific 3D-mesh generation strategy

In [4] the problem of parallel triangular mesh generation via domain decomposition of the 2D-domain to be meshed was solved universally. In other words, provided that the number p of available processors and the involved memory size of the given parallel computer architecture is large enough in every bounded plane domain $\mathbf{\Omega}$, which was decomposed into q single connected subdomains $\mathbf{\Omega}_i$, a sufficiently dense initial triangulation can be generated in parallel. Here the parallelization is performed by the well defined mapping of the q single connected subdomains onto the p processors, where each processor generates adaptively the triangulation in the subdomain its boundary description he got. The boundary curve of every single connected subdomain must be piecewisely consisted of straight lines and arcs of circles or parabolae.

Our applied tetrahedral mesh generation strategy consists in a parallel performed generalized cylindric expansion of even this triangular subdomain meshes called 2D-reference triangulations into the third (z)-dimension of space.

2.1 On the generalized cylindric extension of an 2D-reference domain

Let the domain decomposition of the bounded plane domain $\mathbf{\Omega}$ be given, i.e. we have

$$\bar{\Omega} = \bigcup_{i=1}^{q} \bar{\Omega}_i \quad \subset \quad \mathcal{H} \ .$$

The plane $\mathcal{H}$ in which the domain $\mathbf{\Omega}$ is given divides the three dimensional space into two half spaces. We denote the parallel generated meshes in every single connected subdomain of $\mathbf{\Omega}$ to be 2D-reference triangulations. Starting from the reference triangulations tetrahedral layers are constructed in parallel up to different heights in each case, where the layers are defined orthogonally with respect to the plane $\mathcal{H}$ all of the corresponding subdomains are included in. Figure 1 presents three simple cases for determining the 2D-reference domain decomposed into several single connected subdomains. Finally the 2D-subdomains get the cylindric property by the well specified extension into only one direction of the corresponding half space (body (A)) or into the both half spaces (bodies (B) and (C)) into which the three dimensional space was separated by the reference plane.

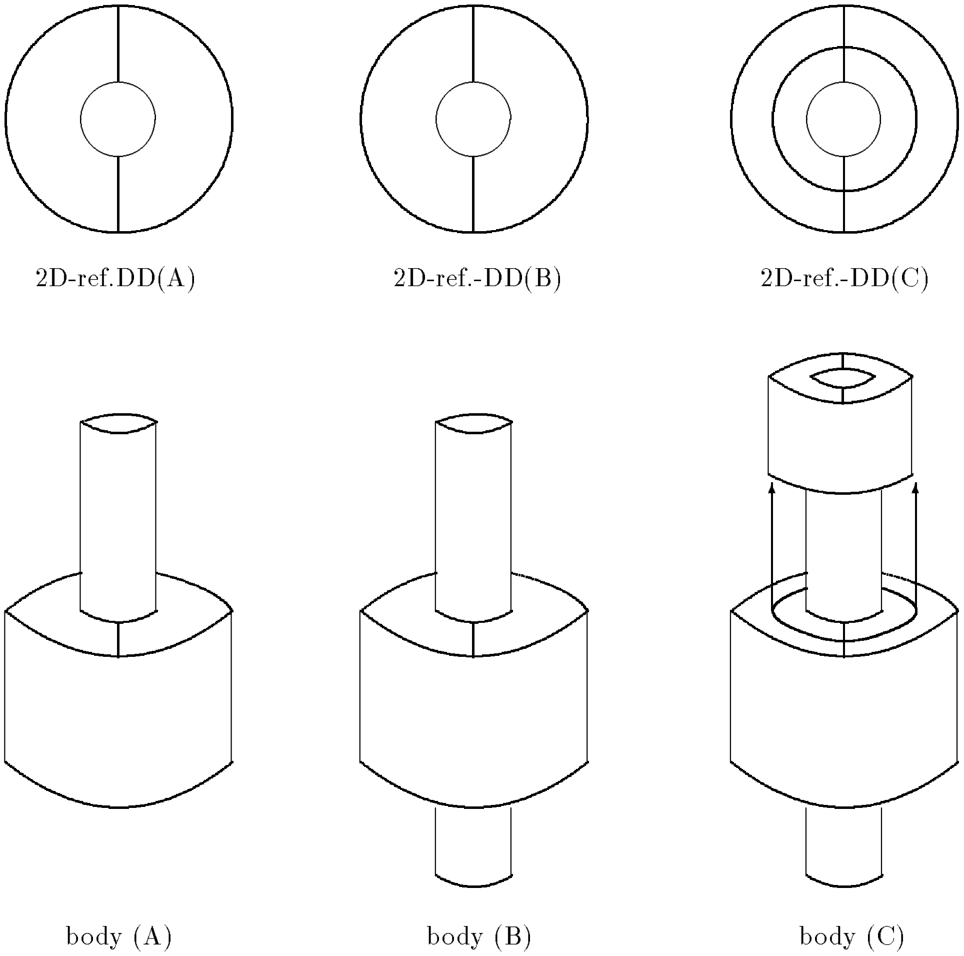
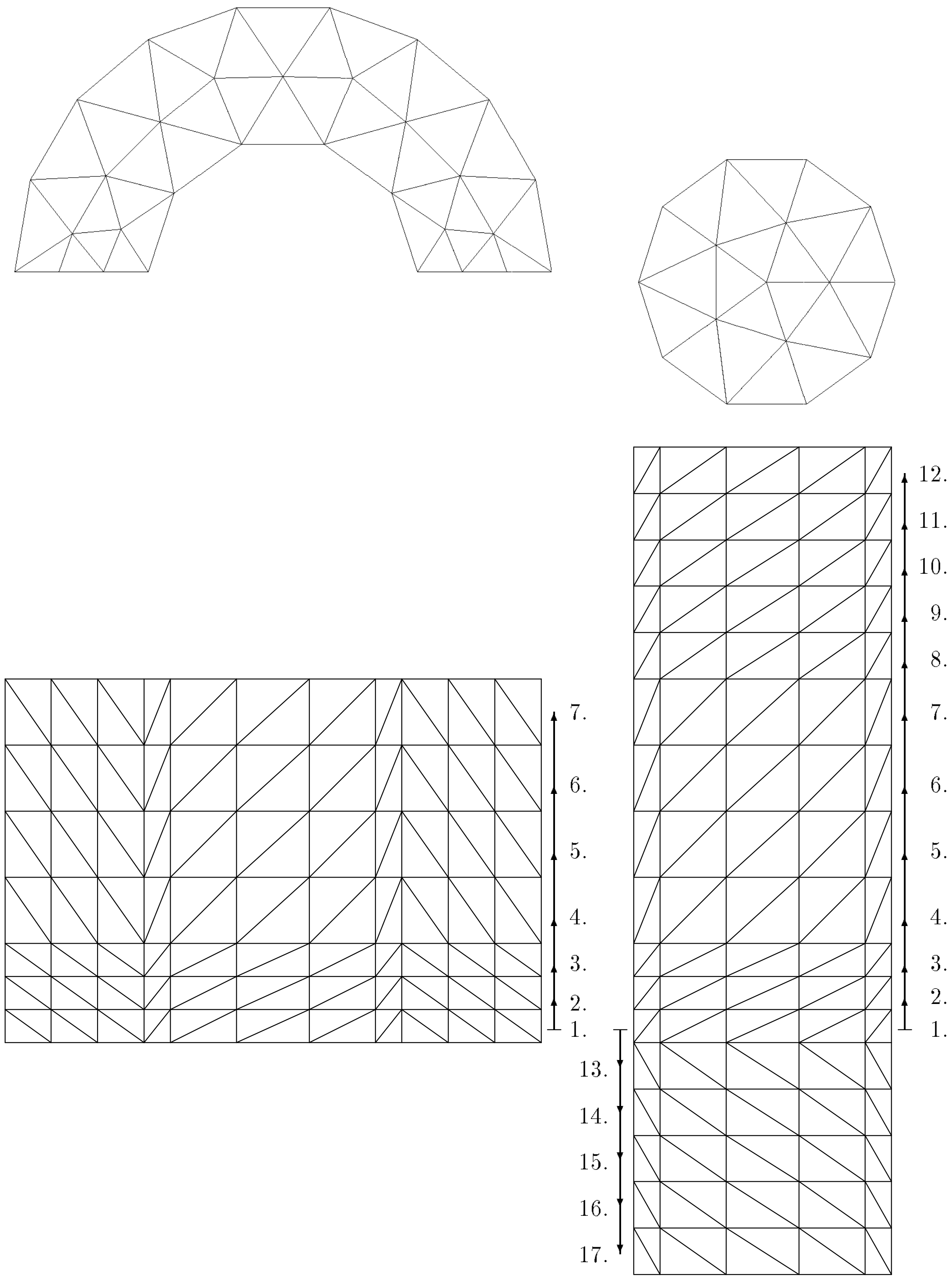


Figure 1: 2D-reference-DD describing generalized cylindric bodies

The strategy for defining the tetrahedral layers can be described as follows. First of all only one pentahedral layer is built up based on its corresponding 2D-reference subdomain triangulation, where the triangles these meshes are consisted of become the lower bases of all of the pentahedrons. Then the pentahedrons are raised orthogonally with respect to its lower bases having the same height given by the geometric data. Now every pentahedron can regularly be divided up into three tetrahedrons, cf. subsection 2.2 . The regularly connected union of the tetrahedrons constructed by dividing the orthogonal pentahedrons as above forms the first tetrahedral layer of the corresponding 3D-subdomain mesh. In the following for generating the next tetrahedral layer the algorithm piles up this first tetrahedral layer, where again the height of the new one must be specified in the geometric data-input.



subdomain 1 subdomain 3

Figure 2: sub-ref.-meshes of body (B) and elevations of tetrahedral layers constructed by

Doing as above the local names uniquelly determining the element connectivity must be numbered appropriately. So, step by step starting from the reference tetrahedral layer connected with its 2D-subdomain triangulation the mesh generator piles up tetrahedral layers one upon

the other until the tetrahedral mesh is fully generated in the generalized cylindric subdomain specified in the geometric data-input. The following adopted convention guarantees the mesh regularity of the tetrahedral layers horizontally, i.e. parallel to the reference plane. As two or more reference subdomains have some part in common (crosspoint, coupling boundary piece) the vertical construction of the tetrahedral layers must be performed uniformly. I.e., the heights of the auxiliary pentahedral layers are equal in the corresponding subdomains. Consequently the first tetrahedral layer called the reference one will have the same height in the whole three dimensional body. For more insight into see Figure 2, too. Obviously the piling up tetrahedral layers also can be performed using two directions, where both are assigned to the corresponding half space. Figure 2 outlines the piling up-method either applied to bodies having cylindric shape extended into one direction only (cf. body (A)) or into both directions otherwise as it is the case e.g. in Figure 1, bodies (B) and (C). Especially in the latter case the succession for the two-directional piling up-method starting from the first layer is given in Figure 2, too, where also non-connected layers may occur; cf. Figure 1, body (C).
The technique applied in order to guarantee the (diagonally) regular finite element connectivity of the tetrahedrons especially those to be placed at the coupling faces between the generalized cylindric subdomains will be presented in the following subsection.

2.2 The principle for generating the tetrahedral layers

Let the subdomain's reference triangulation consisted of NUMEL regularly connected triangles be given. This triangulation can be considered as finite, simple and connected graph consisted of NUMNP nodes and NUMED edges, see e.g. Figure 3. Step by step (j=1(1)NUMEL), for each triangle included in performing well defined orientation and succession of its corresponding three edges we get the simple, directed graph which is uniquely assigned to the reference triangulation. As we know (cf. [4]) the NUMNP nodes of the triangulation are numbered locally per each processor performing the corresponding mesh generation, where the coupling nodes on subdomain's boundary are numbered first of all. Provided that the global names of these coupling nodes here are available from the partitioning of the whole boundary contour the root processor previously made we define the following direction of the edges of 2D-reference mesh.

$$\begin{array}{lcl} CN_1 \Rightarrow CN_2 & \leftrightarrow & g(CN_1) < g(CN_2) \;\; ; \; CN_{1,2} - \text{both are coupling nodes} \\ CN1 \Rightarrow IN_1 & \leftrightarrow & CN_1 - \text{coupling node} \;\; ; \; IN_1 - \text{interior node} \\ IN1 \Rightarrow IN_2 & \leftrightarrow & IN_1 < IN_2 \;\; ; \; IN_{1,2} - \text{both are interior nodes,} \end{array}$$

where the local name of each coupling node is uniquely assigned to its corresponding global one by the function $g(\cdot)$.

The results of the described orientation of triangle's edges are directed triangles having two edges with the same starting point. In the case of no coupling node in the triangle (an interior one) this point coincides with the minimum number of its three vertices. The direction of the third edge is an immediate consequence of the above definition, where from the graph theoretical point of view no circle occurs in the simple directed graph generated as above, see e.g. Figure 3. Concerning the above newdefinition of starting and ending points now the 2D-local mesh data set IED(5,·) containing the description of subdomain's edges is redefined correspondingly. Moreover the sucession of the three triangle's edges the vector IECE(4,·) is consisted of now is determined as follows. Concerning the two edges that have the same starting vertex in common the first edge IECE(1,·) = $\boxed{1}$ is the one having the ending point of the third edge, too. The second edge IECE(2,·) = $\boxed{2}$ is defined by the other and the third one completes IECE(3,·) = $\boxed{3}$ as previously defined, cf. Figure 5. It is easy to see that every subdomain's reference mesh can be uniquely directed in this way.

Each of the auxiliary pentahedrons having the directed-triangular lower base are cut by two interior triangular faces getting the three tetrahedrons (see Figure 5) which must be regularly connected in the FEM-sense to those in the immediate neighbourhood, see e.g. Figure 4. Even the direction of the coupling edges (edges which are located at subdomain's boundary) is globally unique and guarantees the regular FEM-connectivity of the tetrahedrons between the 3D-cylindric subdomains as presented now.

The starting and the ending point of the three diagonal edges $\boxed{10}$, $\boxed{11}$, $\boxed{12}$ (upwards in the case of cylinder's extension into one half space only, cf. body (A) in Figure 1, or upwards/downwards otherwise, see bodies (B),(C), correspond to those of the pentahedron as the direction of the underlying edges in the lower base is determined, see also Figure 5.

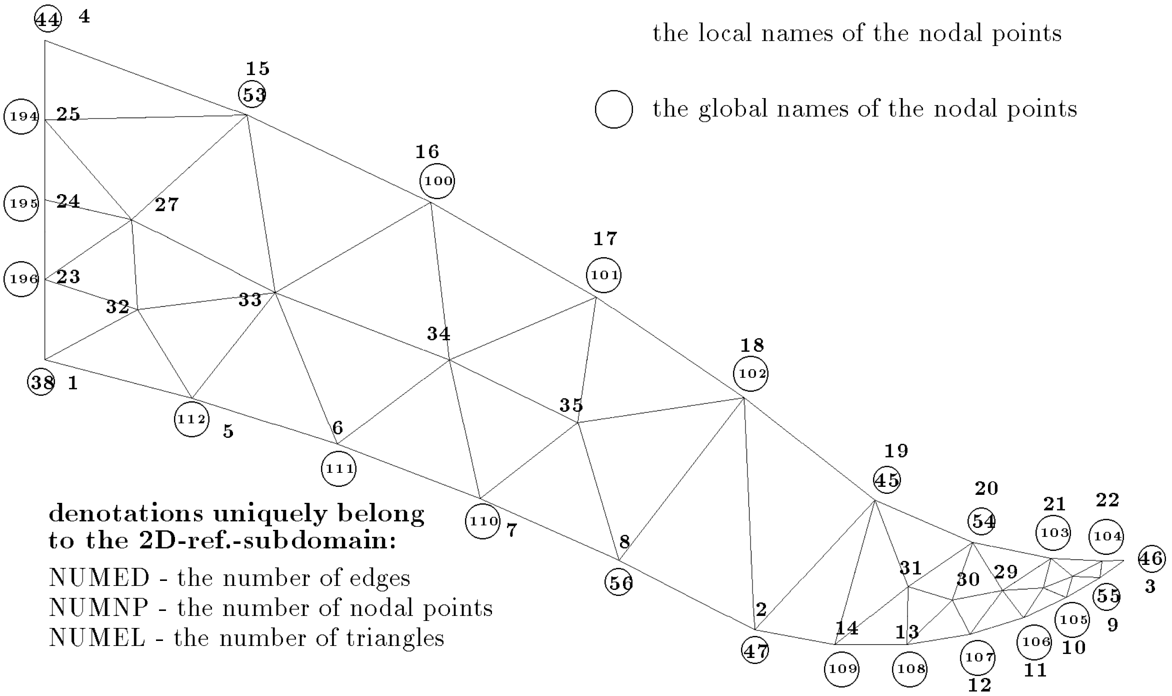
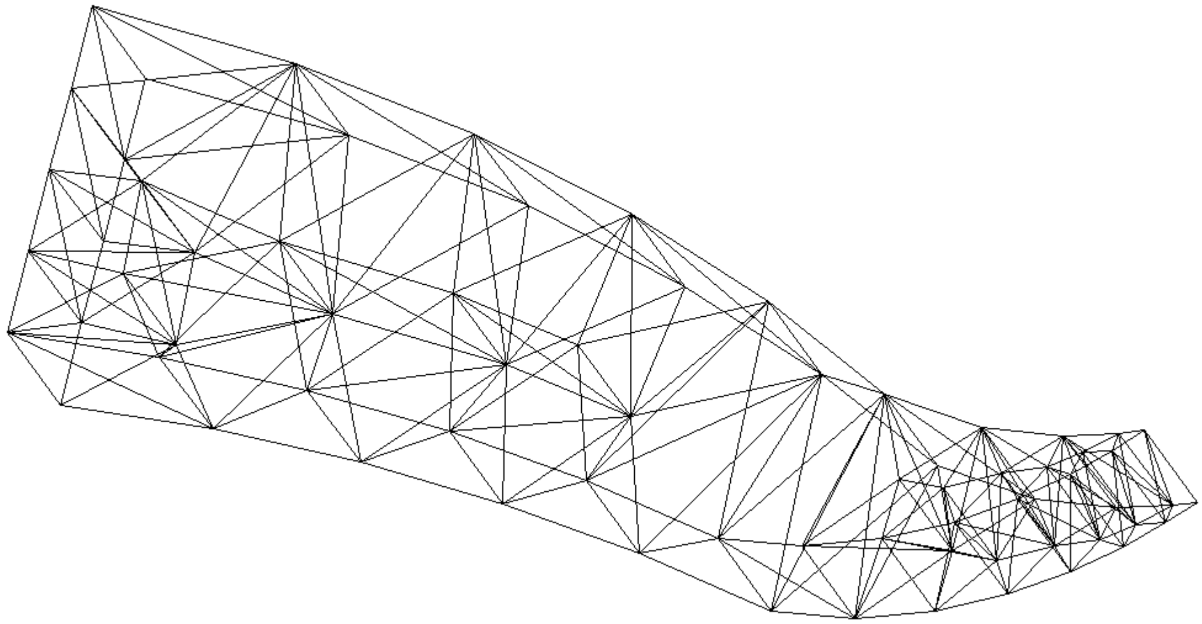


Figure 3: Example for showing the context between subdomain's mesh and its directed graph

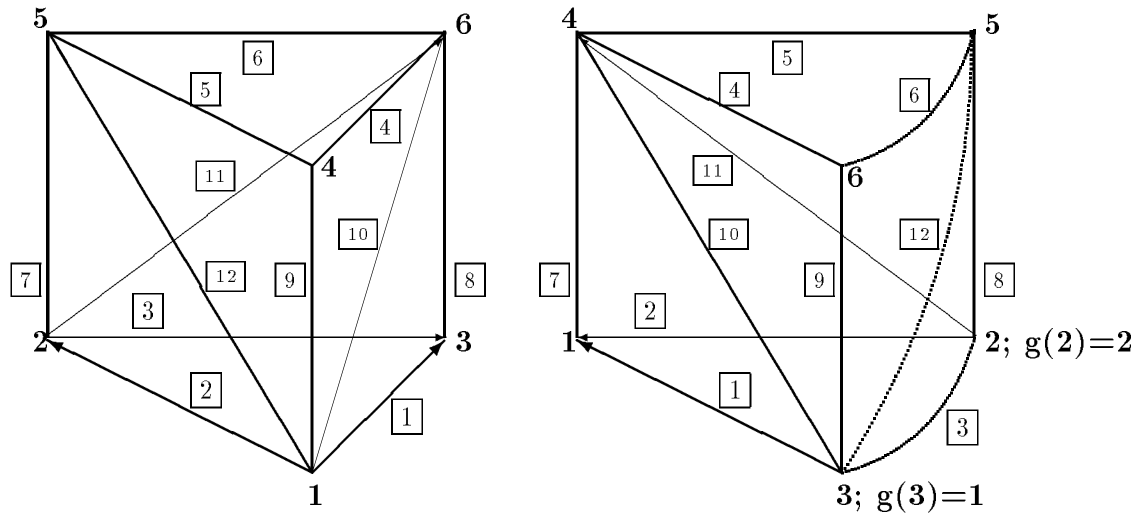


magnitudes determining the memory size per one tetrahedral layer:

the number of edges in the layer := 2*NUMED + NUMED + NUMNP
the number of nodes in the layer := 2*NUMNP
the number of faces the tetraahedrons formed by := 2*NUMEL + 2*NUMEL + 2*NUMED
the number of tetrahedrons in the layer := 3*NUMEL

Figure 4: The reference tetrahedral layer constructed from the above 2D-subdomain

Provided that the data arrays IED(5,·) and IECE(4,·) are prepared as described above in the following we give a survey of program's activities now performed in order to generate the first 3D-subdomain's tetrahedral layer called the reference one. After doing so the next layers are constructed from the above by piling up one upon the other, see section 1.



interior lower base

lower base having the circular coupling edge $\boxed{3}$

Figure 5: Outlines for cutting pentahedrons with directed lower bases $\boxed{1}\boxed{2}\boxed{3}$ and $\boxed{1}\boxed{3}\boxed{2}$

IED(5,.) :	1	3	0	0	1	3	1	0	0	1
	1	2	0	0	1	2	1	0	0	1
	2	3	0	0	1	3	2	1	n	2
	4	6	0	0	1	6	4	0	0	1
	4	5	0	0	1	5	4	0	0	1
	5	6	0	0	1	6	5	2	n	2
	1	4	0	0	1	1	4	0	0	1
	2	5	0	0	1	2	5	0	n	1
	3	6	0	0	1	3	6	0	n	1
	1	6	0	0	1	3	4	0	0	1
	2	6	0	0	1	2	4	0	0	1
	1	5	0	0	1	3	5	3	n	4

IECE(4,.) :	1	2	3	1	1	3	2	1
	4	5	6	1	4	6	5	1
	9	10	1	1	7	10	1	1
	7	10	4	1	9	10	4	1
	9	3	11	1	7	2	11	1
	8	6	11	1	8	5	11	1
	8	12	2	1	8	12	3	2
	7	12	5	1	9	12	6	2
	10	11	2	1	10	11	3	1
	12	10	6	1	12	10	5	1

ITETR(5,1):	1	3	5	9	1	1	3	5	9	1
	7	9	6	10	1	7	9	6	10	1
	8	4	2	10	1	8	4	2	10	1

Figure 6: The data sets of the edges, faces and tetrahedrons belong to the above examples

1. Defining the edges of all of the upper bases of the auxiliary pentahedrons by adding the number NUMNP of subdomain's nodes to the names of the starting and ending vertex of each edge in the reference triangulation we get NUMED new edges of the previously determined type (implemented in the subroutine FACEILL in Figure 8).

2. Determining all of the faces forming the upper bases of the auxiliary pentahedrons by its three edges built up in 1., i.e., adding the number NUMED to each of the three names of the edges the lower base of the pentahedron is given by we get NUMEL new faces (implemented in the subroutine EDGFILL in Figure 8).

3. Defining all of the orthogonal edges between the vertex in the lower base and the corresponding one in the upper base of the pentahedrons we get NUMNP new edges of straight-line type and now we have 2*NUMED + NUMNP edges totally before the following procedure for constructing the reference tetrahedral layer starts.

subroutine's outline for constructing the reference tetrahedral layer :

```
c***************************************************************************
c construction of the reference tetrahedral layer from the underlying
c auxiliary pentahedrons having triangular lower and upper base.
c
c parameter description :
c
c nenk=5,ndf,nen11=2+ndf,nen21=4,nen31=5,ntr=10,nh=3,iproc,list,
c izg   :   distinct (horizontal) array's dimensioning ;
c izeig : array containing scalar magnitudes describing mesh data sets
c ied   : array containing the description of the edges
c iece  : array containing the description of the faces (triangles)
c ibc   : array containing the description of boundary elements
c itetr : array containing the element description
c         (the 4 faces of the tetrahedron and a material code belong to)
c ih    : auxiliary memory for coding already generated edges and faces
c**************************************************************************
      subroutine tetraer(nenk,ndf,nen11,nen21,nen31,ntr,nh,iproc,list,izg,
     *                  izeig,ied,iece,ibc,itetr,ih)
c
      integer*4 ied(nenk,1),ibc(nen11,1),iece(nen21,1),
     *          ih(nh,1),izeig(list,ntr,1),itetr(nen31,1)
c setting scalar magnitudes describing the 2D-mesh data sets :
      numed  = izeig(5,1,izg); numel  = izeig(8,1,izg); numnp  = izeig(1,1,izg)
      numbed = izeig(6,1,izg); nummp  = izeig(9,1,izg); nummph = 2*nummp; icr = 1
      nvkante = 2*numed; nskante = nvkante + numnp; nrel = 0; nsface = 2*numel
      call vicopy(nh*numed,ih,1,0,0)
c
      do 100 i = 1,numel
         ifacetyp1 = 1; ifacetyp2 = 1; ifacetyp3 = 1
         k1 = iece(1,i); k2 = iece(2,i); k3 = iece(3,i); imb = iece(4,i)
         n1a = ied(1,k1); n2a = ied(1,k2); n3a = ied(1,k3)
         n1e = ied(2,k1); n2e = ied(2,k2); n3e = ied(2,k3)
         nvkh1 = nvkante + n1e; nvkh2 = nvkante + n1a; nvkh3 = nvkante + n2e
c the first tetrahedron lying below :
         if (ih(1,k1) .eq. 0) then
             nskante = nskante + 1; nskh1 = nskante; ih(1,k1) = nskh1
             nsface = nsface + 1; nsf1 = nsface; nsface = nsface + 1; nsf3 = nsface
             ih(2,k1) = nsf1; ih(3,k1) = nsf3; ied(1,nskh1) = n1a
             ied(2,nskh1) = n1e + numnp; ied(3,nskh1) = 0
             if (ied(3,k1) .ne. 0) then
c defining the number of the midpoint if a curvilinearly diagonal edge occurs :
                 ied(3,nskh1) = ied(3,k1) + nummph + icr; icr = icr + 1
             endif
             ied(4,nskh1) = ied(4,k1); ied(5,nskh1) = ied(5,k1)
             if (ied(5,nskh1).gt.1) then
                 ifacetyp1 = ied(5,nskh1); ied(5,nskh1) = ied(5,nskh1) + 2
             endif
             if (ied(4,k1) .gt. 0) then
c defining the 2 boundary-faces belong to this edge in the array ibc :
             endif
             iece(1,nsf1) = nvkh1; iece(2,nsf1) = nskh1; iece(3,nsf1) = k1
             iece(4,nsf1) = ifacetyp1; iece(1,nsf3) = nvkh2; iece(2,nsf3) = nskh1
             iece(3,nsf3) = k1 + numed; iece(4,nsf3) = ifacetyp1
                             else
             nskh1 = ih(1,k1); nsf1 = ih(2,k1); nsf3 = ih(3,k1)
         endif
```

```
        if (ih(1,k3) .eq. 0) then
            nskante = nskante + 1; nskh3 = nskante; ih(1,k3) = nskh3
            nsface = nsface + 1; nsf2 = nsface; nsface = nsface + 1; nsf4 = nsface
            ih(2,k3) = nsf2; ih(3,k3) = nsf4; ied(1,nskh3) = n3a
            ied(2,nskh3) = n3e + numnp; ied(3,nskh3) = 0
            if (ied(3,k3) .ne. 0) then
c defining the number of the midpoint if a curvilinearly diagonal edge occurs :
                ied(3,nskh3) = ied(3,k3) + nummph + icr; icr = icr + 1
            endif
            ied(4,nskh3) = ied(4,k3); ied(5,nskh3) = ied(5,k3)
            if (ied(5,nskh3).gt.1) then
                ifacetyp3 = ied(5,nskh3); ied(5,nskh3)=ied(5,nskh3) + 2
            endif
            if (ied(4,k3) .gt. 0) then
c defining the 2 boundary-faces belong to this edge in the array ibc :
            endif
            iece(1,nsf2) = nvkh1; iece(2,nsf2) = k3; iece(3,nsf2) = nskh3
            iece(4,nsf2) = imb; iece(1,nsf4) = nvkh3; iece(2,nsf4) = k3 + numed
            iece(3,nsf4) = nskh3; iece(4,nsf4) = imb
                             else
            nskh3 = ih(1,k3); nsf2 = ih(2,k3); nsf4 = ih(3,k3);
        endif
c
        if (ih(1,k2) .ne. 0) then
            nsf5 = ih(2,k2); nsf6 = ih(3,k2); nskh2 = ih(1,k2);
                             else
            nskante = nskante + 1; nskh2 = nskante; ih(1,k2) = nskh2
            nsface = nsface + 1; nsf5 = nsface; nsface = nsface + 1
            nsf6 = nsface; ih(2,k2) = nsf5; ih(3,k2) = nsf6
            ied(1,nskh2) = n2a; ied(2,nskh2) = n2e + numnp; ied(3,nskh2) = 0
            if (ied(3,k2) .ne. 0) then
c defining the number of the midpoint if a curvilinearly diagonal edge occurs :
                ied(3,nskh2) = ied(3,k2) + nummph + icr; icr = icr + 1
            endif
            ied(4,nskh2) = ied(4,k2); ied(5,nskh2) = ied(5,k2)
            if (ied(5,nskh2).gt.1) then
                ifacetyp2 = ied(5,nskh2); ied(5,nskh2)=ied(5,nskh2) + 2
            endif
            if (ied(4,k2) .gt. 0) then
c defining the 2 boundary-faces belong to this edge in the array ibc :
            endif
            iece(1,nsf5) = nvkh3; iece(2,nsf5) = nskh2; iece(3,nsf5) = k2
            iece(4,nsf5) = imb; iece(1,nsf6) = nvkh2; iece(2,nsf6) = nskh2
            iece(3,nsf6) = k2 + numed; iece(4,nsf6) = imb
        endif
c
c the first interior triangular face :
c
        nsface = nsface + 1; nsfi = nsface; iece(1,nsfi) = nskh1
        iece(2,nsfi) = nskh3; iece(3,nsfi) = k2; iece(4,nsfi) = 1
        nrel = nrel + 1; itetr(1,nrel) = i; itetr(2,nrel) = nsf1
        itetr(3,nrel) = nsf2; itetr(4,nrel) = nsfi; itetr(5,nrel) = imb
c
c the second tetrahedron in the middle, the second inner triangular face :
c
        nsface = nsface + 1; nsfii = nsface; iece(1,nsfii) = nskh2
        iece(2,nsfii) = nskh1; iece(3,nsfii) = k3 + numed
        iece(4,nsfii) = 1; nrel = nrel + 1; itetr(1,nrel) = nsf5
        itetr(2,nrel) = nsfi; itetr(3,nrel) = nsf4
        itetr(4,nrel) = nsfii; itetr(5,nrel) = imb
c
c the third tetrahedron lying above :
c
        nrel = nrel + 1; itetr(1,nrel) = nsf6; itetr(2,nrel) = nsf3
        itetr(3,nrel) = numel + i; itetr(4,nrel) = nsfii; itetr(5,nrel) = imb
  100 continue
c
c setting scalar magnitudes describing the 3D-mesh data sets :
c
      numte = 3*numel; izeig(5,1,izg) = nskante; izeig(8,1,izg) = nsface;
      izeig(12,1,izg) = numte; izeig(6,1,izg) = 2*(numbed+numel)
      izeig(9,1,izg) = nummph + icr - 1
      return
      end
```

2.3 The geometric description of the 3D-domain via ASCII-codes

The point of departure for the generation of tetrahedral meshes in the generalized cylindric domain was the 2D-reference domain decomposition. Hence the structure of the 3D-geometric data file will coincide with the 2D-one with exception of the data block the information for the extension of 2D-subdomains into the space is included in. The input-data file is consisted of five data blocks concerning the global names of domain's geometric data. The detailed explanation of the blocks describing 2D-geometric data was given in [4, 16, 17], whereas in this paper the fourth data block gets slightly modification in its sense and the fifth block is extended with respect to the 3D-cylindric expansion of the 2D-reference domain decomposition. The description of both blocks is fully presented after the following table 1 gave a short summery of the file structure.

Block 1 : The first block (2D-reference scalar data) contains the numbers of :
basic nodes, basic lines, subdomains, basic lines with respect to some boundary conditions that are defined upwards and downwards, respectively, and :
3 values the maximum of the range of the above numbers is restricted by;

Block 2 : The cartesian coordinate block pointwisely consisted of the x- and y-coordinates of the 2D-reference basic nodes;

Block 3 : The block of the description of the basic lines, where its names (NUM), types (LT), the names of its basic nodes and the method (IT) for specifying its uniform or adaptive partitioning with (NR) new nodes to be placed on it are given, see Figure 7 and cf. [4];

Block 4 : The 4-th block describing basic lines with respect to boundary conditions;

Block 5 : The 5-th block of the description of the subdomains at first consisted of
two rows for each 2D-reference subdomain, where the first row contains :

NUM	the name of the 2D-reference subdomain
NPA	the number of basic lines that are bounding it
NMB	the corresponding material code of the 2D-cross section
PI2A	the first criterion-angle (L) defining the reference triangles (default - 0.)
PI24A	the second criterion-angle (L) defining the reference triangles (default - 0.)
PI6A	the third criterion-angle (L) defining the reference triangles (default - 0.)

the second row contains the following :

LB(I), I=1,...,NPA   the names of the corresponding basic lines from block 3

and now per each subdomain a third row containing the number

IL

of further rows here, the 3D-extension is described main-level (LI) by main-level, (I=1(1)IL), in terms of 10+3*NDF real values in each case, where the value NDF is equal to the number of the degrees of freedom given by the underlying p.d.e. to solve on the domain. The IL rows now included in this data block have the following structure :

RNUML1,RLTL1,ZL1L,ZL1M,ZL1U,RITL1,RNRL1,A1L1,A2L1,RMBL1,NDF*BCL1{l,u,s}
RNUML2,RLTL2,ZL2L,ZL2M,ZL2U,RITL2,RNRL2,A1L2,A2L2,RMBL2,NDF*BCL2{l,u,s}
....;

The denotations used at last in the fifth data block are declared as follows, where they belong to the I-th main-level in each case, I=1(1)IL.

RNUMLI	the name of the I-th main level
RLTLI	the type(=1) of the I-th main level (up to now constructed orthogonally)
ZLIL	the z-coordinate of the lower base of the main-level
ZLIM	the z-coordinate of some point in the middle possibly given
ZLIU	the z-coordinate of the upper base of the main-level
RITLI	the type for partitioning the main-level into levels, cf. Figure 7 and [4]
RNRLI	the number of such levels included in the I-th main-level minus one
A1LI	the first ratio the adaptive main-level partitioning is determined by
A2LI	the second ratio the adaptive main-level partitioning is determined by
RMBLI	the material code of the I-th main-level; RMBL1 = NMB holds
BCLI$\{l,u,s\}$	for NDF degrees of freedom in each case the type of the boundary conditions(BC) imposed on the lower (l), upper (u) base of the main-level and on some parts of its lateral surface (s). This is done in the cartesian product-connection with the function pointer specified for basic lines in data block 4. If no BC the value is 0.

Table 1: The general structure of the PARMESH-3D-input-ASCII-data file

As in the 2D-case the geometric input-data file can be quickly created by the graphical editor GRAFED, see e.g. [4, 17].

In data block 4 all of the basic lines are described by means of its orthogonal projection performed upwards or/and downwards edges of faces will produced later that possess some kind of boundary conditions. The description of these basic lines includes its name (NUM), its type (LT, straight line, circular or parabolic arc), the name of its starting, middle and ending basic node, the corresponding material code determined by the reference cross-section and the pointer to some function-program that defines the values of the boundary condition imposed as explained above.

In connection with data block 5 the type of the corresponding boundary conditions is coded. For simplicity here we adopt the convention that this type may change only vertically, i.e., main-level by main-level its lower and upper triangulations are parallel to the reference one included in the plane $\mathcal{H}$, cf. section 1. Therefore the decomposition of the 2D-reference domain into the q single connected subdomains must be appropriately given by corresponding basic lines supporting boundary conditions in the sense of data block 4. As first of all the the geometric shape of the 3D-domain determines the 2D-reference domain decompostion this additional requirement could cause an increased number q of subdomains. The meshes in the subdomains will generate by p processors in parallel after the mapping "subdomain(s) $\leftrightarrow$ processor" is performed as defined in the following section 3. Provided that the number p of available processors is large enough the above assumption can be guaranteed there.

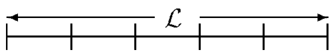
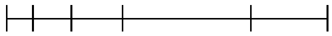
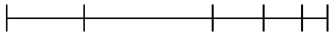
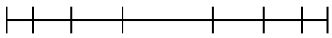
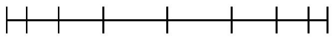
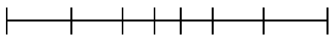
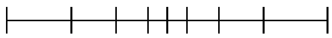
nodal dispersion	(R)IT(LI)	(R)NR(LI)	A1(LI)	A2(LI)
 $\mathcal{L}$, a	1(0) 6	$\mathcal{L}/a-1$ 0	0 a	0 0
 a b	2 7	arbitrary 0	b/a a or b	0 b or a
 b a	3 8	arbitrary 0	b/a a or b	0 b or a
 a b a	4 11	even 0	b/a a or b	0 b or a
 a b b a	4 9	odd 0	b/a a or b	0 b or a
 b a b	5 12	even 0	b/a a or b	0 b or a
 b a a b	5 10	odd 0	b/a a or b	0 b or a
adapted to point singularities (cf. [4])	13 or 14	arbitrary	μ	H_i

Figure 7: The context of the parameters (R)IT(LI), (R)NR(LI), A1(LI), A2(LI) in the data blocks 3 and (5), respectively

By the example shown in the next Figure we gain insight into its given specific 3D-geometric data file structure.

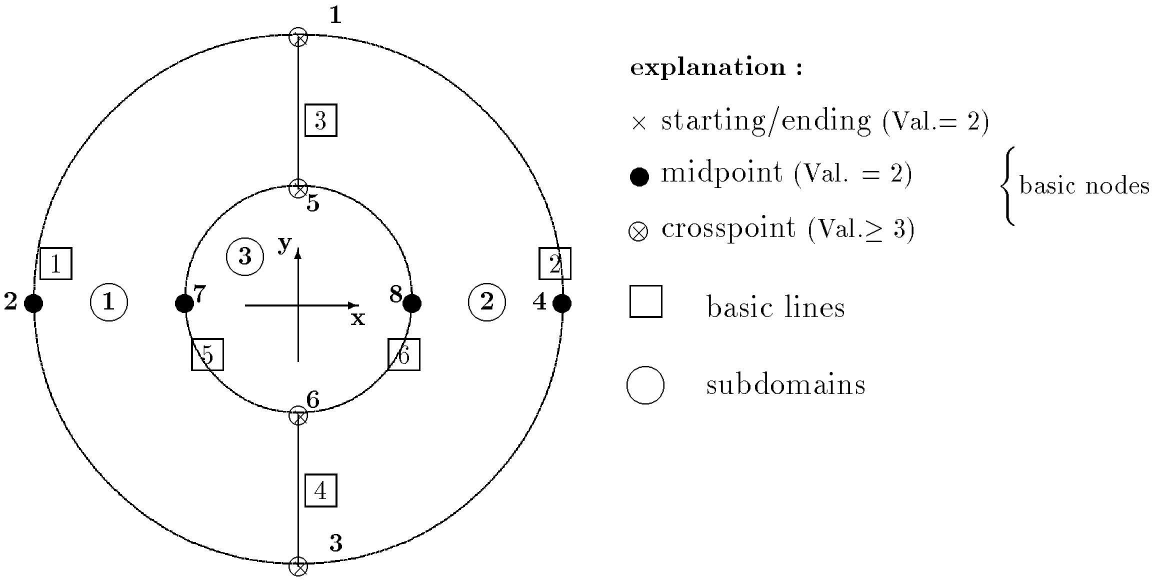


Figure 8: Example of the 2D-reference DD belonging to body (B) in Figure 1

```
 8     6    3    4    0    0    0
 0.0    2.0
-2.0    0.0
 0.0   -2.0
 2.0    0.0
 0.0    1.0
 0.0   -1.0
-1.0    0.0
 1.0    0.0
1    2    1    2    3    1    8    0     0
2    2    3    4    1    1    8    0     0
3    1    1    0    5    1    2    0     0
4    1    3    0    6    1    2    0     0
5    2    5    7    6    1    4    0     0
6    2    6    8    5    1   _4___ 0     0
1    2    1    2    3    1   | 1 |
2    2    3    4    1    1   | 1 |    pointers to some functions
5    2    5    7    6    1   | 2 |    describing boundary conditions
6    2    6    8    5    1   | 2 |
1    4    1    0    0    0   -----
1    4    5    3
2                                    --------------
1.   1.   0.   0.   0.5  1. 2. 0. 0.  | 1. 2. 0. 1. |
2.   1.   0.5   0.  1.5  1. 3. 0. 0.  | 1. 0. 2. 2. |
2    4    1    0    0    0            ---------------
2    4    6    3
2                                     ---------------
1.   1.   0.   0.   0.5  1. 2. 0. 0.  | 1. 2. 0. 1. |  material and
2.   1.   0.5   0.  1.5  1. 3. 0. 0.  | 1. 0. 2. 2. |  boundary codes
3    2    1    0    0    0            ---------------  for the main-
5    6                                                  levels
3                                     ---------------
1.   1.   0.   0.   0.5  1. 2. 0. 0.  | 1. 0. 0. 0. |
2.   1.   0.5   0.  1.5  1. 3. 0. 0.  | 1. 0. 0. 0. |
3.   1.   1.5   0.  5.5  1. 4. 0. 0.  | 1. 0. 2. 2. |
4.   1.   0.   0.  -4.0  1. 4. 0. 0.  | 1. 0. 2. 2. |
                                      ---------------
```

Table 2: The geometric input-data file belonging to body (B)

3 On program's parallelization and capabilities

The parallelization of the mesh generator PARMESH3D consists in the following steps. Let the reference domain decomposition (DD) of the domain $\mathbf{\Omega} \subset \mathcal{H}$ into q 2D-subdomains be given. Reading the input-data file by the root processor, there the basic line partitioning (P) concerning the whole boundary contour will be performed and stored according to its specification given in data block 3, cf. Figure 7. The assigned number of necessary arithmetical operations is of order $O(h_0)$, where the parameter h_0 denotes the average size of the distances in the basic line partitioning. Moreover, here the partitioning of the main-levels must be computed as it is required by its definition given in the corresponding extension of subdomain's data block 5.

Ending the basic line partitioning, well specified subsets $\mathcal{D}_i$, $i = 1, \ldots, q$, in each case containing the data of the $i-$th subdomain (the coordinates of all of the points at its 2D-reference boundary contour, its global names defined by the whole 2D-contour and the 2D,3D-subdomain's description concerning e.g. the 2D-reference meshing control parameters (L) and the coordinates of the lower and upper bases of the tetrahedral layers, the material and BC-information given in data block 5 are included in) will send from the root processor to well determined others, where the implemented static-loadbalanced data division is defined by the following bijection of the data sets $\mathcal{D}_i$ onto the $k-$th processor, $k = 0, \ldots, p-1$:
Let q be the number of subdomains and p be the number of available processors.

- case 1, $(p \geq q)$: We define the mapping $i \leftrightarrow k$, i.e., one and only one data set $\mathcal{D}_{k+1}$ per processor $k = 0, 1, \ldots, q-1$; and $p - q$ processors will do nothing;
- case 2, $(q > p)$: We have $n := q/p$ and $r = mod(q, p)$ and the data set $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_q\}$ is sequentially subdivided into p subsets as follows :
  $\mathcal{S}_0 = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n\}$, $\mathcal{S}_1 = \{\mathcal{D}_{n+1}, \mathcal{D}_{n+2}, \ldots, \mathcal{D}_{2n}\}$, $\ldots$, $\mathcal{S}_{p-1} = \{\mathcal{D}_{n(p-1)+1}, \ldots, \mathcal{D}_{np}\}$, where $card(\mathcal{S}_i) = n \quad \forall i = 0, 1, \ldots, p-1$.
  If $r \neq 0$ then defining the remainder-set $\mathcal{R} = \{\mathcal{D}_{np+1}, \mathcal{D}_{np+2}, \ldots, \mathcal{D}_q\}$ and the sets $\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_{r-1}$ must be sequentielly extended by one and only one data set from $\mathcal{R}$.
  Then, we define the mapping : $\mathcal{S}_k \leftrightarrow k$, $k = 0, 1, \ldots, p-1$.

We emphasize that the number of subdomains its meshes are generated by one and only one processor may not exceed the constant IPROC determined by the program, cf. section 4. Therefore $n + 1 \leq IPROC$ must be fulfilled.
In [4] we proposed two opportunities for performing scientific load balancing such that the amounts of numerical operations for the 2D-mesh generation performed in each processor agree reasonably well. This 2D-loadbalance determines the balance of the corresponding 3D-mesh generation essentially, whereas the amount for constructing the tetrahedral layers is neglectible with exception of that concerning the reference one.

After the above one-to-one mapping was performed by means of only rare communication the program PARMESH3D runs totally parallel such that the 2D-reference triangulations (T_i) in the corresponding subdomains will be generated very efficiently. The internally performed mesh generation frontier-algorithm (F) is based on the leveling and removing process of triangles. Its idea was founded in [20]. The amount of necessary arithmetical operations can be estimated by $Q(F) = C_F(\mathbf{\Omega}_i, P_i, L_i) \cdot h_{0,i}$. Here the magnitude $h_{0,i}$ denotes the average-size of the distances between points at i-th subdomain's boundary. The constant C_F depends on subdomain's size and shape, on the kind of the partitioniong (P_i) and on the mesh generation control parameters (L_i) the leveling of triangles is specified by. In some worse cases of the demanded initial mesh density the amount $Q(F)$ is of order $O(h_{0,i}^{-2})$. The types of the arithmetical operations are substantially those needed for the computation of angles, distances, etc., such that the calls of standard functions like "arctan" and "sqrt" are hidden here.

To improve the shape of the triangles generated in subdomain's reference mesh *it*-times the grid smoothing process (S) (see e.g. [1, 16, 17]) can be applied, where its parallelization is based on the made mapping, too. The corresponding amount of numerical operations is equivalent to $Q(S) = C(T_i) \cdot it \cdot h_{0,i}^{-2}$. The reference triangulations can be refined by the hierarchical mesh generator GIGEHI very fast, see e.g. [4], before the construction of the tetrahedral layers starts. After the 2D-reference mesh generation is completed the constructions of the corresponding tetrahedral layers are performed in parallel as described in section 1. The amount for generating the layers can be estimated by $O(h_{0,i}^{-2})$, where the main part is determined by the reference one, whereas the data of the others are built up by doubling the previous one and adding well defined integers to.

When the mesh in the 3D-subdomain was generated the renumbering (R_3) of the names of its interior nodal points is continued immediately if it was globally demanded by the user. This process is performed totally parallel, too. The underlying algorithm is essentially based on minimal nodal degree ordering, see [3, 4] and the references therein. Its amount of arithmetical operations can be estimated by $Q(R_3) = C(\text{graph}(\text{Tetr.}(\mathrm{T_i})))\cdot$ NUMINP$>$ $O(h^{-4.5})$, where the variable NUMINP represents the number of interior nodal points of the 3D-subdomain. For comparison the amount for nodal renumbering in the 2D-case (R_2) was estimated in [3] to be $Q(R_2) = C(\text{graph}(\mathrm{T_i}))\cdot$ NUMINP $\leq C\cdot$NUMINP$^{3/2}$ $= O(h^{-3})$. The constant $C(\text{graph}(\text{Tetr.}(\mathrm{T_i})))$ the amount $Q(R_3)$ of the algorithm is estimated by depends on properties characterizing the tetrahedral graph constructed from the reference triangulation T_i and exceeds $O(h^{-1.5})$. The impressive results given in [3] have been proven that the renumbering of the nodal points in the coarsest mesh is very efficient in the case of the exact solution of systems of equations discretized there and included in the multilevel method. This method can be used as efficient subdomain solver for the parallel solution of the partial differential equation given on the domain $\mathbf{\Omega}$ discretized by the additive Schwarz-DD-method, cf. also [7, 8, 9].
The following Figure 9 presents the scheme for controlling and operating with the program package PARMESH3D.

calling the program (run ...)

questioning of the 3D-/ 2D-input-data file → (y/n)
input of the input file name
defining the name of an output-data file

2D-mesh generation | 3D-mesh generation

questioning as follows :

- grid smoothing demanded ? → (y/n)
- renumbering of the cross points ? → (y/n)
- renumbering of subdomain's inner nodes ? → (y/n)
- input the number of hierarchical prerefinements ! → $(I * 4 - value \geq 0)$

some questions concerning the
2D-FEM-output-file, see [16, 17]

output of the whole mesh demanded ? →
(y / n)

file-output
END

output of subdomain's mesh ?
(y → selecting subdomains name, file-output, END)
(n → END, or further main-processing in parallel)

Figure 9: The plan for handling with the mesh generator

Ending the 3D-mesh generation the user may require the output of the whole tetrahedral mesh to be fit in the root processor or select the output of one of the subdomain meshes. Otherwise some kind of further parallel main-processing (see e.g. [2, 4, 7, 8, 9] and the references therein) runs locally, where the mesh data structure given in section 4 can be used.

In each case the corresponding output is implemented by writing out mesh data sets into the output file. The structure of the output file is defined analogously to the case of 2D-FEM mesh data, e.g. described in [16, 17]. The coordinates of the vertices of the tetrahedrons, the corresponding standard-FEM connectivity as well as the boundary-face description equipped with the coded boundary conditions are involved in. By means of the output file the tetrahedral meshes can be visualized distinctly using the program GRAPE, see e.g. [5].

On the root processor also the renumbering of all of the nodes in the whole tetrahedral mesh can be performed by the mentioned algorithm before the file-output begins, but it costs the above substantial time effort. Nevertheless this can be useful in order to prepare the efficient solution of the system of equations discretized later on these (cross)points in the case of further sequential or parallel 3D-FEM processing, see e.g. [2, 7, 8, 9].
The following Figure 3 presents the hierarchical tree of the subroutines called by the main program PARMESH3D.

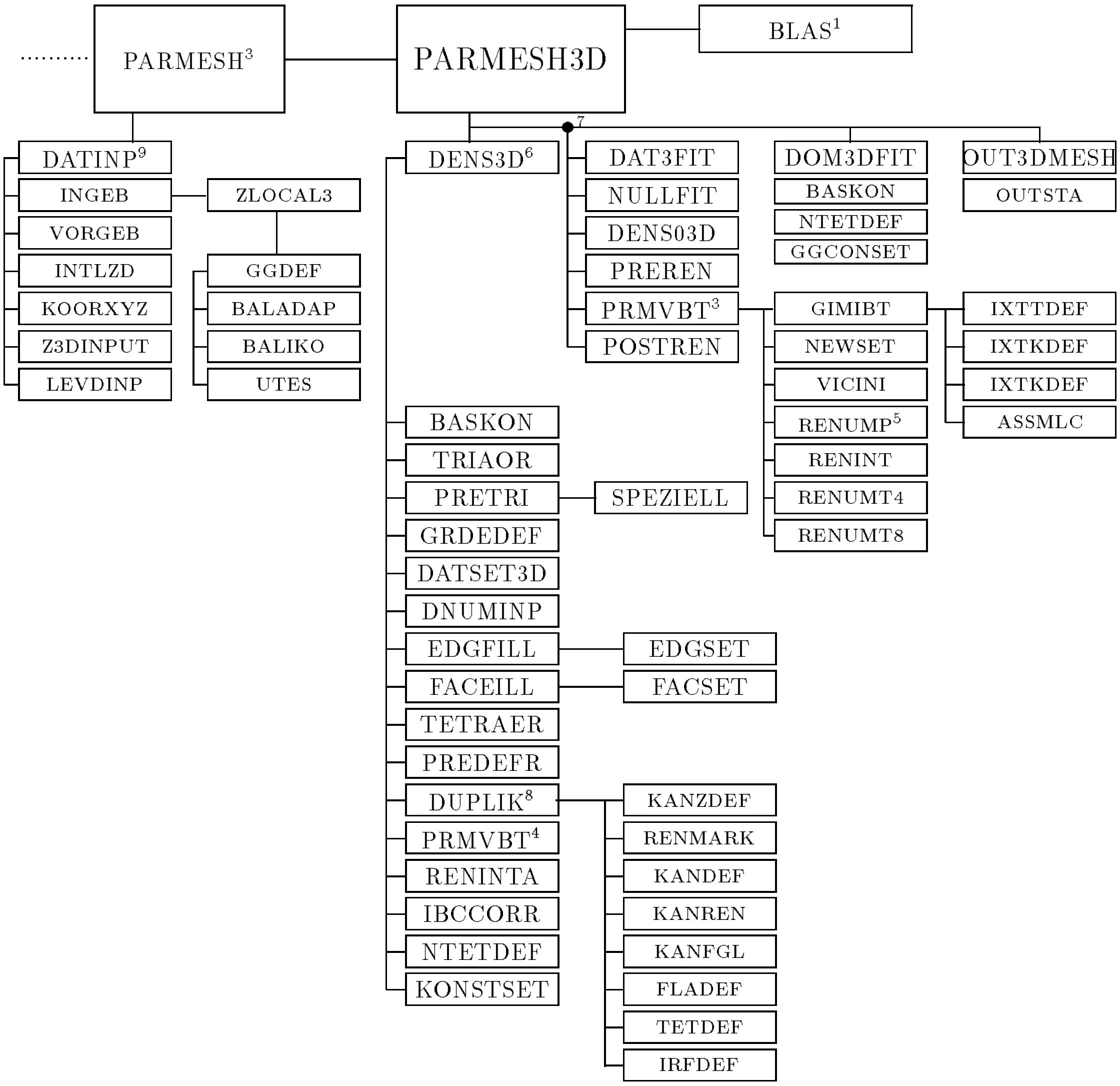


Figure 8 : The subroutines called by PARMESH3D

[1] basic linear algebraic subroutines for vector operation and communication, see e.g. [6]
[2] The other ingredients of the 2D-mesh generator are given in [4]
[3] Controlling of the nodal renumbering of the whole 3D-mesh
[4] Controlling of the inner nodal renumbering in the 3D-subdomains having the same subroutines
[5] The continued nodal renumbering algorithm is described in [3]
[6] Local controlling of the construction of the tetrahedral layers
[7] Preparing and performing the output of the whole tetrahedral mesh and some selected 3D-subdomain part, respectively, by the continued subroutines
[8] Piling up the tetrahedral levels one upon the other
[9] Setting the data of the z-extension of the 2D-subdomains by the root processor

Concluding remarks :
Here we give a short summary that contains several opportunities for developing the program PARMESH3D to become more robust.

- By the automatical balancing of problem's underlying domain decomposition up to now static-heuristically performed we want to improve the efficiency of the 3D-mesh generator to become well balanced; ideas are given e.g. in [4, 18]. Here the dynamic prediction of the memory size necessary for the 3D-subdomain meshes is involved.
- Determining adaptively the type RITLI and the number RNRLI of partitions included in the main-level specification we reflect geometric properties and peculiarities that were already taken into account for defining the 2D-reference boundary contour appropriately (cf. [4]). Here, e.g. the computation of the partitioning of the main-levels into equidistant levels was possible as well as anisotripic elements could occur. During the mesh generation the corresponding size of the level distances could be derived from the sizes of the triangular edges the 2D-reference mesh is given by.
- Our initial mesh generator must be appropriately connected to an hierarchical 3D-mesh generator that is capable of canonical and adaptive refining, too; see e.g. [4, 12, 15] for the adaptive mesh refinement in the 2D-FEM-case and [11] for the canonical refinement of tetrahedrons.
- The implementation of the 3D-grid smoothing in terms of the appropriate transition of the inner nodal points, see e.g. [1, 16, 17], can easily be performed.
- The opportunity for defining the boundary-faces of the subdomains curvilinearly is already included in the main-level description of the data block 5 (RLTLI,ZLIM) but the extension is still performed orthogonally as explained in section 1.

4 The Output Data-Structure of the tetrahedral mesh

In each subdomain the corresponding triangulation generated in parallel is described by the program PARMESH3D in terms of an edge-related data structure. I.e., for the generalized cylindric 3D-subdomain the corresponding processor makes available distinct data blocks its structure is claimed to be very suitable for further parallel computations. The names of all of the data and the size of the sets they belong to are defined to be locally and uniquely there, too. For simplicity the data are summarized and described in the following survey.

1. **scalar data :**

NUMNP :	the total number of nodal points generated in the subdomain (the corresponding number of points on its boundary is included in)
NUMINP:	the number of nodal points in the interior of the subdomain
NUMMP :	the number of midpoints given on the curvilinear edges of the boundary of the corresponding subdomain
NUMCP :	the number of nodal points on the boundary of the subdomain (so called coupling points that are no cross points)
NUMCRP: (2D only)	the number of crosspoints (i.e. the number of basic nodal points that have at least the degree graeter than two). Its size corresponds to the number of 2D-coupling boundary pieces given between two cross points
NUMED :	the number of all of the edges generated in the subdomain
NUMEL :	the number of all of the faces forming the tetrahedrons in the subdomain
NUMTE :	the number of all of the tetrahedrons generated in the subdomain
NUMBED:	the number of the edges on the boundary of the subdomain if some corresponding part of the actual boundary of the domain Ω is described by
NDF :	the number of degrees of freedom the boundary conditions of the whole domain Ω are defined by
NTR :	the number of refinement steps are to be carried out by the hierarchical mesh generator.

Remarks

- Each of the above first eight data item belonging to one and only one subdomain-mesh is uniquely assigned to some i-th element of the pointer-vector IZEIG(i,NTR,IPROC), where the variable IPROC (≥ 1) denotes the maximum number of subdomains one and only one processor can have for the corresponding mesh generation in it. Hence the assignment of the q subdomains to the p available processors as described in section 2.2. is restricted by the size of IPROC. Because PARMESH generates the initial triangulation (NTR = 1) the entries are made in IZEIG(i,1,l), where the index $l, (1 \leq l \leq$ IPROC) is the local name of one of the subdomains mapped to the k-th processor ($k = 1, \ldots, p$). Remembering the adopted convention $n = q/p$ and provided that the case $q > p$ holds at most $l = n + 1$ can be fulfilled.
- Defining further entries in the pointer-vector IZEIG the starting pointers for each of the following array-data are marked by because all of the vectorial data stand on some large array B(.) in our FORTRAN-program package. If we put more than one subdomain into the k-th processsor consequently we get the corresponding entries in IZEIG($\cdot$,1,l), $l > 1$, the starting pointer of the l-th vectorial data set of the same type **(a)** – **(h)** is given.

2. vectorial data :

(a) IED(5,j) : the vector all of the edges of the subdomain, where

$j = 1, \ldots,$NUMED, and the following holds :

IED(1,j) :	the name of the starting point of the j-th edge
IED(2,j) :	the name of the ending point of the j-th edge
IED(3,j) :	the name of the midpoint of the j-th edge, if the edge is of curvilinear type otherwise zero stands here
IED(4,j) :	is set to be zero if the j-th edge is in an interior one and otherwise it is the name of the corresponding coupling boundary piece the j-th edge belong to
IED(5,j) :	= 1 if the type of the edge is a straight line = 2 if the type of the edge is a piece of some circle = 3 if the type of the edge is a piece of some parabola = 4 if the edge is an elliptic piece cut out of a circular cylinder = 5 if the edge is an parabolic piece cut out of a parabolic cylinder

(b) IECE(4,j) : the vector describing the faces the tetrahedrons are bounded by, where

$j = 1, \ldots,$NUMEL, and the following holds :

IECE(1,j):	
IECE(2,j):	are called the names of the 3 edges belong to the j-th triangle
IECE(3,j):	
IECE(4,j):	the j-th face-type (plane(=1), circularly(=2) or parabolicly(=3) cylindrical)

(c) ITETR(5,j) : the element connectivity vector containing tetrahedron's faces in each case, where

$j = 1, \ldots,$NUMTE, and the following holds :

ITETR(1,j):	
ITETR(2,j):	are called the names of the four faces uniquely describing
ITETR(3,j):	the j-th tetrahedron
ITETR(4,j):	
ITETR(5,j):	the material code name belonging to the j-th tetrahedron

Remarks :

* These vectors can be used in order to generate the finite element stiffness matrix quickly. The element connectivity vector that contains the names of the nodal points belonging to the j-th element is typical for finite element computations. Using the pointer-context "element-faces-edges-nodal points" described in [4] this data set is implicitly given.

(d) IBC(2+NDF,j) : the vector for coding the boundary properties, where

$j = 1,\ldots,$NUMBED, and the following holds :

IBC(1,j) :	the name of the face defining the j-th boundary-face
IBC(2,j) :	the material code name of the j-th boundary-face
IBC(3,j) :	is set 1,2 or 3 if Dirichlet, Neumann or boundary conditions
....	of third type are imposed on the j-th boundary-face, where the
IBC(2+NDF,j):	variable NDF (≥ 1) is the number of the degrees of freedom

(e) ICBN(4,j) : the vector defining the coupling boundary pieces of the 2D-ref. subdomains, where

$j = 1,\ldots,$NUMCRP, and the following holds :

ICBN(1,j):	the name of the first coupling point given on the j-th coupling boundary piece of the subdomain
ICBN(2,j):	the number of coupling points on the j-th coupling boundary piece
ICBN(3,j):	are the names of the two cross points that are starting or ending
ICBN(4,j):	points of the j-th coupling boundary piece of the subdomain

Remarks :

$*$ The coupling points that lie on the j-th coupling boundary piece are uniquely numbered in the natural sequence starting with the first name ICBN(1,j) that followed the cross point ICBN(3,j) immediately.

$*$ The finite sequence of the cross points ICBN(3,1),ICBN(4,1),ICBN(3,2),...... ... ,ICBN(3,NUMCRP),ICBN(4,NUMCRP) is closed and given in mathematically positive relation, where the condition ICBN(3,1)= ICBN(4,NUMCRP) is fulfilled.

(f) X(3,j) : the (x, y, z)- coordinate vector of all of the nodal points of the 3D-subdomain, where

$j = 1,\ldots,$NUMNP, and the following holds :

X(1,j) :	the x- coordinate of the j-th nodal point
X(2,j) :	the y- coordinate of the j-th nodal point
X(3,j) :	the z- coordinate of the j-th nodal point

Remark :

$*$ Increasing the index j monotonically the nodal point coordinates are given as follows : At first the coordinates of the crosspoints stand according to its local-sequentially natural numbering, secondly the coordinates of the coupling points stand according to the sequentially natural numbering of the coupling boundary pieces they belong to and now the coordinates of all of the interior nodal points complete the vector.

(g) XM(3,j) : the (x, y, z)- coordinate vector of the midpoints belonging to curvilinear edges possibly participated in defining the boundary of the subdomain, where

$j = 1,\ldots,$NUMMP, and the following holds :

XM(1,j) :	the x- coordinate of the j-th midpoint
XM(2,j) :	the y- coordinate of the j-th midpoint
XM(3,j) :	the z- coordinate of the j-th midpoint

Remark :

$*$ The numbering of these coordinate triples is sequentially natural, too, and mathematically positive related with respect to the z-height in addition to.

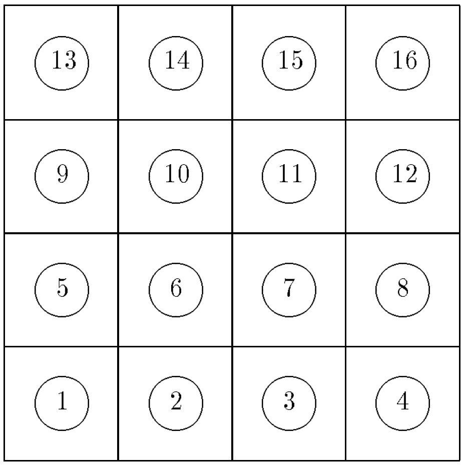
(h) IEDM(j) : the vector that includes specific refinement information coded for each edge of subdomain's tetrahedrons, where $j = 1,\ldots,$NUMED.

5 Numerical examples

The computation of the following two examples are performed on the Multicluster II equipped with 32 T 805 processors (30Mhz, 8Mbyte). The computational times given in the following Table 3 for example 1 and separately depicted in the case of the two others contains neither the time necessary for the rare communication and the cross point renumbering at the beginning of the program nor the time for the possible output of the mesh at its end. Hence, the measured time really indicates the effect of the performed parallelization, where in each case both the time needed for parallel grid smoothing of the 2D-reference meshes and for parallel interior nodal renumbering in all of the 3D-subdomains is involved.

5.1 Example 1 – An academic test problem

The first example is an academic one. The 2D-reference domain Ω is the $(0,4) \times (0,4)$ square divided up into 16 congruent subsquares, see Figure 10. The corresponding basic lines are the edges of the smaller squares. For all basic lines its division into pieces is of equidistant type specified by the corresponding number NR given in the first column of Table 3.



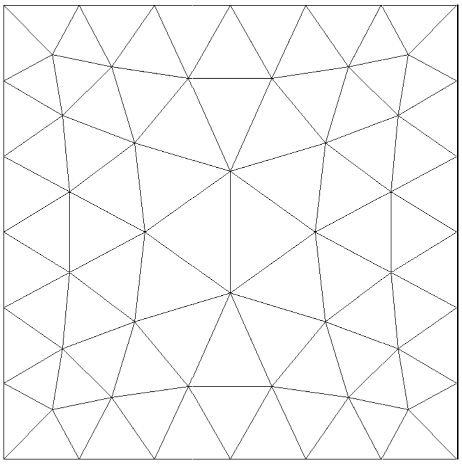


Figure 10: The reference square decomposed into the given 16 subsquares; One initial subdomain reference mesh with 5 divisions per each basic line

Table 3 presents the computational results belonging to the above square. As it was expected because of the totally uniform loadbalance between the 16 2D-subdomains given here and its uniform z-expansion, the performed 3D-mesh generation is very fast.

the number NR of basic line parts in the 2D-ref.-mesh	number of triangles in the 2D ref.-mesh	number of equidistant level (total number of generated tetrahedrons) measured CPU-time (in min:sec)					
		1	2	6	11	21	31
0	32	(96) 0:00,5	(192) 0:00,73	(576) 0:00,86	(1056) 0:00,9	(2016) 0:01,24	(2976) 0:01,5
0	64	(192) 0:00,56	(576) 0:00,7	(1152) 0:00,72	(2112) 0:00,78	(4032) 0:01,17	(5952) 0:01,6
1	256	(768) 0:00,96	(1536) 0:01,0	(4608) 0:01,1	(8448) 0:01,2	(16128) 0:01,3	(23808) 0:01,5
3	640	(1920) 0:01,0	(3840) 0:01,1	(11520) 0:01,2	(21120) 0:01,4	(40320) 0:01,6	(59520) 0:02,0
5	1881	(3762) 0:01,13	(7524) 0:01,27	(22572) 0:01,4	(41382) 0:01,9	(79002) 0:02,29	(116622) 0:02,69
10	4604	(13812) 0:02,04	(27624) 0:02,47	(82872) 0:03,1	(151932) 0:04,41	(290052) 0:05,78	(428172) 0:07,69

Table 3: The CPU-time concerning distinct basic line-partitionings and main-levels

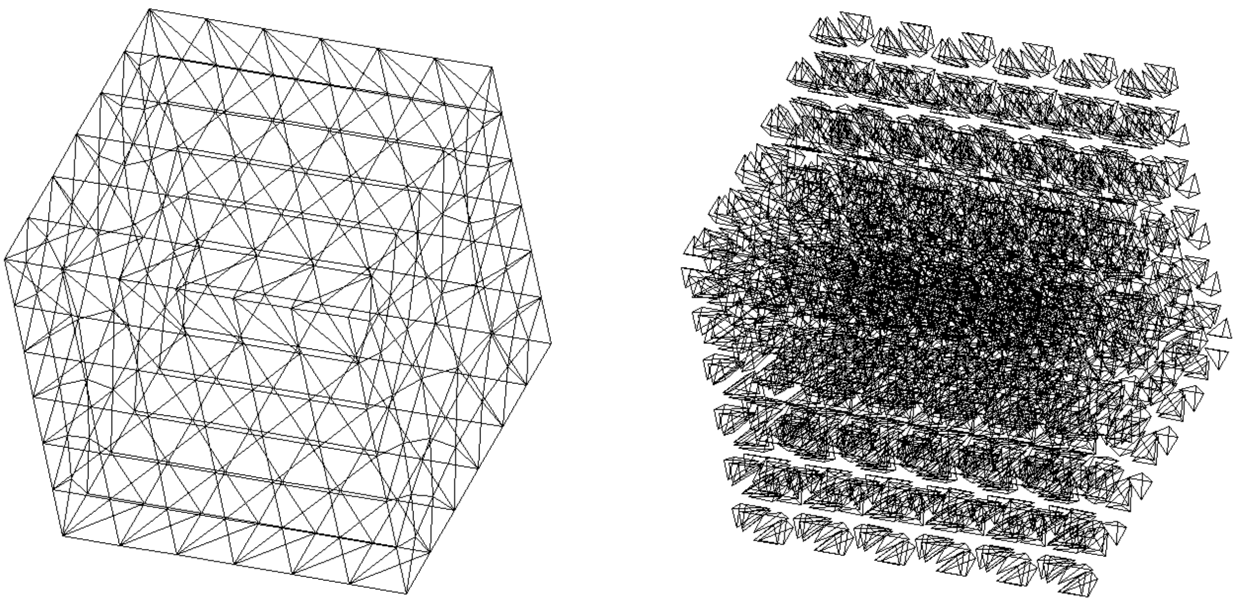


Figure 11: The (un)shrinked 3D-mesh in one of the 16 cubes having the above reference triangulation and 6 tetrahedral levels

5.2 Example 2 – A mechanical cylinder with partially opened sheet

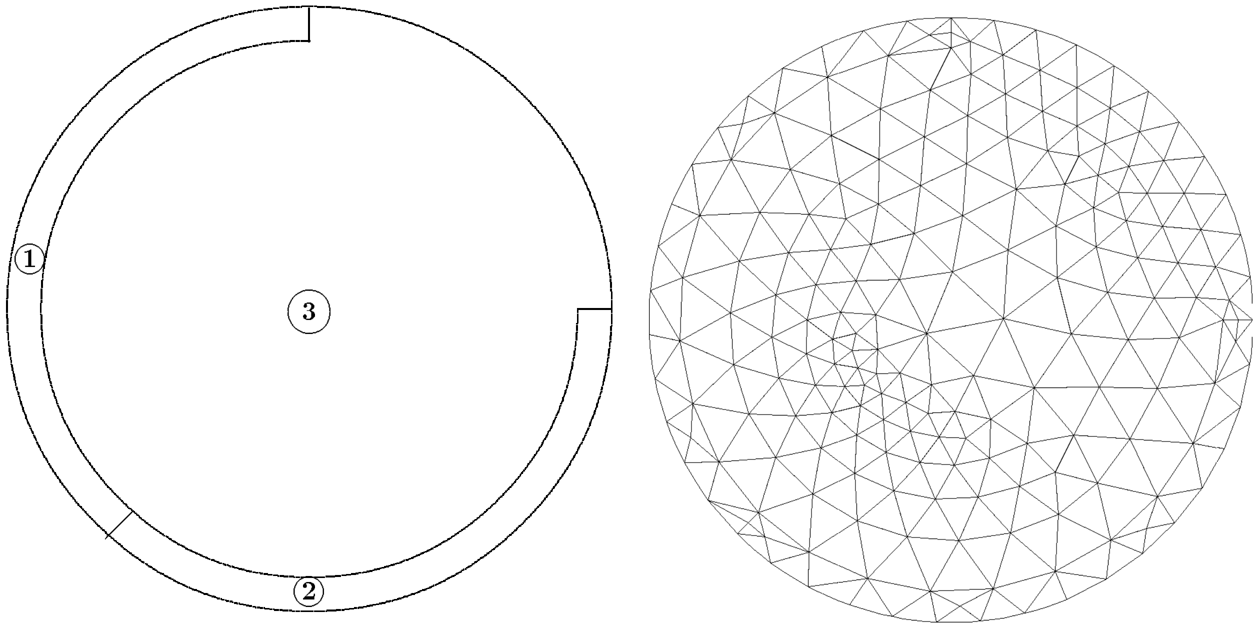
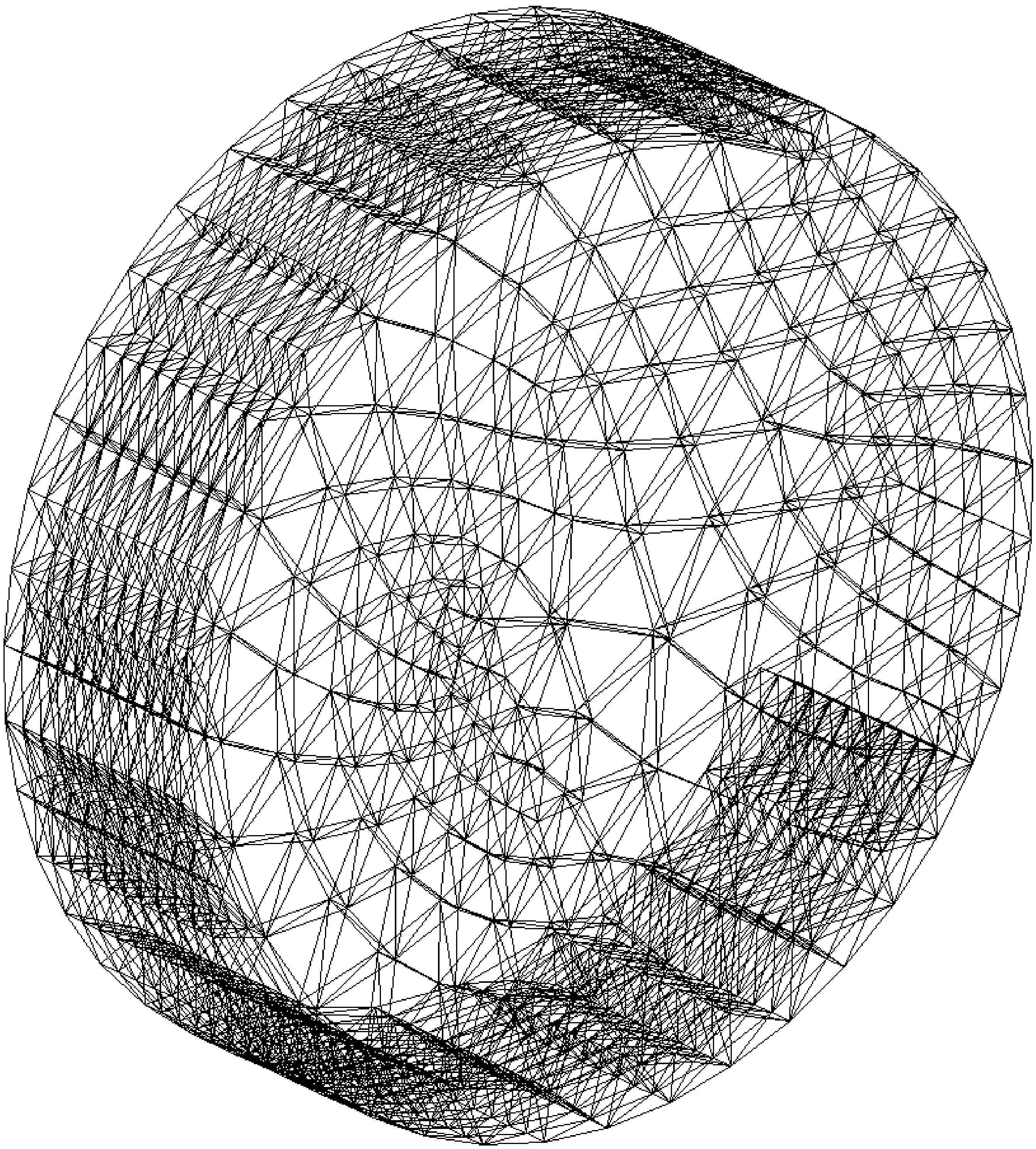


Figure 12: 2D-ref. DD and the corresponding mesh for the mechanical cylinder with cut sheet

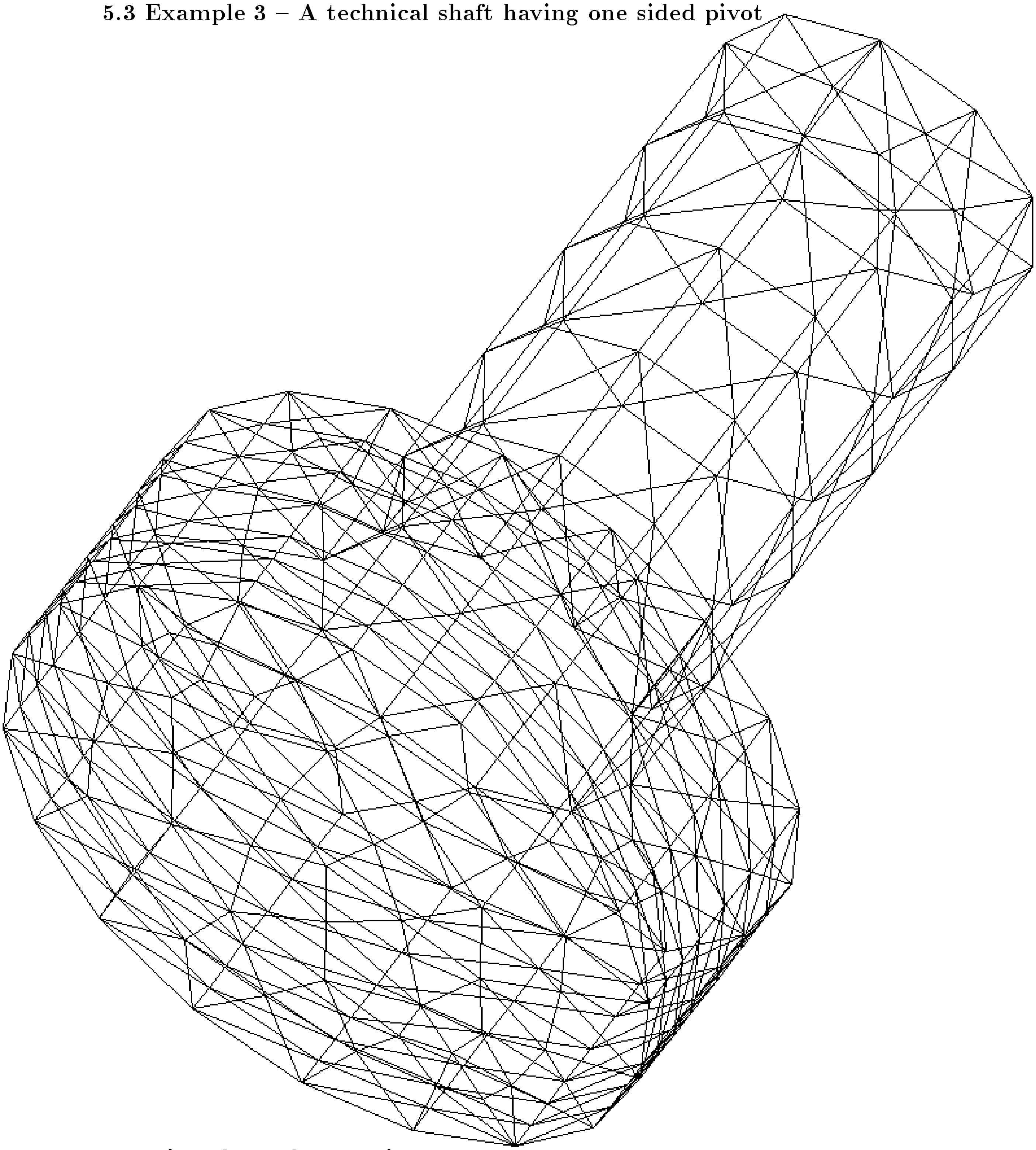


CPU-times for mesh generation per

3 processors : 0:01,10 sec
1 processor : 0:02,92 sec
speed up : 2.65

Figure 13: hzyls_3d.dat; 978 nodes, 2961 tetrahedrons; (1080, 1020, 861 per processor 0, 1, 2)

5.3 Example 3 – A technical shaft having one sided pivot



CPU-times for mesh generation per

3 processors : 0:00,52 sec

1 processor : 0:01,44 sec

speed up : 2.76

Figure 14: welle_3dn.dat; 560 nodes, 2400 tetrahedrons; (840, 840, 720 per processor 0, 1, 2)

References

[1] R.E. Bank, *PLTMG User's Guide (June 1981 version)* (Techn. Report, Dep. of Mathematics, Univ. of California at San Diego, La Jolla, CA, 1982).

[2] G.F. Carey (ed.), *Parallel Supercomputing: Methods, Algorithms and Applications* (John Wiley and Sons, Chichester, New York, Brisbane, 1989).

[3] G. Globisch, *Robuste Mehrgitterverfahren für einige elliptische Randwertaufgaben in zweidimensionalen Gebieten* (Dissertation, Technische Universität Chemnitz, 1992).

[4] G. Globisch, PARMESH – a parallel mesh generator, Preprint-Reihe *der Chemnitzer DFG-Forschergruppe "Scientific Parallel Computing"*, (SPC 93_3, June 1993); submitted to *Parallel Computing"*.

[5] Temporary GRAPE Documentation, *Technical Report*, (Sonderforschungsbereich 256, Universität Bonn, 1993).

[6] G. Haase, T. Hommel, A. Meyer, M. Pester, Bibliotheken zur Entwicklung paralleler Algorithmen, Preprint-Reihe *der Chemnitzer DFG-Forschergruppe "Scientific Parallel Computing"*, (SPC 94_4, February 1994).

[7] G. Haase, U. Langer, A. Meyer, Parallelisierung und Vorkonditionierung des CG-Verfahrens durch Gebietszerlegung, in: *Parallele Algorithmen auf Transputersystemen, Teubner-Scripten zur Numerik III* (Teubner, Stuttgart, 1992).

[8] W. Hackbusch, *Parallel Algorithms for Partial Differential Equations* (Vieweg, Braunschweig, 1991); or in: *Proceedings of the Sixth GAMM-Seminar*, (Kiel, 1990).

[9] B. Heinrich, U. Langer, A. Meyer, M. Pester, Algorithmische Grundlagen der Simulation von angewandten Problemen der Kontinuumsmechanik auf massiv parallelen Rechnern, (Gründungsantrag der DFG-Forschergruppe SPC, TUC Chemnitz, 1992).

[10] K. Ho-Le, Finite element mesh generation methods: a review and classification, *Comput. Aided Des.*, (10, pp. 27-38, 1988).

[11] Th. Hommel, Algorithmische Aspekte des BPX-Vorkonditionierers, in: U. Langer, ed.,*Proceedings zum DFG-Workshop "Implementierung paralleler Algorithmen auf Transputersystemen" (DFG Forschungsschwerpunkt Randelementmethoden)*, (Fachbereich Mathematik, Technische Universität Chemnitz, 1992).

[12] C. Israel, Ein hierarchischer paralleler 2D-Netzgenerator für 6-Knoten-Dreiecke, in: M. Pester, ed., *Proceedings zum Workshop "Paralleles Pre- und Postprocessing"*,(DFG-Forschergruppe "SPC", Fachbereich Mathematik, Technische Universität Chemnitz, Juni 1993).

[13] H. Jin, R.I. Tanner, Generation of unstructured tetrahedral meshes by advancing front technique, *Int. j. numer. methods eng.*, (vol. 36, pp. 1805-1823, 1993).

[14] B.P. Johnston, J.M. Sullivan, jr., A normal offsetting technique for automatic mesh generation in three dimensions, *Int. j. numer. methods eng.*, (vol. 36, pp. 1717-1734, 1993).

[15] M. Jung, R. Wohlgemuth, Generation of hierarchical finite element meshes for interface problems, *Preprint Nr. 202*, (Fachbereich Mathematik, Technische Universität Chemnitz, 1991).

[16] W. Queck, ed., *FEMGP (Finite Element Multigrid Package)*, (Programmdokumentation), Technologieberatungszentrum Parallele Informationsverarbeitung GmbH (TEZ*PARIV), Bernsdorferstr. 210-212, 09126 Chemnitz.

[17] W. Queck, The Finite Element Multigrid Package FEMGP - A software tool for solving boundary value problems on personal computers, in: S. Hengst, ed., *Proceedings of the GAMM-Seminar on Multigrid-Methods, Gosen, Germany, September 21-25, 1992* (IAAS Berlin, Report No. 5, ISSN 0942 - 9077, Berlin, 1993).

[18] H.D. Simon, Partitioning of unstructured problems for parallel processing, *Computing Systems in Engineering*, (2:(2/3), pp. 135-148, 1991).

[19] J.F. Thompson; Z.U.A. Warsi; C.W. Mastin, *Numerical grid generation (Foundation and Applications)*, (North Holland, 1985).

[20] A.G. Tsybenko, N.G. Vashchenko, N.G. Krishchuk, Yu.O. Lavendel, *Avtomatizirovannaya sistema obsluzhivaniya konechno-elementnykh raschyotov* (Golovnoe izdatelstvo isdatelskogo obedineniya "Vishcha shkola", Kiev, 1988).

Author's address :

Dr. rer. nat. G. Globisch
Technical University of Chemnitz–Zwickau
Department of mathematics
PSF 964, **D-09001** Chemnitz
e-mail: gglobisc@mathematik.tu-chemnitz.de