Technische Universität Chemnitz-Zwickau
Sonderforschungsbereich 393
Numerische Simulation auf massiv parallelen Rechnern

Gerhard Globisch*

The hierarchical preconditioning having unstructured threedimensional grids

Preprint SFB393/97_ 25

Abstract

Continuing the previous work in [4] done for the 2D-approach in this paper we describe the Yserentant preconditioned conjugate gradient method as well as the BPX–preconditioned cg–iteration fastly solving 3D-elliptic boundary value problems on unstructured quasi uniform grids. These artificially constructed hierarchical methods have optimal computational costs. In the case of the sequential computing several numerical examples demonstrate their efficiency not depending on the finite element types used for the discretiziation of the original potential problem. Moreover, implementing the methods in parallel first results are given. Our solution strategy can be of enormous importance in the industrial engineering, when often only the nodal coordinates and the element connectivity of the underlying (fine) discretization are available.

Key words: automatical mesh generation, finite element methods, hierarchical preconditioning, multilevel methods, parallel computing, partial differential equations.

AMS(MOS) subject classification: 65N55, 65N50, 65N30, 78A30

* The work of the author is supported by the Deutsche Forschergemeinschaft (DFG).

Preprint-Reihe des Chemnitzer SFB 393

SFB393/97_ 25 **December 1997**

Contents

Authors' address:

Dr. rer. nat. Gerhard Globisch
Faculty for Mathematics
Technical University Chemnitz
D - 09107 Chemnitz, Germany

e-mail: `gerhard.globisch@mathematik.tu-chemnitz.de`

http://www.tu-chemnitz.de/sfb393/people/globisch.html

1 Introduction

We want to solve the following threedimensional boundary value problem having the formally selfadjoint differential operator L in the domain Ω, where Dirichlet as well as Neumann boundary conditions may be imposed on its boundary $\partial\Omega$.

$$\begin{aligned} L\,u &= f \qquad \text{in} \quad \Omega \subset \mathbb{R}^3 \\ l\,u &= g \qquad \text{on} \quad \partial\Omega = \Gamma \end{aligned}$$

Discretizing the problem e.g. by means of the finite element method finally we get a large scale system of linear algebraic equations

$$K\,\underline{u} = \underline{f}\,,$$

where K is the symmetric and positive definite stiffness matrix and $\underline{f}$ the given right hand side vector.

Our aim is the efficient numerical solution of the system by hierarchical methods, although, in practice, we have its unstructured discretization available only. For the 2D-case, in [4, 8] we determined the structured auxiliary problem into which the original one can be embedded. Introducing an operator R we defined the one-to-one correspondence between the N nodes of the unstructured quasiuniform mesh Ω^h and the nodes of the hierarchically discretized square Π^h defining the fictitious space. It is easy to see that the approach can be straightforwardly transfered to 3D-problems. Here, we embed the unstructured threedimensional mesh into the hierarchically discretized cube Π^h consisting of 2^{3J} congruent subcubes $D_{ijk},\ i,j,k = 1,2,\ldots,2^J$ which belong to the level J, where $J = 0,1,2,\ldots$ is a positive integer to be fixed. Hence, this structured hexahedral grid has L^3 nodal points, where $L = 2^J + 1$ is valid. The side length $\bar{h}$ of the subcubes D_{ijk} is chosen appropriately such that every subcube contains at least one vertex of the original grid Ω^h consisting of N grid points. We set $\bar{h} < (2\sqrt{3})^{-1}\min(d_i)\,,\ i = 1,2,\ldots,N$, where d_i represents the maximum of the radii of balls that may be inscribed into the union of finite elements having the i-th node in common. The side length l of the cube Π which must contain the grid Ω^h fulfils $\bar{h} = l\,2^{-J}$ chosing the depth J as small as possible. The minimum union of subcubes $D_{ijk} \subset \Pi^h$ encompassing the unstructured grid Ω^h is called the step form approximation Q^h. Using the grids Ω^h and Q^h, now we are able to define corresponding finite element spaces for applying the fictitious space lemma. Finally, by means of this lemma we derive the corresponding spectral equivalence inequality describing the preconditioning property of the artificially constructed hierarchical preconditioner belonging to the auxiliary grid points.

Considering the effect of our preconditioning we get the following result which was proved in [8] taking distinct boundary conditions imposed on the approximate boundary Γ^h of the domain Ω into account. There are positive constants c_1 and c_2 independent of the mesh size parameter h such that

$$c_1(K^{-1}\underline{u},\underline{u}) \le (C^{-1}_{\Pi^h,\mathrm{bc}(\Gamma^h)}R^*\,\underline{u},\,R^*\,\underline{u}) \le c_2(K^{-1}\underline{u},\underline{u})$$

is fulfilled for all vectors $\underline{u} \in \mathbb{R}^N$ belonging to the original discretization. For simplicity here we identify the number of unknowns (dof) with the number N of vertices in the unstructured grid Ω^h. The underlying theory is presented in detail e.g. in [7, 9] and the references therein. Provided that scalar computing runs the mentioned papers prove the convergence rate of the iterative process to be as fast as it is the case for the conventional hierarchical solution method, i.e., it is (nearly) independent of the mesh size. Performing the BPX–preconditioning artificially within the cg–iteration process the condition number of the operator $RC_{\Pi^h}^{-1}R^*K$ is of order $O(1)$. In the case of the artificially constructed Yserentant preconditioning we may have the condition number $\kappa(RC_{\Pi^h}^{-1}R^*K) = O(2^J)$.

In the next section we discuss essential aspects of the numerical implementation of the new hierarchical methods **art**ificially constructed. We do it in the case of the Yserentant preconditioning (**artYs**) as well as in the more important case of the artificial BPX-preconditioner (**art**BPX).

In the last section we illustrate the efficient implementation of the two hierarchical preconditioners computing several 3D–potential problems discretized using unstructured tetrahedral and hexahedral grids. Moreover, in the case of the tetrahedral grids we are able to compare the artificially constructed hierarchical iteration based on the canonically performed refinement of the coarse and structured user triangulation with the same method using really unstructured fine grids generated by an advancing front mesh generator described in [3]. Finally, first numerical results of the parallel implementation of our approach are given, where the corresponding numerical analysis is yet under consideration. The iteration numbers are satisfactory although the unfair comparison with the parallelized structured methods is rather bad. The basis of the implementation of the unstructured parallel solvers is a non-overlapping domain decomposition data structure (see e.g. [5]) such that they are well-suited for parallel machines with MIMD–architecture. This section is also an impressive performance to demonstrate the practical importance of the designed methods. Often in the industrial engineering boundary value problems have to be solved, where a (rather) fine mesh of the domain and the discretization concept are given sometimes already resulting in the corresponding system of equations. But no fast hierarchical solver can be applied because nothing is known about the grid structure. Using our approach this bottleneck isn't any more.
To do it in advance the following survey sums up both the advantages (marked by "+") and the shortcomings (marked by "-") of the method.

+ Concerning the iteration number we got a robust approach for solving partial differential equations efficiently on sequential computers having the good convergence property of the preconditioned cg-iteration.

+ Based on the modular toolkit the implementation of the method into available software packages is easy, especially in comparison with algebraic multigrid.

+ The solution strategy is also of considerably practical importance in engineering.

+ The discretization of the original problem can be performed independent of the method. Above all, various types of finite elements can be used.

- The memory size additionally required for the method is not negligible especially in the case when the unstructured mesh tends to lose the quasiuniformity. Moreover, the more the quasiuniformity is deteriorated progressively the more the iteration number of our method does increase.

- The computation of the step form approximation and the construction of long BPX-lists do substantially enlarge the CPU-time really needed.

- Having interface problems the fast convergence speed is injured.

- Up to now the convergence property of the implemented parallel version of our method is not satisfactory enough.

2 Aspects of the numerical implementation

Using the number marks in the following commando tool picture we describe the new algorithmic components and their cooperation. Furthermore, we get insight into the handling with our method completing the program package SPC-PM Po 3D, see [1, 2].

```
run -f2 2 2 tet.ppc
run : Creating 2 * 2 descriptor by calling mkdesc.
run : Starting D-Server at mordred link 2.
  # ############################################################## #
  #                                                                #
  #  SSSS  PPPPP   CCCC      PPPPP  M       M   PPPPP       333  #
  # SS  SS PP  PP CC  CC     PP  PP MM     MM   PP  PP     33 33 #
  # SS     PP  PP CC         PP  PP MMM   MMM   PP  PP        33 #
  #  SSSS  PPPPP  CC     ### PPPPP  MM MMM MM   PPPPP 000    333  #
  #     SS PP     CC         PP     MM  M  MM   PP   00 00     33 #
  # SS  SS PP     CC  CC     PP     MM     MM   PP   00 00  33 33 #
  #  SSSS  PP      CCCC      PP     MM     MM   PP    000    333  #
  #                                                                #
  # ############################################################## #
  #                                                                #
  #             Programm-Modul 3D-Potentialprobleme                #
  #                            Version:  3.20                     #
  #                                                                #
  #                   DFG-Forschergruppe "SPC"                     #
  #         TU Chemnitz-Zwickau, Fakultaet fuer Mathematik         #
  #                                                                #
  #     Th.Apel, A.Meyer, M.Meyer, F.Milde, M.Pester, M.Thess     #
  #                                                                #
  #   16-MB-Variante ( 3500000 Worte) - bis zu 1024 Prozessoren   #
  #                          in Benutzung:          4 Prozessor(en) #
  #                          Gelinkt mit bsp.z                     #
  #                                                                #
  # ############################################################## #
       ****************************************************
       *          Belegung der Steuerparameter            *
       * (kann mittels File control.tet angepasst werden) *
       ****************************************************
       *                                                  *
       *  vertvar    =   2       lin_quad   =   1         *
       *  nen2d      =   3       nen3d      =   4         *
       *  femakkvar  =   3       loesvar    =   5         *
       *  nint2ass   =  34       nint3ass   = 121         *
       *  nint2error =  34       nint3error = 531         *
       *  iter       = 500       epsilon    = 0.10E-05    *
       *  ion        =  10       ndiag      = 150         *
       *                                                  *
       *  Verzeichnis fuer Netze :  mesh3/                *
       ****************************************************

Filename: r1c48

GEWUENSCHTE ZAHL VON VERFEINERUNGSSCHRITTEN
   -1 = NEUES NETZ
   -2 = PROGRAMM BEENDEN
EINGABE : 2

EINLESEN DER NETZDATEN AUS : mesh3/r1c48.std

Version 2.0 files sind noch in der Testphase,
Bei Problemen bitte umgehend bei mir melden (Dag)
Wuerfel, Kantenlaenge 2, unten spring. Dirichlet
```

```
Gerhard Globisch
24.02.1995
POISSON
XDB
3D
data_read done
No error

EINLESEN BEENDET, IER=    0

VERTEILUNG DER TETRAEDER DURCH REKURSIVE SPEKTRALBISEKTION.

Anzahl der Elemente in den Prozessoren:
  24  24  24  24
  12  12  12  12

 ==> in    4 Tetr.    4-mal Flaechen getauscht.
NETZ VERFEINERT VFS=1
NETZ VERFEINERT VFS=2

************************************************************
**                      AUSGABEMENUE                      **
************************************************************
*      0  : WEITER                                         *
*      4  : AUSGABE DER NETZDATEN                          *
*      5  : AUSGABE DER RANDKETTENDATEN                    *
*      8  : AUSGABE DER NETZDATEN IN STANDARDFILE          *
************************************************************
 -> EINGABE : 0

 ***********************************************************
 Wollen Sie das Globisch-Nepomnyaschikh-Verfahren?                       (1)
 (Geben Sie j/J/y/Y oder n/N ein) ----->j
 1: full auxiliary hexahedral grid version?
 2:     auxiliary tetrahedral grid version?
 (Bitte geben Sie 1 oder 2 ein) ------->1
 Mehrprocessorverarbeitung: Wollen Sie viele Ausgaben?                   (2)
 (Geben Sie j/J/y/Y oder n/N ein) ----->n
 ***********************************************************

 ICH =            0: h_max / h_min =    1.732051
 Faktor > 0 eingeben ( = 1 ?) --->2                                      (3)
 ICH =   0 distmin =     0.250000E+00   J =  3 L =     9
 ICH =   0 xedge =     0.000000E+00       0.000000E+00       0.000000E+00

 Statt Hilfswuerfel ein Hilfsquader gewuenscht?                          (4)
 (Geben Sie j/J/y/Y oder n/N ein) ----->n

************************************************************
**                      AUSGABEMENUE                      **
************************************************************
*      0  : WEITER                                         *
*      1  : 3D-GRAFIK MIT GRAPE                            *
*      2  : 2D-GRAFIK SCHNITT/OBERFLAECHE                  *
*      4  : AUSGABE DER NETZDATEN                          *
*      5  : AUSGABE DER RANDKETTENDATEN                    *
************************************************************
 -> EINGABE : 0

START GENERIEREN/ASSEMBLIEREN
 Zeiten fuer Warten+Kommunikation [s]
 Prozessor
 log. /phys.    input :    in % :   output:    in % :   gesamt:
    0   0  0      0.00      0.00      0.00      0.00      0.15
    1   1  0      0.00      0.00      0.00      0.00      0.15
    2   0  1      0.00      0.00      0.00      0.00      0.15
    3   1  1      0.00      0.00      0.00      0.00      0.15
 reine Arithmetikzeit (max):      0.15
 Display: PseudoColor  ***   0 colors allocated ***
ASSEMBLIEREN BEENDET

* Probleminformationen (lokal Prozessor P):
 - globale Anzahl Crosspoints   :      27
 - Anzahl der Knoten (lokal)    :     225
 - davon:  lok. Crosspoints     :      12
           Summe der Randketten :     201
           Koppelknoten         :     213
           innere Knoten        :      12
 - Anzahl der Koppelkanten      :      33
 - Anzahl der Koppelflaechen    :      34

* Probleminformationen (  global ):
 - Anzahl der Prozessoren:      4
 - Anzahl der Knoten     :    729
 - davon :  Koppelknoten :    681
 -         interne Knoten :     48
 ->        Gesamtanzahl der Freiheitsgrade :    729

* Start der Simulation: Vorkonditionierung Nr. 5
                                          <enter> v                      (5)
 neue Variante=2
                                          <enter>

 Wuenschen Sie die Betrachtung der BC im Traeger?                        (6)
 (Geben Sie j/J/y/Y oder n/N ein (=n?)) -->n
 Geben Sie rmult fuer R^T ein  (=0?)   --->0                             (7)
 Aufbau der Stufenapproximation; bitte warten.                           (8)
 ICH:           0 card(Q^h) =         235 card(eps) =         628
 Schichtvisualisierung der Stufenapproximation.                          (9)
 (weiter bei Eingabe von ilevel <=0)
 Eingabe Prozessornummer; ilevel ======> 0 0
 L^3 =         2916       Sigma[card(Q^h)] =         920                (10)
 dof(L^3) =         2916

  IT     (r,w)         (As,s)        ALFA          BETA         Eta
   1  3.235708E+06  7.707548E+06 -4.198103E-01  0.000000E+00   1.00
  10  1.187671E+05  2.297546E+05 -5.169301E-01  7.359730E-01   0.69
  20  5.360262E+02  8.915826E+02 -6.012076E-01  6.089517E-01   0.63
  30  1.877404E+00  2.250594E+00 -8.341817E-01  6.308484E-01   0.61
  40  2.859391E-03  4.739238E-03 -6.033442E-01  5.265399E-01   0.59
  50  4.064511E-06  5.393838E-06 -7.535470E-01  4.730400E-01   0.57
 IT=         51
 Zeiten fuer Warten+Kommunikation [s]

 Prozessor
 log. /phys.    input :    in % :   output:    in % :   gesamt:
    0   0  0      0.40     28.32      0.30     21.35      1.41
    1   1  0      0.36     25.43      0.57     40.01      1.41
    2   0  1      0.35     24.60      0.59     41.87      1.41
    3   1  1      0.41     28.77      0.66     47.00      1.41
 reine Arithmetikzeit (max):      0.71
```

```
**********************************************************
**                      AUSGABEMENUE                    **
**********************************************************
*      0  : WEITER                                       *
*      1  : 3D-GRAFIK MIT GRAPE                          *
*      2  : 2D-GRAFIK SCHNITT/OBERFLAECHE                *
*      4  : AUSGABE DER NETZDATEN                        *
*      5  : AUSGABE DER RANDKETTENDATEN                  *
*      6  : AUSGABE DER LOESUNG                          *
*      7  : AUSGABE VON FEHLERNORMEN                     *
**********************************************************
 -> EINGABE : 0

GEWUENSCHTE ZAHL VON VERFEINERUNGSSCHRITTEN
   -1 = NEUES NETZ
   -2 = PROGRAMM BEENDEN
EINGABE : -2
```

(1) After the method for performing the hierarchical preconditioning artificially is ordered at the beginning the user is asked for determining the structure of the auxiliary hierarchical grid. The option "1" is used for generating the hierarchical list of the hexahedral grid later having the mesh size $\bar{h}$, where the parameter LC_DAT included by the file net3ddat.inc is equal to zero. The option "2" produces the hierarchical list of the correspondingly defined auxiliary tetrahedral grid. In this case the parameter LC_DAT is equal to one. The numerical results belonging to the "2"–option are always a bit worser than in the "1"–case. Furthermore, regarding the iteration numbers sometimes the "1"-results were even better than the results got by the conventionally hierarchical method on tetrahedral grids used for the unfair bracket–comparison given in the tables of the last section.

The points of the grids Π^h are numbered linewise from the left to the right starting from the plane below in front going backwards ending at the plane above. The memory size needed to store their auxiliary hexahedral Yserentant–list is equal to $L^3 + 3L_{J-1}^2(L_{J-1}-1)*3 + 3L_{J-1}(L_{J-1}-1)^2*5 + 2^{3(J-1)}*9$ and $L^3 + L^3*4$ otherwise, where $L_{J-1} = 2^{(J-1)}+1$. In both cases the above memory size is consisted of the description vector with L^3 components and the entries defined by son–father–relations. Having the auxiliary hexahedral mesh there are grid points possessing 2 and 4 and 8 fathers, respectively, whereas in the tetrahedral mesh the grid point has at least 2 fathers. The corresponding BPX–lists have equivalent lenghts. When the user gives standard input or $v = 4$ at stage (5) the BPX–list is automatically generated from the Yserentant–list. The message about the corresponding process in action is delivered near the stage (9).

(2) When the parallel version of the program runs a lot of output information belonging to that what happens at each of the used processors can be avoided.

(3) By this real value input the user manipulates the computed depth J of the auxiliary grid. Stretching the corresponding parameter $\bar{h}$ by the given factor the default is equal to one. Computing in parallel the formal zero input does allow to set single input specifically for each processor. Sometimes factors equal to two or to other powers of two help to reduce the required memory size. If the factor was defined too large at the stage (8) the program will warn of violating the one–to–one correspondence between the points in the unstructered and the auxiliary structured grid. The given output distmin is equal to the parameter $\bar{h}$, where the corresponding depth J is added. The tripel (x_m, y_m, z_m) of real values listed here using the denotation xedge is defined as follows: $(x_m, y_m, z_m) = \{(\min(x), \min(y), \min(z)),\ \ (x, y, z) \in \Pi\}$, where the auxiliary cube / cuboid Π is already centered w.r.t. the domain Ω.

(4) Instead of the hierarchical cube Π^h the user may order an auxiliary cuboid, which can be useful especially in such cases when the three coordinate range expansions of the domain Ω differ from each other considerably, where the unstructered grid Ω^h inside is correspondingly discretized by relatively longish and thin elements, respectively. In this case three output parameter distx, disty distz describing the edge lengths of the cell cuboid are given.

(5) Because no coarsest grid is available when the unstructured method is applied in parallel, the types $v = 3$ and $v = 5$ which initialize the preconditioning at the cross points directly solving the corresponding system are automatically switched to 2 (artYs-method) and 4 (artBPX-method), respectively.

(6) According to the theoretical approach given in [4, 8] the hierarchical preconditioner is defined distinctly taking the boundary conditions into account when throughout the auxiliary constructed grid hierarchy the supports of the corresponding grid functions are considered w.r.t. the discrete boundary Γ^h. At the given stage the user may order the option which is numerically expensive. Doing so, often it does not improve the convergence behaviour. However, the corresponding subroutine is not yet fully developed for the general case of having arbitrary shape of Ω including the boundary.

(7) The input value influences the construction of the step form approximation. Moreover, it weights the Jacobi- preconditioning matrix specifically. The default (rmult =0) gives order for performing the Jacobi-preconditioning as well known, cf. [4]. In most cases by this option we get a good result. However, computing linear elasticity problems often rmult = 2.2 was the better parameter especially in the case of the artBPX-method.

(8) Here, using the artificial grid Π^h the step form approximation Q^h encompassing the unstructured mesh Ω^h is computed for defining the Jacobi- preconditioning matrix, cf. [4]. Relating it to the corresponding processor the output information includes the number card(Q^h) of points in Q^h and the total number card(eps) of "critical" decisions made in the immediate neighbourhood of the element boundaries ∂e_h.

(9) By means of the little tool behind the given stage the user can get insight into the step form approximation of the domain Ω visualizing cross sections of Π^h vertically defined throughout this auxiliary grid inside the cube. The first integer input is the number of the processor containing parts of the discretization when we do compute in parallel. The second number k, $k = 1, 2, \dots, L$ determines the location of the cross section through Π^h defined from below to above w.r.t. the z-direction. When this number is specified less than zero the program is continued.

(10) The number L^3 of grid points in Π^h as well as the total number of points in Q^h are given. In the case of the artBPX-method the number L^3(BPX) of list–components is added. The corresponding number of unknowns dof(L^3) included in the long correction vector $\underline{v}$ completes the output data. In the parallel case these values are the corresponding maxima after the cubedo-operation is done throughout the processors.

Additionally considering the algoritmic description in the next section the number of numerical operations hidden in (1)–(10) is equivalent to the number N of unknowns belonging to the original discretization. Thus, we may conclude as follows: In the case of "artYs" as well as in the case of "artBPX" the preconditioner has an optimal computational cost, i.e. the number of arithmetic operations required for their implementation is proportional to the number of unknowns in the problem.

3 Short description of the modular toolkit

The picture gives the connexion of the new subroutines included in the libraries of the program package SPC-PM Po 3D to make the "Globisch-Nepomnyaschikh"-preconditioner available. Most of them complete the source code in the subdirectory Solve, cf. [1, 2]. Sometimes the presented subroutine calls further subroutines which are not outlined since their task is not so essential.

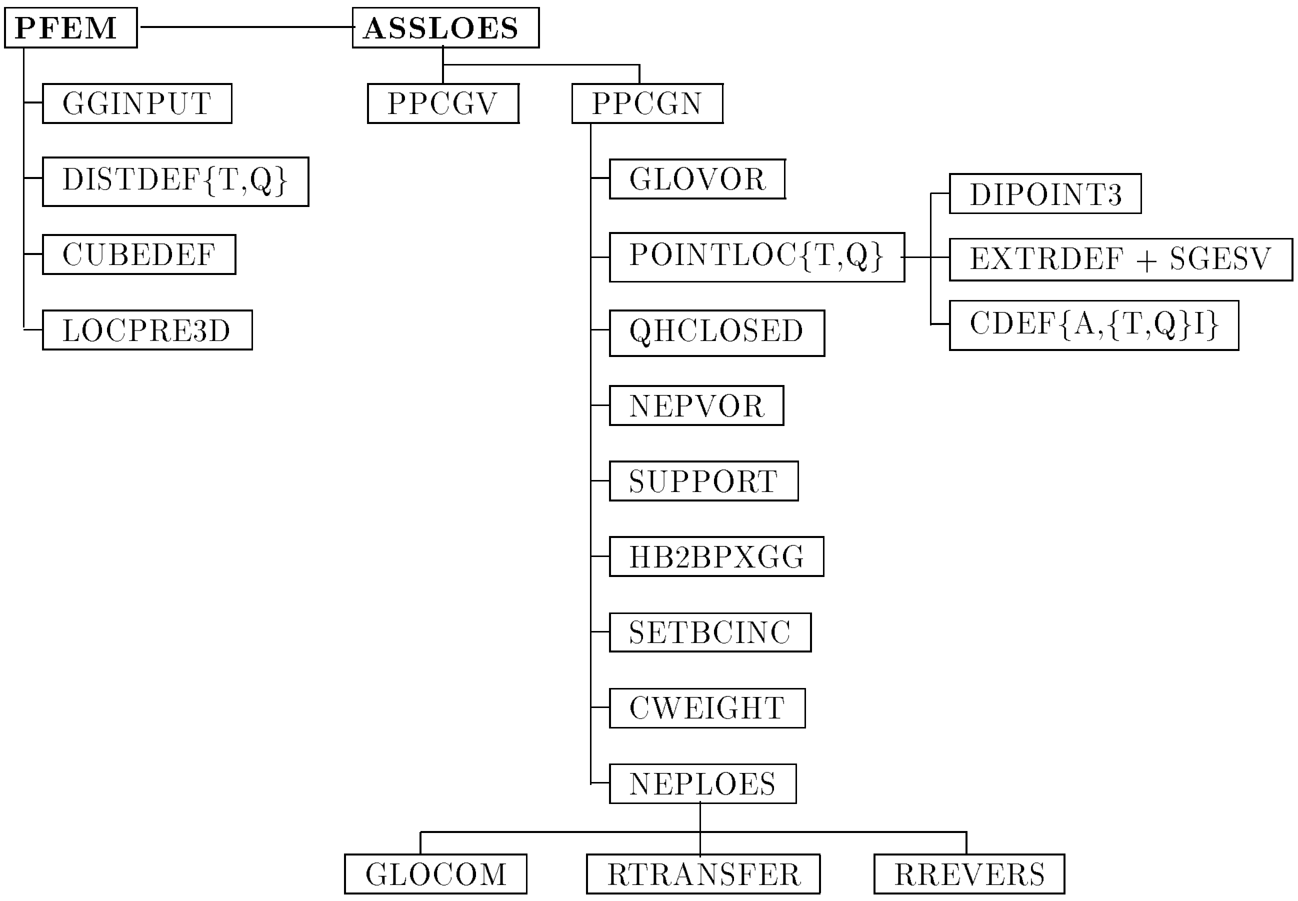


Figure 1: Scheme of new routines called by the main program **PFEM** and by **ASSLOES**

In the following the short decription of the new routines let be given. The subroutine

GGINPUT asks for the "Globisch-Nepomnyaschikh"-preconditioner and reads a few initial input for if it is selected. The user may specify whether the auxiliary grid hierarchy should be either of hexahedral (input 1) or tetrahedral (input 2) kind.

DISTDEF{T,Q} computes the mesh size parameter $\bar{h}$ both in the case **T** of an unstructured tetrahedral grid and in the case **Q** of an unstructured hexahedral mesh. Taking the vertices of the tetrahedron the perpendiculars w.r.t. the opposite face are used for defining the distance between the vertex and the face in each case. For all of the tetrahedra searching the minimum d of the distances the parameter $\bar{h}$ is defined to be $\bar{h} = \sqrt{3}\, d/2$. Having unstructured hexahedral meshes at first each of the hexahedra is devided up into six auxiliary tetrahedra to be concerned as above for computing the parameter $\bar{h}$. To become more robust manipulating with the auxiliary grid size parameter the user can specify a factor for multiplying $\bar{h}$ with. For saving memory size, often the factor may be larger than one without disturbing the one to one correspondence between the grid points of the unstructured mesh and those of the auxiliary hexahedral mesh. Running the program we get notice when the factor is too large and the mapping is violated.

CUBEDEF defines the auxiliary cube encompassing the unstructured mesh. The length l of the cube side is calculated taking the parameter $\bar{h}$ and the appropriately chosen depth J of the auxiliary hexahedral grid inside into account such that we have $\bar{h} = l\,2^{-J}$. Moreover, the location of the cube is centered w.r.t. the x, y, and z ranges of the original domain. Instead of the cube, the user is asked for a cuboid encompassing the mesh as above and having the maximum side length l. The side lengths of the cuboid are set according to the coordinate ranges of the domain, where no centering is made.

LOCPRE3D defines the hierarchy inside of the auxiliary hexahedral grid meshing the cube as well as the cuboid having the hierarchical depth J. Finally, this results in computing the auxiliary hierarchical lists in the case "1" and "2" correspondingly.

PPCGV asks for two input data and prepares corresponding actions depending on the (in)homogeneity of the problem to be solved. The first input is the question whether the boundary conditions of the problem must be incorporated into the definition of the auxiliary preconditioner (yes or no), and the second input is a non-negative real value rmult (default = 0.) specifically weighting the Jacobi-preconditioning.

PPCGN is the adapter-module for performing the preconditioned conjugate gradient method using the artificially constructed approach as the subroutine **PPCGM** does it conventionally in the case of the original method.

GLOVOR extracts and modifies the main diagonal of the original stiffness matrix K appropriately for the auxiliary preconditioning, where also some previously needed steps are performed.

POINTLOC{T,Q} performes the one-to-one correspondence between the grid points in the original mesh and the points of the auxiliary grid uniquely chosen according to the adopted mapping principle. Finally, this results in setting the vector R^* as well as in defining the auxiliary closure Q^h. Here, step by step, all of the tetrahedral elements (**T**) as well as hexahedral elements (**Q**) belonging to the corresponding unstructured mesh are considered.

DIPOINT3 helps to compute R^* locating the grid point of the original unstructured mesh w.r.t. the corresponding cell-cube of the auxiliary structured hexahedral grid.

EXTRDEF defines the minimum hull-cuboid consisting of cells of Π^h and containing the tetrahedron/hexahedron of the original mesh, where the four/eight vertices of the element are taken into account.

SGESV performes the decision whether a grid point Z of the above hull-cuboid is outside or inside of the tetrahedron under consideration. Provided that x^i are the coordinates of the four vertices of the tetrahedron, we test the tetrahedral representation of $Z = \sum_{i=1}^{4} \lambda_i x^i$, where $\sum_{i=1}^{4} \lambda_i = 1$ checking $\lambda_i \geq 0$, $i = 1,2,3,4$ for being an interior point. The 4×4 systems of linear equations are solved by the implemented LU–decomposition.

CDEF{A,{T,Q}I} computes the auxiliary main diagonal of the auxiliary operator A_{Q^h} (see [4]) which is designed to perform the interior Jacobi-preconditioning implemented between the two hierarchical multiplications provided that we have a potential problem without jumping coefficient functions. Otherwise, in most cases we must use the outer Jacobi–preconditioning merely multiplying with the diagonal matrix derived from the main diagonal of the stiffness matrix K as usually. The subroutine marked by **A** and **{T,Q}I** implements the arithmetical mean approximation (input: rmult $\in [0,2]$ and the weigted distance approximation (input: rmult $\in (2, 2.5)$ in the case of tetrahedral and hexahedral grids, respectively (cf. [4]).

QHCLOSED is called when the closure of the step form approximation Q^h is ordered by the user giving the input rmult = 2.

NEPVOR calculates the square root of the auxiliary main diagonal diag(A_Q) to be specifically inverted according to the chosen method artYs and artBPX, respectively.

SUPPORT markes nodal points in the artificially constructed hierarchical list when the support of the corresponding grid function is specially influenced by the kind of the boundary condition imposed on $\partial\Omega$. When doing so, according to user's input red by **PPCGV** a lot of additional computational effort is required, which hardly results in improving the convergence speed.

HB2BPXGG computes the hierarchical BPX–list from the artificially constructed hierarchical Yserentant-list belonging to the hierarchical grid Π^h in the cube.

SETBCINC defines finally the auxiliary preconditioner $C^{-1}_{\Pi^h, bc(\Gamma^h)}$ in the case of considering the boundary conditions calling the subroutine **SUPPORT**.

CWEIGHT weights the interior Jacobi-preconditioning specifically according to the given input rmult. In most cases the default rmult = 0 defining no weighting is recommendable.

NEPLOES solves the preconditioning system, i.e. $\underline{w} := R[C^{-1}_{\Pi^h}]R^* \underline{r}$ distinctly applying the artYs-method and the artBPX-method, respectively.

GLOCOM performes the communication step w.r.t. the correction values belonging to the coupling nodes. In the parallel version of the method the routine is called before applying the hierarchical multiplication using the long vector $\underline{v}$ as well as afterwards.

RTRANSFER performes the mapping $\underline{v} := R^* \underline{r}$.

RREVERS performes the revers mapping $\underline{w} := R\underline{v_p}$, where the long vector $\underline{v_p}$ already contains the solution of the auxiliary hierarchical preconditioning.

4 Numerical results

The two subsections present the numerical tests computing potential problems sequentially on a large HP workstation, and, in parallel using the GCPowerPlus multiprocessor computer, respectively.
The tables contain the results for the cg–algorithm preconditioned by the "artYs"–method as well as by the "artBPX"–method computing the itemized test example. Chosing the option "1" for all of the examples the used auxiliary hierarchical grid is consisted of hexahedra. The subcolumn marked by "struct. grid" means that we perform computations using a coarse structured initial grid successively refined canonically as the level depth J increases but embedded in the corresponding auxiliary grid Π^h consisting of L^3 points.[1] For comparison the subcolumn marked by "unstr. grid" contains the results belonging to really unstructured grids generated by the mesh generator given in [3] having (nearly) the same number N of degrees of freedom. Here, both the number of cg–iterations and the corresponding CPU–time (in sec) are given which were needed to get the relative error of the cg–iteration less than the previously defined accuracy $\epsilon = 10^{-6}$.[2] The relative error was measured in the $KC^{-1}K$-norm. In the first column indicating the depth J sometimes two numbers divided by the symbol "/" are given which differ from each other. Then, the first number belongs to the auxiliary grid depth due to the canonical refinement of the structured initial mesh and the second one is the depth of the auxiliary grid having some inhomogeneities causing the different J by means of the computation of the tetrahedron heights. Naturally, here we have also the corresponding other number of L^3 given below. The percentages of the CPU–time which are needed for performing the operations indicated by R^* and R do not exceed 12% in each case such that the corresponding part for the preconditioning $C^{-1}_{\Pi^h}$ within the cg–iteration including the amount of the cg–iteration itself is the main one being of near 80%, see also [4]. The percentages are measured on an average w.r.t. the given depths J of the auxiliary grids. Taking this percentages into account we finally discover that the artificially constructed hierarchical methods using only the nodal coordinates and the element connexion need the numerical effort which is approximately 1.6 times more than the effort of the original hierarchical approach having a lot of additional mesh data information to be input. Therefore the application of our new methods is a good practice, especially, for the industrial engineering.

4.1 Sequential Computing

The results are computed by means of the HP 9000/889 K460-workstation using large memory size (1GigaByte) and on an average 7MFlop performance. The executable programs are called "tet.HPPA" in the case of tetrahedral meshes and "quad.HPPA" in the case of using the hexahedral discretization, respectively. Having UNIX for making the programs available the user has to specify the "makefile"-operation by the options ggtet and ggquad, respectively. The information about the background of the underlying software package including tools of the pre- and postprocessing is contained e.g. in [1, 2].

[1]In every table changed, in the columns marked by "struct. grid", using scriptsize the added brackets include the iteration number and the corresponding CPU-time for the real structured hierarchical methods.

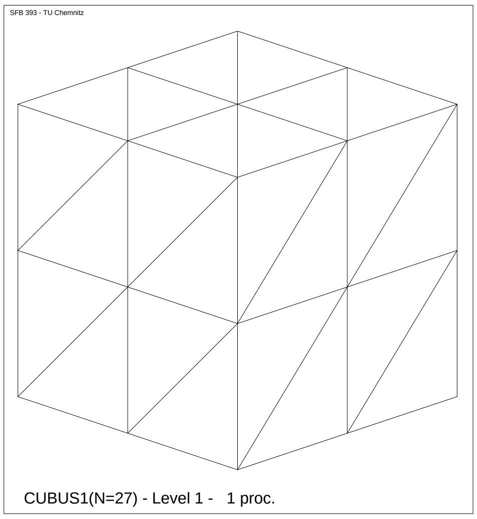
[2]In the given CPU-time neither the times for computing the hierarchical lists of the auxiliary grid Π^h and the step form approximation Q^h inside nor the time for considering the support of the corresponding grid functions w.r.t. the boundary conditions on Γ^h are incorporated. In practice this hidden amount does enlarge the real CPU-time substantially.

1. *Laplace equation in the cube:*

$$-\Delta u = 0 \quad \text{in} \quad \Omega = (0,10)\times(0,10)\times(0,10)$$

$$u = \begin{cases} 0\,, & \text{on } \Gamma_{01} = \{x = (x_1,x_2,x_3)^T : x_3 = 0\} \\ 1\,, & \text{on } \Gamma_{02} = \{x : x_3 = 10\}\ , \end{cases}$$

$$\text{where} \quad \Gamma_0 = \Gamma_{01} \cup \Gamma_{02}\ ; \quad \text{and} \qquad \partial u/\partial N = 0 \quad \text{on} \quad \Gamma_1 = \partial\Omega \backslash \Gamma_0\ .$$



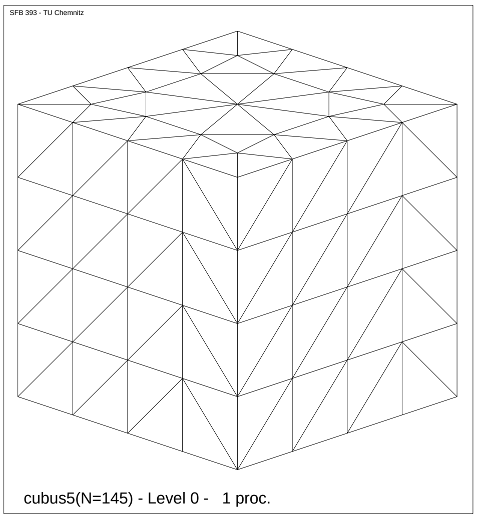


Figure 2: structured and unstructured terahedral grid in the cube

Tetrahedral grids :

			artYs		artBPX	
J	N	L^3	struct. grid	unstr. grid	struct. grid	unstr. grid
1/2	27/ 39	27	[10 (0.00)] 10 (0.00)	13 (0.01)	[10 (0.00)] 10 (0.00)	11 (0.02)
2/3	125/ 145	125	[27 (0.01)] 23 (0.01)	32 (0.10)	[19 (0.01)] 16 (0.01)	17 (0.06)
3/4	729/ 783	729	[48 (0.13)] 33 (0.14)	47 (1.21)	[24 (0.07)] 16 (0.07)	23 (0.54)
4/5	4913/ 5321	4913	[75 (1.30)] 46 (1.42)	64 (11.83)	[27 (0.51)] 15 (0.88)	26 (5.10)
5/6	35937/ 39105	35937	[106 (18.15)] 61 (22.10)	87 (141.04)	[28 (5.10)] 15 (4.22)	27 (43.12)
6/7	274625/ 299585	274625	[144 (218.93)] 77 (181.70)	116 (1396.3)	[29 (46.46)] 14 (34.38)	27 (352.20)

Hexahedral grids:

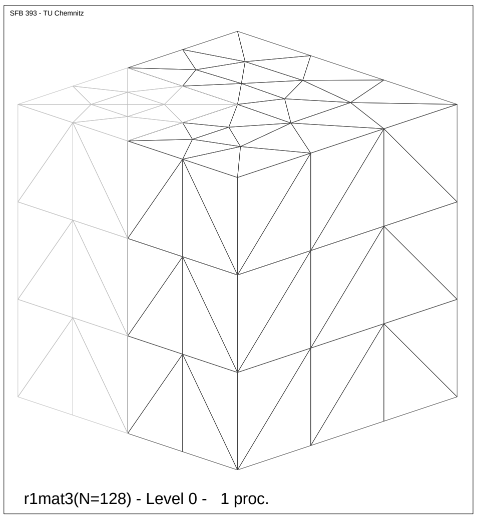
			artYs		artBPX	
J	N	L^3	struct. grid	unstr. grid	struct. grid	unstr. grid
2	27	125	[2 (0.00)] 5 (0.00)		[2 (0.00)] 5 (0.01)	
3	125	729	[2 (0.00)] 25 (0.08)	no mesh	[2 (0.00)] 13 (0.05)	no mesh
4	729	4913	[2 (0.00)] 39 (0.82)	generator	[2 (0.02)] 17 (0.43)	generator
5	4913	35937	[2 (0.05)] 49 (8.75)	available	[2 (0.07)] 19 (3.56)	available
6	35937	274625	[3 (0.73)] 63 (105.24)		[3 (0.77)] 19 (54.11)	
7	274625	2146689	[4 (8.42)] 77 (1205.8)		[4 (8.58)] 19 (255.05)	

2. *Material depending potential problem in the cube:*

$$-\mathrm{div}(a(x)\mathrm{grad}(u)) = 0 \quad \text{in} \quad \Omega = (0,2)^3\,, \quad \text{where} \;\; a(x) = \begin{array}{ll} 10^6 & \text{in} \quad (0,1)\times(0,1)\times(0,2) \\ 1 & \text{in} \quad \Omega \setminus (0,1)^3\,, \end{array}$$

$$u = \begin{cases} 0\,, & \text{on } \Gamma_{01} = \{x = (x_1,x_2,x_3)^T : 0 < x_1 < 1\,,\ x_2 = 0\,,\ 0 < x_3 < 1\} \\ 1\,, & \text{on } \Gamma_{02} = \{x : (x_1,x_2,x_3)^T \in (0,2)\times\{0\}\times(0,2) \setminus \Gamma_{01}\}\,, \quad \text{and} \end{cases}$$

$$\partial u/\partial N = 0 \quad \text{on} \quad \Gamma_1 = \partial\Omega \setminus \Gamma_0\,, \quad \text{where} \quad \Gamma_0 = \Gamma_{01} \cup \Gamma_{02}\,.$$



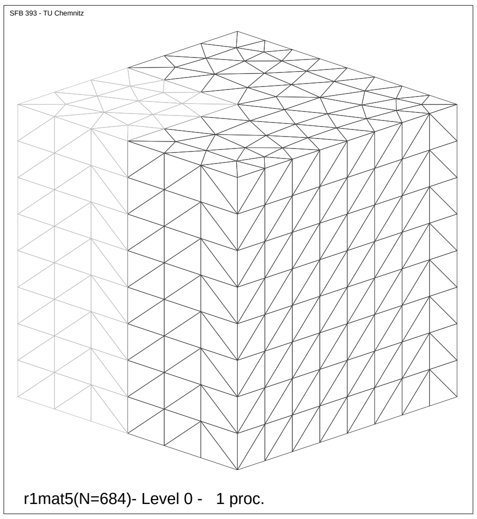


Figure 3: subsequence of unstructured tetrahedral grids in the material cube

Tetrahedral grids :

			artYs		artBPX	
J	N	L^3	struct. grid	unstr. grid	struct. grid	unstr. grid
1/3	27/ 39	27	[11 (0.00)] 16 (0.00)	20 (0.07)	[11 (0.00)] 12 (0.00)	17 (0.06)
2/4	125/128	125	[25 (0.02)] 27 (0.02)	29 (0.58)	[19 (0.02)] 21 (0.02)	27 (0.65)
3/4	729/684	729	[50 (0.16)] 42 (0.20)	40 (0.88)	[25 (0.09)] 25 (0.13)	32 (0.73)
4/5	4913/ 5280	4913	[76 (1.56)] 73 (2.64)	71 (13.10)	[26 (0.59)] 27 (1.00)	37 (7.79)
5/6	35937/38843	35937	[109 (21.76)] 138 (39.73)	144 (267.92)	[28 (5.43)] 29 (17.21)	44 (98.27)
6/7	274625/297741	274625	[144 (245.35)] 143 (607.88)	208 (3216.5)	[30 (63.92)] 32 (89.10)	50 (637.73)

Hexahedral grids:

			artYs		artBPX	
J	N	L^3	struct. grid	unstr. grid	struct. grid	unstr. grid
2	27	125	[7 (0.00)] 13 (0.02)		[7 (0.00)] 8 (0.00)	
3	125	729	[18 (0.02)] 29 (0.10)	no mesh	[10 (0.05)] 17 (0.09)	no mesh
4	729	4913	[30 (0.13)] 47 (1.08)	generator	[13 (0.05)] 22 (0.57)	generator
5	4913	35937	[41 (1.23)] 83 (16.59)	available	[14 (0.48)] 25 (6.63)	available
6	35937	274625	[59 (28.91)] 123 (232.07)		[14 (6.48)] 30 (49.17)	
7	274625	2146689	[83 (197.20)] 165 (2917.0)		[15 (32.35)] 33 (411.52)	

3. *Poisson equation in the FEM-domain resulting in the variational problem:*

Find $u \in V_0$ such that

$$\int\limits_{\Omega} \nabla^T v(x)\, \nabla u(x)\, dx = \int\limits_{\Omega} (-6x_3)\, v(x)\, dx \quad \forall\, v \in V_0$$

holds with $V_0 = \{u \in H^1(\Omega)\ :\ u = 0 \text{ on } \Gamma_0\}$.

$\Gamma_0 = \{x = (x_1, x_2, x_3)\ :\ x_3 = 0\}$ (reverse basis)

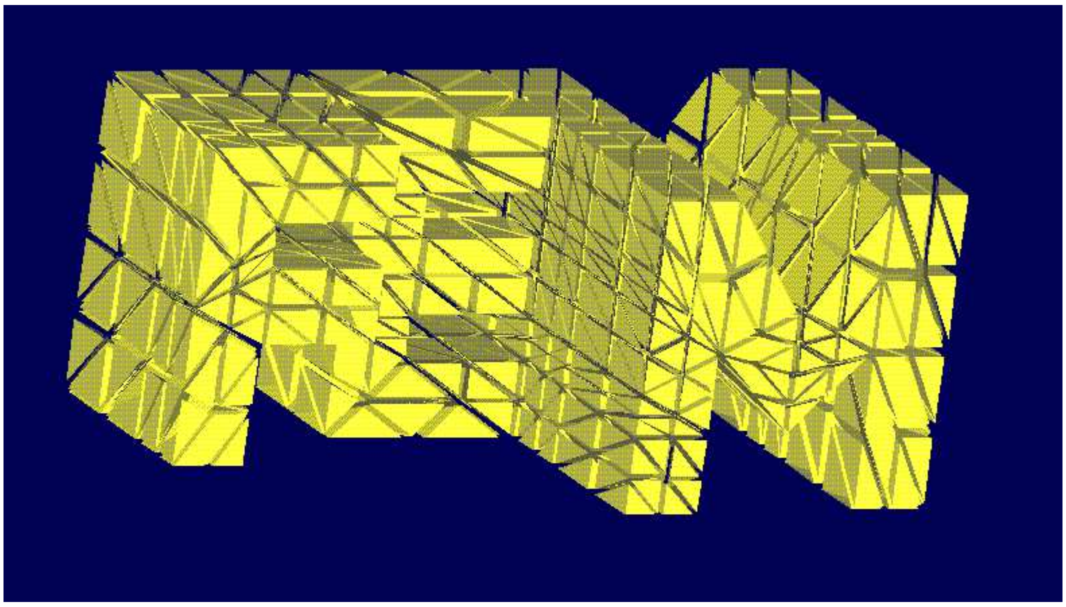


Figure 4: struct. tetr. grid (N=122) in the FEM-domain tends to lose the quasiuniformity

Tetrahedral grids:

			artYs		artBPX	
J	N	L^3	struct. grid	unstr. grid	struct. grid	unstr. grid
4	122	4913	[37 (0.01)] 54 (1.19)	–	[37 (0.01)] 34 (0.98)	–
5	631/648	35937	[64 (0.13)] 84 (15.72)	72 (11.79)	[49 (0.12)] 58 (12.16)	50 (9.44)
6	3848/4032	274625	[95 (1.36)] 141 (256.89)	141 (211.25)	[59 (0.94)] 85 (138.86)	71 (180.99)
7	26386/28026	2146689	[143 (18.16)] 198 (3599.4)	252 (3318.5)	[65 (9.00)] 122 (1694.8)	96 (1253.1)

4. *Laplace equation in a specific radiator slab:*

$$-\Delta u = 0 \quad \text{in } \Omega$$
$$u = \tfrac{z}{10} \quad \text{on } \Gamma_0 \quad \text{(both front ends marked by F)}$$
$$\partial u/\partial N = 0 \quad \text{on} \quad \Gamma_1 = \partial\Omega \backslash \Gamma_0 \;.$$

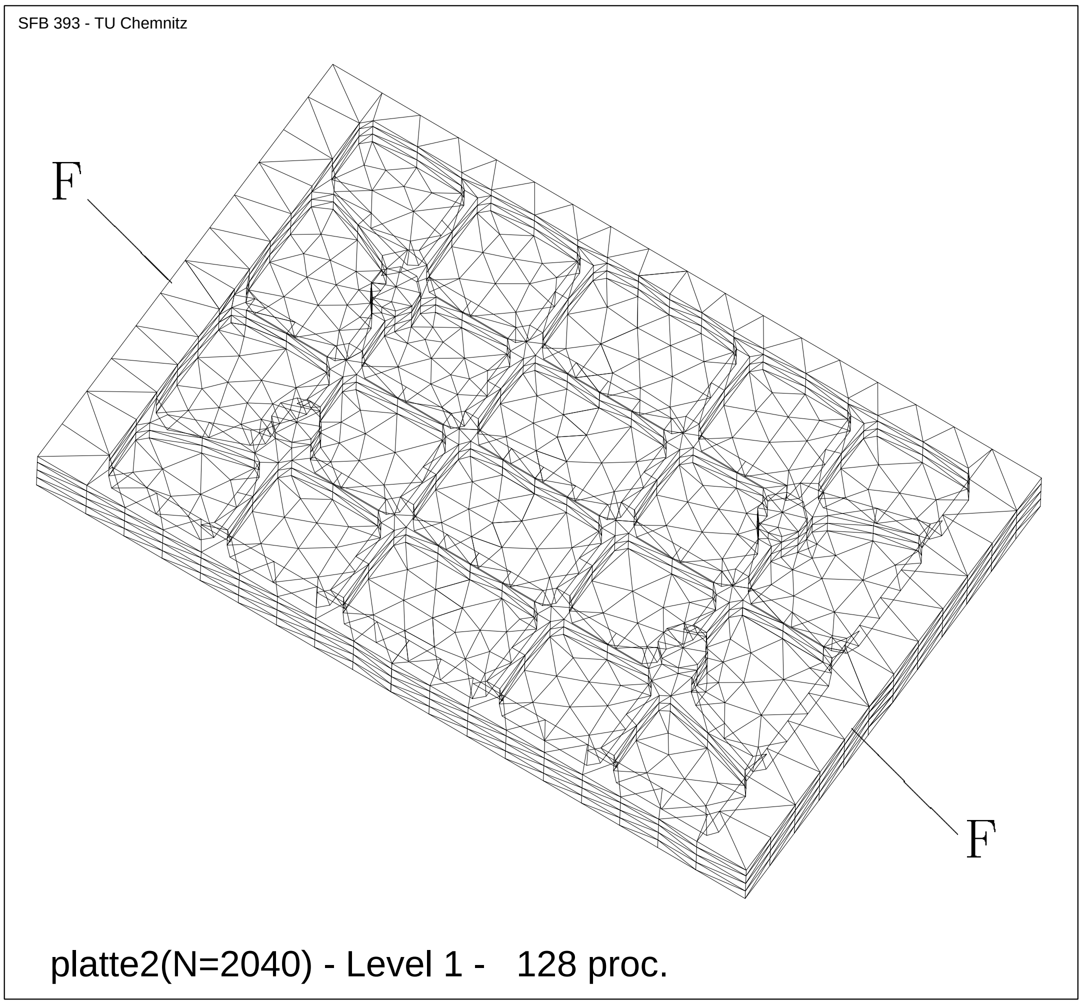


Figure 5: unstructured tetrahedral grid in the slab-domain losing the quasiuniformity

Tetrahedral grids:

			artYs		artBPX	
J	N	L^3	struct. grid	unstr. grid	struct. grid	unstr. grid
6	2040	35937	[166 (0.58)] −−	180 (288.39)	[131 (0.90)] −−	140 (233.55)
7	12391	274625	[214 (12.05)] −−	289 (3221.8)	[193 (22.69)] −−	254 (2874.3)

For comparison here: Parallel computing (data distr.) using 128 proc.:

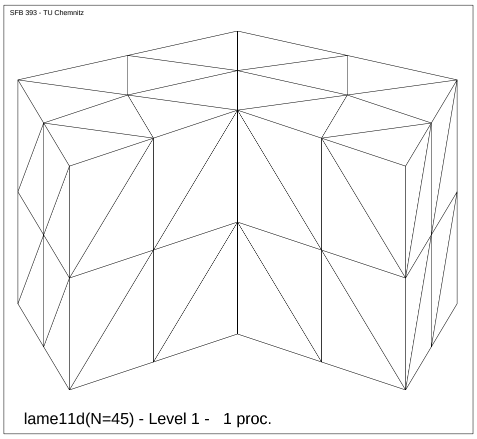
			artYs		artBPX	
J	N	L^3	struct. grid	unstr. grid	struct. grid	unstr. grid
6	2040	35937	[176 (36.58)] −−	230 (168.65)	[136 (31.76)] −−	171 (64.09)

5. *Problem of linear elasticity in the edge–block domain:*

$$-E[\tfrac{1}{2+2\nu}\Delta\vec{u} - \tfrac{2(1+\nu)}{1-2\nu}\mathrm{grad}(\mathrm{div}\ \ \vec{u})] = 0 \qquad \text{in } \ \Omega\,,$$

$$\text{where} \quad E = 200000 \qquad \text{and} \qquad \nu = 0.3$$

$$\vec{u} = (\tfrac{z}{10}, \tfrac{z}{10}, \tfrac{z}{10})^T \qquad \text{on } \ \partial\Omega$$



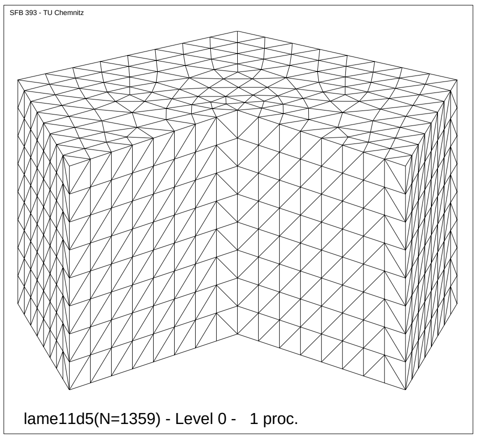


Figure 6: structured and unstructured tetrahedral grid in the edge–block domain

Tetrahedral grids:

			artYs		artBPX	
J	$3*N$	$3*L^3$	struct. grid	unstr. grid	struct. grid	unstr. grid
3	135	3*729	[9 (0.00)] 10 (0.06)	8 (0.06)	[9 (0.01)] 9(0.06)	8 (0.06)
4	675	3*4913	[28 (0.07)] 33 (1.77)	32 (1.55)	[21 (0.06)] 19 (1.04)	19 (1.10)
5	4131/4077	3*35937	[55 (1.11)] 51 (20.75)	42 (22.27)	[31 (0.67)] 25 (20.97)	25 (20.29)
6	28611/28203	3* 274625	[84 (26.59)] 83 (458.40)	59 (208.26)	[38 (7.50)] 30 (114.83)	29 (101.84)
7	212355/209187	3*2146689	[124 (185.31)] 134 (4090.4)	80 (2263.8)	[42 (61.39)] 36 (1034.7)	36 (1019.2)

Hexahedral grids:

			artYs		artBPX	
J	$3*N$	$3*L^3$	struct. grid	unstr. grid	struct. grid	unstr. grid
3	135	3*729	[6 (0.00)] 8 (0.05)		[6 (0.01)] 8 (0.06)	
4	675	3*4913	[26 (0.10)] 26 (1.19)	no mesh	[18 (0.07)] 17 (0.81)	no mesh
5	4131	3*35937	[47 (1.28)] 51 (19.37)	generator	[27 (0.77)] 21 (11.02)	generator
6	28611	3* 274625	[73 (17.45)] 60 (189.32)	available	[32 (7.84)] 28 (93.47)	available
7	212355	3*2146689	[103 (225.48)] 115 (2924.3)		[36 (72.87)] 37 (957.38)	

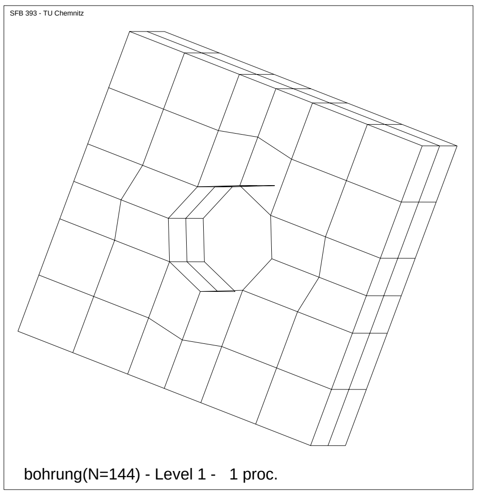
6. *Problem of linear elasticity in the block domain with drill hole:*

$$-E[\tfrac{1}{2+2\nu}\Delta\vec{u} - \tfrac{2(1+\nu)}{1-2\nu}\mathrm{grad}(\mathrm{div}\ \ \vec{u})] = 0 \qquad \text{in } \Omega$$

where $E = 200000$ and $\nu = 0.3$,

$\vec{u} = (0,0,0)^T$ on Γ_0 (basis below); $\dfrac{\partial\vec{u}}{\partial N} = (0,0,-1)^T$ on Γ_{11} (basis above)

$\dfrac{\partial\vec{u}}{\partial N} = (0,0,0)^T$ on $\Gamma_{12} = \Gamma \setminus (\Gamma_0 \cup \Gamma_{11})$ (rest of the surface)



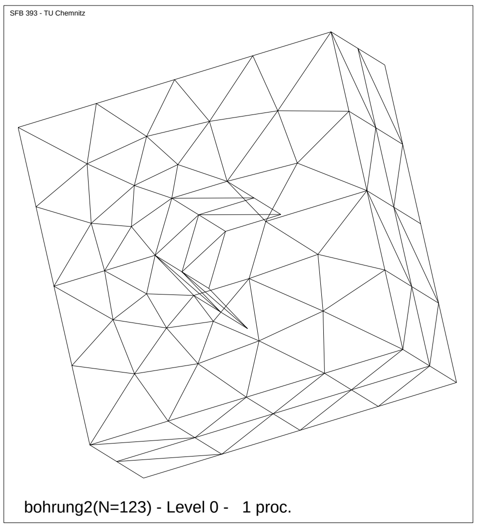


Figure 7: struct. hexahedral and unstruct. tetrahedral grid in the drilled domain

Tetrahedral grids:

			artYs		artBPX	
J	$3*N$	$3*L^3$	struct. grid	unstr. grid	struct. grid	unstr. grid
3	84	3*125	[27 (0.00)] 30 (0.20)		[27 (0.01)] 26 (0.22)	
4	378/369	3*729	[57 (0.09)] 70 (3.87)	66 (3.30)	[44 (0.07)] 51 (3.05)	49 (2.70)
5	2100/2130	3*35937	[97 (0.96)] 120 (58.42)	106 (47.73)	[60 (0.64)] 80 (40.88)	71 (32.92)
6	13608/14148	3*274625	[103 (13.20)] 106 (674.38)	170 (750.02)	[82 (12.51)] 119 (463.89)	87 (414.30)
8	97104/102408	3*2146689	[277 (210.39)] mem.ex.	mem. ex.	[118 (92.82)] mem.ex.	mem. ex.

Hexahedral grids:

			artYs		artBPX	
J	$3*N$	$3*L^3$	struct. grid	unstr. grid	struct. grid	unstr. grid
2	96	3*125	[8 (0.00)] 20 (0.03)		[8 (0.00)] 18 (0.04)	
3	432	3*729	[33 (0.07)] 49 (0.40)	no mesh	[22 (0.05)] 30 (0.25)	no mesh
5	2400	3*35937	[63 (0.85)] 93 (39.21)	generator	[31 (0.45)] 44 (18.25)	generator
6	15552	3*274625	[103 (13.20)] 152 (487.10)	available	[38 (5.03)] 55 (183.26)	available
7	110976	3*2146689	[155 (203.20)] 236 (7051.3)		[43 (87.16)] 62 (3314.2)	

4.2 First results of the Parallel Computing

As already discussed in [4] for the 2D–case we get a parallelizable preconditioner performing one communication before applying the hierarchical multiplication $\mathbf{Q}^T$ and one communication after $\mathbf{Q}$ was completed. Thus, e.g. for the Yserentant hierarchical preconditioning we have

$$\underline{w}_s = \| \; C_s^{-1} \; \| \, \underline{r}_s \; := \; \sum_{s=1}^{p} H_s \, [R_s \; C_{\Pi_s^h}^{-1} \; R_s^* (\sum_{s=1}^{p} \; H_s^T \underline{r}_s)] \; ,$$

where the accumulation matrices H_s symbolically handle the communication w.r.t. the residual vectors $\underline{r}_s$, $s = 1, \ldots , p$, distributed to p processors having L_s^3 components there. To get the results of the subsection we used the well known Parsytec parallel computer GCPowerPlus having 32MByte memory at each processor node and a peak performance of 80MFlop. The executable programs are called "tet.ppc" having tetrahedral meshes and "quad.ppc" when hexahedral grids are used. For installing the programs using PARIX the makefile must have the options ggtet and ggquad, respectively. For more details describing the related software tools see also [1, 2, 6]. For the parallelization the FEM–mesh–data are distributed to the 8 processors used for computing the next examples. The parameters J and L^3 in the following tables are the maxima $J = \max_{s=1,8}(J_s)$ and $L^3 = \max_{s=1,8}(L_s^3)$, respectively.

7. *Laplace equation in the cube (problem no. 1) using 8 processors:*

Tetrahedral grids :

			artYs		artBPX	
J	N	L^3	struct. grid	unstr. grid	struct. grid	unstr. grid
1/2	27/ 39	27	[10 (0.10)] 10 (0.14)	14 (0.19)	[10 (0.12)] 10 (0.17)	14 (0.19)
2/3	125/ 145	125	[27 (0.26)] 32 (0.53)	35 (0.75)	[19 (0.27)] 24 (0.36)	27 (0.56)
3/4	729/ 783	729	[48 (0.53)] 49 (1.01)	56 (4.50)	[24 (0.45)] 37 (0.49)	36 (2.81)
4/5	4913/ 5321	4913	[75 (1.73)] 74 (4.90)	72 (23.87)	[27 (0.93)] 61 (3.78)	47 (16.44)
5/–	35937/–	35937	[107 (11.30)] 128 (42.42)	mem. ex.	[28 (3.61)] 114 (49.30)	mem. ex.

Hexahedral grids:

			artYs		artBPX	
J	N	L^3	struct. grid	unstr. grid	struct. grid	unstr. grid
2	125	125	[2 (0.03)] 23 (0.39)	no mesh	[2 (0.07)] 21 (0.40)	no mesh
3	729	729	[2 (0.05)] 45 (1.09)	generator	[2 (0.12)] 36 (0.85)	generator
4	4913	4913	[2 (0.22)] 74 (4.82)	available	[2 (1.96)] 56 (3.43)	available
5	35937	35937	[3 (0.38)] 128 (46.38)		[3 (0.43)] 97 (34.59)	

8. *Material depending problem no. 2 in the cube using 8 processors:*

Tetrahedral grids :

			artYs		artBPX	
J	N	L^3	struct. grid	unstr. grid	struct. grid	unstr. grid
1/3	27/ 39	27	[11 (0.14)] 10 (0.11)	28 (0.48)	[11 (0.00)] 21 (0.28)	26 (0.53)
2/4	125/128	125	[27 (0.34)] 31 (0.59)	78 (4.12)	[19 (0.30)] 31 (0.66)	52 (1.24)
3/4	729/684	729	[50 (0.71)] 51 (1.31)	149 (10.75)	[25 (0.09)] 48 (1.21)	114 (8.44)
4/5	4913/ 5280	4913	[76 (1.78)] 98 (6.38)	348 (122.80)	[26 (0.91)] 86 (5.45)	175 (57.04)
5/–	35937/–	35937	[108 (9.46)] 212 (78.85)	mem. ex.	[28 (3.09)] 184 (64.65)	mem. ex.

Hexahedral grids:

			artYs		artBPX	
J	N	L^3	struct. grid	unstr. grid	struct. grid	unstr. grid
1	27	27	[7 (0.06)] 11 (0.16)		[7 (0.07)] 11 (0.19)	
2	125	125	[18 (0.19)] 30 (0.46)	no mesh	[10 (0.13)] 23 (0.37)	no mesh
3	729	729	[29 (0.40)] 51 (1.17)	generator	[13 (0.25)] 36 (0.94)	generator
4	4913	4913	[40 (0.90)] 91 (5.04)	available	[14 (0.46)] 56 (3.39)	available
5	35937	35937	[59 (6.41)] 188 (67.51)		[14 (1.73)] 109 (39.69)	

9. *Linear elasticity problem no. 4 in the drill hole domain using 8 processors:*

Tetrahedral grids:

			artYs		artBPX	
J	$3 * N$	$3 * L^3$	struct. grid	unstr. grid	struct. grid	unstr. grid
2	84	3*125	[27 (0.25)] 39 (0.58)		[23 (0.25)] 32 (0.50)	
3	378/369	3*729	[57 (0.76)] 79 (2.41)	66 (2.18)	[44 (0.74)] 60 (1.86)	49 (1.68)
5/4	2100/2130	3*4913	[95 (2.35)] 134 (78.36)	121 (12.65)	[60 (0.64)] 90 (57.24)	79 (8.65)
–/5	13608/14148	3*35937	[166 (10.67)] mem.ex.	264 (161.09)	[82 (6.57)] mem.ex.	156 (102.88)

Using the parallelization of the method the subhull–cubes may have a specific grid size parameter $\bar{h}_s\,, s = 1, \ldots, 8$. The corresponding input of factors can be made especially when this example is computed.

Hexahedral grids:

			artYs		artBPX	
J	$3 * N$	$3 * L^3$	struct. grid	unstr. grid	struct. grid	unstr. grid
1	96	3*27	[8 (0.08)] 8 (0.11)		[8 (0.10)] 8 (0.12)	
2	432	3*125	[33 (0.49)] 51 (1.08)	no mesh	[22 (0.41)] 44 (0.93)	no mesh
3	2400	3*729	[63 (1.48)] 97 (10.11)	generator	[31 (0.99)] 68 (7.03)	generator
4	15552	3*4913	[104 (9.20)] 184 (116.49)	available	[38 (4.02)] 115 (78.48)	available
5	110976	3*35937	[155 (87.49)] 264 (164.61)		[43 (24.89)] mem.ex.	

Acknowledgement: Many thanks to Dr. Sergej V. Nepomnyaschikh of the Computing Centre, Siberian Branch of the Russian Academy of Science for his helpful comments concerning the presented work.

References

[1] T. Apel, SPC-PM Po 3D — User's manual. Preprint SPC 95_33, TU Chemnitz-Zwickau, December 1995.

[2] T. Apel, F. Milde, and M. Theß, SPC-PM Po 3D — Programmer's manual. Preprint SPC 95_34, TU Chemnitz-Zwickau, December 1995.

[3] G. Globisch, On an automatically parallel generation techniques for tetrahedral meshes. Parallel Computing, **21**, no. 3 (1995), pp. 1979–1995.

[4] G. Globisch and S.V. Nepomnyaschikh, The hierarchical preconditioning having unstructured grids. Preprint SFB393/97_11, TU Chemnitz-Zwickau, April 1997.

[5] G. Haase, U. Langer and A. Meyer, Parallelisierung und Vorkonditionierung des CG–Verfahrens durch Gebietszerlegung. in: G. Bader, R. Rannacher, G. Wittum (eds.), Numerische Algorithmen auf Transputer–Systemen, Teubner-Skripten zur Numerik, Teubner–Verlag, Stuttgart 1992.

[6] G. Haase, T. Hommel, A. Meyer and M. Pester, Bibliotheken zur Entwicklung paralleler Algorithmen. Preprint SPC 95_20, TU Chemnitz-Zwickau, June 1995.

[7] S.V. Nepomnyaschikh, Fictitious space method on unstructured meshes. East–West J. Numer. Math., **3** (1995), no. 1, pp. 71–79.

[8] S.V. Nepomnyaschikh, Preconditioning operators on unstructured grids. Preprint No. 178, Weierstraß–Institut für Angewandte Analysis und Stochastik, ISSN 0946–8633, Berlin 1995.

[9] S.V. Nepomnyaschikh, Preconditioning operators on unstructured grids. In N.D. Melson, T.A. Manteuffel, S.F. McCormick, and C.C. Douglas, editors. *Seventh Copper Mountain Conference on Multigrid Methods*, number 3339 in NASA Conference Publication, pages 607–621, 1996. (also available as Preprint 178, Weierstraß–Institut, Berlin, 1995).