Technische Universität Chemnitz-Zwickau

Sonderforschungsbereich 393

Numerische Simulation auf massiv parallelen Rechnern

Gerhard Globisch*

The hierarchical preconditioning on locally refined unstructured grids

Preprint SFB393/98_30

Abstract

We present the implementation of two hierarchically preconditioned methods for the fast solution of mesh equations that approximate 2D-elliptic boundary value problems on unstructured quasi uniform triangulations constisting of N nodes and having locally refined regions. Based on the fictitious space approach the original problem can be adaptively embedded into an auxiliary one in which hanging nodes occur, where the hierarchical grid information and the preconditioner are well defined. We provide the artificial Yserentant preconditioned conjugate gradient method as well as the BPX–preconditioned cg–iteration having optimal computational costs. However, for the previous step that maps the unstructured locally refined grid appropriately onto the rectangular hierarchy using an octal tree numerical operations of order $O(\log(N)N)$ are necessary. Several numerical examples demonstrate the efficiency of the artificially constructed hierarchical methods which can be of importance in industrial engineering, where often only the nodal coordinates and the element connectivity of the underlying (fine) discretization are available.

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

AMS(MOS) subject classification: 65N55, 65N22, 65N30, 78A30

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

Preprint-Reihe des Chemnitzer SFB 393

SFB393/98_30 **October 1998**

Contents

Authors' address:

Gerhard Globisch
Faculty for Mathematics
University of Technology 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 twodimensional boundary value problem determined by the selfadjoint differential operator $\mathcal{L}$ in the domain Ω, where Dirichlet as well as Neumann boundary conditions may be imposed on its boundary $\partial\Omega$.

$$\begin{aligned} \mathcal{L}\, u &= f \qquad \text{in} \quad \Omega \subset \mathbb{R}^2 \\ l\, u &= g \qquad \text{on} \quad \partial\Omega = \Gamma \end{aligned} \tag{1}$$

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 modern methods preconditioning hierarchically, although, in practice, we have an unstructured mesh discretization of Ω available only. For real-life problems (1), adaptively refined and a-priori graded meshes are usually used. Continuing the previous work done in [11, 12] for the artificial preconditioning using the fully hierarchical scheme of depth J meshing the fictitious square, in the next section we adaptively construct the structured auxiliary problem into which the original one can be embedded. Hence, in this case the mapping technique described in [11] is modified, successively refining the auxiliary hull-square by only chosing grid cells which contain more than one node of the unstructured grid. Finally, hanging nodes will occur in the auxiliary grid. However, the method reduces the number of auxiliary unknowns substantially, and less storage is required. For handling with hanging nodes, the corresponding approach is given e.g. in [13, 24]. Moreover, based on the results given in [12] for the artificially full scheme the adaptive method can be easily transfered to real life 3D–simulations, where we consider a hierarchically 26-tree instead of an octal tree performing the mapping. Having the 3D-complexity of the problem (1) this appropriate passage is a must to diminish the storage memory as well as the larger number of the artificial unknowns.

The fictitious space lemma is applied in section 2 to derive the spectral equivalence inequality describing the preconditioning property of the artificially constructed hierarchical preconditioner which belongs to the auxiliary grid points, see also [23, 24]. The convergence rate of the iterative process was proved to be as fast as for the conventionally hierarchical solution method, i.e., it is (nearly) independent of the mesh size.

We discuss several aspects of the numerical implementation of the method in section 3 preparing section 4. Furthermore, we present how the program runs getting control by the UNIX-commando tool. In section 4 we give a short survey of the produced software involving the available numerical toolkit. Furthermore, we describe essential subroutines incorporated into a box diagram.

In section 5 we illustrate the efficiency of the two adaptive algorithms computing several 2D–potential problems. Most of them were already used in [11, 12] for being test examples. Therefore, we recommend to read the paper being faced with the previous numerical results given therein. We evaluate the corresponding hierarchically preconditioned cg–iteration computing test series based on two kinds of discretizations. The first type utilizes unstructured fine grids generated by the parallel advancing front mesh generator in [10] which are embedded in the artificially constructed full scheme consisting of the rectangular hierarchy. For comparison, the examples given in [11] were computed using the artificially constructed triangular hierarchy that seems to provide worser results than the corresponding quadrilateral one. The second type includes the main contents, i.e. the results of the hanging-node-approach are opposed adaptively performing the mapping of the same unstructured grid. Now, the given CPU-times which are necessary to perfom the iterative solution have the same order as those of the conventional hierarchical methods, where the time for constructing the mapping appropriately is hidden. If the depth J of the artificial hierarchy is greater than 10 this amount can not be neglected. The user has to wait up to several hours while the mapping process runs. Moreover, our method is compared with the cg-black-box solver which uses the modified incomplete Cholesky factorization preconditioner (MIC), see [28]. In this case the effect of the preconditioning depends substantially on the numbering of the nodes in the unstructured mesh. We also kept out of sight the time for reducing bandwith and profile of the stiffness matrix K by the nodal renumbering method described in [9]. We only mention that the corresponding efforts considerably exceed those for preparing the adaptively artificial method.

Numerical results of the parallel implementation are also given, although the numerical analysis is still under consideration. The iteration numbers are satisfactory although the comparison with the parallelized methods on structured grids is rather optimistic. The basis for implementing the parallel solvers is a non-overlapping domain decomposition data structure (see e.g. [15, 19]) which is well-suited to parallel machines with MIMD–architecture. Section 5 also indicates the practical importance of our method. Often, in industrial engineering boundary value problems have to be solved, where a fine mesh in the domain and the discretization concept are given, see e.g. [27]. However, no fast hierarchical solver can be applied as nothing is known about the grid structure. Using our approach this bottleneck is overcome.

2 The technique for locally refined grids

In this section we consider a triangulation Ω^h of the domain Ω

$$\Omega^h = \bigcup_{i=1}^{M} \tau_i$$

and assume Ω^h is regular but not quasi-uniform, i.e. there exists a constant s, independent of h, such that

$$\frac{r_i}{\rho_i} \leq s, \; i = 1, \ldots, M$$

where r_i and ρ_i are the radii of the circumscribed and the inscribed circles of the triangle τ_i, respectively. It means that the mesh Ω^h can be locally refined. For this triangulation Ω^h, we define the space $H_h(\Omega^h)$ of real continuous functions which are linear on each triangle τ_i of Ω^h. For the sake of simplicity, we only consider the Dirichlet boundary condition and assume that the functions from $H_h(\Omega^h)$ vanish at Γ^h.

If we introduce an uniform fictitious grid Q^h, then it is possible to modify the operators R and T which were defined in [11] in the case of the full hierarchical scheme. Now they must be appropriated to the locally refined triangulation Ω^h. However, the numerical implementation of the corresponding preconditioner will be more expensive.

Let us embed the domain Ω in a square Π and start with a coarse uniform grid Π_0^h. We refine Π_0^h several times

$$\Pi_0^h, \; \Pi_1^h, \ldots$$

The grid Π_l^h consists of cells $D_{ij}^{(l)}$. Let Q_0^h denote the minimum figure that consists of cells $D_{ij}^{(o)}$ and contains Ω^h. Denote by I_0 a set of indices (i,j) such that

$$Q_0^h = \bigcup_{(i,j)\in I_0} D_{ij}^{(0)}$$

We define grids $Q_1^h, Q_2^h, \ldots$ in the following way. Denote by I_l a set of indices (i,j) such that the cell $D_{ij}^{(l)}$ contains more then one vertex of the triangulation Ω^h. We devide $D_{ij}^{(l)}$ and all neighbouring cells (which have at least one node with the cell $D_{ij}^{(l)}$ in common) into four congruent subcells by connecting the midpoints of the edges. We denote the new cells by $D_{ij}^{(l+1)}$ and the resulting grid by $Q_{l+1}^h, l = 0, 1, \ldots$, which is the minimum figure that contains Ω^h. We stop this process when each cell contains no more than one vertex of Ω^h specifying the final grid by $Q_J^h = \tilde{Q}^h$. The one to one correspondence between the locally refined unstructured grid and the hierarchical grid $\Pi_J^h = \tilde{\Pi}^h$ having hanging nodes is outlined in Figure 1.

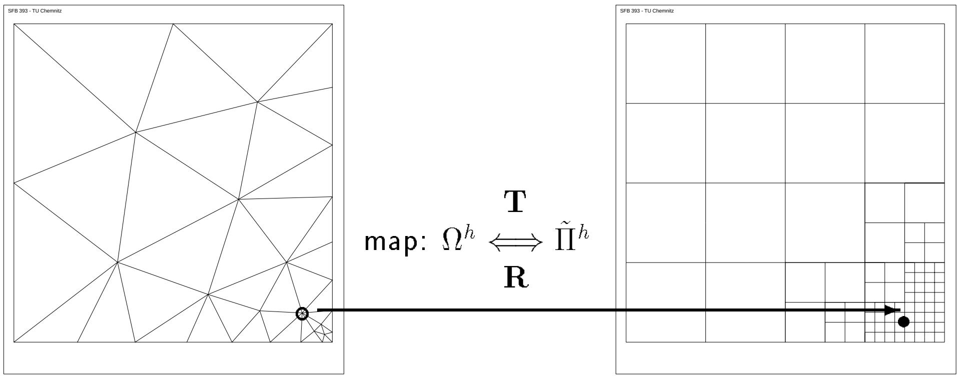


Figure 1: Locally refined grid Ω^h and auxiliary hierarchical grid $\tilde{\Pi}^h$ with hanging nodes

We define a finite element space $H_h(Q_J^h)$ as follows

$$H_h(Q_J^h) = \{ \sum_{\mathrm{supp}\Phi_k^{(0)} \subset Q_J^h} \alpha_k^{(0)} \Phi_k^{(0)} + \sum_{l=0}^{J-1} \sum_{(i,j)\in I_l} \sum_{\mathrm{supp}\ \Phi_k^{(l+1)} \cap D_{ij}^{(l)} \neq \emptyset} \alpha_k^{(l+1)} \Phi_k^{(l+1)} \mid \alpha_k^{(l)} \in \mathbb{R}^1 \} .$$

From [11] we take over the projection operator R and the extension operator T.

$$R : H_h(Q_J^h) \longrightarrow H_h(\Omega^h) , \qquad T : H_h(\Omega^h) \longrightarrow H_h(Q_J^h) .$$

In the following, we define a preconditioning operator in $H_h(Q_J^h)$ for any $U^h \in H_h(Q_J^h)$:

$$C_{\tilde{\Pi}^h}^{-1} U^h = \sum_{\mathrm{supp}\Phi_k^{(0)} \subset Q_J^h} (U^h, \phi_k^{(0)})_{L_2(Q_J^h)} \Phi_k^{(0)} + \sum_{l=0}^{J-1} \sum_{(i,j)\in I_l} \sum_{\mathrm{supp}\ \Phi_k^{(l+1)} \cap D_{ij}^{(l)} \neq \emptyset} (U^h, \phi_k^{(l+1)})_{L_2(Q_J^h)}\ \Phi_k^{(l+1)}$$

Theorem There exist positive constants $c_1$and c_2 which are independent of h such that

$$c_1(K^{-1}u, u) \leq (R C_{\tilde{\Pi}^h}^{-1} R^* u, u) \leq c_2(K^{-1}u, u) \qquad \forall u \in \mathbb{R}^N$$

Proof. We use the equivalence of the H^1-norms of finite element functions in the spaces $H_h(\Omega^h)$ and $H_h(Q_J^h)$ and the difference counterparts of these norms and the multilevel technique, see [22, 24] for more details, especially for the fictitious space lemma.

3 Aspects of the numerical implementation

The main difference between the implementation of the artificially constructed full-hierarchical method (see [11, 12]) and their adaptive version described here is the mapping principle. Up to now, the mesh size parameter $\bar{h}$ was computed in advance defining the depth J of the artificially hierarchical rectangular Π^h. In the 2D-case it is consisted of all of the $L^2 = (2^J+1)^2$ vertices. Considering the nodal coordinate set $X(2,i), i = 1, 2, \ldots, N$ of the unstructured grid Ω^h one time only, for $i = 1, 2, \ldots, N$, the one-to-one correspondence between selected points in Π^h and all of the points in Ω^h was performed. Now, we also fix the depth J in advance. However, for $l = 1, 2, \ldots, J$ we run through the grid cells D^l_{ij} of a transiently available Π^h belonging to the current stage of l really doing as it was given in the previous section. I.e., in the case of more than one point of the unstructured mesh inside of the cell D^l_{ij} we refine D^l_{ij} and their neighbourhood cells using successively the octal tree description of the grid hierarchy. An auxiliary vector IH defined to be of the Integer∗2-type to save the memory having $(2^J+1)^2$ transient entries marks the points of Π^h which are in $\tilde{\Pi}^h$ as follows. In the case of a regular point it is marked by 2 in the case of a hanging node by 1. Otherwise, if the point does not belong to $\tilde{\Pi}^h$ the corresponding entry (i, j) in IH remains equal to zero. Running through the loop $l = 1, 2, \ldots, J$ at each stage the nodal coordinate vector $X(2, i)$ of the unstructured grid must be checked for assigning uniquely these points to the auxiliary vertex in the grid $\tilde{\Pi}^h$, which is, in general, the point in the left-below of the corresponding cell D^l_{ij}. In comparison with the L^2 artificial nodes in the fully hierarchical grid Π^h we get merely $\tilde{L}$ auxiliary unknowns summing up the points in IH which are marked by 2 or 1, respectively. Since we have unstructured grids with locally refined regions, in general we arrive at $\tilde{L} \ll L^2$. Hence, the amount of the artificially performed hierarchical preconditioning using the father-son connexion of $\tilde{L}$ artificial unknowns is nearly as fast as in the case of the classically hierarchical method which can't be applied because nothing is known about the structure of the grid Ω^h. The above mapping process which is performed by the routine **LOCPRE2X** costs $(O(N \log(N))$ numerical operations. Having a relatively large J, say $J > 10$, the assigning is numerically expensive waiting several hours for. To define the Jacobi-preconditioning matrix $\mathbf{J}$ for all of their entries belonging to the $\tilde{L}$ points we use the approximation which was already given in [11].

By means of the following commando-tool version we get insight into the controlling of the program artYs-BPX.{HPPA,ppc,parix}.px, where we also describe numerical techniques that are hidden here.

```
helix% run16 pmhi.parix.px                                                  (0)
run : Creating 4 * 4 descriptor by calling mkdesc.
run : Starting D-Server at helix link 0.
  # ############################################################# #
  #                                                               #
  #  SSSS  PPPPP   CCCC      PPPPP  M       M   PPPPP        222  #
  # S    S P    P C    C     P    P MM     MM   P    P      2   2 #
  # S      P    P C          P    P M M   M M   P    P          2 #
  #   SS   PPPPP  C      === PPPPP  M  M M  M   PPPPP ooo    222  #
  #      S P      C          P      M   M   M   P    o   o  2     #
  # S    S P      C    C     P      M       M   P    o   o  2     #
  #  SSSS  P       CCCC      P      M       M   P     ooo   22222 #
  #                                                               #
  # ############################################################# #
  #                                                               #
  #             Programm-Modul 2D-Potentialprobleme              #
  #                         Version:  2.00                        #
  #                                                               #
  #                DFG-Sonderforschungsbereich 393                #
  #             TU Chemnitz, Fakultaet fuer Mathematik            #
  #                                                               #
  #           A.Meyer, Chr.Israel, M.Pester, G.Globisch           #
  #                                                               #
  #    4-MB-Variante ( 580000 Worte) - bis zu  128 Prozessoren    #
  #                       in Benutzung:         16 Prozessor(en)  #
  #                                                               #
  # ############################################################# #
 Display: PseudoColor  ***  14 colors allocated ***
```

```
*****************************************************************
Wollen Sie Globisch-Nepomnyaschikh-vorkonditionieren.?
Falls nicht, so  erhalten  Sie die folgenden Optionen:
1. Klass. hier. Vorkonditionierung nach Yserentant/BPX                    (1)
*****************************************************************
(Geben Sie j/J/y/Y oder n/N ein) ----->j
1: Full triangular  auxiliary grid version?
2: The rectangular auxiliary  grid version?                               (2)
(Bitte geben Sie 1 oder 2 ein) ------->2

*****************************************************************
Typeingabe (full=1, hanging node=2) --->2                                 (3)
Kuenstl. Unbekannte  (Faktor zu N)? --->15                                (4)
artYs/artBPX? (Yserentant=1, BPX=2) --->1                                 (5)

*****************************************************************

--> Auswahl des Elementtyps :

               3 : Dreiecke
               4 : Vierecke
         (ENTER = Dreiecke) :
adsquare.net  d2.net        keule16.net   qu16.net
adtest.net    jinjan.net    kr4.net       x1004.net
Filename for User-Net:adsquare

EINGABE : Anzahl der Levels:1
lineare Finite Elemente ? (j/n)

 * Netz : (Prozessor 0)
 - lokale Anz. der Knoten   :             0
 - lokale Anz. der Kanten   :             0
 - lokale Anz. der Elemente :             0

Ges. lok. Knotenzahl :                0
davon :  glob.Crosspoints :         101
         lok. Crosspoints :           0
         Summe der Randketten :       0
         Koppelknoten :               0
         innere Knoten :              0
 Anz. der Randketten   :              0

grafische Darstellung ? - [j]/n : n

*******************************************************************
ICH =   4, di =       0.215455E+00                                        (6)
ICH =   1, di =       0.387959E-01
ICH =   5, di =       0.588012E+00
ICH =  12, di =       0.481422E-01
ICH =   8, di =       0.158798E+00
ICH =   3, di =       0.244106E+00
ICH =   2, di =       0.277854E+00
ICH =   9, di =       0.167542E+00
ICH =  13, di =       0.390644E+00
ICH =   0, di =       0.000000E+00
ICH =   7, di =       0.609501E-01
ICH =  15, di =       0.157121E+00
ICH =   6, di =       0.350719E-01
ICH =  11, di =       0.714978E-01
ICH =  14, di =       0.143623E+00
ICH =  10, di =       0.527156E-01

Faktor > 0 eingeben ( = 1 ?) --->1                                        (7)
Mehrprozessorarbeit: Wollen Sie viele Ausgaben?                           (8)
(Geben Sie j/J/y/Y oder n/N ein) ----->y
Optimierungsgrad  der Abbildung  festlegen.
(opt<0: auf Teufel komm raus Knoten sparen)                               (9)
(Geben Sie -3,-2,-1,0,1, ... ,5  ein) --->-1
Statt Hilfsquadrat ein Hilfsrechteck gewuenscht?                         (10)
(Geben Sie j/J/y/Y oder n/N ein) ----->n
Proz.:  0 max. Tiefe J =  8 L^2 =           66049                        (11)

*******************************************************************

* Assemblierung der Steifigkeitsmatrix :
 - Anzahl der Nichtnull-Elemente :
      bei A :                 0

Proz.:           0 nline ist Null, bin arbeitslos.

Aufbau der Abbildung; etwas Geduld bitte.
Haeng. Zushg. zeigen?  (Eing. Proz.-Nr.) ==>16                           (12)
maximale Tiefe des kuenstlichen Gitters:           8
Proz.:  4 --> L^2 =          1089, aber L~ =           170               (13)
Proz.: 12 --> L^2 =         66049, aber L~ =           472
Proz.:  8 --> L^2 =          1089, aber L~ =           329
Proz.:  5 --> L^2 =           289, aber L~ =            81
Proz.:  1 --> L^2 =           289, aber L~ =            69
Proz.: 13 --> L^2 =           289, aber L~ =           132
Proz.:  9 --> L^2 =          1089, aber L~ =           216
Proz.:  7 --> L^2 =         66049, aber L~ =           447
Proz.:  6 --> L^2 =          1089, aber L~ =           177
Proz.:  3 --> L^2 =          1089, aber L~ =           177
Proz.: 15 --> L^2 =          1089, aber L~ =           190
Proz.:  2 --> L^2 =          4225, aber L~ =           332
Proz.: 11 --> L^2 =          4225, aber L~ =           341
Proz.: 14 --> L^2 =          1089, aber L~ =           161
Proz.: 10 --> L^2 =          1089, aber L~ =           245
Proz.:  0 --> L^2 =             0, aber L~ =             0
```

```
Aufbau der Stufenapproximation; Bitte etwas Geduld.
Visualisierung (Eingabe Prozessornummer) ==>16

ICH =   0; Sum L^2 =       150127; Sum card(Q^h) =        9011; J =  8   (14)
Zeiten fuer Warten+Kommunikation [s]

Prozessor
log. /phys.    input :    in % :   output:    in % :   gesamt:
   0   0  0      0.00      0.00    229.06     99.95    229.18
   1   1  0      0.03      0.01    228.25     99.59    229.18
   2   3  0      0.02      0.01    205.27     89.57    229.18
   3   2  0     20.08      8.76    205.26     89.57    229.18
   4   0  1      0.02      0.01    225.52     98.40    229.18
   5   1  1      2.69      1.17    225.51     98.40    229.18
   6   3  1      0.02      0.01    225.19     98.26    229.18
   7   2  1      0.02      0.01     23.52     10.26    229.18
   8   0  3      0.02      0.01    223.16     97.37    229.18
   9   1  3      1.49      0.65    223.05     97.33    229.18
  10   3  3      0.02      0.01    224.85     98.11    229.18
  11   2  3      0.03      0.01    208.29     90.89    229.18
  12   0  2      0.02      0.01      0.01      0.01    229.18
  13   1  2    227.96     99.47      0.01      0.00    229.18
  14   3  2    224.59     98.00      0.04      0.02    229.18
  15   2  2    224.70     98.04      0.00      0.00    229.18
reine Arithmetikzeit (max):    229.14                                  (15)

benutzter Speicher:           2 ...        5634 WORTE
freier Speicher:         579998 ...      574366 WORTE
* Probleminformationen (lokal Prozessor 0):
- Anzahl der Knoten     :      0
- davon :  Koppelknoten :      0
-        interne Knoten :      0

* Probleminformationen (  global ):
- Anzahl der Prozessoren:     16
- Anzahl der Knoten     :    361
- davon :  Koppelknoten :    241
-        interne Knoten :    120
->       Gesamtanzahl der Freiheitsgrade :    361

* Start der Simulation
                                <enter>
Geben Sie rmult fuer R^T ein --------->0                               (16)

 IT       (r,w)         (As,s)          ALFA          BETA
  1   1.784472E+04   4.455678E+04  -4.004939E-01   0.000000E+00
  2   6.913020E+03   1.933661E+04  -3.575094E-01   3.873987E-01
  3   3.640377E+03   1.107323E+04  -3.287547E-01   5.265972E-01
  4   1.789120E+03   4.218368E+03  -4.241262E-01   4.914656E-01
  5   8.423368E+02   1.636132E+03  -5.148342E-01   4.708106E-01
  6   4.809581E+02   8.316169E+02  -5.783409E-01   5.709808E-01
  7   3.731538E+02   7.936708E+02  -4.701619E-01   7.758551E-01
  8   2.208358E+02   2.837288E+02  -7.783342E-01   5.918090E-01
  9   1.201522E+02   2.090880E+02  -5.746488E-01   5.440793E-01
 10   1.087616E+02   2.858571E+02  -3.804756E-01   9.051990E-01
 11   4.709540E+01   8.012380E+01  -5.877829E-01   4.330148E-01
 12   2.504550E+01   4.310150E+01  -5.810819E-01   5.318036E-01
 13   1.109942E+01   2.206038E+01  -5.031384E-01   4.431702E-01
 14   4.043865E+00   6.676073E+00  -6.057251E-01   3.643312E-01
 15   2.628543E+00   5.479189E+00  -4.797321E-01   6.500076E-01
 16   1.451768E+00   2.893943E+00  -5.016574E-01   5.523091E-01
 17   7.138102E-01   1.394144E+00  -5.120061E-01   4.916834E-01
 18   3.706404E-01   6.902177E-01  -5.369906E-01   5.192422E-01
 19   1.941290E-01   3.855149E-01  -5.035577E-01   5.237664E-01
 20   1.028034E-01   1.923278E-01  -5.345221E-01   5.295626E-01
 21   5.771207E-02   1.058960E-01  -5.449883E-01   5.613826E-01
 22   2.280583E-02   3.317241E-02  -6.874938E-01   3.951657E-01
 23   1.140175E-02   1.783604E-02  -6.392536E-01   4.999492E-01
 24   5.230066E-03   8.132724E-03  -6.430891E-01   4.587071E-01
 25   2.825409E-03   4.668480E-03  -6.052097E-01   5.402243E-01
 26   1.264276E-03   2.293823E-03  -5.511654E-01   4.474665E-01
 27   5.953633E-04   1.044797E-03  -5.698366E-01   4.709124E-01
 28   2.390452E-04   3.716761E-04  -6.431545E-01   4.015114E-01
 29   9.227201E-05   3.716761E-04  -6.431545E-01   3.860024E-01
* ENDE ,           29 Iterationen
Zeiten fuer Warten+Kommunikation [s]

Prozessor
log. /phys.    input :    in % :   output:    in % :   gesamt:
   0   0  0      1.33     66.93      0.37     18.70      1.99
   1   1  0      0.34     16.96      1.39     69.93      1.98
   2   3  0      0.35     17.52      0.72     36.12      1.98
   3   2  0      0.93     46.90      0.53     26.94      1.98
   4   0  1      1.07     54.11      0.37     18.83      1.98
   5   1  1      0.36     18.26      1.33     67.14      1.98
   6   3  1      0.99     49.88      0.40     20.37      1.98
   7   2  1      0.59     30.02      0.15      7.75      1.98
   8   0  3      0.61     30.59      0.40     20.31      1.98
   9   1  3      0.28     14.09      1.05     53.20      1.98
  10   3  3      0.82     41.30      0.44     22.08      1.98
  11   2  3      0.94     47.35      0.09      4.34      1.98
  12   0  2      0.48     24.13      0.20     10.02      1.98
  13   1  2      0.45     22.70      1.07     54.14      1.98
  14   3  2      0.65     32.65      0.80     40.31      1.98
  15   2  2      1.32     66.53      0.08      4.23      1.98
reine Arithmetikzeit (max):      1.30
grafische Darstellung ? - [j]/n :

******************************************************************
Wollen Sie Globisch-Nepomnyaschikh-vorkonditionieren.?                 (17)
```

```
Falls nicht, so  erhalten  Sie die folgenden Optionen:
1. Klass. hier. Vorkonditionierung nach Yserentant/BPX
*************************************************************
(Geben Sie j/J/y/Y oder n/N ein) ----->n

*************************************************************
Yserentant oder BPX?  (Ys=1, BPX=2) --->1

*************************************************************

EINGABE : Anzahl der Levels:-2
lineare Finite Elemente ? (j/n)

run : Terminating with result = 0.
```

Figure 2: The commando tool while the program runs in parallel

(0) The older parallel parix-version of the program is initialized using 16 processors T800 of the Parsytec-GmbH-cluster.

(1) At this stage the "Globisch-Nepomnyaschikh"-preconditioning for constructing the hierarchical preconditioner artificially w.r.t. a given (unstructured) discretization of the partial differential equation can be ordered. Otherwise, the user gets the opportunity to solve with the cg-algorithm preconditioned by the classical hierarchical methods provided that the mesh hierarchy is given. In the case of having only one processor, i.e. performing the sequential computing, the option for solving the problem applying factorized preconditioned cg-methods like IC or MIC can be selected, see [28]. Here, the previously performed nodal renumbering process costs several hours time provided that a large scale problem is given, see [9, 11].

(2) The user may order the element-type of the artificially constructed hierarchical mesh into which the original one can be embedded. The triangular hierarchy (input=1) and the rectangular hierarchy (input=2) are available. Finally, this results in defining the corresponding hierarchical lists which is performed within the subroutine **LOCPRE3D**. The numerical tests indicate that the rectangular hierarchy almost ever provides better results than the triangular hierarchy does.

(3) At this stage the user defines the type of the artificial preconditioning. The input equals to "1" does fix the full-hierarchical scheme whereas the input equals to "2" orders the hanging node version which is available now. Since the fully artificial scheme requires larger auxiliary memory size than the hanging node approach in this case the user may select a memory-saving branch of the program by giving the input equals to "3".

(4) The user has to specify the integer factor by which the given number N of nodes in the unstructured mesh is multiplied to predict an upper bound for the number $\tilde{L}$ of artificial unknowns in the auxiliary system preconditioning artificially having hanging nodes in the hierarchy. In the case of the parallel computing the intermediate zero-input allows to set the factor individually per processor $i = 1, 2, \ldots, p$. Otherwise, the given positive input is uniformly spread over the processors.

(5) The user may fix the type of the preconditioning. He can order the Yserentant hierarchical preconditiong (input=1) as well as the BPX-preconditioning (input=2). Omitting the stages (2)-(4) this input is also necessary in the case of the hierarchical preconditioning performed conventionally. In general, the following actions of the program are well known, see e.g. [1, 2]. In our example (see Fig. 2) the root processor has got nothing to do.

(6) Here the program informs the user about the parameter d_i belonging to the processor $i = 1, \ldots, p$ which was computed using the heights of the triangles in the unstructured mesh, see [11].

(7) The user may put in a real factor ($f \leq 1/2; f \geq 2$) multiplying the parameter d_i for all i, $i = 1, \ldots, p$, to get a finer and a coarser artificial hierarchy, respectively. The default factor is equal to 1. By giving input equal to zero in the following the user may specify the factor f_i per processor distinctly evaluating the data given at the stage (6), see also [12].

(8) In the case of the parallel computing the user may order several helpful information per processor including error messages during the program runs (input="y"). Otherwise (input="n"), this can be avoided.

(9) The user can "optimize" the adaptive mapping described in the previous section. The larger the given input integer (iopt $= -3, -2, \ldots, 5$) is the "smoother" the hanging node connexion is spreaded over the hierarchy. Negative input tries to save auxiliary unknowns as much as possible. The input at this stage plays together with the factor f as it is specified larger and lower at the stage (7), respectively.

(10) Instead of the hull square the user may order a hull rectangular embedding the original discretization; for more details see also [12].

(11) Here, the maximum depth J of the artificial hierarchies as well as the related number of auxiliary unknowns L^2 are set to be output taking the corresponding values per processor $i = 1, \ldots, p$ into account.

(12) If the input parameter iopt at stage (9) was defined to be greater than -2, the user can order the vizualization of the hanging node pattern (cf. vector IH) related to the processor number $i = 0, 1, \ldots, p$–1 set to be input. If this number is not in the corresponding integer range the program continues.

(13) At this stage the given output compares the full number L^2 with the adaptively defined number $\tilde{L}$ based on the hanging node approach of the mapping.

(14) After the step form approximation was generated (see [11]) the output gives the overlapping parameter Sum card(Q^h) of the corresponding points got by summing up the values related to each of the processors. If the full scheme is used for the hierarchical preconditioning, at this stage the user is asked for the visualization of the step form approximation per processor. The number for the corresponding processor $i = 0, 1, \ldots, p$–1 is required to be input. If the input is equal to the number p of processors the program continues without showing the 0–1 pattern.

(15) If the user has ordered the auxiliary output in the case of the parallel computing (see (8)) here all of the CPU-times are given per processor which were necessary to perform the adaptive mapping. In the case of the artificial BPX-preconditioning between the stage (14) and the next stage (15) an information on the erection of the BPX-list from the Yserentant-list will be given. Provided that the depth J is less than 10 the corresponding process comes fastly to the end. Otherwise, the user has to wait for patiently. However, the BPX-list for the full rectangular hierarchy ($J = 7, \ldots, 12$) can be ordered being an input-file after the artBPX-preconditioning is choosen at stage (5). These compressed files are available in the AFS-directory gglobisc/svnep/listen.

(16) At this stage the user provides a non-negative real input parameter **rmult** for weighting the mapping vector R^T (see [12]) specifically. The default is set to be zero, i.e. the factor is equal to one.

(17) In the following the program can be restarted using another input file (Level number input = -1) as well as it may be stoped (Level number input = -2), see also [1, 2].

The numerical result using 16 processors for the parallel computation of the Laplace equation in the unit square imposing the linear Dirichlet boundary condition ($u = x_1 + x_2 + 1$) on the boundary was shown by the above commando tool picture, see Fig. 2. We used the locally refined mesh discretization of the square which is the data-file *adsquare.net* in the **mesh3**-directory. To get insight into the second refinement of the unstructured grid in Fig. 1 as well as in the filled isoline picture for the solution u we present Fig. 3.

To compare the above result with the sequential computing which is still more effective than the parallel version under theoretical consideration and to brighten the options "artYs" or "artBPX" (see (5)) playing together with the full scheme "F_J" and the hanging node scheme "H_J" both depending on the specified grid depth J (see (3)) as well as with the choice of the element type ("T"=triangle, "Q"=rectangle) in the auxiliary hierarchy (see (2)) now the next survey is provided computing the test example. The time epitomized by "(t_{map})" enlightens the amount for performing the adaptive mapping. Furthermore, there are rows in Table 1 which contain the results of the cg-algorithm preconditioned by the IC, MIC, and the MAF-method described in [28]. Moreover, the presented brackets contain the results of the classically hierarchical iteration which we could apply since the corresponding refinements of the initial mesh (see Fig. 1) were stored. The computations are performed on a HP 9000/889 K460- workstation.

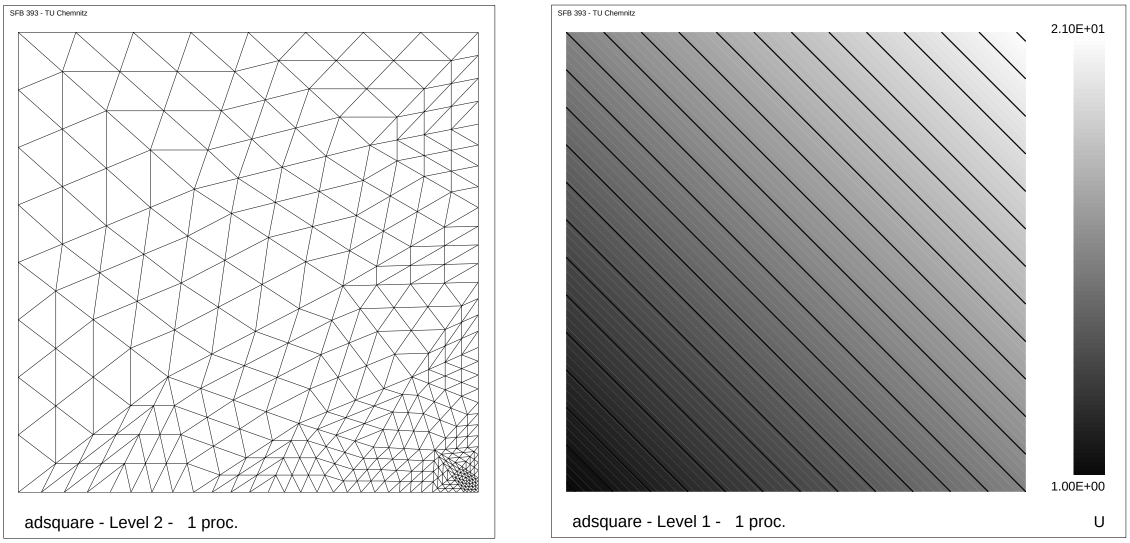


Figure 3: Locally refined unstructured mesh and isolines

Further test problems containing locally refined mesh regions e.g. such as given by the files *f.net*, *ad*1*.net*, *lbenchdi*, $i = 1, 2, \ldots, 9$ illustrate impressively the nice effect of mapping the original mesh adaptively to the embedding grid using hanging nodes. The test examples in section 5 of the paper provide an additional contrast between the superior hanging node approach and the full scheme results which were presented in [11] computing the same.

Table 1: #cg–iterationens, CPU–times (in sec) for the Laplace equation in "adsquare.net"

	N	$L^2/\tilde{L}(t_{\mathrm{map}})$	artYs T	artYs Q	artBPX(rmult $=0.5$) T	artBPX(rmult $=0.5$) Q
			[--]		[--]	
H_7	31	141 (0.73)	8 (0.00)	8 (0.00)	8 (0.01)	8 (0.01)
			[12 (0.01)]		[10 (0.00)]	
H_8	101	398 (3.61)	13 (0.02)	11 (0.01)	13 (0.04)	12 (0.04)
IC	361			{34 (0.72)}		
MIC	361			{ 9 (0.10)}		
MAF	361			{11 (0.14)}		
F_9	361	263169/	14 (4.20)	12 (16.03)	13 (8.82)	11 (11.57)
			[17 (0.03)]		[14 (0.02)]	
H_9	361	1067 (48.69)	15 (0.04)	14 (0.04)	14 (0.07)	13 (0.09)
IC	1361			{66 (9.44)}		
MIC	1361			{12 (0.40)}		
MAF	1361			{19 (0.50)}		
F_{10}	1361	1050625/	18 (30.04)	16 (96.69)	17 (51.02)	14 (19.81)
			[21 (0.13)]		[18 (0.12)]	
H_{10}	1361	3740 (170.8)	17 (0.13)	16 (0.23)	16 (0.29)	15 (0.30)
			[23 (0.21)]		[20 (0.19)]	
H_{11}	5281	14628 (2115.3)	19 (0.55)	19 (1.14)	17 (2.85)	16 (3.09)
			[26 (2.91)]		[21 (1.68)]	
H_{12}	20801	55865 (9.44 h)	23 (6.13)	21 (6.84)	20 (9.28)	19 (10.60)
			[28 (9.54)]		[22 (6.47)]	
H_{13}	82561	214658 (1.82 d)	26 (27.13)	24 (25.14)	22 (24.28)	20 (23.71)

4 The description of the program

In the following the short decription of several new routines should be given, where their connexion tree is presented in Fig. 4. The source codes are included in the directory gglobisc/HangNode belonging to the Client/Server system of the SFB393 in Chemnitz. The subroutine

GGINPUT asks for the "Globisch-Nepomnyaschikh"-preconditioner and reads the initial input (2)-(5) for if those are selected; see the commando tool picture Fig. 2.

HEIGHTDEF computes the mesh size parameter $\bar{h}$ for an unstructured triangular grid using the triangle heights; see [11] for more details.

RECTADEF defines the auxiliary square encompassing the unstructured mesh. The length l of the square side is calculated taking the parameter $\bar{h}$ and the appropriately chosen depth J of the auxiliary hierarchical grid inside into account such that we have $\bar{h} = 12^{-J}$. Moreover, the location of the square is centred w.r.t. the x, y, and z ranges of the original domain. Instead of the square, sometimes the user is asked for a rectangle encompassing the mesh as above and having the maximum side length l. The side lengths of the rectangle are set according to the coordinate ranges of the domain, where no centralization is made.

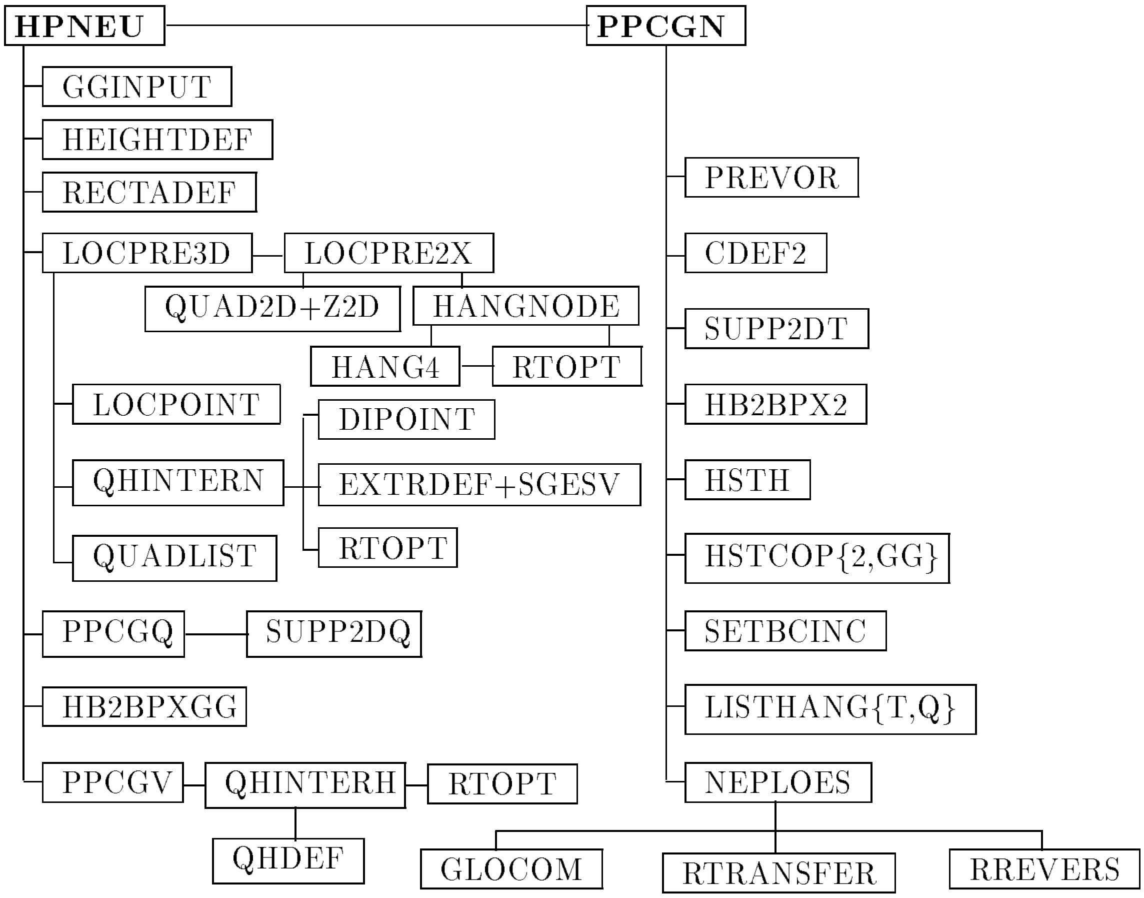


Figure 4: Scheme of (new) important routines called by the main program **HPNEU**

LOCPRE3D defines the full/adaptive hierarchy inside of the auxiliary triangular/rectangular grid meshing the hull square as well as the hull rectangle having the hierarchical depth J. Finally, this adapter results in computing the auxiliary hierarchical lists in the four cases ({full scheme, hanging node scheme} combined with {"artYs", "artBPX"}) correspondingly.

LOCPRE2X is the module for performing the hanging node mapping in the 2D-case, where the nodal coordinate vector is an essential input.

QUAD2D+Z2D defines a rectangular cell in the artificial hierarchy of depth J as well as, in general, their eight neighbour cells such that we have an octal tree available for performing the mapping. Every cell is identified by the name of the centred (son-)vertex, the names of the four related fathers and the names of the four midpoints between them. Furthermore, the coordinates of these points are computed and an information as to whether and how the cell has a non-empty intersection with the boundary of the hull-square/rectangle is coded.

HANGNODE handles the octal tree within the loop $l = 1, 2 \ldots, J$ as described in section 2.

HANG4 operates specifically with the four little subcells which are contained in all of the cells belonging to the last level $l = J$ of the hierarchy having the identifying centre vertex in common.

RTOPT prepares the one-to-one correspondence between the nodes in the unstructured grid and the corresponding vertices in the artificial hierarchy optimizing the mapping by determining a smaller and larger cell-neighbourhood-connexion, respectively, cf. (9). Roughly speaking, the octal tree can be extended up to the "24-tree".

LOCPOINT computes the names of the fathers and the current depth l of an input nodal name within an artificial square-grid hierarchy of depth J. We note that the L^2 points in the 2D-hierarchy are numbered linewise from the left to the right starting at the lower base of the square moving up.

QHINTERN generates the step form approximation $\tilde{Q}^h$ in the case of having only $\tilde{L}$ artificial unknowns, see [11]. The step form approximation is necessary for performing the inner Jacobi-preconditioning, see also [11]. This routine uses the algorithm for performing the inner-outer decision basing on the convex polyhedral element description, see [11], *Computing* article.

PPCG{V,Q} asks the user for the decision as to whether the distinct boundary conditions of the problem must be taken into account for defining the hierarchical preconditioner artificially in the case of the full hierarchy consisting of triangular/rectangular elements, see [11, 24]. Furthermore, the parameter **rmult** for the weighting of the mapping R^T is a real input value (default = 0.). Moreover, the subroutine performes corresponding actions depending on the (in)homogeneity as well as on the material properties of the problem to be solved.

SUPP2D{T,Q} prepares the boundary depending artificial preconditioner defined by the fully triangular and the fully rectangular hierarchy, respectively. In this case the preconditioner depends on the kind of boundary conditions if the above answer into **PPCG{V,Q}** was "yes". Here, the artificial unknowns are marked if their support is specifically related to the boundary conditions imposed on the original discretization, where a lot of additional computational effort is required. However, the amount hardly results in improving the convergence speed.

QHINTERH generates the step form approximation Q^h in the case of having all of the L^2 artificial unknowns, see [11]. The implemented "cut-algorithm" is a bit faster than the convex polyhedral method.

QHDEF is the heart of the routine **QHINTERH** performing the cut between the horizontal line and the element boundary scoring the number of cut points, cf. [11], *preprint* SFB97_11.

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

PREVOR extracts and modifies the main diagonal of the original stiffness matrix K appropriately for the auxiliary preconditioning, where some steps are prepared, which are also necessary for.

DIPOINT helps to compute the matrix R^* locating the grid points of the original unstructured mesh w.r.t. the corresponding cell-rectangle of the auxiliary hierarchical grid Π^h; see [11] for more details.

EXTRDEF defines the minimum hull-rectangle consisting of all of the cells of the grid Π^h that contain the corresponding triangle of the original mesh, where the three vertices of the element are taken into account.

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

CDEF2{A,I} computes the auxiliary main diagonal of the auxiliary operator $A_{Q^h/\tilde{Q}^h}$ (see [11]) which is designed to perform the interior Jacobi-preconditioning implemented between the two hierarchical multiplications provided that a potential problem without discontinuous coefficient functions must be solved. Otherwise, in most cases we have to use the outer Jacobi–preconditioning solely multiplying with the diagonal matrix derived from the main diagonal of the stiffness matrix K as usually. The subroutine marked by the suffix **A** and **I**, respectively implements the arithmetical mean approximation (input: rmult $\in [0, 2]$ and the weighted distance approximation (input: rmult $\in (2, 2.5)$, see [11]).

HB2BPX{GG,2} computes the hierarchical BPX–list from the artificially constructed hierarchical Yserentant-list belonging to the hierarchical grids Π^h and $\tilde{\Pi}^h$, respectively. The subroutine marked by the suffix "**2**" is the old version based on the list-structure (vertex, father1, father2, bit-code) without the description vector. This routine is called provided that the fully hierarchical scheme consisting of triangles is input, where the routines **HSTH** and **HSTCOP2** complete the corresponding transformation. Otherwise, having the hierarchy consisting of rectangular elements both in the full scheme case and in the case of hanging nodes the subroutine with the suffix "**GG**" is called, where the new compact data structure is implemented using the description vector at the top of the list. We note that the hanging nodes are handled well transforming the list. However, the existence of two routines for transfering the hierarchical list to the BPX-list depending on different data structures isn't fine. Hence, a redesign of the software could be a task in the future.

SETBCINC finally defines the auxiliary preconditioner $C^{-1}_{\Pi^h, bc(\Gamma^h)}$ in the case of the consideration of the boundary conditions calling the subroutine **SUPP2D{T,Q}**.

LISTHANG{T,Q} weights the interior Jacobi-preconditioning specifically according to the given input rmult. In most cases the default rmult = 0. defining no weighting is recommendable. Moreover, in the case of the hanging node approach the corresponding coefficient in the intermediate correction vector $Q^T \underline{v}$ is set to be zero, cf. [11].

NEPLOES solves the preconditioning system, i.e. $\underline{w} := R[C^{-1}_{\Pi^h/\tilde{\Pi}^h}]R^* \underline{r}$ distinctly applying the artYs-method and the artBPX-method, respectively; see also [11].

GLOCOM performes the communication step w.r.t. the correction values belonging to the coupling nodes. Computing in parallel 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.

5 Numerical results

The tables present the results for the cg–algorithm preconditioned by the artificially constructed Yserentant preconditioner "artYs" as well as by the arificially constructed BPX–preconditioner "artBPX" computing the itemized test examples. The subcolumn marked by "full scheme" means that computations are performed using the unstructured mesh generated by the mesh generator in [10] and embedded in the full rectangular hierarchy of depth J. For comparison, the full scheme consisting of the triangular hierarchy provided worser results, cf. [11]. For more insight into the entity of the method, in this columns the brackets include the iteration number and the CPU-time for the conventionally hierarchical methods on structured grids. By this we make an unfair confrontation which is nevertheless quite satisfactory regarding the iteration numbers. To provide a more realistic comparison in braces we give the results of the cg-algorithm preconditioning the matrix A by their Modified Incomplete Cholesky factorization (MIC), where the previous nodal renumbering is substantial and much more expensive than the preliminary steps within the "art-methods". In this case we have the condition number $\kappa(C^{-1}A) = O(h^{-1})$. The nodal renumbering is performed by an algorithm which improves minimal degree ordering and nested dissection combining both techniques, see [9]. For this the corresponding CPU-time is excluded. The subcolumn marked by "hanging nodes" contains the results based on the same unstructured grids but using the new adaptive mapping technique to get rid the "full-density effect" caused by locally refined mesh regions. Both the number of cg–iterations and the corresponding CPU–time (in sec)[1] are given which were necessary to get the relative error of the cg–iteration less than the previously defined accuracy $\epsilon = 10^{-4}$. This error is measured in the $AC^{-1}A$-norm.
At the bottom of the tables the percentages of the CPU–time are given which were necessary for performing the operations indicated by R^*, R, and the preconditioning $C^{-1}_{\Pi^h}$ within the cg–iteration, respectively, where the third percentage includes also the amount of the cg–iteration itself. The percentages belong to the "hanging node part" of the tables and they are measured on an average w.r.t. the given depths J of the auxiliary grids. Comparing these percentages with those of the full scheme (see [11]) the portion of R^* and R is slightly degraded.

Nevertheless, provided that the time for performing the adaptive mapping is excluded the artificially hierarchical strategy using the nodal coordinates and the element connexion does only need an amount which is approximately 1.5 times more than the effort of the originally hierarchical technique requiring structured mesh data. Hence, the application of our method could be a good practice especially in the industrial engineering. Moreover,

[1]In the given CPU-time neither the times for generating the adaptive mapping and the hierarchical lists of the auxiliary grid $\tilde{\Pi}^h$ nor the time for the construction of the step form approximation $\tilde{Q}^h$ inside are incorporated. In practice this hidden amount does enlarge the real CPU-time substantially.

our method can be easily implemented into existing software since the algorithms are based on a modular toolkit.
Regarding the artBPX-method in several examples their convergence seems to be slightly dependent on the mesh size parameter h. Due to [26] the independency of h can be expected to have the starting point at a sufficiently fine level. In particular for interface problems with different material properties the mentioned convergence behaviour is distinct. Here, the outer Jacobi-preconditioning damps the high frequencies insufficiently only in the points of the original grid.
Computing example no. **4** and no. **5** we observed the following. The more the quasi-uniformity of the unstructured mesh is deteriorated progressively the more the iteration number of the artificially hierarchical preconditioned cg–method does increase.

5.1 Sequential Computing

The results are computed by means of the HP 9000/889 K460-workstation using large memory size (1GigaByte) and on an average 15MFlop performance.

1. *Preconditioning having the potential problem in the square:*

$$-\Delta u = 0 \quad \text{in} \quad \Omega = (0,4) \times (0,4)$$

$$u = \begin{cases} 0\,, & \text{on } \Gamma_{01} = \{x = (x_1, x_2)^T : x_1 = 0, x_2 < 1\} \cup \{x : x_2 = 0\,, 0 < x_1 \leq 4\} \\ 1\,, & \text{on } \Gamma_{02} = \{x : x_1 = 0, 1 \leq x_2 \leq 4\}\,, \end{cases}$$

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

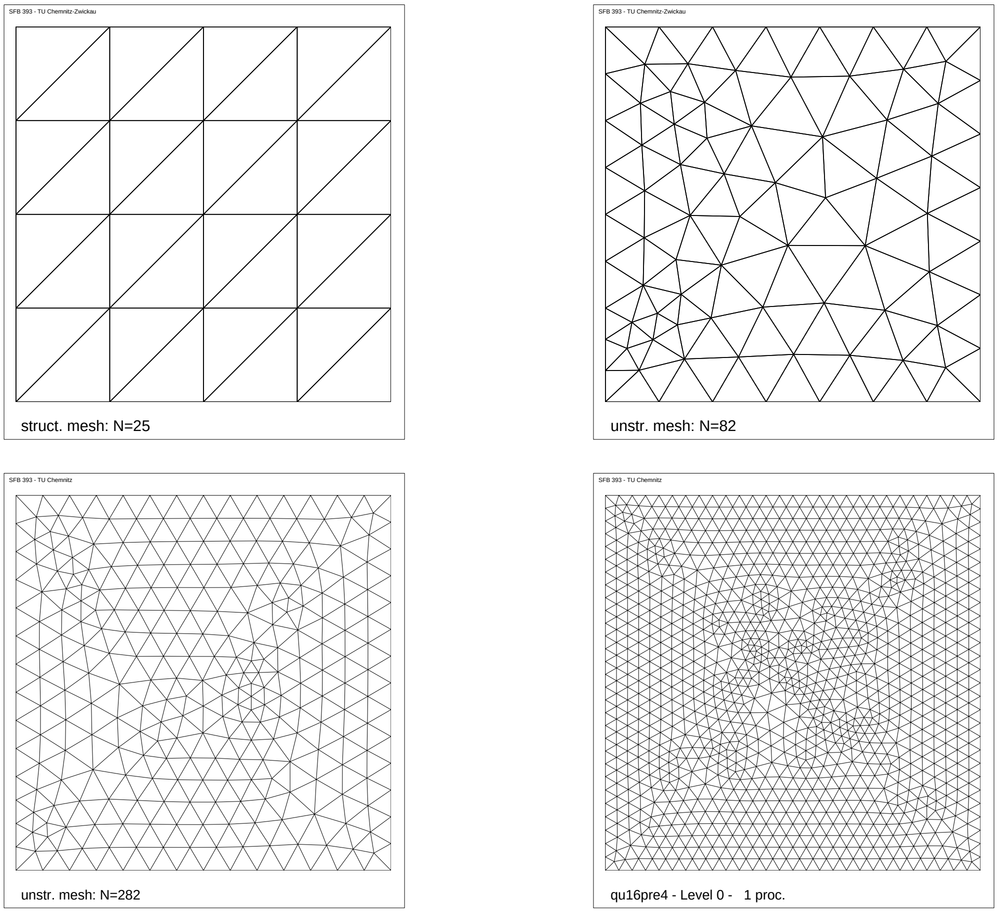


Figure 5: Struct. mesh (N=25); subsequence of unstruct. meshes (N = 82, 282, 1094)

Table 2: #cg–iterations and CPU–times for the computing in the square

J	N	$L^2/\tilde{L}$	artYs full scheme	artYs hanging nodes	artBPX full scheme	artBPX hanging nodes
4	25	289/81	[9 (0.00)] 10 (0.01)	{ 6 (0.00)} 11 (0.00)	[9 (0.00)] 9 (0.01)	10 (0.01)
5	82	1089/266	[13 (0.01)] 13 (0.03)	{ 7 (0.00)} 13 (0.01)	[11 (0.01)] 10 (0.03)	12 (0.02)
6	282	4225/906	[17 (0.01)] 16 (0.15)	{10 (0.01)} 18 (0.04)	[13 (0.01)] 11 (0.12)	15 (0.04)
7	1094	16641/3007	[20 (0.03)] 17 (0.66)	{13 (0.03)} 18 (0.14)	[14 (0.03)] 12 (0.53)	14 (0.19)
8	4260	66049/9634	[24 (0.14)] 19 (3.28)	{22 (0.20)} 20 (0.77)	[15 (0.10)] 12 (2.61)	15 (0.70)
9	16811	263169/34899	[26 (0.79)] 18 (14.22)	{30 (2.77)} 20 (2.19)	[15 (0.51)] 13 (12.49)	16 (3.46)
10	66789	1050625/132423	[28 (4.36)] 21 (62.42)	{37 (6.85)} 22 (11.19)	[15 (2.56)] 16 (124.25)	18 (17.17)
11	266249	4198401/519777	[29 (21.13)] 21 (262.70)	{70 (56.73)} 26 (56.20)	[15 (12.09)] 18 (279.77)	22 (121.81)
12	1063185	16785409/2078777	[29 (86.18)] 28 (1382.29)	{121 (445.41)} 30 (292.76)	[15 (48.95)] 19 (1037.1)	22 (244.46)
13	4249121	67125249/8072229	[29 (348.20)] mem.ex.	{t_{ren}.ex.} 33 (1264.1)	memory exceeded	
R^*:				20		21
R:				18		18
$C_{\Pi^h}^{-1}$:				62		61

2. *Preconditioning having the potential problem in the club shaped domain:*

$$-\Delta u = 0 \quad \text{in} \ \ \Omega\,, \quad \text{(see Fig. 6)}$$
$$u = \begin{cases} \ \ 1\,, & x \in \Gamma_{01} \text{ marked by (1) in Figure 6} \\ -1\,, & x \in \Gamma_{02} \text{ marked by (2) in Figure 6}\,, \end{cases}$$
$$\partial u/\partial N = 0 \quad \text{on} \quad \Gamma_1 = \partial\Omega \setminus (\Gamma_{01} \cup \Gamma_{02})\,.$$

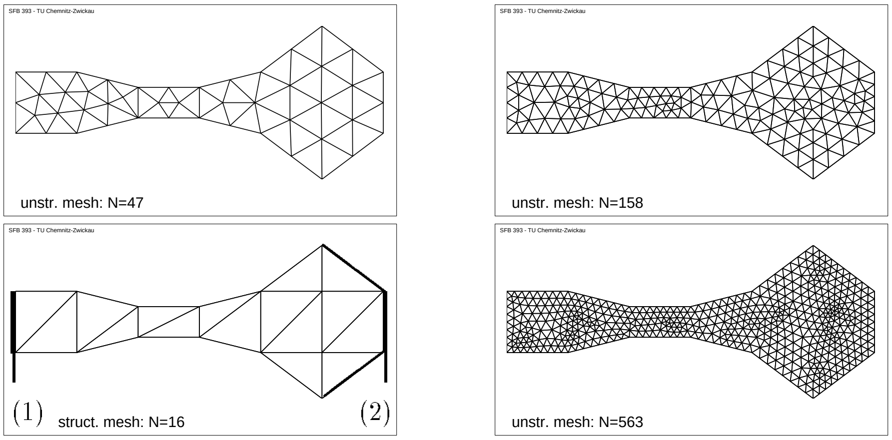


Figure 6: Struct. mesh (N=16) and subsequence of unstruct. meshes (N=47, 158, 563)

Table 3: #cg–iterations and CPU–times for the computing in the club shaped domain

J	N	$L^2/\tilde{L}$	artYs full scheme	artYs hanging nodes	artBPX full scheme	artBPX hanging nodes
5	16	1089/150	[9 (0.00)] 7 (0.02)	{6 (0.00)} 12 (0.01)	[9 (0.00)] 7 (0.02)	11 (0.01)
6	47	4225/383	[15 (0.00)] 14 (0.14)	{8 (0.00)} 15 (0.02)	[14 (0.00)] 13 (0.17)	15 (0.03)
7	158	16641/830	[19 (0.01)] 17 (0.70)	{10 (0.02)} 25 (0.05)	[17 (0.01)] 14 (0.65)	20 (0.09)
8	563	66049/2291	[22 (0.02)] 19 (4.84)	{14 (0.04)} 28 (0.16)	[19 (0.02)] 15 (3.16)	21 (0.22)
9	2182	263169/6467	[26 (0.07)] 20 (14.21)	{19 (0.07)} 25 (0.42)	[19 (0.07)] 17 (15.44)	24 (0.70)
10	8279	1050625/24204	[28 (0.36)] 23 (68.03)	{26 (0.57)} 35 (2.47)	[19 (0.27)] 20 (71.73)	25 (5.73)
11	32637	4198401/86816	[30 (2.00)] 27 (310.82)	{39 (3.57)} 41 (11.37)	[19 (1.49)] 22 (319.02)	30 (16.82)
12	132529	16785409/344571	[30 (11.01)] 36(1676.5)	{64 (27.59)} 46 (50.30)	[19 (7.04)] 24 (1318.9)	33 (48.86)
R^*:				18		13
R:				9		8
$C_{\Pi^h}^{-1}$:				73		79

3. *Preconditioning having the problems* (a) *and* (b) *in the circular domain:*

$$-\mathrm{div}(a(x)\mathrm{grad}(u(x))) = 0 \quad \text{in } \Omega = \{x : x_1^2 + x_2^2 < 1\}$$

where (a) : $a(x) = 1\,, \quad x \in \Omega\,,$

$$\text{and} \quad (\mathrm{b}) : a(x) = \begin{cases} 1\,, & x \in \Omega_1 = \Omega \setminus \bar{\Omega}_2 \\ 10^6\,, & x \in \Omega_2 \text{ marked by (2) in Figure 7} \end{cases}$$

$$u = \begin{cases} 100\,, & \text{on } \Gamma_{01} = \{x : \; x_1^2 + x_2^2 = 1\,, \; -1 \le x_1 \le -\frac{\sqrt{2}}{2}\,, \; 0 \le x_2 \le \frac{\sqrt{2}}{2}\,\} \\ 0\,, & \text{on } \Gamma_{02} = (\partial\Omega \backslash \Gamma_{01})\,. \end{cases}$$

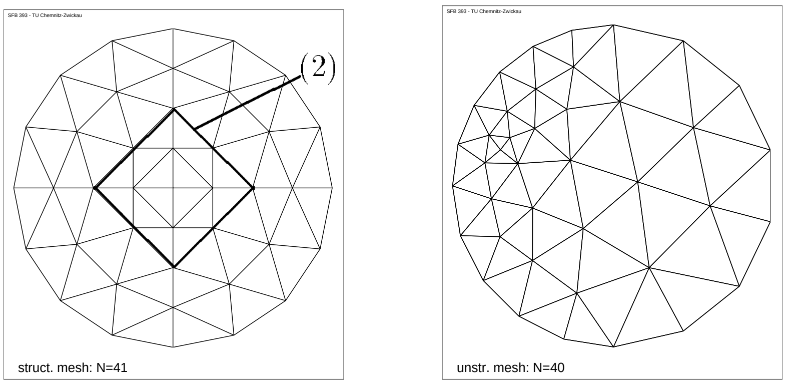


Figure 7: Struct. mesh (N=41) and unstruct. mesh ($N = 40$) in the circular domain

Table 4: #cg–it. and CPU–times for the homogeneous problem in the circular domain

J	N	$L^2/\tilde{L}$	artYs full scheme	artYs hanging nodes	artBPX full scheme	artBPX hanging nodes
4	15	81/55	[5 (0.00)] 5 (0.00)	{ 3 (0.00)} 5 (0.00)	[5 (0.00)] 5 (0.01)	5 (0.00)
5	40	289/217	[9 (0.00)] 10 (0.09)	{ 5 (0.00)} 13 (0.01)	[8 (0.00)] 9 (0.02)	11 (0.01)
5	146	1089/490	[14 (0.00)] 11 (0.04)	{ 8 (0.00)} 16 (0.03)	[11 (0.00)] 11 (0.04)	14 (0.03)
6	545	4225/1736	[18 (0.02)] 15 (0.14)	{11 (0.01)} 20 (0.08)	[13 (0.01)] 14 (0.16)	16 (0.11)
7	2115	16641/5123	[21 (0.06)] 18 (0.71)	{15 (0.07)} 23 (0.30)	[15 (0.05)] 11 (0.52)	18 (0.35)
8	8340	66049/18942	[24 (0.32)] 20 (3.45)	{23 (0.41)} 27 (1.47)	[15 (0.22)] 13 (3.74)	19 (1.78)
9	33123	263169/70805	[25 (1.70)] 24 (20.55)	{36 (3.43)} 31 (7.60)	[16 (1.25)] 15 (16.27)	21 (8.17)
11	132021	4198401/422562	[26 (9.68)] 26 (456.58)	{55 (22.27)} 40 (50.73)	[16 (6.19)] 18 (269.69)	24 (57.64)
12	525313	16785409/1658587	[26 (41.19)] 40(2790.0)	{81 (255.71)} 51 (289.38)	[16 (27.21)] 22 (1743.3)	26 (187.31)
R^*:				23		22
R:				20		21
$C_{\Pi^h}^{-1}$:				57		57

Table 5: #cg–iterations and CPU–times for the material problem in the circular domain

J	N	$L^2/\tilde{L}$	artYs full scheme	artYs hanging nodes	artBPX full scheme	artBPX hanging nodes
4	16	289/65	[5 (0.00)] 6 (0.04)	{ 6 (0.00)} 6 (0.00)	[5 (0.00)] 6 (0.00)	6 (0.00)
5	40	1089/154	[18 (0.00)] 23 (0.06)	{11 (0.00)} 22 (0.01)	[15 (0.00)] 18 (0.05)	20 (0.01)
6	144	4225/601	[33 (0.02)] 37 (0.35)	{15 (0.01)} 47 (0.09)	[23 (0.01)] 30 (0.36)	36 (0.11)
7	547	16641/2007	[45 (0.04)] 47 (1.71)	{21 (0.03)} 62 (0.32)	[28 (0.04)] 41 (1.76)	47 (0.41)
7	2115	16641/5123	[53 (0.15)] 21 (0.71)	{32 (0.16)} 28 (0.33)	[35 (0.12)] 18 (0.69)	24 (0.42)
8	8340	66049/18942	[63 (0.81)] 25 (4.01)	{48 (0.80)} 36 (1.78)	[38 (0.57)] 22 (4.13)	28 (2.13)
9	33123	263169/70805	[72 (4.93)] 31 (22.55)	{79 (6.82)} 35 (7.55)	[41 (3.24)] 21 (17.27)	25 (8.30)
10	132021	1050625/291214	[80 (30.45)] 49 (144.00)	{123 (44.07)} 44 (88.41)	[47 (23.89)] 35 (119.82)	29 (43.11)
11	527145	4198401/1077678	[88 (243.07)] 62(1185.6)	{212 (679.29)} 59 (416.96)	[51 (92.45)] 50 (728.82)	36 (221.94)
12	2106705	16785409/4307078	[94 (620.26)] 81(4162.4)	{t_{ren}.ex.} 67 (1762.1)	[53 (383.96)] mem.ex	34 (980.21)

The real unstructured meshes for computing the inhomogeneous problem were also generated by the advancing front algorithm in [10], where the interfaces can be taken into account. To abbreviate the section we forego presenting corresponding grids.

4. *Preconditioning having the problems* (a) *and* (b) *in the "SFB–domain":*

$$
\begin{aligned}
&-\mathrm{div}(a(x)\mathrm{grad}(u(x))) = 0 \quad \text{in} \quad \Omega = SFB\,, \quad \text{see Figure 1}\,,\\
&\text{where (a): } a(x) = 1\,, \quad x \in \Omega = SFB\,,\\
&\text{and} \quad \text{(b): } a(x) = \begin{cases} 1\,, & x \in S \\ 10^3\,, & x \in F \\ 10^6\,, & x \in B \end{cases}\\
&u = x_1 + x_2 + 1 \quad \text{on} \quad \Gamma_0 \quad = \quad \text{exterior part of } \partial\Omega\,,\\
&\partial u/\partial N = 0 \quad \text{on} \quad \Gamma_1 = \partial\Omega \setminus \Gamma_0 \quad = \quad \text{3 interior boundary pieces}\,.
\end{aligned}
$$

For this problem the unstructured mesh having $N = 163$ nodes was already shown in Figure 1. For completing the structured part of the tables below we used the initial mesh given in Figure 2 ($N = 50$) consecutively refining it canonically.

Table 6: #cg–iterations and CPU–times for the homogeneus problem in the "SFB-domain"

			artYs		artBPX	
J	N	$L^2/\tilde{L}$	full scheme	hanging nodes	full scheme	hanging nodes
5	50	1089/333	[7 (0.00)] 7 (0.02)	{ 4 (0.00)} 8 (0.00)	[8 (0.00)] 6 (0.02)	8 (0.01)
6	163	4225/697	[15 (0.01)] 12 (0.10)	{ 7 (0.00)} 17 (0.03)	[16 (0.03)] 11 (0.11)	14 (0.04)
7	532	16641/2320	[22 (0.03)] 14 (0.45)	{10 (0.01)} 22 (0.11)	[21 (0.04)] 11 (0.42)	13 (0.09)
8	2001	66049/6771	[28 (0.10)] 22 (3.25)	{15 (0.06)} 29 (0.42)	[25 (0.10)] 15 (2.76)	17 (0.37)
9	7744	263169/22105	[33 (0.39)] 33 22.46)	{23 (0.36)} 37 (1.98)	[28 (0.45)] 20 (17.98)	19 (1.55)
10	30450	1050625/79311	[38 (2.85)] 43 (118.40)	{39 (3.46)} 35 (8.11)	[30 (2.48)] 24 (87.73)	25 (8.51)
11	120742	4198401/301655	[41 (14.93)] 48(542.61)	{60 (21.50)} 47 (58.54)	[31 (13.38)] 27 (405.26)	29 (45.94)
12	453694	16785409/1088705	[44 (67.55)] 57(2700.6)	{93 (127.25)} 57 (250.53)	[32 (52.05)] 30(1781.0)	32 (163.15)
R^*:				25		23
R:				23		20
$C_{\Pi^h}^{-1}$:				52		57

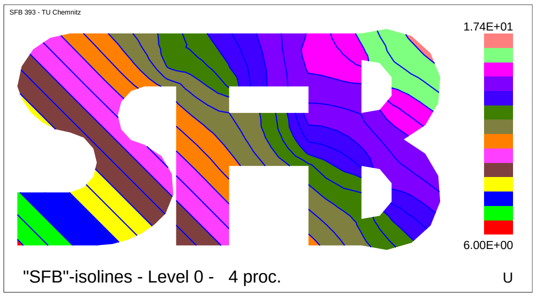


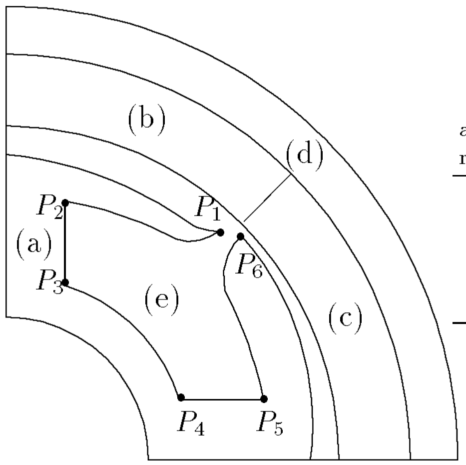
Figure 8: The filled "SFB"–isoline picture delivered by our postprocessing

Table 7: #cg–it. and CPU–times for the material problem in the "SFB–domain"

J	N	$L^2/\tilde{L}$	artYs full scheme	artYs hanging nodes	artBPX full scheme	artBPX hanging nodes
5	50	1089/333	[8 (0.00)] 7 (0.02)	{ 4 (0.00)} 8 (0.00)	[8 (0.01)] 7 (0.02)	7 (0.01)
6	163	4225/697	[16 (0.01)] 13 (0.11)	{ 7 (0.00)} 14 (0.02)	[14 (0.03)] 11 (0.11)	13 (0.04)
7	532	16641/2320	[22 (0.03)] 17 (0.54)	{10 (0.01)} 23 (0.12)	[20 (0.03)] 14 (0.52)	16 (0.12)
8	2001	66049/6771	[27 (0.08)] 24 (3.46)	{16 (0.06)} 30 (0.44)	[24 (0.10)] 20 (3.46)	21 (0.46)
9	7744	263169/22105	[33 (0.37)] 33 (21.84)	{24 (0.37)} 37 (1.98)	[26 (0.44)] 30 (25.57)	29 (2.38)
10	30450	1050625/79311	[37 (2.55)] 56 (152.83)	{38 (3.16)} 51 (11.60)	[28 (2.29)] 49 (193.95)	39 (13.04)
11	120742	4198401/301655	[40 (13.98)] 73 (778.73)	{63 (22.35)} 80 (84.83)	[28 (11.39)] 61 (971.21)	59 (89.56)
12	453694	16785409/1088705	[43 (65.61)] 108(4916.2)	{96 (130.59)} 90 (416.90)	[28 (45.73)] 70 (3186.9)	67 (301.51)
R^*:				23		22
R:				22		21
$C_{\Pi^h}^{-1}$:				55		57

5. *Magnetic field computation in an electronic motor:*

The example is of practical interest, see [9, 10, 17] for details. The domain $\mathbf{\Omega}$ is the fourth of the cross section of an electronic motor in which magnetic field computation is performed. Figure 9 presents the motor's geometry with distinct material properties additionally connected with geometric peculiarities causing solution's singularities in several indicated points P_i, $i = 1, \dots, 6$.



abs. permeability : $\mu_0 = 1.257 * 10^{-6} Vs/Am$
rel. permeability and the given materials :

(a)	iron rotor	$\mu_r = 1694$
(b), (c)	permanent magnet	$\mu_r = 1.15$
(d)	sheet–metal shell	$\mu_r = 2488$
(e)	air gap	$\mu_r = 1$

Figure 9: The fourth-cross section of the electronic motor containing 4 materials

By Maxwell's laws the magnetic field problem defined on the motor's cross section can be rewritten in the following variational formulation, cf. also [9, 17] :

Find the function $u \in \overset{\circ}{H}{}^1$ such that for all $v \in \overset{\circ}{H}{}^1$ holds :

$$\int\limits_{\Omega} \frac{1}{\mu_0 \mu_r(x)} \nabla^T u \, \nabla v \, dx_1 \, dx_2 = \int\limits_{\Omega} \frac{1}{\mu_0 \mu_r(x)} \left(\frac{\partial v}{\partial y} B_{0x_1} - \frac{\partial v}{\partial x} B_{0x_2} \right) \, dx_1 \, dx_2 \ ,$$

where B_{0x_1} and B_{0x_2} denote the remanent inductions of the permanent magnet in x_1 and in x_2 direction, respectively.
Because of the complicate inner geometry no structured grids for discretizing the domain are available. Moreover, by the automatical mesh generator in [10] the unstructured mesh can be initially adapted to the given point singularities, see Figure 10.

Table 8: #cg–iterations and CPU–times for the magnetic field problem

J	N	$L^2/\tilde{L}$	artYs full scheme	artYs hanging nodes	artBPX full scheme	artBPX hanging nodes
8	469	66049/1834	[– –] 42 (6.05)	{22 (0.02)} 58 (0.24)	[– –] 40 (6.89)	51 (0.44)
9	1831	263169/5964	[50 (0.68)] 67 (43.60)	{33 (0.17)} 91 (1.20)	[48 (0.41)] 60 (52.22)	71 (2.02)
10	7237	1050625/20671	[54 (0.93)] 100 (263.88)	{48 (0.70)} 156 (7.82)	[49 (1.20)] 76 (317.98)	121 (14.49)
11	28777	4198401/76340	[59 (5.25)] 97 (1025.7)	{69 (5.48)} 119 (26.72)	[51 (5.48)] 100 (1386.1)	90 (47.80)
12	114769	16785409/292997	[68 (29.93)] 149 (7003.8)	{113 (38.82)} 202 (173.46)	[52 (18.47)] 123 (6640.1)	157 (160.74)
R^*:				27		23
R:				23		20
$C_{\Pi^h}^{-1}$:				50		57

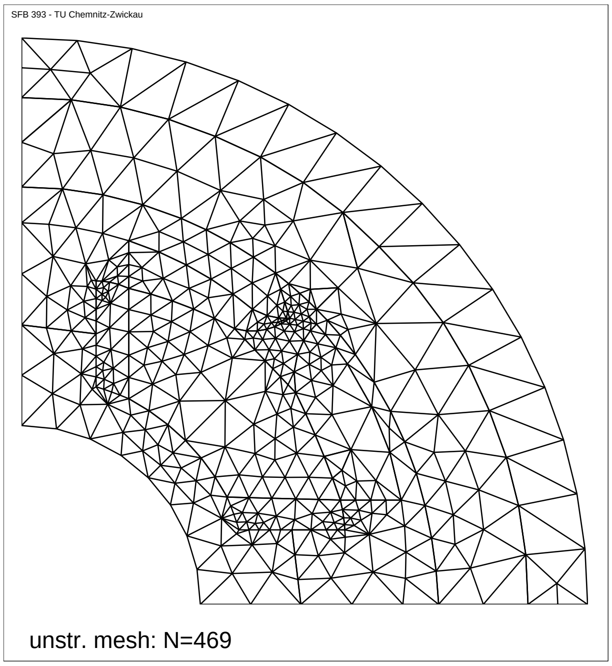


Figure 10: The adaptive mesh of the motor's fourth losing the quasiuniformity

Computing the inhomogeneous problems the weaker increasing of the iteration numbers starting at a certain stage of J is due to the better approximation of the interfaces as it is made more precisely in the form of steps. This can be observed e.g. in Table 4. Moreover, to overcome the difficulties caused by jumping coefficient functions still occuring in the case of the adaptive mapping with hanging nodes at present we discuss the shifting of appropriately choosen nodes of the domain $\tilde{Q}^h_J$ onto the interfaces resulting in a locally irregular grid $\tilde{\tilde{Q}}^h_J$.

5.2 First results of the parallel computing

To get the results we used the well known Parsytec parallel computer GCPowerPlus having 32MByte memory at each processor node and a peak performance of 80MFlop. For more details describing the related software tools see also [16].
The next examples are computed using 16 processors in each case. The parallelism for the conventional hierarchical methods resulting in the data given in brackets is based on the domain decomposition (DD) given by the meshes in the leftabove part of Figure 5 and 6, respectively. Computing in the square we have the 16 subsquares consisting of the two initial triangles shown in Figure 5. Computing in the club shaped domain we have the 16 subtraingles given in the leftbelow of Figure 6. For the parallelization of the "real unstructured grid"–computations (full scheme and hanging node approach) the FE–data are distributed by recursively spectral bisection after the mesh generation was done by the parallel mesh generator in [10]. Further DD-based experiments are contained e.g. in [14]. The parameter L^2, $\tilde{L}$ and J are the maxima $L^2 = \max(L_s^2)$, $\tilde{L} = \max(\tilde{L}_s)$, $s = 1, \ldots, p$ and $J = \max(J_s)$, $s = 1, \ldots, p$, respectively.

1. *The decomposed problem no.* **1.** *of the previous subsection:*

Table 9: #cg–it. and CPU–times, where data distribution was made

J	N	$L^2/\tilde{L}$	artYs full scheme	artYs hanging nodes	artBPX full scheme	artBPX hanging nodes
4	25	9	[10 (0.00)] 10 (0.04)	16 (1.14)	[9 (0.00)] 12 (0.13)	16 (1.30)
5	84	1089/204	[13 (0.18)] 22 (0.56)	26 (0.50)	[11 (0.20)] 22 (0.60)	30 (0.66)
6	295	4225/432	[17 (0.22)] 36 (1.82)	37 (1.33)	[13 (0.26)] 41 (2.29)	44 (1.61)
7	1069	16641/840	[20 (0.28)] 61 (8.08)	61 (4.07)	[14 (0.30)] 65 (10.07)	69 (4.95)
8	4260	66049/2520	[24 (0.36)] 104 (49.68)	120 (24.16)	[15 (0.37)] 111 (64.05)	122 (26.03)
9	16811	263169/5959	[26 (0.47)] 170 (238.19)	203 (44.08)	[15 (0.47)] 128 (197.23)	193 (48.11)
10	65217	1050625/15724	[26 (0.47)] mem.ex.	328 (64.60)	[15 (0.47)] mem.ex.)	244 (50.86)
11	193620	4198401/55216	[26 (0.47)] mem.ex.	425 (167.23)	[15 (0.47)] mem.ex.	mem. ex.
R^*:				21		22
R:				20		20
$C^{-1}_{\Pi^h}$:				59		58

2. *The decomposed problem no.* **2.** *of the previous subsection:*

Table 10: #cg–it. and CPU–times, where data distribution was made

J	N	$L^2/\tilde{L}$	artYs		artBPX	
			full scheme	hanging nodes	full scheme	hanging nodes
??2	16	25	[10 (0.00)] 16 (0.07)	8 (0.00)	[6 (0.00) 6 (0.29)	12 (0.07)
3	55	81/58	[15 (0.19)] 24 (0.45)	26 (0.47)	[14 (0.24)] 25 (0.47)	26 (0.49)
6	157	4225/363	[19 (0.24)] 34 (1.40)	36 (0.89)	[17 (0.31) 37 (1.68)	41 (1.36)
7	570	16641/623	[22 (0.29)] 42 (4.66)	58 (2.41)	[19 (0.37) 45 (5.81)	66 (3.04)
8	2130	66049/1866	[26 (0.37)] 83 (32.76)	117 (13.45)	[19 (0.41) 89 (43.32)	126 (15.26)
9	8279	263169/3979	[28 (0.43)] 157 (204.80)	191 (24.46)	[19 (0.51) 171 (85.06)	207 (28.53)
10	32637	1050625/9598	[30 (0.71)] mem.ex.	323 (50.89)	[19 (0.72)] mem.ex.	330 (59.29)
11	129593	4198401/26658	[31 (1.77)] mem.ex.	613 (161.28)	mem. ex.	mem. ex.
R^*:				18		16
R:				10		10
$C_{\Pi^h}^{-1}$:				74		72

3. *The parallel magnetic field computation in the fourth of the motor:*

Table 11: #cg–it. and CPU–times, where data distribution was made

J	N	$L^2/\tilde{L}$	artYs		artBPX	
			full scheme	hanging nodes	full scheme	hanging nodes
8	469	66049/642	[––] 60 (20.38)	73 (7.14)	[––] 49 (20.83	63 (6.46)
9	1831	263169/1382	[51 (2.92)] 117 (129.66)	140 (14.06)	[48 (4.71)] 131 (55.94)	114 (12.49)
10	7237	1050625/3075	[54 (3.14)] mem.ex.	232 (25.44)	[49 (6.72)] mem.ex.	196 (24.85)
11	28777	4198401/8429	mem. ex.	314 (42.97)	mem. ex	255 (47.58)
R^*:				24		23
R:				20		19
$C_{\Pi^h}^{-1}$:				56		58

Finally, let us give the following remarks comparing the results of the three tables presented here. If all of the subdomain meshes Ω_s^h , $s = 1, \ldots, p$, into which the whole mesh Ω^h is decomposed coincide with the auxiliary square grids Π_s^h , $s = 1, \ldots, p$, the computation in parallel is very efficient as it was expected, see Table 8. Otherwise, the step form approximation of the coupling boundaries defined by the domains Q_s^h / $\tilde{Q}_s^h$ which are subsets of the overlapped grids Π_s^h / $\tilde{\Pi}_s^h$, $s = 1, \ldots, p$, deteriorates the convergence of the preconditioned parallel cg–method substantially. We still seek a remedy to recover the fast convergence of the artificially preconditioned cg–methods in the general case of their parallelization. Moreover, the rectangular full scheme does rather degrade the parallel results computing the examples, whereas the scalar computation did improve the iteration numbers obviously. Despite the difficulties consisting in the efficient parallelization of our approach which remain yet to be solved the parallelized hanging node scheme seems to be the better method in the case of unstructured meshes really having locally refined regions specifically adapted to (geometric) peculiarities, see the parallel computation of the electronic motor. Nevertheless, the efficiency of the parallel methods does decrease as the number of processors does increase.

Acknowledgement *Many thanks to Mr. S.V. Nepomnyaschikh (Russian Academy of Sciences, Novosibirsk) for his helpful comments concerning the presented work.*

References

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

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

[3] Aubin, J.P. (1972): Approximation of elliptic boundary value problems. Wiley–Interscience, New York London Sydney Toronto.

[4] Bank, R.E., Xu, J. (1994): The hierarchical basis multigrid method and incomplete LU decomposition. Contemporary Mathematics **180**, 163–174.

[5] Bank, R.E., Xu, J. (1996): An algorithm for coarsening unstructured meshes. Numer. Math. **73**(1), 1–36.

[6] Bramble, J.H., Pasciak, J.E., Xu, J. (1990): Parallel multilevel preconditioners. Math. Comp. **55**, 1–22.

[7] Chan, T.F., Smith, B.F. (1994): Domain Decomposition and Multigrid algorithms for elliptic problems on unstructured meshes. Contemporary Mathematics **180**, 175–190.

[8] Ciarlet, Ph. (1977): The Finite Element Method for Elliptic Problems. North–Holland, Amsterdam.

[9] Globisch, G. (1993): Robuste Mehrgitterverfahren für einige elliptische Randwertaufgaben in zweidimensionalen Gebieten. Technische Universität Chemnitz, Dissertation, Chemnitz.

[10] Globisch, G. (1995): PARMESH – a parallel mesh generator. Parallel Computing **21**(3), 509–524.

[11] Globisch, G., Nepomnyaschikh, S.V. (1997): The hierarchical preconditioning having unstructured grids. Preprint SFB393/97_11, Technische Universität Chemnitz, Chemnitz, April 1997
accepted for publishing in Computing.

[12] Globisch, G. (1997): The hierarchical preconditioning having unstructured threedimensional grids. Preprint SFB393/97_25, Technische Universität Chemnitz, Chemnitz.

[13] Groh, U. (1997): FEM auf irregulären hierarchischen Dreiecksnetzen. Preprint SFB393/97_05, Technische Universität Chemnitz, Chemnitz, December 1997.

[14] Haase, G., Heise, B., Kuhn, M., Langer, U. (1997): Adaptive domain decomposition methods fof finite and boundary element equations. in: Wendland, W. (ed.) Boundary element topics. Proceedings of the conference of the priority research program Boundary Element Methods 1989–1995 of the German Research Fondation, October 2–4, 1995 in Stuttgart, Germany. Berlin: Springer-Verlag, 121–147.

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

[16] Haase, G., Hommel, Th., Meyer, A., Pester, M. (1995): Bibliotheken zur Entwicklung paralleler Algorithmen. Preprint SPC 95_20, Technische Universität Chemnitz–Zwickau, Chemnitz.

[17] Heise, B. (1994): Analysis of a fully discrete finite element method for a nonlinear magnetic field problem. SIAM J. Numer. Anal. **31**(3), 745–759.

[18] Matsokin, A.M., Nepomnyaschikh, S.V. (1993): The fictitious domain method and explicit continuation operators. Zh. Vychisl. Mat. Mat. Fiz. **33**, 45–59.

[19] Meyer, A. (1990): A parallel preconditioned conjugate gradient method using domain decomposition and inexact solvers on each subdomain. Computing **45**, 217–234.

[20] Meyer, A., Pester, M. (1994): Verarbeitung von Sparse-Matrizen in Kompaktspeicherform KLZ/KZU. Preprint SPC 94_12, Technische Universität Chemnitz-Zwickau, Chemnitz.

[21] Nepomnyaschikh, S.V. (1991): Method of splitting into subspaces for solving elliptic boundary value problems in complex-form-domains. Sov. J. Numer. Anal. Math. Model. **6**(2), 151–168.

[22] Nepomnyaschikh, S.V. (1991): Mesh theorems of traces, normalization of function traces and their inversion. Sov. J. Numer. Anal. Math. Model. **6**(3), 223–242.

[23] Nepomnyaschikh, S.V. (1995): Fictitious space method on unstructured meshes. East–West J. Numer. Math. **3**(1), 71–79.

[24] Nepomnyaschikh, S.V. (1996): Preconditioning operators on unstructured grids. in: Nelson, N.D., Manteuffel, T.A., McCormick, S.F., Douglas, C.C. (eds.), Proceedings of the Seventh Copper Mountain Conference on Multigrid Methods, no. 3339 in NASA-Conference Publication, 607–621.

[25] Oganesyan, L.A., Ruchovets, L.A. (1979): Variational Difference Methods for Solving Elliptic Equations. Izdat. Akad. Nauk Arm. SSR, Erevan, (Russian).

[26] Oswald, P (1994): Multilevel Finite Element Approximation: Theory and Applications. Teubner Skripten zur Numerik. B. G. Teubner Stuttgart.

[27] Queck, W. (ed.), (1993): FEMGP (Finite Element Multigrid Package). Programmdokumentation, Technologieberatungszentrum Parallele Informationsverarbeitung GmbH (TBZ*PARIV), Bernsdorfers Str. 210–212, D–09126 Chemnitz.

[28] Rjasanow, S. (1986): Dokumentation und theoretische Grundlagen zum Programm SOLKLZ. Preprint Nr. 15, Sektion Mathematik, Technische Universität Karl-Marx-Stadt.

[29] Xu, J. (1992): Iterative methods by space decomposition and subspace correction. SIAM Review **34**(4), 581–613.

[30] Xu, J. (1996): The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. Computing **56**(3), 215–235.

[31] Yakovlev, G.N. (1967): On traces of piecewise smooth surfaces of functions from the space W_p^l. Mat. Sbornik **74**, 526–543.

[32] Yserentant, H. (1986): On the multi–level splitting of finite element spaces. Numer. Math. **49**, 379–412.

Other titles in the SFB393 series:

96-01 V. Mehrmann, H. Xu. Chosing poles so that the single-input pole placement problem is well-conditioned. Januar 1996.

96-02 T. Penzl. Numerical solution of generalized Lyapunov equations. January 1996.

96-03 M. Scherzer, A. Meyer. Zur Berechnung von Spannungs- und Deformationsfeldern an Interface-Ecken im nichtlinearen Deformationsbereich auf Parallelrechnern. March 1996.

96-04 Th. Frank, E. Wassen. Parallel solution algorithms for Lagrangian simulation of disperse multiphase flows. Proc. of 2nd Int. Symposium on Numerical Methods for Multiphase Flows, ASME Fluids Engineering Division Summer Meeting, July 7-11, 1996, San Diego, CA, USA. June 1996.

96-05 P. Benner, V. Mehrmann, H. Xu. A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils. April 1996.

96-06 P. Benner, R. Byers, E. Barth. HAMEV and SQRED: Fortran 77 Subroutines for Computing the Eigenvalues of Hamiltonian Matrices Using Van Loans's Square Reduced Method. May 1996.

96-07 W. Rehm (Ed.). Portierbare numerische Simulation auf parallelen Architekturen. April 1996.

96-08 J. Weickert. Navier-Stokes equations as a differential-algebraic system. August 1996.

96-09 R. Byers, C. He, V. Mehrmann. Where is the nearest non-regular pencil? August 1996.

96-10 Th. Apel. A note on anisotropic interpolation error estimates for isoparametric quadrilateral finite elements. November 1996.

96-11 Th. Apel, G. Lube. Anisotropic mesh refinement for singularly perturbed reaction diffusion problems. November 1996.

96-12 B. Heise, M. Jung. Scalability, efficiency, and robustness of parallel multilevel solvers for nonlinear equations. September 1996.

96-13 F. Milde, R. A. Römer, M. Schreiber. Multifractal analysis of the metal-insulator transition in anisotropic systems. October 1996.

96-14 R. Schneider, P. L. Levin, M. Spasojević. Multiscale compression of BEM equations for electrostatic systems. October 1996.

96-15 M. Spasojević, R. Schneider, P. L. Levin. On the creation of sparse Boundary Element matrices for two dimensional electrostatics problems using the orthogonal Haar wavelet. October 1996.

96-16 S. Dahlke, W. Dahmen, R. Hochmuth, R. Schneider. Stable multiscale bases and local error estimation for elliptic problems. October 1996.

96-17 B. H. Kleemann, A. Rathsfeld, R. Schneider. Multiscale methods for Boundary Integral Equations and their application to boundary value problems in scattering theory and geodesy. October 1996.

96-18 U. Reichel. Partitionierung von Finite-Elemente-Netzen. November 1996.

96-19 W. Dahmen, R. Schneider. Composite wavelet bases for operator equations. November 1996.

96-20 R. A. Römer, M. Schreiber. No enhancement of the localization length for two interacting particles in a random potential. December 1996. to appear in: Phys. Rev. Lett., March 1997

96-21 G. Windisch. Two-point boundary value problems with piecewise constant coefficients: weak solution and exact discretization. December 1996.

96-22 M. Jung, S. V. Nepomnyaschikh. Variable preconditioning procedures for elliptic problems. December 1996.

97-01 P. Benner, V. Mehrmann, H. Xu. A new method for computing the stable invariant subspace of a real Hamiltonian matrix or Breaking Van Loan's curse? January 1997.

97-02 B. Benhammouda. Rank-revealing 'top-down' ULV factorizations. January 1997.

97-03 U. Schrader. Convergence of Asynchronous Jacobi-Newton-Iterations. January 1997.

97-04 U.-J. Görke, R. Kreißig. Einflußfaktoren bei der Identifikation von Materialparametern elastisch-plastischer Deformationsgesetze aus inhomogenen Verschiebungsfeldern. March 1997.

97-05 U. Groh. FEM auf irregulären hierarchischen Dreiecksnetzen. March 1997.

97-06 Th. Apel. Interpolation of non-smooth functions on anisotropic finite element meshes. March 1997

97-07 Th. Apel, S. Nicaise. The finite element method with anisotropic mesh grading for elliptic problems in domains with corners and edges.

97-08 L. Grabowsky, Th. Ermer, J. Werner. Nutzung von MPI für parallele FEM-Systeme. March 1997.

97-09 T. Wappler, Th. Vojta, M. Schreiber. Monte-Carlo simulations of the dynamical behavior of the Coulomb glass. March 1997.

97-10 M. Pester. Behandlung gekrümmter Oberflächen in einem 3D-FEM-Programm für Parallelrechner. April 1997.

97-11 G. Globisch, S. V. Nepomnyaschikh. The hierarchical preconditioning having unstructured grids. April 1997.

97-12 R. V. Pai, A. Punnoose, R. A. Römer. The Mott-Anderson transition in the disordered one-dimensional Hubbard model. April 1997.

97-13 M. Thess. Parallel Multilevel Preconditioners for Problems of Thin Smooth Shells. May 1997.

97-14 A. Eilmes, R. A. Römer, M. Schreiber. The two-dimensional Anderson model of localization with random hopping. June 1997.

97-15 M. Jung, J. F. Maitre. Some remarks on the constant in the strengthened C.B.S. inequality: Application to h- and p-hierarchical finite element discretizations of elasticity problems. July 1997.

97-16 G. Kunert. Error estimation for anisotropic tetrahedral and triangular finite element meshes. August 1997.

97-17 L. Grabowsky. MPI-basierte Koppelrandkommunikation und Einfluß der Partitionierung im 3D-Fall. August 1997.

97-18 R. A. Römer, M. Schreiber. Weak delocalization dueto long-range interaction for two electrons in a random potential chain. August 1997.

97-19 A. Eilmes, R. A. Römer, M. Schreiber. Critical behavior in the two-dimensional Anderson model of localization with random hopping. August 1997.

97-20 M. Meisel, A. Meyer. Hierarchically preconditioned parallel CG-solvers with and without coarse-matrix-solvers inside FEAP. September 1997.

97-21 J. X. Zhong, U. Grimm, R. A. Römer, M. Schreiber. Level-Spacing Distributions of Planar Quasiperiodic Tight-Binding Models. October 1997.

97-22 W. Rehm (Ed.). Ausgewählte Beiträge zum 1. Workshop Cluster-Computing. TU Chemnitz, 6./7. November 1997.

97-23 P. Benner, Enrique S. Quintana-Ortí. Solving stable generalized Lyapunov equations with the matrix sign function. October 1997

97-24 T. Penzl. A Multi-Grid Method for Generalized Lyapunow Equations. October 1997

97-25 G. Globisch. The hierarchical preconditioning having unstructured threedimensional grids. December 1997

97-26 G. Ammar, C. Mehl, V. Mehrmann. Schur-like forms for matrix Lie groups, Lie algebras and Jordan algebras. November 1997

97-27 U. Elsner. Graph partitioning - a survey. December 1997.

97-28 W. Dahmen, R. Schneider. Composite Wavelet Bases for Operator Equations. December 1997.

97-29 P. L. Levin, M. Spasojević, R. Schneider. Creation of Sparse Boundary Element Matrices for 2-D and Axi-symmetric Electrostatics Problems Using the Bi-orthogonal Haar Wavelet. December 1997.

97-30 W. Dahmen, R. Schneider. Wavelets on Manifolds I: Construction and Domain Decomposition. December 1997.

97-31 U. Elsner, V. Mehrmann, F. Milde, R. A. Römer, M. Schreiber. The Anderson Model of Localization: A Challenge for Modern Eigenvalue Methods. December 1997.

98-01 B. Heinrich, S. Nicaise, B. Weber. Elliptic interface problems in axisymmetric domains. Part II: The Fourier-finite-element approximation of non-tensorial singularities. January 1998.

98-02 T. Vojta, R. A. Römer, M. Schreiber. Two interfacing particles in a random potential: The random model revisited. February 1998.

98-03 B. Mehlig, K. Müller. Non-universal properties of a complex quantum spectrum. February 1998.

98-04 B. Mehlig, K. Müller, B. Eckhardt. Phase-space localization and matrix element distributions in systems with mixed classical phase space. February 1998.

98-05 M. Bollhöfer, V. Mehrmann. Nested divide and conquer concepts for the solution of large sparse linear systems. March 1998.

98-06 T. Penzl. A cyclic low rank Smith method for large, sparse Lyapunov equations with applications in model reduction and optimal control. March 1998.

98-07 V. Mehrmann, H. Xu. Canonical forms for Hamiltonian and symplectic matrices and pencils. March 1998.

98-08 C. Mehl. Condensed forms for skew-Hamiltonian/Hamiltonian pencils. March 1998.

98-09 M. Meyer. Der objektorientierte hierarchische Netzgenerator Netgen69-C++. April 1998.

98-10 T. Ermer. Mappingstrategien für Kommunikatoren. April 1998.

98-11 D. Lohse. Ein Standard-File für 3D-Gebietsbeschreibungen. – Definition des Fileformats V 2.1 –. April 1998.

98-13 L. Grabowsky, T. Ermer. Objektorientierte Implementation eines PPCG-Verfahrens. April 1998.

98-14 M. Konik, R. Schneider. Object-oriented implementation of multiscale methods for boundary integral equations. May 1998.

98-15 W. Dahmen, R. Schneider. Wavelets with complementary boundary conditions - Function spaces on the cube. May 1998.

98-16 P. Hr. Petkov, M. M. Konstantinov, V. Mehrmann. DGRSVX and DMSRIC: Fortran 77 subroutines for solving continuous-time matrix algebraic Riccati equations with condition and accuracy estimates. May 1998.

98-17 D. Lohse. Ein Standard-File für 3D-Gebietsbeschreibungen. - Datenbasis und Programmschnittstelle `data_read`. April 1998.

98-18 A. Fachat, K. H. Hoffmann. Blocking vs. Non-blocking Communication under MPI on a Master-Worker Problem. June 1998.

98-19 W. Dahmen, R. Schneider, Y. Xu. Nonlinear Functionals of Wavelet Expansions - Adaptive Reconstruction and Fast Evaluation. June 1998.

98-20 M. Leadbeater, R. A. Römer, M. Schreiber. Interaction-dependent enhancement of the localisation length for two interacting particles in a one-dimensional random potential. June 1998.

98-21 M. Leadbeater, R. A. Römer, M. Schreiber. Formation of electron-hole pairs in a one-dimensional random environment. June 1998.

98-22 A. Eilmes, U. Grimm, R. A. Römer, M. Schreiber. Two interacting particles at the metal-insulator transition. August 1998.

98-23 M. Leadbeater, R. A. Römer, M. Schreiber. Scaling the localisation lengths for two interacting particles in one-dimensional random potentials. July 1998.

98-24 M. Schreiber, U. Grimm, R. A. Römer, J. X. Zhong. Energy levels of quasi-periodic Hamiltonians, spectral unfolding, and random matrix theory. July 1998.

98-25 V. Mehrmann, H. Xu. Lagrangian invariant subspaces of Hamiltonian matrices. August 1998.

98-26 B. Nkemzi, B. Heinrich. Partial Fourier approximation of the Lamé equations in axisymmetric domains. September 1998.

98-27 V. Uski, B. Mehlig, R. A. Römer, M. Schreiber. Smoothed universal correlations in the two-dimensional Anderson model. September 1998.

98-28 D. Michael, M. Meisel. Some remarks to large deformation elasto-plasticity (continuum formulation). September 1998.

98-29 V. Mehrmann, H. Xu. Structured Jordan Canonical Forms for Structured Matrices that are Hermitian, skew Hermitian or unitary with respect to indefinite inner products. October 1998.

The complete list of current and former preprints is available via
`http://www.tu-chemnitz.de/sfb393/preprints.html`.