PREPRINT
NOT FOR REVIEW

ma
the
ma
tisch

cen
trum

MC

AFDELING NUMERIEKE WISKUNDE
(DEPARTMENT OF NUMERICAL MATHEMATICS)

NW 65/79 MEI

P.W. HEMKER

ON THE STRUCTURE OF AN ADAPTIVE MULTI-LEVEL ALGORITHM

Preprint

amsterdam 1979

5794. 820

stichting
mathematisch
centrum

MC

AFDELING NUMERIEKE WISKUNDE
(DEPARTMENT OF NUMERICAL MATHEMATICS)

NW 65/79 MEI

P.W. HEMKER

ON THE STRUCTURE OF AN ADAPTIVE MULTI-LEVEL ALGORITHM

Preprint

2e boerhaavestraat 49 amsterdam

BIBLIOTHEEK MATHEMATISCH CENTRUM
AMSTERDAM

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

AMS(MOS) subject classification scheme (1970): 65B05, 65N05, 65Q05, 65R05

On the structure of an Adaptive Multi-Level Algorithm *

by

P.W. Hemker

ABSTRACT

The structure of Multi-Level Adaptive Algorithms is explained. Their recursive character is exposed by means of the formal language ALGOL 68. The basic structure is given in two forms: coarse grid corrections to fine grids and fine grid corrections to coarse grids. The latter description is more convenient for the treatment of fully automatic adaptive grids. A new description of the FAS FMG algorithm of Brandt is given and the last section concludes with an ALGOL 68 procedure for the solution of boundary value problems with adaptive mesh-refinement.

KEY WORDS & PHRASES: *Multi-Grid Algorithms, Multi-Level Technique, Adaptive mesh-selection, Defect Correction Principle.*

---

* This report will be submitted for publication elsewhere.

1. INTRODUCTION

In recent papers, BRANDT [1977, 1978, 1979] considers Multi-Level Adaptive Techniques (MLAT) for the computation of solutions to partial differential equations. The essential feature of these techniques is that the solution of one continuous problem on a region Ω is obtained by different discretizations of the problem on a hierarchy of finer and finer grids. The accuracy of the numerical solution finally obtained corresponds to the accuracy that can be attained with the discretization on the finest grid in the hierarchy and the iterative process for the solution of the discrete system on the finest grid largely benefits from the interaction with the discretization on the coarser ones.

There is overwhelming evidence (cf. e.g. BRANDT [1977a], MIRANKER [1977], NICOLAIDES [1978]), that the iterative method thus obtained is among the most efficient ones for a very large class of linear and non-linear problems. The point is that the apparent redundancy of introducing the coarser grids appears to be profitable when the final efficiency of the iterative process is considered. It is an additional advantage of the Multi-Grid approach that it allows for adaptive local grid-refinement, i.e. during the computation a mesh-refinement can be generated in those parts of Ω where this refinement is required for the accurate representation of the solution.

These are all arguments in favour of the Multi-Grid (MG-)method, but a disadvantage is that its implementation is rather intricate (cf. BRANDT 1977b). A reason for this is that one wants a program to be efficient both with respect to computer storage and computer operations. Another reason may be that the fundamental MG-algorithm has a much more complex structure than most discretization and iterative solution methods. In this paper it is our intention to clarify this basic structure of multi-level algorithms, such that it will be easier in future both to analyze them mathematically and to implement them in a well structured way.

In Section 2 we give the basic principles of the MG-algorithm and we give an ALGOL 68 representation of it. In Section 3 we give an ALGOL 68 representation of the FAS FMG algorithm of BRANDT [1979].

In Section 4 we give the dual representation of the MG-algorithm and in Section 5 we give a representation of the algorithm in the case of locally refined meshes and adaptive mesh-refinement. Further we indicate how adaptive order selection can be implemented.

2. BASIC PRINCIPLES

It is not our intention to give here the heuristic background or a theoretical account for the MLAT-technique. For this we refer to BRANDT [1977a] or the clear and comprehensive description in BRANDT [1979]. However, we give here an overview of the essentials of the theory such that some insight can be obtained into the structure of the algorithm.

We consider the approximate solution of the (*nonlinear*) operator equation

$$Lu = f \tag{2.1}$$

on a domain Ω with boundary $\delta\Omega$. The domain Ω is covered by a sequence of finer and finer grids $\mathbb{G}_0,\mathbb{G}_1,\ldots,\mathbb{G}_k,\ldots$ with mesh width $h_0,h_1,\ldots,h_k$. Usually $h_{j+1} = h_j/2$. There exist operators I_h^H (the prolongation from $\mathbb{G}^H$ to $\mathbb{G}^h$) and I_H^h (the restriction from $\mathbb{G}^h$ to $\mathbb{G}^H$) which transfer grid functions defined on a coarse grid $\mathbb{G}^H$ to grid-functions defined on the next finer grid $\mathbb{G}^h$ (vice versa), so that $I_H^h\, I_h^H$ is the identity operator on $\mathbb{G}^H$.

If, on the grid $\mathbb{G}^h$, we have an approximate solution y^h to the $\mathbb{G}^h$-discretization of (2.1)

$$L^h u^h = f^h, \tag{2.2}$$

then we can get better approximations, i.e. approximations with smaller residuals

$$z^h = f^h - L^h y^h, \tag{2.3}$$

either be relaxation methods on the grid $\mathbb{G}^h$ (Jacobi, Gauss-Seidel, SOR etc.), which are efficient for reducing the high-frequency components of

the residual, or by solving the residual equation on a coarser grid $\mathbb{G}^H$; i.e. solving

$$L^H U^H = I_H^h f^h - I_H^h L^h y^h + L^H I_H^h y^h, \tag{2.4}$$

using $I_H^h y^h$ as a starting approximation. A better approximation to u^h is then given by

$$y^h + I_h^H (U^H - I_H^h y^h). \tag{2.5}$$

These coarse grid corrections are efficient for reducing the low-frequency components of the residual. Equation (2.2) now can be solved efficiently by using both methods successively (in any order, whichever - at a particular stage of the iteration process - is the more relevant with respect to the frequencies in the residual).

We can also consider the relation between both grids from another point of view. If we solve

$$L^H u^H = f^H, \tag{2.6}$$

we obtain a $\mathbb{G}^H$-approximation to the solution of (2.1). As usual, the truncation error, τ^H, of (2.6) is defined by

$$L^H I_H u = f^H + \tau^H, \tag{2.7}$$

where u denotes the solution of (2.1) and I_H denotes the restriction operator which restricts a continuous function on Ω to a grid function on $\mathbb{G}^H$. Analogously, the relative truncation error, τ_h^H, between $\mathbb{G}^H$ and $\mathbb{G}^h$ can be defined by

$$L^H I_H^h u^h = f^H + \tau_h^H, \tag{2.8}$$

where u^h is the solution of (2.2). Hence, once having obtained

$$\tau_h^H \approx L^H I_H^h y^h - I_H^h L^h y^h, \tag{2.9a}$$

```
MODE FUNCTION  = #some procedure mode which represents a continuous (vector-
                  valued) function IR^k → IR^n, where k is the dimension of Ω;
                  e.g. with k = 2, n = 1, we have
                  MODE FUNCTION = PROC (REAL, REAL) REAL#
MODE NET       = #some structure to represent a net function on G^h,
                  e.g. with k = 2, n = 1,
                  MODE NET = REF [,] REAL#
OP LH          =  (NET yh) NET:
                 #some representation of the operator L^h in eq. (2.2)#
OP RESTRICT    =  (NET yh) NET:
                 #some representation of the operator I_H^h#
OP PROLONGATE  =  (NET yh) NET:
                 #some representation of the operator I_h^H#
OP INTERPOLATE =  (NET yh) NET:
                 #some interpolation operator from G^H onto G^h, possibly
                  more accurate than I_h^H#
OP RESTRICTION =  (INT k, FUNCTION f) NET:
                 #some representation of the operator I_{h_k} (see eq. (2.7))#
FUNCTION f     = #some representation of the right hand side function f in
                  equation (2.1)#
PROC relax     =  (REF NET yh, NET rh,...) VOID:
                 #a procedure which performs one step of the relaxation
                  iteration of L^h y^h = r^h; in ... it delivers information
                  about the size of the residual and the speed of conver-
                  gence#
PROC solve directly
               =  (REF NET yh, NET rh) VOID:
                 #solves the discrete equation L^h y^h = r^h on the coarsest
                  grid G_0. Since L^h is essentially non-linear this procedure
                  in general needs an initial approximant to the solution
                  (as input in yh)#
```

Text 1. The modes, operators and procedures used in Text 1b.

The algorithmic structure is given in Text 1b in two procedures. solve upto level and solve at level; solve upto level is the driver procedure to

which the number of levels m and a start approximation on the zero-level grid $\mathbb{G}_0$ are given. It delivers the solution on a set of finer and finer grids $\mathbb{G}_0, \mathbb{G}_1, \ldots, \mathbb{G}_m$ that all cover the whole domain Ω. Most of the work is done by the recursive procedure solve on level, in which relaxation and coarse grid corrections interchange with each other in order to reduce the residual.

```
PROC solve upto level = (INT m, REF [] NET yh, NET start approximation) VOID:
BEGIN NET rh = 0 RESTRICTION f;
      yh[0] := start approximation;
      solve directly (yh[0],rh);
      FOR k TO m
      DO NET rh = k RESTRICTION f;
         yh[k] := INTERPOLATE yh[k-1];
         solve on level (k,yh[k],rh)
      OD
END;

PROC solve on level = (INT k, REF NET yh, NET rh) VOID:
IF   k = 0
THEN solve directly (yh,rh)
ELSE WHILE relax (yh, rh, converged, slow convergence);
           NOT converged
     DO    IF slow convergence
           THEN NET yhc := RESTRICT yh;
                NET rhc = RESTRICT rh
                    -RESTRICT LH yh + LH RESTRICT yh;
                solve on level (k-1,yhc,rhc);
                yh := yh +
                    PROLONGATE (yhc - RESTRICT yh)
           FI
     OD
FI;
```

Text 1b. The structure of the basic MG-algorithm.

(Legend to Text 1b)

The work done in the procedures is the following. First a starting approximation on the coarsest grid $\mathbb{G}_0$ is used to solve the problem on $\mathbb{G}_0$. Since $\mathbb{G}_0$ only contains a small number of points, we assume that this problem can be solved efficiently by some "direct method". This can be either the direct solution of a linear system for linear problems or the use of e.g. some Newton process for nonlinear problems. The solution on $\mathbb{G}_0$ is interpolated onto $\mathbb{G}_1$ to obtain a starting approximation on $\mathbb{G}_1$ and the problem is solved on $\mathbb{G}_1$. Thus we get a solution on level 1, which is interpolated onto $\mathbb{G}_2$, etc.. On each level k (k = 1,2,...,m) the problem is essentially solved by the successive use of relaxation sweeps (reducing the high-frequency components in the residual) and the computation of coarse grid corrections on level k-1 (reducing the low-frequency components in the residual). Thus, in [0:m] NET yh, one finally obtains the sequence $\{y^{h_j}\}_{j=0,\ldots,m}$ of solutions to the discrete problems $L^{h_j} y^{h_j} = f^{h_j}$ on $\mathbb{G}_j$, j = 0,...,m.

3. THE FAS FULL MULTI-GRID ALGORITHM

From the different variants of the MLAT algorithm (Cycle A, B and C in BRANDT [1976, 1977] for linear problems and Cycle C FAS (BRANDT 1977) and FAS FMG (BRANDT 1979) for nonlinear problems) the FAS FMG (full approximation storage, full multi-grid) algorithm shows the best complete structure. With the basic structure of Section 2 in mind we are now able to translate the FAS FMG flowchart into two ALGOL 68 procedures. This translation is given in Text 2. The algorithm given here is essentially the same as the one given in Text 1, but now some strategy has been implemented (1) to decide on slow convergence, and (2) to solve the discrete systems of equations with a residual error which is of the same order of magnitude as the truncation error.

Hence, in this algorithm, we meet three parameters η, α and δ in order to tune the strategy; η is a parameter which decides on slow convergence, its value depends on the difference scheme and the relaxation method used; α is a parameter which controls the factor by which a finer grid solution should be more accurate than a coarse grid solution, its value depends on

```
PROC solve upto level = (INT m, REF [] NET yh) VOID:
BEGIN REAL tolk := 0.0001 * alfa;
      solve directly (yh[0], 0 RESTRICTION f);
      FOR k TO m
      DO  NET rh := k RESTRICTION f;
          yh[k] := INTERPOLATE yh[k-1];
          tolk := tolk*alfa;
          solve on level (k,yh[k],rh,tolk,tolk)
      OD
END;

PROC solve on level = (INT k, REF NET yh, NET rh, REAL tol, REF REAL
                       nexttol) VOID:
BEGIN REAL error, olderror := maxreal;
      WHILE relax (yh, rh, error);
            error ≥ tol
      DO    IF k = 0 OR error/olderror ≤ eta
            THEN olderror := error
            ELSE NET yc := RESTRICT yh;
                 NET residual = rh - LH yh;
                 NET rc = LH yc + RESTRICT residual;
                 REAL tau = norm(rc - (k-1)RESTRICTION f);
                 REAL tolc := delta * error;

                 solve on level (k-1,yc,rc,tolc,tolc);

                 yh := yh + PROLONGATE (yc - RESTRICT yh);
                 nexttol := alfa * tau;
                 olderror := maxreal
            FI
      OD
END;
```

Text 2. The FAS FMG-algorithm.

An ALGOL 68 translation of the algorithm published by BRANDT [1979]. In this text the procedure relax(yh,rh,error) performs a relaxation iteration step in the solution of $L^h y^h = r^h$ and delivers in error an estimate of the norm of its residual: error $\approx \|r^h - L^h y^h\|$.

the mesh ratio and the order of accuracy of the discretization method; δ is a parameter which tunes the requirement that a coarse grid correction should not be computed much more accurate then the relaxation corrections with which it shares the task of reducing the residual. More details about these parameters can be found in BRANDT [1979].

4. THE DUAL REPRESENTATION OF THE MG-ALGORITHM

In the procedure `solve on level` in Section 2 the MG-algorithm can clearly be recognized from the point of view where coarse grid problems serve as fine grid correctors. Having obtained the fine grid solution on $\mathbb{G}_m$, we trivially can obtain $\mathbb{G}_m$-accurate solutions on lower levels by

```
FOR k FROM m BY -1 TO 1
DO y[k-1] := RESTRICT y[k] OD; .
```

However we can also reorganize the algorithm such that (i) these grid-functions are obtained immediately, and (ii) fine grid solutions can be recognized as generating corrections - in the form of corrected solutions and relative truncation errors - to the coarse grid solutions. To this end we have to turn the recursive part of the algorithm inside out. Thus we find the basic algorithm in its dual representation. *It computes the fine grid solution* `y[m]` *and its restrictions to coarser grids essentially the same way as in the representation in Text 1 in Section 2.*

The use of the dual representation is that it will enable us to introduce partial mesh-refinement (see Section 5). In that case finer grids do not extend over the whole range of Ω (the "fine grid solution" is not defined on Ω) but finer grids are introduced on parts of Ω only.

We can understand the recursive part of the procedure `correct for finer nets` in Text 3 by noting that for each call of the form

```
correct for finer nets (k,y,r)
```

we have

```
PROC solve upto level = (INT m, REF [] NET y, NET startapproximation) VOID:
BEGIN [0:m] NET r;
      FOR i TO m DO y[i] := NIL OD;
      y[0] := startapproximation;
      r[0] := 0 RESTRICTION f;
      WHILE solve directly (y[0],r[0]);
        NOT correct for finer nets (0,y,r)
      DO SKIP OD;
END;

PROC correct for finer nets = (INT k, REF [] NET y, r)BOOL:
IF   k = UPB r
THEN TRUE
ELSE BOOL converged; REAL olderror, error := maxreal;

     IF   NET (y[k+1]) :=: NET(NIL)
     THEN #enter level k+1 for the first time#
          y[k+1] := INTERPOLATE y[k];
          r[k+1] := (k+1) RESTRICTION f
     ELSE y[k+1] := y[k+1] + PROLONGATE (y[k] - RESTRICT y[k+1])
     FI;

     WHILE NOT
       IF   error := olderror; relax (y[k+1], r[k+1], error);
            converged := error < tolerence;
            converged
       THEN correct for finer nets (k+1, y, r)
       ELSE (error/olderror) ≤ eta #i.e. slow convergence#
       FI
     DO SKIP OD;

     y[k] := RESTRICT y[k+1];
     r[k] := RESTRICT r[k+1] - RESTRICT LH y[k+1]+ LH RESTRICT y[k+1];
     converged
FI;
```

Text 3. The dual representation of the basic MG-algorithm.
In this text the meaning of the procedure relax is the same as in Text 2.

(1) at entrance: the coarse grid (level k) solution has been found, since:

(4.1) norm(rh[k] - LH yh[k]) ≤ tolerence;

(2) on exit with the value TRUE:

y[m], y[j], r[j], j = k,k+1,...,m-1 have been changed, and

r[m], y[j], r[j], j = 0,1,...,k-1 have not;

y[m] contains the solution of the problem discretized on $\mathbb{G}_m$, moreover,

y[j] = RESTRICT y[j+1], j = k,k+1,...,m, (i.e. at all levels the $\mathbb{G}_m$-accurate solution is delivered);

r[j], j = k,k+1,...,m-1,m, contains the relative truncation error of the $\mathbb{G}_j$-discretization with respect to the $\mathbb{G}_m$-discretization;

(3) on exit with the value FALSE only

r[k], y[k] and y[k+1] have been changed and we shall not return at the level k before y[k] has been adapted - by coarse grid corrections - such that with the given r[k] the inequality (4.1) has been satisfied.

REMARK. Notice that the ALGOL 68 phrase "WHILE a; NOT b DO SKIP OD" can be understood as "REPEAT a UNTIL b".

5. PARTIAL AND ADAPTIVE GRID REFINEMENT

The discretization applied in connection with the MG-technique mostly uses uniform nets (either for finite difference or finite element methods). This implies that the set-up of the discrete equations is simple and requires very little administration of geometrical details. Mesh-refinement is obtained by adding finer and finer mesh-levels $\mathbb{G}_j$. If a partial mesh-refinement is sufficient, this refinement can be restricted to (smaller and smaller) parts of the original domain Ω. The only requirements are (i) that a few (e.g. two) coarse grids cover the original domain Ω and (ii) that the domain of each grid is covered by the domain of the next coarser one. A coarse-grid "father" can have different grid "sons", i.e. finer grids that extend over a subdomain. We restrict ourselves to the case where "brother"-grids (sons of the same father) extend over disjoint subregions of the father-region. Thus we obtain a tree-structure of net-functions that together form a grid-family. The data structure for such a grid

family is in ALGOL 68 easily represented by

```
MODE GRIDF = STRUCT (NET net, REF [] GRIDF sons).
```

Each object of the mode GRIDF contains a NET as in Section 2 and an array of references to its sons, which themselves are of the mode GRIDF again. If this array of sons is of length zero, there are no sons, i.e. there are no finer grids in the region covered by this GRIDF.

The only tool we further need in order to formulate our MG-algorithm on a grid-family instead of on a set of coextensive grids $\mathbb{G}_0, \mathbb{G}_1, \ldots, \mathbb{G}_m$, is an operator SUBREGION so that the expression gridson SUBREGION gridfather denotes the grid which is the restriction of the grid gridfather to the region that is covered by the grid gridson. Such an operator

```
OP SUBREGION = (GRIDF gridson, REF GRIDF gridfather)
                    REF GRIDF: ... ,
```

the details of which depend on the explicit structure of a NET, can be implemented in ALGOL 68 without difficulties. An auxiliary operator:

```
OP <:= = (REF GRID a, GRID b)VOID: net OF a := net OF b;
```

is convenient for the assignment of net-values of grid b to grid a.

With the aid of the mode GRIDF and the operators SUBREGION and <:=, we describe the MG-algorithm for an arbitrary grid-family by the two procedures in Text 4.

In order not to repeat the small complication of the first interpolation (initialization) on a new level for y, in the last procedure we have identified the two interpolation operators INTERPOLATE and PROLONGATE; a distinction between both operators would lead to an algorithmic structure for the initialization similar to the one in Section 4.

We notice that for a PDE the boundary conditions for a subregion

```
PROC solve = (REF GRIDF y, NET startapproximation)VOID:
BEGIN GRIDF r := y RESTRICTION f;
      initialize at zero all nets of the family (y);
      net OF y := startapproximation;

      WHILE solve directly (net OF y, net OF r);
            NOT correct for finer grids (y,r)
      DO SKIP OD
END;

PROC correct for finer grids = (REF GRIDF yhf, rhf)BOOL:
BEGIN BOOL ready := TRUE;

      FOR i TO UPB (sons OF yhf)
      DO REF GRIDF yh = (sons OF yhf)[i],
                   rh = (sons OF rhf)[i];
         yh := yh + PROLONGATE (yh SUBREGION yhf - RESTRICT yh)
         WHILE NOT
            IF   relax(net OF yh, net OF rh,...); converged
            THEN correct for finer nets (yh,rh)
            ELSE slow convergence
            FI
         DO SKIP OD;

         yh SUBREGION yhf <:= RESTRICT yh;
         rh SUBREGION rhf <:= RESTRICT rh
                                 - RESTRICT LH yh + LH RESTRICT yh;
         ready := ready AND converged
      OD; ready
END;
```

Text 4. The MG-algorithm for tree-structured partially refined grids. This algorithm solves the equation Lu = f on the trees of grids called GRIDF y. In this procedure the operators PROLONGATE, RESTRICT and LH, working on a GRIDF gridf, have the same effect on net OF gridf as their analogs working on a NET in the previous sections. The expression gridf RESTRICTION f creates a grid-family similar to (congruent with) gridf and initializes all its net-functions with the corresponding values of the FUNCTION f.

$\Omega_\alpha \subset \Omega$ are determined by either a part of the boundary conditions of Ω itself (if Ω_α is partly along the boundary of Ω) or by the most recently obtained approximation to the solution on its gridfather (which - at convergence - takes the same values as the gridson itself!)

In contrast with the representation given in Section 2, in the representation of the algorithms given in Section 4 and 5, at any time before the finest grid solution has finally been obtained, the gridfamily can be extended in any subregion and at any level, in order to adapt the grid to the solution. This means that both y and r should be extended in a similar way and that the offspring of y should be initialized at approximate values of the solution and the corresponding offspring of r should be initialized at the corresponding function values of f. In the course of the computation the algorithm automatically corrects all values at all levels.

We illustrate this in Text 5, which is part of an actual ALGOL 68 program which solves a (two-point) boundary-value problem. Beside the extension of the tree of grids, in this program - as in Text 2 - some strategy has been implemented to decide on convergence and slow convergence. The program has been used to find and to resolve boundary layers in linear and nonlinear problems of the form

$$\varepsilon y'' + N(x,y,y') = f \qquad \text{on } (a,b),$$

$$y(a) = ya, \quad y(b) = yb.$$

Numerical results of the MG-method for this problem will be reported later. (We do not mean that in practice two-point boundary-value problems should be solved by iterative means (!) but Text 5 can easily be extended for 2- and 3-dimensional problems.)

Beside adaptive grids it is also easily possible to implement adaptive order of convergence. To this end one should have an operator LH of which the order of accuracy of the discretization depends on the mesh-size and/or on the subregion of Ω where it is applied. We notice that the high order of accuracy of the solution is obtained by evaluation of the discrete operator LH on the finest grids only (i.e. only by the computation of relative truncation errors). There is no need to solve any more complex

algebraic system than for a low order discretization, since the discrete system to solve and the iterative method to solve it are not affected by the order of discretization. In this sense the method used is very similar to the method of deferred corrections (cf. STETTER 1978).

```
PROC solve = (REF GRIDF y, REAL tol)VOID:
BEGIN REAL tolc := 0.1, alfa := 0.25, eta := 0.75;
  PROC solve directly = (REF GRIDF y, r)VOID:
  BEGIN REAL error;
        WHILE relaxation(y, r, error);
          NOT (error < tol)
        DO SKIP OD
  END;
  PROC correct for finer grids = (REF GRIDF yhf, rhf)BOOL:
  BEGIN BOOL ready := TRUE;
    FOR i OF UPB (sons OF yhf)
    DO BOOL converged; GRIDF tau;
       REAL taunorm, error, olderror := maxreal;
       REF GRIDF yh = (sons OF yhf)[i],
                 rh = (sons OF rhf)[i];
       yh := yh + PROLONGATE(yh SUBREGION yhf - RESTRICT yh);
       WHILE
          WHILE NOT
             IF   relaxation(yh, rh, error);
                  converged := error < tolc
             THEN correct for finer grids (yh, rh)
             ELSE (error/olderror) > eta #i.e. slow convergence#
             FI
          DO olderror := error OD;
          tau := LH RESTRICT yh - RESTRICT LH yh;
          IF   UPB (sons OF yh) ≠ 0
          THEN FALSE
          ELIF taunorm := NORM tau;
               (tolc := taunorm * alfa; tolc < tol | tolc := tol);
```

```
            error > tolc
        THEN converged
        ELIF converged := TRUE;
            taunorm * alfa ** 2 > tol
        THEN create sons(yh, tau, tol/alfa ** 2);
            rh := yh RESTRICTION f;
            (tolc *:= alfa; tolc < tol | tolc := tol);

            NOT correct for finer levels (yh, rh)
        ELSE FALSE
        FI
      DO olderror := error OD;

      yh SUBREGION yhf <:= RESTRICT yh;
      rh SUBREGION rhf <:= RESTRICT rh + tau;

      ready := ready AND converged
  OD; ready
 END;

GRIDF r := y RESTRICTION f;
WHILE solve directly (y, r);
   NOT correct for finer grids (y,r)
DO SKIP OD

END;
```

Text 5. An ALGOL 68 procedure for a Multi-Level Algorithm with Adaptive Mesh-refinement for the solution of Lu = f.
A solution y is obtained such that $\|Ly - f\| < tol$. The procedure call create sons (yh, tau, t) creates sons for the GRIDF yh on those parts of Ω where net OF tau > t, i.e. where the values of the netfunction tau are greater than a given value t.

ACKNOWLEDGEMENT

The author is grateful to Dr. H. te Riele and H. Schippers for the careful reading of the manuscript and for many helpful remarks.

REFERENCES

BRANDT, A., (1977a), *Multi-Level Adaptive Solutions to Boundary Value Problems,* Math. Comp. 31 (1977) 333-390.

BRANDT, A., (1977b), *Multi-Level Adaptive Solutions to Partial Differential Equations - Ideas and Software,* Proceedings of Symposium on Mathematical Software. (Mathematics Research Center, University of Wisconsin, March 1977), (John Rice, ed.) pp. 277-318, Academic Press, New York.

BRANDT, A., (1979), *Multi-Level Adaptive Techniques (MLAT) for Singular Perturbation Problems.* (In: Numerical Analysis of Singular Perturbation Problems, P.W. Hemker and J.J.H. Miller eds., Academic Press, London, 1979.)

MIRANKER, W.L., (1977), *Hierarchical relaxation,* IBM Res. Rept. RC 6884.

NICOLAIDES, R.A., (1978), *On the observed rate of convergence of an iterative method applied to a model elliptic difference equation,* Math. Comp. 32 (1978) 127-133.

STETTER, H.J., (1978), *The Defect Correction Principle and Discretization Methods,* Num. Math. 29 (1978) 425-443.

VAN WIJNGAARDEN et al. Eds. (1976), *Revised Report on the Algorithmic Language ALGOL 68,* Springer-Verlag, New York, Heidelberg, Berlin (1976).