Numerical Linear Algebra with Applications,

Downwind Gauß-Seidel Smoothing for Convection Dominated Problems [1]

Wolfgang Hackbusch, Thomas Probst

Praktische Mathematik
Christian-Albrechts-Universität zu Kiel
D-24098 Kiel, Germany

In the case of convection dominated problems, multi-grid methods require an appropriate smoothing to ensure robustness. As a first approach we discuss a Gauß-Seidel smoothing with a correct numbering of the unknowns and if necessary a special block partitioning. Numerical experiments show that, in the case of general convection directions, the multi-grid algorithms obtained in this way have the same properties as in the model situation. If the graph arising from the convection part is acyclic, we describe a numbering algorithms which is valid for all spatial dimensions. Cycles give rise to special blocks for a blockwise Gauß-Seidel smoothing. We describe an algorithm for the two-dimensional case. The proposed algorithms require a computational work of optimal order (linear in the size of the problem).

Subject Index. AMS 05C20, 05C38, 05C85, 65N30, 65N55

KEY WORDS multi-grid, dominating convection, robustness, planar graphs, cycles

1. Introduction

1.1. The Convection-Diffusion Equation

A typical feature of flow problems is the dominating convection. A corresponding model equation is the convection-diffusion equation

$$-\varepsilon\Delta u + \langle b, \operatorname{grad} u\rangle = f \tag{1.1}$$

with a positive but small ε subject to certain boundary conditions. The equation is elliptic, but the hyperbolic character appears when ε approaches zero. It is wellknown that the discretisation of (1.1) requires special care in order to produce stable systems (cf. Chap. 10.2 in [4]). In the following we assume that the principal part is discretised by standard

[1] submitted to *Numerical Linear Algebra with Applications.*

1070–5325/93/010001–07$8.50

Received
Revised

linear finite elements in a triangulation of the domain. Concerning the convection term, we consider the simplest upwind formulation.

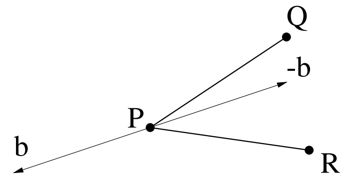

Figure 1. upwind scheme

Let P (see Fig. 1) be the nodal point for which the stiffness entries are to be defined. If, by accident, the direction $-b$ coincides with a direction PQ of a triangle side, $\langle b, \operatorname{grad} u\rangle$ is approximated by the upwind difference $d_{PQ} := |b|\frac{u(P)-u(Q)}{|PQ|}$. In the general case, $-b$ lies between two sides PQ and PR of the triangle PQR. Then, $\langle b, \operatorname{grad} u\rangle$ is a linear combination of d_{PQ} and d_{PR} with positive coefficients (cf. [7]).

1.2. The Multi-Grid Solution

Although ellipticity ensures a multi-grid convergence rate independent of the dimension (grid size), the rate may depend on the size of ε. Therefore, all singular perturbation problems like (1.1) give rise to ask for *robust* multi-grid versions, whose convergence rate is bounded independently of the perturbation parameter ε. The convection-diffusion equation is discussed since the beginning of multi-grid methods (cf. [2], [8]). However, the discussion is more or less focussed to the difference method in a regular grid under the assumption that the direction b lies in a certain angle interval. If, e.g., the x-component of b is non-negative, the lexicographical ordering is appropriate for the Gauß-Seidel smoothing, since in the limit case of $\varepsilon = 0$, the matrix is lower triangular and therefore the Gauß-Seidel iteration becomes an exact solver. Analogously, one needs an anti-lexicographical ordering, if b has a negative x-component. Thus, the appropriate ordering must follow the downwind direction. Difficulties arise when the direction b is a variable coefficient with quite different orientations in the domain. Another difficulty appears when the regular grid of the difference method is replaced by a general finite-element triangulation. In this case, it is not at all obvious what the counterpart of the simple lexicographical ordering in the regular finite-difference case is.

We mentioned that in the limit case $\varepsilon = 0$, the Gauß-Seidel smoothing with downwind numbering becomes an exact solver. If ε is of the size $\mathcal{O}(1)$, the coarse-grid correction is well working and ensures good convergence rates. However, with decreasing ε the coarse-grid correction becomes less effective since due to the diminishing ellipticity the correction can be assumed to smooth only in the convection direction b, but not normal to b (a possible remedy is the frequency decomposition multi-grid version, see [3]).

A very promising approach to problems including Eq. (1.1) is the Schur-complement multi-grid version of Reusken [10]. In the future, we want to combine our numbering techniques with the algorithm of Reusken.

1.3. Numbering and Block Partitioning

The main subject of this paper is the presentation of the numbering techniques. If no cycles arise, one gets a perfect ordering for the *pointwise* Gauß-Seidel process. This case is discussed in §2. §3 is devoted to the more general case when cycles appear. We present efficient algorithms which combine the numbering with the construction of block partitionings. The blocks are simple enough to allow the exact solution of the diagonal block. Therefore, the *block*-Gauß-Seidel smoothing is cheaply performable. The properties of this smoothing procedure are discussed in §4. Other choices of block partitionings are mentioned in §5.

Although we only discuss the Gauß-Seidel smoothing, these results are also of great interest for the incomplete LU smoothers since they depend in the similar way on an appropriate ordering of the unknowns.

2. Downwind Numbering

2.1. Graph of the Convection Matrix

Before posing the problem, we recall the definition of the graph of a matrix (cf. §6.2 in [5]). We consider a *linear system* $A\,x = b$ with matrix entries a_{PQ} and unknowns x_P, where P is in the *index* set V. Due to the upwind difference scheme we define an *inverse* matrix graph which is opposite to the usual one: The matrix graph $G = G(A)$ consists of the *vertex set* $V = V(G)$ and the *edges* $E = E(G) = \{(P,Q) \in V \times V \ : \ a_{QP} \neq 0\}$. This notation ensures that the edges E point in the same direction as the convection does, in this case from P to Q. All graphs used in the following are *directed* graphs.

Let the matrix A arise from Eq. (1.1) by the discretisation described in §1. When we restrict our considerations to the scalar case and linear finite elements (in triangles or tetrahedra, respectively), the vertices of $V(G)$ correspond to the vertices of the (finite) elements, while the edges in G are the edges of the elements. Correspondingly, discretisations of problems in two spatial dimensions lead to a *planar* graph G.

Because of its symmetry, the diffusion term $-\varepsilon\Delta u$ yields symmetric edges (i.e., edges in both directions), whereas the convection term $\langle b, \operatorname{grad} u\rangle$ discretised as above generates edges with only one direction.

2.2. Strong Convection

As long as the convection is moderate (i.e., b/ε of size $\mathcal{O}(1)$ or smaller), multi-grid methods with simple smoothing iterations are successful (cf. [2]). When, however, convection becomes dominant, the convergence may become worse and good convergence rates require a certain ordering of unknowns. Several proposals for "robust smoothers" have been made (cf. §10.4 in [2]). We now follow another line: A simple Gauß-Seidel method leads to excellent results if the ordering of the unknowns follows the convection direction (the downwind direction $b \in \mathbb{R}^2$).

The ideal case would be that the unknowns can be ordered so that x_i depends only on x_j with $j < i$. In that case, A forms a lower triangular matrix and one Gauß-Seidel step is an exact solver. This ideal situation corresponds to an acyclic graph G. Here, we use the following notation. The tuple $(P, Q, R, \ldots, Y, Z)$ is a *path* in G, if $P, \ldots, Z \in V(G)$ and $(P,Q), (Q,R), \ldots, (Y,Z) \in E(G)$. The path is a *cycle*, if $P = Z$. G is *acyclic*, if no cycles exist in G.

The matrix A of the discrete version of problem (1.1) has a natural splitting into to $A^D + A^C$, where the terms correspond to the diffusion and convection part of Eq. (1.1) . Since the multi-grid method works well without or with moderate convection, one has not to care about A^D and it suffices to consider the graph of A^C.

In order to suppress small convection, one can define the graph of dominant convection by edges $(P,Q) \in V \times V$ with

$$|a^C_{QP}| > \kappa |a^D_{QP}| \tag{2.1}$$

where κ may depend on the mesh size. It is also reasonable to consider only the strongest connections:

$$(P,Q) \in E(G) \Leftrightarrow (2.1) \quad \text{holds and} \quad |a^C_{QP}| \geq \frac{1}{2} \max\{|a^C_{QR}| : |a^C_{QR}| > \kappa |a^D_{QR}|\} \tag{2.2}$$

This defines the graph G that we discuss in the following. If G is acyclic, the ordering mentioned above (and constructed in §2.1) does not lead to an exact solver, but the multi-grid method with Gauß-Seidel smoothing works very fast, as demonstrated in §2.4.

The sparsity of the finite element matrix corresponds to a uniformly limited degree of the vertices of the graph.

Remark 2.1. *It is a characteristic requirement for finite element triangulations that the number of edges associated to a nodal point (the degree of the vertex) is bounded by a constant (the usual formulation is: the angle of the triangles must be* $\geq \alpha > 0$, *cf. Theorem 8.4.4 in [4]).*

2.3. *Numbering Algorithm*

In the following, we make no assumptions on G except that

$$\text{G is finite.} \tag{2.3}$$

In particular, G may not be planar. The algorithm described below has two purposes. Firstly, it determines a suitable ordering of the vertices $V(G)$ if G is acyclic. Secondly, in the general case, we characterise those vertices involved in cycles.

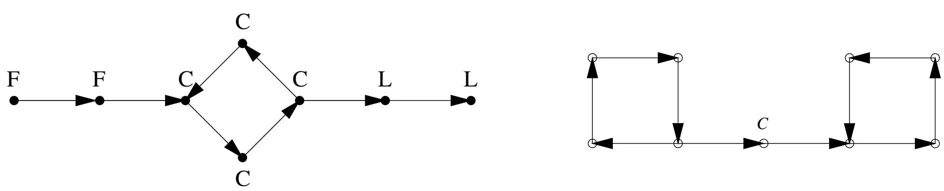

Figure 2. Attributes {F,L,C}

We recall the aim of the numbering procedure. We would like to number a vertex P before Q, if the value at Q depends on that at P ($a_{QP} \neq 0$). Therefore, the associated numbers have to increase in the direction of the edges $(P,Q) \in E(G)$.

The following Pascal-like procedures have vertices $P \in V(G)$ as parameters and determine attributes. The values of $P.attr$ are in the set $\{F, L, C\}$. Attribute F [or L] means that these vertices are to be numbered first [or last], whereas C-vertices are associated with cycles (cf. Fig. 2). The precise (recursive) definition is as follows: $P \in V(G)$ has attribute F $[L]$ if all predecessor [successors] have attribute F $[L]$. Here, Q is called *successor* [*predecessor*] of P if $(P, Q) \in E(G)$ $[(Q, P) \in E(G)]$.

After the initialisation (2.5a) the call (2.5b) follows:

```
for P ∈ V(G) do P.attr :=C;                                          (2.5a)
for P ∈ V(G) do SetAttr(P);                                          (2.5b)
```

The procedure SetAttr is composed by SetF and SetL:

```
procedure SetF(P);                                                   (2.6a)
begin if "all predecessors of P have attribute F" then               (2.6b)
      begin  P.attr := F;                                            (2.6c)
             for Q∈ {successor of P} do if Q.attr=C then SetF(Q)     (2.6d)
      end    end;

procedure SetL(P);                                                   (2.7a)
begin if "all successors of P have attribute L" then                 (2.7b)
      begin  P.attr := L;                                            (2.7c)
             for Q∈ {predecessor of P} do if Q.attr=C then SetL(Q)   (2.7d)
      end    end;

procedure SetAttr(P);                                                (2.8a)
begin if P.attr = C then SetF(P);                                    (2.8b)
      if P.attr = C then SetL(P)                                     (2.8c)
     end;
```

The following conclusions are trivial:

Remark 2.2.
a) The amount of work is proportional to $\#G := \#V(G) + \#E(G)$.

Families of sparse matrices lead to graphs G *with* $\#E(G) < const \cdot \#V(G)$, *where the constant does not depend on the size of* $V(G)$.

b) Let $P_1, \ldots, P_i$ *be the vertices with attribute* F *in the ordering as they are found by SetF. Then all predecessors* Q *of* P_p, $1 < p < i$, *have a smaller index:* $Q = P_q$ *with* $q < p$.

c) Let $P_j, \ldots, P_n$ $(n = \#V(G))$ *be the vertices with attribute* L *in the reverse ordering as they are found by SetL. Then all successors* Q *of* P_p , $j < p < n$, *have a larger index:* $Q = P_q$ *with* $q > p$.

d) If G *is acyclic, no vertex has attribute* C. *Either all vertices have attribute F* $(i = n)$ *or* L $(j = 1)$ *and* $P_1, \ldots, P_n$ *is the desired ordering.*

e) On the other hand, if G *has cycles, there must be vertices with attribute* C. *Each* $P \in V(G)$ *with P.attr=C has at least one successor and at least one predecessor of the same attribute.*

We note that, in general, the resulting attributes F and L are not unique and may depend on the (arbitrary) ordering in the do-loop in (2.5b).

As mentioned in Remark 2.2.d), the algorithm (2.5a, 2.5b) not only determines the numbering, but also checks whether G is acyclic. If G turns out to be cyclic, we can apply the techniques of §§3-4.

2.4. *Numerical Examples: Computational Expense and Rates of Convergence*

The last paragraph has shown that the computational work, which is required to find the right order, grows linearly with the size of the graph. This section defines an acyclic test problem and describes our implementation. It is shown that the expense of cpu-time for the determination of the right ordering is less than the cpu-time for 3 Gauß-Seidel iteration steps, which are a common work unit. Furthermore we state convergence rates for the multi-grid method with Gauß-Seidel smoothers.

Let $\Omega = (0,1)^2 \subset \mathbb{R}^2$ be the domain, $\Gamma := \partial\Omega$ its boundary and

$$L = -\varepsilon\Delta + b_x(x,y)\frac{\partial}{\partial x} + b_y(x,y)\frac{\partial}{\partial y} + a(x,y)$$

a second order differential operator.

We discretise the partial differential equation $Lu = f$ with Dirichlet boundary condition $u|_\Gamma = \varphi$. For our test problems we choose $v(x,y) := x^2 + y^2$, $\varphi := v|_\Gamma$ and $f := Lv$. In this case $u = v$ is the exact solution of the continuous problem and we can control the quality of the discrete solution. We use a standard 5-point finite difference stencil with an upwind scheme on a regular (i.e., equidistant) grid (cf. Chap. 5.1.4 in [4]). This gives a system $L_h u_h = f_h$ of linear equations with an M-matrix L_h and a discrete right-hand side f_h.

As a test problem we choose the flow field like a labyrinth with 6 turns and $a(x,y) := 0$, cf. Fig. 3 b). The domain Ω is divided in 25 squares and in each square the convection (b_x, b_y) is defined separately. We call this test problem LABYRINTH. In order to compare the convergence rates with a standard problem, we define the problem XLINE with $b(x,y) = (1,0)$, i.e. flow in x-direction.

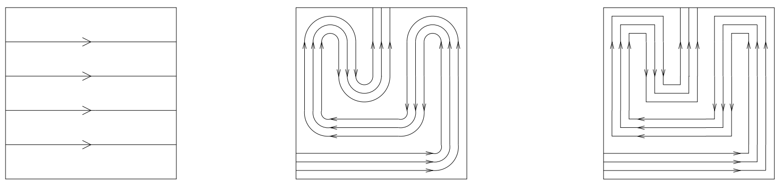

Figure 3. a) XLINE b) LABYRINTH c) STRAIGHT LABYRINTH

2.4.1. Computational Expense Our implementation uses a double linked list to store the vertices and the matrix entries. Thus the ordering of the vertices is given by their position in this list.

Since the ordering algorithm contains nearly no arithmetic operations, we compare the expense by comparing the time for Gauß-Seidel steps and for the ordering. The code was written in C and run on a SUN Sparc 10 workstation.

The numbering procedure consists of three parts: it starts with the definition of the graph $E(G)$ according to (2.1) and (2.2). This step costs about 0.7 to 1.0 times the time of a Gauß-Seidel step.

The next task is the determination of the attributes $\{L, F, C\}$ by algorithm (2.5). The procedures SetF and SetL find the vertices in the desired ordering (cf. Remark 2.2. b,c)). The determination of the attributes is relatively cheap (ca. 0.2 Gauß-Seidel steps), the following reordering of the list and specifying of the upper and lower triangular part of the matrix costs about 0.7 to 0.9 Gauß-Seidel steps. Adding the expense for the ordering, one gets less than 2.2 Gauß-Seidel steps as the total amount of time.

2.4.2. Rates of Convergence In order to compare our test problem with the test problem XLINE, in which the convection follows exactly the grid lines, we define another test problem: STRAIGHT LABYRINTH is similar to the LABYRINTH, but has flow only parallel to grid lines (cf. Fig. 3 c). We present convergence rates for the three test problems for a W-cycle multi-grid method with 2 symmetric Gauß-Seidel smoothing iterations in lexicographical ordering (denoted by GS) and 4 forward Gauß-Seidel steps in downwind ordering (DGS).

Table 1. convergence rates for acyclic flow

Problem	h	Smoother	$\varepsilon = 1$	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-5}$
XLINE	h= 1/64	GS	0.0651	0.0965	0.0001
		DGS	0.0634	0.0746	$< 10^{-6}$
	h= 1/128	GS	0.0625	0.105	0.001
		DGS	0.0611	0.092	$< 10^{-5}$
	h= 1/256	GS	0.0606	0.0977	0.0096
		DGS	0.0589	0.0931	0.0002
STRAIGHT LABYRINTH	h= 1/64	GS	0.0645	0.2403	0.0414
		DGS	0.0547	0.1900	$< 10^{-4}$
	h= 1/128	GS	0.0622	0.2010	0.0548
		DGS	0.0537	0.1693	0.0030
	h= 1/256	GS	0.0604	0.0949	0.1135
		DGS	0.0510	0.0898	0.0452
LABYRINTH	h= 1/64	GS	0.0646	0.1592	0.0994
		DGS	0.0571	0.1457	0.0005
	h= 1/128	GS	0.0622	0.1038	0.1499
		DGS	0.0546	0.0941	0.0064
	h= 1/256	GS	0.0604	0.0920	0.3846
		DGS	0.0530	0.0875	0.0730

Although Fig. 3b) suggests a cyclefree flow, the matrix graph G might contain cycles: Due to the finite difference discretisation there might be more than one convection edge at one vertex: on a vertex where the flow doesn't follow a grid line (or a triangle side when finite elements are used) the standard upwind discretisation yields two matrix entries; there are flow directions where the edges are equally strong.

If one intends to build a cylcefree graph (in order to use algorithm (2.5), which is easy to implement, and not the method of cycle detection of §3), a remedy is to replace the factor $1/2$ in (2.2) by a parameter $\lambda \in (0, 1)$:

$$(P, Q) \in E(G) \Leftrightarrow (2.1) \quad \text{holds and} \quad |a^C_{QP}| \geq \lambda \max\{|a^C_{QR}| : |a^C_{QR}| > \kappa |a^D_{QR}|\} \tag{2.8}$$

If λ is very small, nearly all edges are defined as strong and belong to $E(G)$. With $\lambda = 1$, only the strongest connection is used (cf. §5). It is obvious that there are cases where graph contains cycles indepently of the choice of λ (cf. test problem SQUARE in §4).

At this point we want to present a test problem with a cyclefree graph. Thus we have to choose λ large enough to have no cycles in the graph. On our test problem LABYRINTH it turns out that for $\lambda = 0.145$ the graph G is still cyclefree, but for smaller λ cycles occur.

On the other hand, λ determines the size of matrix entries in the upper triangular part: if the right ordering has been determined by algorithm (2.5), all matrix entries a^C_{QP}, which correspond to an edge $(P, Q) \in E(G)$, are in the lower triangular part, and the entries a^C_{ST} in the upper part fulfil $|a^C_{ST}| < \lambda \max\{|a^C_{SR}| : |a^C_{SR}| > \kappa|a^D_{SR}|\}$ or $|a^C_{ST}| \leq \kappa|a^D_{ST}|$. Thus, with smaller λ, one expects better convergence rates.

Since we want to present results for circlefree graphs, the convergence rates in Tab. 1 for the LABYRINTH problem have been calculated with $\lambda = 0.145$. The standard choice $\lambda = 0.5$ gives a different graph G, another ordering and worse convergence rates (for $h = 1/256, \varepsilon = 10^{-5}$ one gets the rate 0.133).

3. Construction of Concentric Cycles

If G is cyclic, a numbering as in §2 is not possible, however, one can try to find appropriate blocks for a blockwise Gauß-Seidel iteration. Such blocks are the "concentric" cycles described in the following. The name "concentric" refers to the simplest situation, where one cycle surrounds the foregoing one (cf. Fig. 7). Different from the discussion in §2, the following algorithm is restricted to a planar graph, i.e., to the case of two spatial dimensions. Furthermore, we suppose a so-called "one-flow-direction condition" for the graph (see §3.1) which is natural in our situation. In particular, this condition holds, if the outward edges of the graph correspond to two sides of one triangle as in Fig. 1.

3.1. The One-Flow-Direction Condition

In the case of a planar graph, the edges E_i associated with a vertex $P \in V(G)$ can be ordered periodically with respect to the angle (cf. Fig. 4a). Let $\{E_1, E_2, \ldots, E_k\}$ be such a list. We say that the *one-flow-direction condition* is satisfied if for all $P \in V(G)$ there is a list $\{E_1, E_2, \ldots, E_k\}$ such that for some $l \in \{0, \ldots, k\}$ the edges E_i $(1 \leq i \leq l)$ are outward, while E_i $(l < i \leq k)$ are inward. Geometrically, this corresponds to the situation that there are two sectors around P such that the convection enters the second sector and leaves P through the first one. Since $l = 0$ or $l = k$ are admitted, a sector may be empty. A situation forbidden by the condition is shown in Fig. 4 b).

The required condition is natural in our situation and simplifies the following considerations. For modifications of the algorithm without this condition we refer to [6].

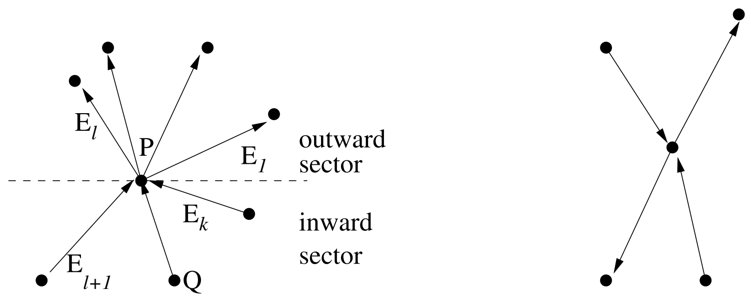

Figure 4. a) one-flow-direction-condition b) forbidden situation

3.2. Planarity

Usually the planarity of the graph is defined by the embedding in the plane $\mathbb{R}^2$. In our situation it is even more advantageous to replace $\mathbb{R}^2$ by a sphere as explained in

Remark 3.1. *The sphere is a more natural two-dimensional manifold in which the graph can be embedded, since then a cycle divides the sphere into two pieces with equal properties, whereas in the case of $\mathbb{R}^2$ the distinction between a bounded and unbounded part may induce wrong interpretations of the terms interior and exterior. Given an embedding of the graph in $\mathbb{R}^2$, one can identify $\mathbb{R}^2$ with $\mathbb{C}$ and replace $\mathbb{C}$ by its compactification $\mathbb{C} \cup \{\infty\}$ corresponding to the Riemann sphere.*

3.3. The Graph G_C

In the following, only those parts of the graph are of interest whose vertices have the attribute C (see Fig. 2). Removing all vertices with attributes F or L from G, we obtain the graph G_C consisting only of C-vertices. The performance of the Gauß-Seidel process, which will be discussed in detail in §4, will follow the strategy sketched in

Remark 3.2. *The Gauß-Seidel process consists of three parts: (a) Pointwise relaxation at the vertices with attribute F in the ordering as described in Remark 2.2. b. (b) Treatment of the C-vertices (vertices of G_C) as in §4. (c) Finally, pointwise relaxation at the vertices with attribute L in the order as described in Remark 2.2.c.*

Concerning the notation, we remark that $G \setminus Z$ for a subset $Z \subset V(G)$ or a subgraph Z of G (e.g., a cycle Z in G) consists of the vertices of G without those of Z, while all edges in G associated with vertices of Z are deleted.

We list a number of properties of G_C.

Remark 3.3. *a) In G_C, each $P \in V(G_C)$ has at least one successor and at least one predecessor.*

b) A cycle in G is also a cycle in G_C, i.e., all vertices involved in a cycle have attribute C.

c) In general, the union of all cycles is smaller than G_C.

Proof Part a) follows from Remark 2.2.e). Since a vertex of a cycle never fulfils (2.6b) or (2.7b), part b) is shown. For part c) use the counterexample in Fig. 2. ■

In the following, only the new graph G_C is considered. For ease of notation, we replace the name G_C again by G.

3.4. Connectivity

The following discussion becomes easier if we reduce ourselves to the case of strongly connected graph. Here, we emphasise that the algorithm is not restricted to this case.

G is called *(weakly) connected* if for any $P, Q \in G$ there is a path either from P to Q or Q to P. G is *strongly connected* if for any $P, Q \in G$ there is a path from P to Q. For any $P \in G$, the set $\{Q$: there are paths between Q and P in both directions $\}$ defines the *strongly connected component* containing P. Obviously, G can be regarded as a union of disjoint strongly connected components.

Remark 3.4. *Any cycle in G is contained in only one strongly connected component.*

Because of this remark, considerations concerning cycles can be reduced to the strongly connected components, separately. Weakly connected graphs will be considered in §3.7.

3.5. The Reduced Graph

Thanks to the "one-flow-direction condition" from §3.1. and because of Remark 3.3. a), for each $P \in V(G)$ there is a unique tuple $\{E_1, \ldots, E_l\}$ of $l \geq 1$ outward edges ordered in the anti-clockwise orientation (cf. Fig. 4). E_1 is the rightmost outward edge of P, while E_l is the leftmost one. Another characterisation is given in

Remark 3.5. *Let $E = (Q, P)$ be any edge directed towards $P \in V(G)$. Running through the edges of P in the [anti-]clockwise orientation, the first outward edge is E_l [E_1].*

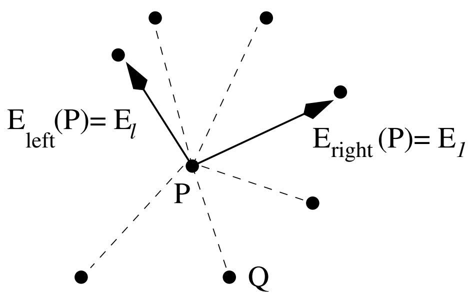

Figure 5. $E_{right}(P)$ and $E_{left}(P)$

Denote the edges E_1 and E_l from above by $E_{right}(P)$ and $E_{left}(P)$, respectively (cf. Fig. 5). The *reduced graph* G^+ consists of the same vertices as G, but the edges are reduced to the smaller set $E_{left}(P)$:

$$V(G^+) := V(G) \quad E(G^+) := \{E_{left}(P) : P \in V(G)\}. \tag{3.1}$$

Analogously, we could define G^- by

$$V(G^-) := V(G), \quad E(G^-) := \{E_{right}(P) : P \in V(G)\}. \tag{3.2}$$

If necessary, we will call edges in G^+ (which are also edges in G) *strong* edges, whereas those in G are named *weak* edges. Successors (predecessors) in G^+ are called *strong* ones, those in G *weak* ones.

The interesting properties of G^+ are gathered in

Remark 3.6. *a) Each $P \in V(G^+)$ has exactly one successor in G^+.*

b) $P \in V(G^+)$ may have no or more than one predecessor in G^+.

c) Any cycle in G^+ is also a cycle in G but not vice versa.

d) There is at least one cycle in G^+ (except when $G^+ = \emptyset$). In case that G is not restricted to C-vertices, this statement reads as follows: If G is not acyclic, there is at least one cycle in G^+, i.e., a minimal cycle (see §3.6.) in G.

e) Different cycles in G^+ are disjoint.

Proof Part a) holds by definition. Since strong successors are also weak ones, a ”strong cycle” is also a ”weak” one. This fact proves part c). The construction in §3.6. will prove part d). Also part e) is postponed to §3.6. ∎

3.6. Minimal Cycles

A cycle in G^+ is in particular a cycle in G (cf. Remark 3.2.). Because of the properties stated in Theorem 3.2., such a cycle in G^+ will be called a *minimal cycle* of G.

Let C^+ be the set of cycles in G^+. In the following, we construct a surjective mapping from $V(G^+)$ onto C^+. Each $P \in V(G^+)$ will be mapped to the cycle $\zeta \in C^+$ which we denote by $\zeta = \zeta(P)$. Since by Remark 3.6.a) every vertex has exactly one successor, the recursion

$$P_0 := P; \quad P_{i+1} := \text{successor of } \ P_i \tag{3.3}$$

defines a uniquely defined infinite sequence of vertices in G^+. Since $\#G < \infty$ (cf. (2.3)), there must be a first index j such that $P_j = P_i$ for some $i < j$. Hence, the path $\{P_i,\ P_{i+1}, \ldots, P_j = P_i\}$ forms a cycle from C^+. This is the cycle named $\zeta(P)$.

Since for any $\zeta \in C^+$ and any $P \in \zeta$, one verifies that $\zeta(P) = \zeta$, the mapping $P \in V(G^+) \mapsto \zeta(P) \in C^+$ is surjective. For the proof of Remark 3.6. e) choose $P \in \zeta \cap \zeta'$ for two cycles $\zeta, \zeta' \in C^+$. Since $\zeta(P) = \zeta$ as well as $\zeta(P) = \zeta'$, $\zeta = \zeta'$ follows.

The basic part of the partitioning algorithm (3.6) from §3.8. is described by (3.3). The remaining part of this subsection is not needed for the algorithm, but explains the terms ”minimal cycle”, ”exterior” and ”interior”.

The structure of the reduced graph G^+ is studied in

Theorem 3.1. *G^+ decomposes into disjoint subgraphs $G_\zeta, \zeta \in C^+$:*

$$G^+ = \biguplus \{G_\zeta : \zeta \in C^+\}. \tag{3.4}$$

Each G_ζ is (weakly) connected and contains no other cycle than ζ. It consists of ζ and possibly of further trees connected with ζ (cf. Fig. 6).

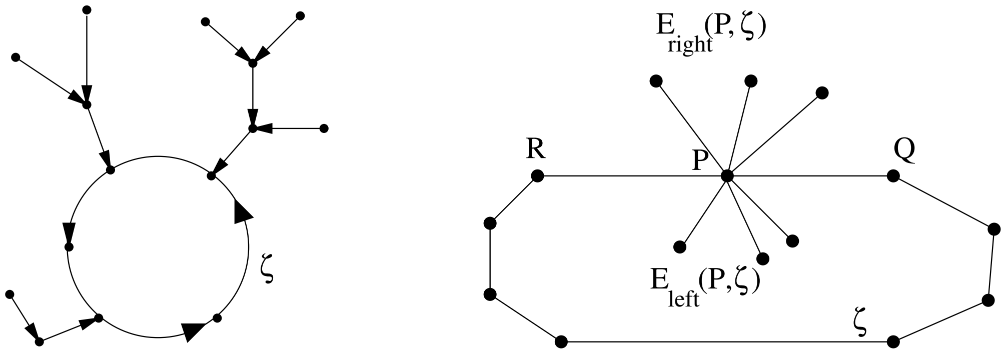

Figure 6. G_ζ Left and right neighbours (ζ: general cycle)

The proof of the theorem is left to the reader since the result is not used in the following.

We name cycles ζ in G^+ *minimal* ones since no further ("smaller") cycles exist in their interior (see Theorem 3.2. below). For this purpose, we have to define the interior $Int(\zeta)$ and the exterior $Ext(\zeta)$ of a cycle ζ. Here, we need that G is a planar graph. Let $P \in \zeta$ be a vertex of ζ with predecessor $Q \in \zeta$ and successor $R \in \zeta$. All edges in the angle between the edges $(P,R) \in E(G)$ and $(Q,P) \in E(G)$ (anti-clockwise orientation) are called *left* edges. They form the set $E_{left}(P,\zeta)$ (cf. Fig. 6). The edges on the other side yield $E_{right}(P,\zeta)$.

Remark 3.7. *Let G be strongly connected. Let G_i be the connected components of $G \setminus \zeta$: $G \setminus \zeta = \dot{\bigcup} G_i$. Each G_i has a vertex $P_i \in V(G_i)$ not contained in ζ but connected with ζ:*

$$\exists P_i \in V(G_i) \;\; \exists P \in \zeta : \;\; (P_i, P) \in E(G). \tag{3.5}$$

If $(P_i, P) \in E_{left}(P,\zeta)$, also any other pair (P_i, P) satisfying (3.5) lies in $E_{left}(P,\zeta)$. Hence, the subgraph G_i can be attached to the *left side* of ζ which we call the *interior* of ζ . Correspondingly, G_i belongs to the *exterior*, if $(P_i, P) \in E_{right}(P,\zeta)$. $Int(\zeta)$ is the union of all left G_i, while $Ext(\zeta)$ is the union of all right G_i.

Theorem 3.2. *Let G be strongly connected. All $\zeta \in C^+$ satisfy $Int(\zeta) = \emptyset$.*

Proof Let $Q' \in Int(\zeta)$ and $P' \in \zeta$. Since G is strongly connected, there is a path from P' to Q'. This sequence of edges contains an edge (P,Q) with $P \in \zeta$ and $Q \in Int(\zeta)$. Since the edge (P,Q) is on the left of (P,P'') ($P'' = E_{left}(P)$: successor of P in G^+), this contradicts the definition of $E_{left}(P)$. ■

3.7. Weak Connectivity

The strong connectivity of G in §3.5. is supposed "without loss of generality". Assume that G is only *weakly connected*. Then we replace $(P_i,P) \in E(G)$ in (3.5) by "$(P_i,P) \in E(G)$ or $(P,P_i) \in E(G)$". Components G_i, which are not weakly connected with ζ , belong neither to $Int(\zeta)$ nor $Ext(\zeta)$.

If $G = \dot{\bigcup} G_i$ is (weakly) connected and G_i are the strongly connected components, this gives rise to an ordering $G_1,\ldots,G_k$ in the sense that for $i < j$ there are only paths from

$P \in G_i$ to $Q \in G_j$ but not in the other direction. Then the later block partitioning can use this fact so that blocks contained in G_i are to be numbered before those of G_j $(i < j)$.

Theorem 3.2. is not true when G is assumed only to be weakly connected, but it can be used to detect further strongly connected components in G.

Remark 3.8. *Let $\zeta \in C^+$. For any $P \in \zeta$, look for a left edge $E \in E_{left}(P, \zeta)$. If it exists it must be of the form $E = (Q, P)$ with $Q \in Int(\zeta)$, $P \in \zeta$. The vertex Q and its predecessors must belong to a connected component G_0 which is only weakly connected with $G \setminus G_0$. By Remark 3.7., G and $G \setminus G_0$ can be investigated separately.*

Proof The proof of Theorem 3.2. shows that there is no weak successor of any $P \in \zeta$ in $Int(\zeta)$. Hence, E must be directed from $Q \in Int(\zeta)$ to $P \in \zeta$. ■

A similar check can be made in the exterior of ζ.

Remark 3.9. *Let ζ be any cycle in G. If all edges in $E_{right}(P, \zeta)$ for all $P \in \zeta$ have the same direction (i.e., all from ζ into $Ext(\zeta)$ or all from $Ext(\zeta)$ to ζ), then $Ext(\zeta)$ is not strongly connected with ζ.*

3.8. Concentric Cycles

Let G be strongly connected and $\zeta \in C^+$ any (minimal) cycle. By Theorem 3.2., $Ext(\zeta)$ and $G \setminus \zeta$ coincide. Rename G and ζ by $G_0 := G$, $\zeta_0 := \zeta$, and construct $G_1' := G \setminus \zeta$. Applying the algorithm from §2.3. to G_1', we may find vertices in G_1' with attribute F (or L), since their predecessors (or successors) are removed from G_0 (cf. Fig. 7). If no vertex of attribute C remains, G_1' is acyclic. Otherwise, we repeat the process: Let G_1 be the restriction to all vertices from G_1' with attribute C. Define the corresponding reduced graph G_1^+ (cf. §3.1.) and determine a (minimal) cycle $\zeta_1 \in C_1^+$ in G_1^+. This process terminates when G_i is empty:

$G_0 := G, \quad \zeta_0 := \zeta,$	(start) (3.6a)
$G_i' := G_{i-1} \setminus \zeta_{i-1},$	(remove previous cycle) (3.6b)
determine attributes in G_i'	(3.6c)
$G_i :=$ restriction to C-vertices of G_i',	(remove F and L vertices)(3.6d)
stop if G_i is empty; otherwise, choose some $P \in G_i$,	(3.6e)
determine the cycle $\zeta_i := \zeta(P)$ in G_i.	(new minimal cycle) (3.6f)

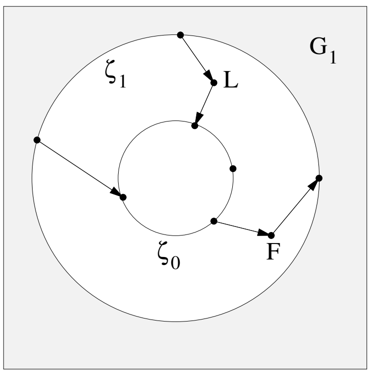

Figure 7. Vertices with attributes F, L belonging to G_1' but not to G_1

In order to understand that the global work is still proportional to the size of G, we discuss the steps c-f in more detail.

Step (3.6 c) must consume a work not larger than proportional to the cardinality of $G_{i-1} \setminus G_i$. For that purpose, we describe a local version of the algorithm (2.5-8) and use the fact that a change in the attribute can only be caused by the removal of vertices from ζ_{i-1}. The initialisation (2.5a) is not necessary. (2.5b) becomes

$$\text{for } P \in \zeta_{i-1} \text{ do SetAttr(P);} \tag{3.7}$$

where the terms "predecessor/successor" in (2.6b) and (2.7b) are to be understood with respect to the smaller graph G_i', whereas (2.6d/7d) refer to G_{i-1}. This means that lines (2.6b/7b) become:

begin if "P in ζ_{i-1} or all predecessors of P in G_1' have attribute F" then ... (3.8a)

begin if "P in ζ_{i-1} or all successors of P in G_1' have attribute L" then ... (3.8b)

while (2.6d/7d) is to be read as

$$\text{for } Q \in \{\text{successors of P in } G_{i-1}\} \text{ do ...}$$

$$\text{for } Q \in \{\text{predecessors of P in } G_{i-1}\} \text{ do ...}$$

The cheapest implementation of (3.6 e,f) is as follows. The very first construction of $\zeta_0 := \zeta$ uses the path $\{P_0, P_1, \ldots, P_i\}$ from (3.3). Deleting vertices in the steps (3.6 b,d), the path is shortened, i.e., only $\{P_0, P_1, \ldots, P_{i'}\}$ with $i' < i$ remains. Then choose $P = P_{i'}$ in step (3.6e). If the empty set remains, since the whole path $\{P_0, P_1, \ldots, P_i\}$ is removed by (3.6 b,d), choose any P for which the Boolean expressions in (3.8a,b) results in FALSE. Such a P has still the attribute C and belongs to G_i. On the other hand, such a P must exist as long as G_i is not empty. These considerations prove the following theorem.

Theorem 3.3. *The algorithm (3.6) for computing the set of cycles $\{\zeta_0, \zeta_1, \ldots\}$, the so-called concentric cycles, requires a work proportional to $\#G$.*

4. Smoothing by Block-Gauß-Seidel

If one uses block-Gauß-Seidel smoothers, it is not sufficient to determine the right ordering by the methods of §2 (resp. §3 in the cyclic case): The vertices on a characteristic have to be gathered to blocks. It turns out that this "blocking" is cheap.

In the cyclic case the algorithm (3.6) detects and removes the cycles; each cycle forms a block. The vertices of each cycle form a block. If the corresponding block matrix contains the strong edges only, it is cyclic tridiagonal. Matrices of this type are easy to invert by a LU-decomposition. Due to the entries in the upper right and the lower left corner of the matrix a slight fill-in occurs during the LU-decomposition (in contrast to the standard tridiagonal case).

The algorithm (2.6) find the vertices with attributes F and L in the appropriate order, while algorithm (3.6) detects (concentric) cycles. During this processes vertices with attributes F and L can occur (e.g., if G_i' in (3.6c) is acyclic). These are to be treated before or after the cycles.

The Gauß-Seidel smoothing consists of pointwise and blockwise iterations: pointwise iteration on the single vertices (F and L) and blockwise iteration on the cycles.

The blocking procedure contains the detection of cycles, the grouping of their vertices to blocks and the list's/matrix' reorganisation and costs about 0.8 to 1.2 Gauß-Seidel steps. The reduction form $E(G)$ to $E^+(G)$ (cf. §3) takes about 0.2 of the time of a Gauß-Seidel steps. The total amount remains less than 2.7 Gauß-Seidel steps.

As test problem we choose a counter-clockwise circular flow (see Fig. 8a) and flow along squares (Fig. 8b) which mimics the XLINE test problem with the flow only on grid lines.

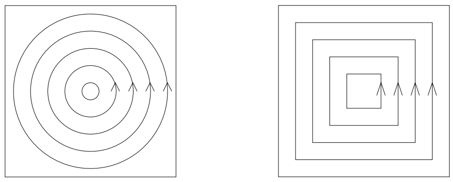

Figure 8. a) CIRCLE b) SQUARE

Since for $\varepsilon \to 0$ the convection problem becomes singular (the characteristics are not linked to the boundary), it is stabilised by a constant term:

$$b(x,y) = \left(\frac{-(y-0.5)}{\sqrt{(x-0.5)^2+(y-0.5)^2}}, \frac{(x-0.5)}{\sqrt{(x-0.5)^2+(y-0.5)^2}} \right), \quad a(x,y) = 1.$$

For the test problem SQUARE one observes for $\varepsilon/h \to 0$ very small convergence rates (e.g. $\varrho = 0.0025$ for $h_{fine} = 1/256$, $h_{coarse} = 1/8$, $\varepsilon = 10^{-5}$, W-cycle with 4 block-Gauß-Seidel presmoothing steps, squares ordered from the boundary to the centre).

It is not a priori clear in which order the cycles are found (from the boundary to the centre or vice versa; this depends on the definition of the reduced graph (G^+ or G^-)).

Since the block-Gauß-Seidel smoother depends on the ordering of the blocks, the rate of convergence can be improved by the selection of an appropriate order: The blocks are considered as vertices, connections between blocks are considered as edges. With algorithm (2.5) the blocks are marked by the attributes F, C and L. In the case that some blocks keep the attribute C, these blocks can be treated by a symmetric block-Gauß-Seidel iteration (cf. Fig. 7: the cycles ζ_0 and ζ_1 are connected mutually). In the test problem CIRCLE each circle has edges to both neighbour circles (interior and exterior); this explains the fact that for small ε the symmetric blocksmoothing gives the best results.

The following table shows the performance of the multi-grid method (W-cycle) with three different type of smoothers: 2 symmetric Gauß-Seidel presmoothing steps with lexicographical ordering (GS), 4 block-Gauß-Seidel-forward steps (BGSF) and 2 symmetric block-Gauß-Seidel steps (BGSS). The forward block ordering is from the boundary to the centre.

Table 2. convergence rates for cyclic flow CIRCLE

h	Smoother	$\varepsilon = 1$	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-5}$
h= 1/64	GS	0.065	0.129	0.141
	BGSF	0.026	0.095	0.141
	BGSS	0.051	0.098	0.094
h= 1/128	GS	0.063	0.112	0.192
	BGSF	0.024	0.079	0.207
	BGSS	0.049	0.081	0.129
h= 1/256	GS	0.061	0.091	0.302
	BGSF	0.023	0.064	0.456
	BGSS	0.047	0.070	0.163

5. Alternative Choices of Cycles

As already mentioned in §4, the choice of the reduced graph (G^+ or G^-) decides whether the concentric cycles are found from the centre to the boundary or in the reverse order.

Another criterion for the definition of the reduced graph might be the size of an edge entry: prefer (P,Q) to (R,Q) if $|a^C_{Q,P}| > |a^C_{Q,R}|$, because the value at P is "more important" for Q than the value at R. This can be achieved in certain test problems by choosing $\lambda = 1$ in (2.8); in this case $E(G)$ contains only strong edges.

REFERENCES

1. J. Bey and G. Wittum. Downwind numbering: A robust multigrid method for convection diffusion problems on unstructured grids. Bericht 95/2, Institut für Computeranwendungen der Universität Stuttgart, 1995.
2. W. Hackbusch. *Multi–grid Methods and Applications*, volume 4 of *Computational Mathematics*. Springer–Verlag, Berlin, 1985.
3. W. Hackbusch. A new approach to robust multi–grid solvers. In J. McKenna and R. Temam, editors, *ICIAM'87: Proceedings of the First International Conference on Industrial and Applied Mathematics*, pages 114–126. SIAM, Philadelphia, 1988.
4. W. Hackbusch. *Elliptic Differential Equations*, volume 18 of *Computational Mathematics*. Springer–Verlag, Berlin, 1992.
5. W. Hackbusch. *Iterative Solution of Large Sparse Systems of Equations*. Springer–Verlag, Berlin, 1993.
6. W. Hackbusch. On a feedback vertex set problem for a planar graph. Bericht 9503, Christian-Albrechts-Universiät Kiel, 1995.
7. W. Hackbusch. Analysis of discretizations by the concept of discrete regularity. In J. Whiteman, editor, *The Mathematics of Finite Elements and Applications IV*, volume IV of *MAFELAP*, pages 369–376, Uxbridge, April-May 1981. Academic Press, London,1981.
8. W. Hackbusch and U. Trottenberg. *Multigrid Methods*, volume 960 of *Lecture Notes in Mathematics*. Springer–Verlag, Berlin, 1982.
9. G. Reinelt. *The linear ordering problem: algorithms and applications*, volume 48 of *Research and Exposition in Mathematics*. Heldermann–Verlag, Berlin, 1985.
10. A. Reusken. Fourier analysis of a robust multigrid method for convection-diffusion equations. *Numer. Math.*, 71:365–398, 1995.
11. E. Speckenmeyer and P. Heusch. Solving feedback vertex problems in diagraphs. a new algorithm based on markov chains. To appear.
12. H. Stamm. Graph-theoretical concepts in computer science. In *Lecture Notes in Computational*

Science 484, pages 79–89. Springer–Verlag, Berlin, 1992.