VARIABLE TWO-STAGE RELAXATION BASED ON THE PRINCIPLE OF AGGREGATION MULTI-GUID

R. Haelterman *, J. Vierendeels †, D. Van Heule ‡

*, † Royal Military Academy, Dept. of Mathematics (MWMW)
Renaissancelaan 30, B-1000 Brussels, Belgium
- e-mail: Robby.Haelterman@rma.ac.be
- web page: http://www.rma.ac.be
† Ghent University, Dept. of Flow, Heat and Combustion Mechanics
St.-Pietersnieuwstraat 41, B-9000 Gent, Belgium
- e-mail: Jan.Vierendeels@UGent.be
- web page: http://www.floheacom.UGent.be

Key words: Iterative methods, non-stationary, multi-grid methods, step-length method

Abstract. Iterative methods are the preferred solution for solving very large algebraic systems of equations. Well-known schemes are damped Jacobi and Successive Over-Relaxation (SOR). While the optimal choice of the relaxation parameter for these schemes has been established for particular examples of system matrices, it is often by trial and error that these need to be determined in more general cases. Iterative methods are also used as smoothers in multi-grid methods. It has been shown that a multi-grid method using the Galerkin construction will converge whenever the relaxation method converges [15]. Therefore the choice of the relaxation parameter that will ensure this convergence is very important, but sometimes difficult for arbitrary matrices.
In this paper we try to automatize the choice of the parameter. This will be done repetitively throughout the iteration, adapting itself to optimal values (with respect to the energy norm) in a non-stationary fashion. The basis of the method is formed by an aggregation-type procedure into a single aggregate and is meant to be applicable as a patch to existing codes using iterative solvers. Its effect turns out to be interpretable as a step-length method.
We focus our attention to iterative Jacobi and related methods as stand-alone applications and in a multi-grid context.

1 INTRODUCTION

Large algebraic systems of equations, as resulting from the discretization of (systems of) PDEs for instance, are routinely solved by iterative methods when the number of unknowns (n) becomes very large. Well-known examples of these methods are named after Gauss, Seidel and Jacobi. Variants exist that use an over- (or under-) relaxation parameter giving rise to Successive Over-Relaxation methods or damped Jacobi. The choice of
the optimal parameter is based upon the spectrum of the error operator, which can only be established in some specific cases (e.g. for the Poisson equation [3]). For arbitrary matrices, this choice is much more difficult. For highly "skewed" operators, one can turn to distributive relaxation schemes, like the Cimmino method [6].

When one has determined a good value of the relaxation parameter most of these schemes can be tuned to have a good initial convergence speed. Still, they tend to slow down after a while. It has long been known that this is due to the way the error operator handles the different Fourier modes of the error. As a result the algebraically "smooth" components (associated with eigenvalues \( \approx 1 \) of the error operator) are only reduced very slowly. Two-grid methods counter this by eliminating these components on a coarser grid, where they appear less "smooth" and can thus be more efficiently reduced. This can be done recursively, thus creating true multi-grid methods. Such was the reasoning introduced by Fedorenko in 1964 [8], and Brandt in 1973 [4]. The multi-grid process can be implemented geometrically or algebraically. In the latter case, the code creates coarse grid matrices and transfer operators based solely on the elements of the fine grid matrix [16]. Perhaps the most basic scheme in this context is the Aggregation (or Agglomeration) scheme [5, 18], where clusters of fine-grid variables are grouped (aggregated) to form coarse-grid variables. As the transfer operators are only piecewise constant, this approach has some limitations, which have partially been overcome by "smoothing" these operators (giving rise to smoothed aggregation [17]) or by multiplying the correction on the coarser grids by a factor [2, 14].

In [15] it was shown that a multi-grid method using the V-cycle and the Galerkin construction will converge whenever the relaxation method used therein converges. The latter will depend on the value of the relaxation parameter. In this paper we propose a procedure to automatize the choice of the parameter used for relaxation schemes in a dynamic way. We are initially interested in damped Jacobi and related methods as stand-alone solvers and as smoothers for a multi-grid method. In the latter case the smoothed aggregation variant of algebraic multi-grid is used. The idea is to add a small number of lines to an existing code as a patch. Emphasis is put on simplicity and low cost (preferably \( \mathcal{O}(n) \)) and applicability to general cases of matrices to guarantee convergence independently of the input relaxation parameters the user has specified or at least reduce its influence.

In section 2 we give a review of the most common iterative schemes and the basics of multi-grid. In section 3 we explain the proposed modification. Finally, in section 4, we try different variations of the method on the advection-diffusion equation.

2 The basic model

2.1 Notation

Throughout this paper, vectors will be written in boldface and matrices in capital letters. The identity matrix will be written as \( I \). Its dimensions will be clear from the context. The Euclidian scalar product of two vectors \( \mathbf{u}, \mathbf{v} \) of \( \mathbb{R}^{n \times 1} \) is given by \( \langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T \mathbf{v} \), and
the Euclidian (L2) norm of a vector \( \mathbf{u} \) is given by \( \| \mathbf{u} \|_2 = \sqrt{\mathbf{u}^T \mathbf{u}} \). If \( A \) is a hermitian positive-definite matrix then \( \sqrt{\langle \mathbf{u}, A \mathbf{u} \rangle} = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle_A} \) also denotes a norm which we will call the energy norm and write as \( \| \mathbf{u} \|_A \).

### 2.2 Iterative methods

We are interested in solving algebraic equations of the type

\[ A \mathbf{u} = \mathbf{f} \]  

(1)

as resulting, for instance, from the discretization of partial differential equations (PDE). We assume that \( A \in \mathbb{R}^{n \times n} \) is non-singular; \( \mathbf{u}, \mathbf{f} \in \mathbb{R}^{n \times 1} \). We will solve this system with an iterative procedure (also called relaxation scheme). The latter typically splits the matrix \( A \) in two parts: \( A = M - N \), where \( M \) is non-singular. The solution is found by iteratively applying

\[ \mathbf{u}^{m+1} = B \mathbf{u}^m + \mathbf{c} \]  

(2)

starting from an initial guess \( \mathbf{u}^0 \), where \( B = M^{-1}N, \mathbf{c} = M^{-1}\mathbf{f} \). Equation (2) is called the first normal form of the relaxation scheme and will be symbolically written as

\[ \mathbf{u}^{m+1} = S(\mathbf{u}^m) \]  

(3)

The second normal form is given by

\[ \mathbf{u}^{m+1} = \mathbf{u}^m + M^{-1}(\mathbf{f} - A \mathbf{u}^m) \]  

(4)

\( \mathbf{f} - A \mathbf{u}^m \) is also called the residual and written as \( \mathbf{r}^m \). \( B \) is called the iteration matrix. Note that, if we define the error as \( \mathbf{e}^m = \mathbf{u} - \mathbf{u}^m \), then

\[ \mathbf{e}^{m+1} = B \mathbf{e}^m = B^{m+1} \mathbf{e}^0 \]  

(5)

The convergence factor (or contraction number) is given by \( \| B \| \) because

\[ \| \mathbf{e}^{m+1} \| = \| B \mathbf{e}^m \| \leq \| B \| \| \mathbf{e}^m \| \]  

(6)

If we define the spectral radius of \( B \) by \( \rho(B) \), then it serves as a lower bound for the convergence factor and equals the asymptotic convergence factor [11]. We therefore require \( \rho(B) < 1 \). If \( B \) is (not) constant throughout the iterations, the method is called (non-)stationary.

The best-known iteration methods start from a decomposition of \( A \) as \( A = D + L + U \), where \( D \) is the diagonal part of \( A \), and \( L(U) \) are the strictly lower (upper) triangular parts of \( A \). For an illustration of the most classical examples we refer to [1, 7]. We only recall that for the damped Jacobi method the first normal form is given by

\[ \mathbf{u}^{m+1} = \left( \frac{1}{\omega} D \right)^{-1} \left( \left( \frac{1}{\omega} D \right) - A \right) \mathbf{u}^m + \left( \frac{1}{\omega} D \right)^{-1} \mathbf{f} \]  

(7)
where \( \omega \in ]0, 1[ \). Therefore, from equation (2) we have

\[
B = \left( \frac{1}{\omega}D \right)^{-1} \left( \left( \frac{1}{\omega}D \right) - A \right) \quad (8)
\]

\[
c = \left( \frac{1}{\omega}D \right)^{-1} f \quad (9)
\]

In [20] a thorough discussion is presented about the convergence of most classical relaxation schemes. Necessary and sufficient conditions are presented under which convergence can be achieved. Under specific circumstances the optimal parameter can easily be derived, while for others the derivation is quite cumbersome or even impossible.

### 2.3 Two-grid and multi-grid methods: the correction scheme

Multi-grid methods are a subclass of the multi-scale methods that were mainly developed to accelerate the solution process with iterative solvers. Assume we have obtained the discrete equation on the finest grid \( \Omega_h \) with grid parameter \( h \):

\[
Au = f \quad (10)
\]

As we have mentioned, iterative solvers like Gauss-Seidel or Jacobi will smooth the high frequency errors very quickly while the low frequency errors are damped very slowly [3]. A solution to this problem would be to transfer the problem to a coarser grid \( \Omega_H, H > h \) where the low frequency errors effectively become higher frequency errors as pioneered by Fedorenko and Brandt [8, 4]. The transfer from fine grid to coarse grid is done by a matrix operator \( R \) (called the restrictor) and the transfer from coarse to fine by the prolongator \( P \).

The two-grid scheme is basically composed of two major building blocks:

1. A smoothing procedure, e.g. with an iterative solver (relaxation method), that returns an improved solution. We define the smoothing operator symbolically with \( \mathcal{S} \). This operation will damp out the high frequency content of the error.

2. A coarse grid correction to reduce the low frequency content of the error.

A typical algorithm is the following [19]:

1. Execute \( \nu_1 \) iterations (smoothing) on the \( m \)-th approximation \( u^m : u^m \leftarrow \mathcal{S}^m (u^m) \)
2. Compute the residual : \( r^m = f - Au^m \)
3. Restrict the residual: \( r_H = Rr^m \)
4. Compute the coarse grid correction: \( v_H = A_H^{-1} r_H \), where \( A_H \) is the coarse grid equivalent of \( A \).
5. Prolong the coarse grid correction: \( v = P v_H \)
6. Update the fine scale solution \( u^m \leftarrow u^m + v \)
7. Execute \( \nu_2 \) iterations (smoothing) on \( u^m : \ u^{m+1} = S^{\nu_2} (u^m) \)
8. Repeat

Step 2 until 6 are called the coarse grid correction or *defect correction* (DC).

In the above algorithm we have assumed that in step 4 (called the defect equation) \( v_H \) can be computed exactly by a direct solver. If not, we can solve \( A_H v_H = r_H \) iteratively with \( \gamma \) iteration steps (note that it is the same type of equation as the original problem).

If in this iterative process we invoke even coarser grids, we have effectively constructed a multi-grid algorithm. If \( \gamma = 1 \) the result is called a V-cycle. Note that we will always assume that \( A_H \) is non-singular and that \( P \) and \( R \) have full rank.

The error propagation matrix for the defect correction is defined by

\[
B_{DC} = I - P A_H^{-1} R A
\]  

(11)

which is rank deficient. Moreover, it can be easily shown that \( B_{DC} = (B_{DC})^2 \), and as \( B_{DC} \neq 0, B_{DC} \neq kI (k \in \mathbb{R}) \) we can conclude that \( m_{B_{DC}}(x) = x^2 - x \) is the minimal polynomial of \( B_{DC} \). Knowing that the minimal polynomial has the same zeroes as the characteristic polynomial, we can thus conclude that the spectrum of \( B_{DC} \) is given by \( \sigma(B_{DC}) = \{0, 1\} \), meaning that the modes are either eliminated completely or left unaltered.

We will only interest ourselves in a specific type of algebraic multi-grid called aggregation (or agglomeration) multi-grid as described in [5, 18] with a coarsening factor of 2. In this method different nodes are clustered into aggregates that fulfill the role of coarse grid variables. The basic prolongation \( P = [p_{ij}] \) and restriction \( R \) is then piecewise constant and given by

\[
p_{ij} = \begin{cases} 
1 & \text{if node } i \text{ belongs to the } j\text{-th aggregate} \\
0 & \text{otherwise}
\end{cases} \quad (12)
\]

\[
R = P^T \quad (13)
\]

To define \( A_H \) we use the Galerkin approach

\[
A_H = R A P 
\]  

(14)

Smoothed aggregation [17] is obtained when we replace \( P \) by

\[
P_s = \left( I - \frac{2}{3} D^{-1} A \right) P
\]  

(15)
3 Dynamic over-/under-relaxation (DOUR)

3.1 The modification

We suppose that we have an undamped relaxation scheme given by (3). In its damped guise, with the user-defined parameter \( \omega \), this becomes

\[
\mathbf{u}^{m+1} = \mathbf{u}^m + \omega(S(\mathbf{u}^m) - \mathbf{u}^m)
\]

We propose the following predictor-corrector routine to dynamically modify \( \omega \) into an effective relaxation parameter \( \omega_e \):

\[
\tilde{\mathbf{u}}^m = \mathbf{u}^m + \omega(S(\mathbf{u}^m) - \mathbf{u}^m)
\]

\[
\mathbf{u}^{m+1} = \tilde{\mathbf{u}}^m + P'(R'AP')^{-1}R'(\mathbf{f} - A\tilde{\mathbf{u}}^m)
\]

where

\[
P' = \tilde{\mathbf{u}}^m - \mathbf{u}^m = \Delta \mathbf{u}^m
\]

\[
R' = (P')^T
\]

It results that

\[
\mathbf{u}^{m+1} = \mathbf{u}^m + \omega_e(S(\mathbf{u}^m) - \mathbf{u}^m)
\]

where \( \omega_e = \omega (1 + \kappa) \). We can distinguish the following situations

- \( \kappa > 0 \): over-relaxation w.r.t. the standard damped relaxation with \( \omega \)
- \( \kappa > \frac{1}{\omega} - 1 \): genuine over-relaxation
- \( \kappa \in ]-1, 0]\): under-relaxation w.r.t. the standard damped relaxation with \( \omega \)
- \( \kappa = -1 \): \( \mathbf{u}^{m+1} = \mathbf{u}^m \)

- \( \kappa < -1 \): the scheme ”goes the other way”. Which is possible if the unmodified scheme was divergent.

**Remark**: Equation (18) is the same formulation as would be used by a two-grid method using aggregation into a single aggregate.

3.2 Convergence of DOUR

From equations (18), (19) and (20) we can deduce that \( \kappa \) can be written as

\[
\kappa = \frac{\langle \Delta \mathbf{u}^m, (\mathbf{f} - A\tilde{\mathbf{u}}^m) \rangle}{\langle \Delta \mathbf{u}^m, A\Delta \mathbf{u}^m \rangle} = \frac{\langle \Delta \mathbf{u}^m, A\tilde{\mathbf{e}}^m \rangle}{\langle \Delta \mathbf{u}^m, A\Delta \mathbf{u}^m \rangle}
\]

When \( A \) is definite-positive this can also be written as

\[
\kappa = \frac{\langle \Delta \mathbf{u}^m, \tilde{\mathbf{e}}^m \rangle_A}{\langle \mathbf{u}^m, \mathbf{u}^m \rangle_A}
\]
Theorem 1
The application of one modified relaxation step as defined by (17) and (18) will return the value of \( \omega_e \) that minimizes the error \( e^{n+1} \) in the energy norm, for any underlying smoother \( S \) as long as \( A \) is definite-positive (and thus induces a norm).

Figure 1 illustrates the principle. We can conclude that the method can be interpreted as a step-length method [11].

![Graphical representation of the DOUR-algorithm](image)

Figure 1: Graphical representation of the DOUR-algorithm

Lemma 1
If the scheme mentioned in (17) and (18) is asymptotically convergent in the energy norm, it is asymptotically convergent in the Euclidean norm and vice versa.

Theorem 2
Suppose that a value of the relaxation parameter \( \omega^* \in \mathbb{R}_o \) exists that allows (16) to be convergent in the energy norm. In that case the DOUR relaxation scheme as defined by (17) and (18) will converge in the energy norm for any relaxation scheme irrespective of the input relaxation parameter \( \omega \) as long as

- \( A \) is hermitian positive-definite (and thus induces a norm)
- a linearly independent set of \( n \) eigenvectors \( \in \mathbb{R}^{n \times 1} \) of \( B \) can be found
- \( \omega \neq 0 \)

Corollary
If the scheme mentioned in (17) and (18) is convergent in the energy norm, it is convergent in the Euclidean norm. This follows immediately from Lemma 1.

The proof of the theorems above will be presented in a forthcoming paper [10].

Theorem 3
Assume \( \nu > 0 \) smoothing steps before the defect correction. If \( A \) minimizes a functional
over a set of functions spanned by the fine grid, then \( \text{RAP} \) minimizes the same functional over the smaller set of functions spanned on the coarser level (variational principle):
\[
\| BS''(e) \|_A = \min_e \| S''(e) - P e_H \|_A
\]  
(24)

**Corollary**
As a consequence the convergence of a two-level (and recursively: \( m \)-level) V-cycle is ensured whenever the relaxation method converges for any interpolation method and smoother as long as the Galerkin construction is used.
For more details and the proof of this theorem and the corollary we refer to [15, 13]

### 3.3 DOUR in a multi-grid context

A priori there is no reason to assume that the method will be a good smoother, as it will try to reduce as much of the error as possible in one step. As such, it will mainly reduce the dominant components, be they smooth or rough.
Tests in section 4 will show if the method is applicable in a multi-grid context.

### 3.4 Cost

The dominant cost of an iterative scheme is the matrix-vector (M-V) multiplication. In most applications the matrix is sparse, hence the exact cost depends on how good the software handles the sparsity. We will allocate one workunit (WU) to a M-V multiplication and neglect the cost of the scalar products. We can then quantify the cost of a damped relaxation as 2 WU (predictor step). The corrector step (\( \kappa \)-modification) also requires two M-V multiplications. The cost of the multi-grid defect correction is more difficult to quantify, depending on the real cost of the transfer operators. As we are only interested in the robustness (convergence) of the scheme, and all variants use the same two-grid defect correction, the exact cost is a moot point and can be estimated as 3 WU.

### 4 Results

The equations under considerations are the two dimensional advection-diffusion equations:
\[
\epsilon (a \cdot \nabla u) + \nabla^2 u = f
\]  
(25)

With \( f \) a (randomly chosen) source term. The advection vector is taken as \( a = \left( \frac{1}{\sqrt{3}}, \frac{3}{\sqrt{3}} \right) \).
(With \( \epsilon = 0 \) we obtain the Poisson equation.) All the tests in this section use the LDA-version of the Residual Distribution Method for the space discretization [9] on a 676 node (structured and unstructured) triangular mesh, and a combination of damped Jacobi relaxations as the iterative solver or smoother unless otherwise stated.
We denote by \( J(\omega_i) \) a Jacobi relaxation with relaxation parameter \( \omega_i \) and by \( J(\omega_i, \kappa) \) a DOUR Jacobi relaxation with input parameter \( \omega_i \). Note that, for the latter, the input parameter is not expected to influence the results. We draw attention to the fact that
we compare with the baseline stencil that contains two relaxation sweeps with possibly different relaxation parameters (schemes SA1 and MG1). Strictly, these are also non-stationary iterative solvers, but with a periodicity of 2, which can be shown to correspond to a two-stage Runge-Kutta style Jacobi relaxation or a semi-iterative method. In the latter guises the Jacobi scheme is still widely used [12], although the basic scheme has long been surpassed. The main advantage of the scheme is the ease with which it can be parallelized, and its insensitive to node ordering, which is especially important for unstructured grids.

Although for \( \epsilon \neq 0 \) the discretization matrix \( A \) is not symmetric positive definite, it might be interesting to see how the DOUR handles these cases. We require the scheme to reduce the L2-norm error to 1 % of the initial value for the stand-alone solver and to 0.01 % for the multi-grid solvers.

4.1 Validation: DOUR as a stand-alone solver

We compare schemes with the following stencils

- **SA1**: \( J(\omega_1) - J(\omega_2) \)
- **SA2**: \( J(\omega_1) - J(\omega_2, \kappa) \)
- **SA3**: \( J(\omega_1, \kappa) - J(\omega_2, \kappa) \)
- **SA4**: a DOUR, not on a single relaxation step, but on the difference between the input and output of 2 consecutive relaxation steps: \( (J(\omega_1) - J(\omega_2)) (\kappa) \)
- **SA5**: a DOUR, on a single relaxation step and on the difference between the input and output of 2 consecutive relaxation steps: \( (J(\omega_1, \kappa) - J(\omega_2, \kappa)) (\kappa) \)
- **SA6**: as previously, but: \( (J(\omega_1) - J(\omega_2, \kappa)) (\kappa) \)
- **SA7**: as previously, but: \( (J(\omega_1, \kappa) - J(\omega_2, \kappa)) (\kappa) \)

We try all possible combinations of \( \omega_1 \) and \( \omega_2 \) between 0 and 2 with a step of 0.05.

Note that for SA2 we might as well have applied the \( \kappa \)-correction to the first iteration, which would asymptotically be the same. Our focus in these results is on structured grids, as these eliminate the influence of the grid itself. Nevertheless, we have run the tests on an unstructured grid as well for validation.

In the tables below we give the optimal results (w.r.t. WU needed for convergence) and the values of \( \omega_1 \) and \( \omega_2 \) for which this was reached. Where the parameter \( \omega_i \) was modified with DOUR we don’t give the optimal value, as the results proved to be independent of the input value, as was expected.

We study the cases for which \( \epsilon = 0 \) (Poisson), 1, 100 and 1000.

We also give the results if we require \( \omega_1 = \omega_2 \).
Table 1: Optimal results for the different stand-alone schemes on the Poisson problem. *: end of range

<table>
<thead>
<tr>
<th>Scheme</th>
<th>SA1</th>
<th>SA2</th>
<th>SA3</th>
<th>SA4</th>
<th>SA5</th>
<th>SA6</th>
<th>SA7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal WU</td>
<td>796</td>
<td>798</td>
<td>1920</td>
<td>756</td>
<td>296</td>
<td>360</td>
<td>1270</td>
</tr>
<tr>
<td>Optimal $\omega_1$</td>
<td>0.65</td>
<td>0.60</td>
<td>N/A</td>
<td>0.75</td>
<td>N/A</td>
<td>0.55</td>
<td>N/A</td>
</tr>
<tr>
<td>Optimal $\omega_2$</td>
<td>2.00</td>
<td>N/A</td>
<td>N/A</td>
<td>1.35</td>
<td>0.65</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 2: Optimal results for the different stand-alone schemes on the Poisson problem if $\omega_1 = \omega_2$.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>SA1</th>
<th>SA2</th>
<th>SA3</th>
<th>SA4</th>
<th>SA5</th>
<th>SA6</th>
<th>SA7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal WU</td>
<td>1056</td>
<td>798</td>
<td>1920</td>
<td>762</td>
<td>296</td>
<td>360</td>
<td>1270</td>
</tr>
<tr>
<td>Optimal $\omega_1$</td>
<td>1.00</td>
<td>0.60</td>
<td>N/A</td>
<td>1.00</td>
<td>0.65</td>
<td>0.55</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 3: Optimal results for the different stand-alone schemes on the advection-diffusion equation with $\epsilon = 1$.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>SA1</th>
<th>SA2</th>
<th>SA3</th>
<th>SA4</th>
<th>SA5</th>
<th>SA6</th>
<th>SA7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal WU</td>
<td>532</td>
<td>528</td>
<td>1275</td>
<td>510</td>
<td>312</td>
<td>288</td>
<td>840</td>
</tr>
<tr>
<td>Optimal $\omega_1$</td>
<td>0.65</td>
<td>0.60</td>
<td>N/A</td>
<td>0.70</td>
<td>N/A</td>
<td>1.20</td>
<td>N/A</td>
</tr>
<tr>
<td>Optimal $\omega_2$</td>
<td>2.00</td>
<td>N/A</td>
<td>N/A</td>
<td>1.85</td>
<td>0.65</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 4: Optimal results for the different stand-alone schemes on the advection-diffusion equation with $\epsilon = 1$ if $\omega_1 = \omega_2$.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>SA1</th>
<th>SA2</th>
<th>SA3</th>
<th>SA4</th>
<th>SA5</th>
<th>SA6</th>
<th>SA7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal WU</td>
<td>56</td>
<td>96</td>
<td>120</td>
<td>108</td>
<td>120</td>
<td>120</td>
<td>160</td>
</tr>
<tr>
<td>Optimal $\omega_1$</td>
<td>0.65</td>
<td>0.95</td>
<td>N/A</td>
<td>0.65</td>
<td>N/A</td>
<td>1.20</td>
<td>N/A</td>
</tr>
<tr>
<td>Optimal $\omega_2$</td>
<td>1.10</td>
<td>N/A</td>
<td>N/A</td>
<td>1.80</td>
<td>1.25</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 5: Optimal results for the different stand-alone schemes on the advection-diffusion equation with $\epsilon = 100$.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>SA1</th>
<th>SA2</th>
<th>SA3</th>
<th>SA4</th>
<th>SA5</th>
<th>SA6</th>
<th>SA7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal WU</td>
<td>56</td>
<td>96</td>
<td>120</td>
<td>108</td>
<td>120</td>
<td>120</td>
<td>160</td>
</tr>
<tr>
<td>Optimal $\omega_1$</td>
<td>0.95</td>
<td>0.95</td>
<td>N/A</td>
<td>1.00</td>
<td>1.25</td>
<td>1.20</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 6: Optimal results for the different stand-alone schemes on the advection-diffusion equation with $\epsilon = 100$ if $\omega_1 = \omega_2$.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>SA1</th>
<th>SA2</th>
<th>SA3</th>
<th>SA4</th>
<th>SA5</th>
<th>SA6</th>
<th>SA7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal WU</td>
<td>60</td>
<td>114</td>
<td>135</td>
<td>108</td>
<td>120</td>
<td>120</td>
<td>150</td>
</tr>
<tr>
<td>Optimal $\omega_1$</td>
<td>0.70</td>
<td>0.85</td>
<td>N/A</td>
<td>0.60</td>
<td>N/A</td>
<td>1.05</td>
<td>N/A</td>
</tr>
<tr>
<td>Optimal $\omega_2$</td>
<td>0.90</td>
<td>N/A</td>
<td>N/A</td>
<td>1.05</td>
<td>1.25</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 7: Optimal results for the different stand-alone schemes on the advection-diffusion equation with $\epsilon = 1000$.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>SA1</th>
<th>SA2</th>
<th>SA3</th>
<th>SA4</th>
<th>SA5</th>
<th>SA6</th>
<th>SA7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal WU</td>
<td>60</td>
<td>114</td>
<td>135</td>
<td>108</td>
<td>120</td>
<td>120</td>
<td>150</td>
</tr>
<tr>
<td>Optimal $\omega_1$</td>
<td>0.80</td>
<td>0.85</td>
<td>N/A</td>
<td>0.95</td>
<td>1.25</td>
<td>1.05</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 8: Optimal results for the different stand-alone schemes on the advection-diffusion equation with $\epsilon = 1000$ if $\omega_1 = \omega_2$.

4.1.1 Discussion

Without going into detail, we can summarize the results as follows: We remark that the baseline scheme (SA1) gets a better relative performance for the optimal combination of parameters when the advection becomes more dominant, but that it is always subject to constraints in the input parameter. Away from the optimal values
(which are not always very obvious \textit{a priori}) its performance degrades quickly; the same goes for SA4. Scheme SA5 has very similar results to SA6 (which is not surprising given their construction).

We take a closer look at the situation where $\omega_1 = \omega_2$.

For the Poisson problem, SA5 might have the better optimum, but it has a worse conditioning with respect to the input (the curve is less flat), and generally SA6 will be somewhat better. SA2 has a broad convergence domain, but is steadily outperformed by other schemes. As could be expected SA3 and SA7 give a constant output, SA3 being the better of the two. With increasing advection, its relative performance increases, and while it might not have the best overall performance, it has a constant output that betters that of the other schemes outside their respective optimum parameter-domain. We illustrate these results for $\epsilon = 0$ and $\epsilon = 1000$.

![Figure 2: Results for different SA schemes. $\omega_1 = \omega_2$. $\epsilon = 0$ (top) and $\epsilon = 1000$ (bottom)](image)

When both parameters take their values independently (semi-iterative method), we noted that SA1 only has a small region w.r.t. $(\omega_1, \omega_2)$ where it is stable. All other schemes, except SA4, are stable over a much wider range (some over the whole range under investigation). Moreover, their sensitivity to the input is much reduced. In the neighborhood of its optimum, SA1 is cheaper than the modified schemes, but this appeared to be only due to the extra cost of computing $\kappa$. It was also noted that, after an initial transient phase, $\kappa$ only varied slowly, raising the possibility to reduce cost by only periodically computing
its value. We illustrate this with scheme SA5 for the Poisson problem. We see that it is better almost everywhere (Figure 3), except in a very narrow parameter-zone, where its advantage is relatively small. On the other hand, where SA1 is worse than SA5 it can be so by a very wide margin. A similar performance is noted for SA6. This performance carries over to small values of $\epsilon$. For higher values of $\epsilon$ the possible gain of the modified scheme is much reduced.

![SA1− SA5 for Poisson (Darker = SA5 less performant)](image)

Figure 3: Performance difference between SA1 and SA5 in WU for the Poisson equation.

We noted that SA4 has a large zone where it is not convergent. It has been verified that it does not diverge as such, but stagnates at a point, stopping the divergence of the original, unmodified, scheme. Tests on an unstructured grid showed that the modified schemes were very insensitive to the composition of the grid.

### 4.2 DOUR in a multi-grid context

While we still use the Jacobi relaxation as previously, we augment it with a 2-grid method using smoothed aggregation, denoted by $DC$.

We compare the following schemes with stencils:

- **MG1**: $J(\omega_1) - DC - J(\omega_2)$
- **MG2**: $J(\omega_1) - DC - J(\omega_2, \kappa)$
- **MG3**: $J(\omega_1, \kappa) - DC - J(\omega_2, \kappa)$
- **MG4**: a DOUR, not on a single relaxation step, but on the difference between the input and output of the V-cycle: $(J(\omega_1) - DC - J(\omega_2))(\kappa)$
- **MG5**: $(J(\omega_1, \kappa) - DC - J(\omega_2))(\kappa)$
- **MG6**: $(J(\omega_1) - DC - J(\omega_2, \kappa))(\kappa)$
• MG7: \((J(\omega_1, \kappa) - DC - J(\omega_2, \kappa))(\kappa)\)

where we try all possible combinations of \(\omega_1\) and \(\omega_2\) between 0 and 2 with a step of 0.05. Again we take the following values of \(\epsilon\): 0, 1, 100, 1000.

<table>
<thead>
<tr>
<th>minimal WU</th>
<th>MG1</th>
<th>MG2</th>
<th>MG3</th>
<th>MG4</th>
<th>MG5</th>
<th>MG6</th>
<th>MG7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>112</td>
<td>90</td>
<td>154</td>
<td>108</td>
<td>99</td>
<td>99</td>
<td>143</td>
</tr>
<tr>
<td>optimal (\omega_1)</td>
<td>0.50</td>
<td>0.55</td>
<td>N/A</td>
<td>0.50</td>
<td>N/A</td>
<td>0.90</td>
<td>N/A</td>
</tr>
<tr>
<td>optimal (\omega_2)</td>
<td>1.35</td>
<td>N/A</td>
<td>N/A</td>
<td>1.60</td>
<td>0.95</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 9: Optimal results for the different multi-grid schemes on the Poisson problem.

<table>
<thead>
<tr>
<th>minimal WU</th>
<th>MG1</th>
<th>MG2</th>
<th>MG3</th>
<th>MG4</th>
<th>MG5</th>
<th>MG6</th>
<th>MG7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>112</td>
<td>90</td>
<td>154</td>
<td>108</td>
<td>99</td>
<td>99</td>
<td>143</td>
</tr>
<tr>
<td>optimal (\omega)</td>
<td>0.80</td>
<td>0.55</td>
<td>N/A</td>
<td>0.80</td>
<td>0.95</td>
<td>0.90</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 10: Optimal results for the different multi-grid schemes on the Poisson problem if \(\omega_1 = \omega_2\).

<table>
<thead>
<tr>
<th>minimal WU</th>
<th>MG1</th>
<th>MG2</th>
<th>MG3</th>
<th>MG4</th>
<th>MG5</th>
<th>MG6</th>
<th>MG7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>112</td>
<td>99</td>
<td>154</td>
<td>99</td>
<td>110</td>
<td>110</td>
<td>143</td>
</tr>
<tr>
<td>optimal (\omega_1)</td>
<td>0.55</td>
<td>0.55</td>
<td>N/A</td>
<td>0.50</td>
<td>N/A</td>
<td>0.65</td>
<td>N/A</td>
</tr>
<tr>
<td>optimal (\omega_2)</td>
<td>1.40</td>
<td>N/A</td>
<td>N/A</td>
<td>1.50</td>
<td>0.80</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 11: Optimal results for the different multi-grid schemes on the advection-diffusion equation with \(\epsilon = 1\).

<table>
<thead>
<tr>
<th>minimal WU</th>
<th>MG1</th>
<th>MG2</th>
<th>MG3</th>
<th>MG4</th>
<th>MG5</th>
<th>MG6</th>
<th>MG7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>112</td>
<td>99</td>
<td>154</td>
<td>108</td>
<td>110</td>
<td>110</td>
<td>143</td>
</tr>
<tr>
<td>optimal (\omega)</td>
<td>0.80</td>
<td>0.55</td>
<td>N/A</td>
<td>0.80</td>
<td>0.80</td>
<td>0.65</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 12: Optimal results for the different multi-grid schemes on the advection-diffusion equation with \(\epsilon = 1\) if \(\omega_1 = \omega_2\).

<table>
<thead>
<tr>
<th>minimal WU</th>
<th>MG1</th>
<th>MG2</th>
<th>MG3</th>
<th>MG4</th>
<th>MG5</th>
<th>MG6</th>
<th>MG7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>245</td>
<td>360</td>
<td>605</td>
<td>261</td>
<td>330</td>
<td>319</td>
<td>390</td>
</tr>
<tr>
<td>optimal (\omega_1)</td>
<td>0.50</td>
<td>0.15</td>
<td>N/A</td>
<td>0.55</td>
<td>N/A</td>
<td>0.70</td>
<td>N/A</td>
</tr>
<tr>
<td>optimal (\omega_2)</td>
<td>0.70</td>
<td>N/A</td>
<td>N/A</td>
<td>1.90</td>
<td>0.10</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 13: Optimal results for the different multi-grid schemes on the advection-diffusion equation with \(\epsilon = 100\).

<table>
<thead>
<tr>
<th>minimal WU</th>
<th>MG1</th>
<th>MG2</th>
<th>MG3</th>
<th>MG4</th>
<th>MG5</th>
<th>MG6</th>
<th>MG7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>245</td>
<td>360</td>
<td>605</td>
<td>261</td>
<td>330</td>
<td>319</td>
<td>390</td>
</tr>
<tr>
<td>optimal (\omega)</td>
<td>0.60</td>
<td>0.15</td>
<td>N/A</td>
<td>0.90</td>
<td>0.70</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 14: Optimal results for the different multi-grid schemes on the advection-diffusion equation with \(\epsilon = 100\) if \(\omega_1 = \omega_2\).
R. Haelterman, J. Vierendeels, D. Van Heule

Table 15: Optimal results for the different multi-grid schemes on the advection-diffusion equation with $\epsilon = 1000$.  

<table>
<thead>
<tr>
<th>Scheme</th>
<th>MG1</th>
<th>MG2</th>
<th>MG3</th>
<th>MG4</th>
<th>MG5</th>
<th>MG6</th>
<th>MG7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal WU</td>
<td>287</td>
<td>432</td>
<td>979</td>
<td>306</td>
<td>385</td>
<td>374</td>
<td>468</td>
</tr>
<tr>
<td>Optimal $\omega_1$</td>
<td>0.50</td>
<td>0.10</td>
<td>N/A</td>
<td>0.55</td>
<td>N/A</td>
<td>0.70</td>
<td>N/A</td>
</tr>
<tr>
<td>Optimal $\omega_2$</td>
<td>0.65</td>
<td>N/A</td>
<td>N/A</td>
<td>1.70</td>
<td>0.75</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 16: Optimal results for the different multi-grid schemes on the advection-diffusion equation with $\epsilon = 1000$ if $\omega_1 = \omega_2$.  

<table>
<thead>
<tr>
<th>Scheme</th>
<th>MG1</th>
<th>MG2</th>
<th>MG3</th>
<th>MG4</th>
<th>MG5</th>
<th>MG6</th>
<th>MG7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal WU</td>
<td>294</td>
<td>432</td>
<td>801</td>
<td>306</td>
<td>385</td>
<td>374</td>
<td>468</td>
</tr>
<tr>
<td>Optimal $\omega$</td>
<td>0.55</td>
<td>0.10</td>
<td>N/A</td>
<td>0.85</td>
<td>0.75</td>
<td>0.70</td>
<td>N/A</td>
</tr>
</tbody>
</table>

4.2.1 Discussion

Space precludes us from giving all numerical results, but we can summarize them as follows:

The baseline scheme (MG1) is very sensitive to the input parameters. The other schemes (except for MG4) are stable over a wider range of parameters. Schemes MG5, MG6 and MG7 are particularly well conditioned. Note that none of the schemes, including the baseline, perform particularly well for the high advection case. This is most likely due to the fact that the defect correction is not particularly suited to these kind of problems (which was not under investigation).

We noted that the computed values of $\kappa$ vary only slowly after an initial transient phase. As the computation of $\kappa$ is responsible for an extra cost of 25 to 100 % over the baseline scheme, we will try to gain some efficiency by only computing the value once in a while. We also drop schemes MG2 and MG3 because of their low performance and MG4 because it is not unconditionally convergent.
4.3 Multi-grid DOUR with periodically updated $\kappa$

When we plot the evolution of the $\kappa$-parameters for schemes MG5 and MG7 (for $\epsilon = 1$), we see that after a small number of iterations their value is almost constant. (By $\kappa_i$ we mean the DOUR modification on the $i$-th Jacobi iteration, and by $\kappa_t$ a DOUR applied to the complete MG-cycle.) For MG5 we take the initial values $(\omega_1, \omega_2) = (1, 0.65)$, for MG7 these are $(0.1, 0.1)$. (Note that only $\omega_2$ for MG5 has any actual influence on the convergence.)

![Figure 5: The evolution of the $\kappa$-parameters for schemes MG5 and MG7. $\epsilon = 1$]
While we did not exactly determine the length of the transient phase for every single case, it appeared to be close to 4-5 iterations. For that reason, we tried to fix the values of $\kappa$ after a small number of iterations for MG5, MG6 and MG7. We called the modified schemes MG5e, MG6e and MG7e respectively. While for MG5e and MG6e we gained in cost for most combinations of $\omega_1$ and $\omega_2$, we completely lost the robustness, i.e. for a small number of input parameters the convergence was very bad. MG7e, being independent of the input parameters, gave very good results when we fixed the transient phase to 4 iterations. Only for a very narrow range of parameters could the baseline scheme beat it, and then only by less than 10 %. We show the results for $\omega_1 = \omega_2$ in figure 6.

![Comparison of MG1 and MG7e](image)

**Figure 6:** Results for MG7e vs MG1. $\omega_1 = \omega_2$. $\epsilon = 0, 1, 100, 1000$

### 4.4 DOUR with Gauss-Seidel smoother

As the scheme is applicable to any damped smoother we also try it on damped Symmetric Gauss-Seidel instead of Jacobi. The schemes with the most stable output were MG3,5 and 7. Again they would need to have the computation of $\kappa$ frozen after some iterations to make them computational competitive with MG1 using the optimal combinations of the relaxation parameters.

## 5 CONCLUSIONS

In this first attempt to automatize the relaxation parameters of an iterative scheme we used a simple aggregation procedure using a single aggregate and a prolongation vector based on the output of a preceding relaxation sweep. We applied the scheme to a damped Jacobi and damped Gauss-Seidel iterative method, as a stand-alone solver and as a smoother for multi-grid, allowing for two different relaxation parameters during the iteration in a periodic fashion, thus creating a semi-iterative relaxation. We found that by varying the exact application of the method we could make the scheme independent or almost independent of the user defined input parameters. A theoretical explanation for these results was presented without proof.
We also found that, while $\omega < 1$ is needed in order to ensure convergence of a single damped Jacobi relaxation sweep, this is no longer the case when we modify it to a semi-iterative relaxation. Indeed, combinations including $\omega > 1$ sometimes gave the best results. More often than not, using $\omega_1 \neq \omega_2$ gave superior results over $\omega_1 = \omega_2$.

REFERENCES


