BIT
2002, Vol. 42, No. 3, pp. 541–560
0006-3835/02/4203-0541 $16.00

MATRIX MULTILEVEL METHODS AND PRECONDITIONING *

THOMAS HUCKLE and JOCHEN STAUDACHER †

Institut für Informatik, Technische Universität München, D-80290 München, Germany. email: huckle@in.tum.de, staudacj@in.tum.de

Abstract.

The Matrix Multilevel approach is based on a purely matrix dependent description of multigrid methods. The formulation of multilevel methods as singular matrix extensions via generating systems leads to the description of the method as a preconditioned iterative scheme, and illuminates the significance of the used prolongation and restriction operator for the related preconditioner. We define the matrix dependent black box restriction C by shifting the original matrix A in the form $B = \alpha I - A$ and picking out every second column to $C = B(:, 2 : 2 : n)$. Here, α has to be chosen as a rough upper estimate of the largest eigenvalue of A. By this mapping the related preconditioner enlarges the small eigenvalues while the maximum eigenvalue remains nearly unchanged. Although we derive our method in an additive setting, we can also use the new prolongations/restrictions in multiplicative algorithms. Our test results are very promising: We give various numerical examples where multigrid with standard prolongation/restriction deteriorates whereas the new method shows optimal behaviour. We also notice that in many cases using $B = abs(A)$ instead of $B = \alpha I - A$ gives equally good results. We mainly consider symmetric positive definite matrices in one and two dimensions, but the results can be generalized to higher dimensional problems.

AMS subject classification: 65F10.

Key words: Algebraic multigrid, elliptic pde, iterative methods, preconditioning.

1 The additive two-level method.

We consider a linear equation $Ax = b$ with a sparse ill-conditioned $n \times n$ matrix A. The aim is to design a purely algebraic multilevel method that can be applied to any matrix in order to reduce the condition number. Here we restrict ourselves to the symmetric positive definite case.

Multigrid methods allow the fast $O(n)$ solution of linear equations arising from elliptic PDE (see [2, 3, 13, 14]). The technique uses a sequence of grids, and the restrictions and prolongations between the original problem formulated on the different grids. We can consider such methods purely algebraically based on the matrix A without any geometrical information—the well-known approach of algebraic multigrid (see [17, 11]). We start in an additive setting.

Let us first consider only the transfer operators without any smoothing. In the symmetric case our method is based on a mapping C for the restriction and

*Received May 1998. Communicated by Petter E. Bjørstad.
†The work of this author was partially supported by the DAAD via grant D/00/20283.

prolongation of the original linear system on a coarser problem. Then we get C^TAC, e.g. as an $n/2 \times n/2$ matrix related to the original problem formulated on a coarse grid. Following Griebel [8, 9] and the idea of understanding multigrid algorithms as iterative methods on generating systems we can write the sequence of matrices on different levels also as a sequence of matrix extensions of the form

$$A^{(1)} = A, \qquad A^{(2)} = \begin{pmatrix} A & AC \\ C^TA & C^TAC \end{pmatrix} = \begin{pmatrix} I \\ C^T \end{pmatrix} A \begin{pmatrix} I & C \end{pmatrix}. \tag{1.1}$$

Let us first analyse the relation between the original equation $Ax = b$ and the extended matrix $A^{(2)}$. If $(y^T\ z^T)^T$ is a solution of the extended system

$$\begin{pmatrix} A & AC \\ C^TA & C^TAC \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix},$$

we have to set $a = C^Tb$, and then $x = y + Cz$ gives us the solution of the original problem $Ax = b$. Furthermore, in view of (1.1), the kernel of $A^{(2)}$ is spanned by the vectors that fulfill $y = -Cz$, and hence the kernel is given by all vectors of the form $\begin{pmatrix} -C \\ I \end{pmatrix} z$. Similarly with (1.1) the range of $A^{(2)}$ is of the form $\begin{pmatrix} I \\ C^T \end{pmatrix} x$.

The convergence of an iterative method applied on the extended linear system depends on the generalized condition number—the quotient of λ_{max} over the smallest nonzero eigenvalue. To describe the nonzero eigenvalues of $A^{(2)}$ we consider the Rayleigh Quotient relative to the space that is orthogonal to the null space $y = \begin{pmatrix} I \\ C^T \end{pmatrix} x$. With

$$\frac{y^T \begin{pmatrix} I \\ C^T \end{pmatrix} A \begin{pmatrix} I & C \end{pmatrix} y}{y^T y} = \frac{x^T \begin{pmatrix} I & C \end{pmatrix} \begin{pmatrix} I \\ C^T \end{pmatrix} A \begin{pmatrix} I & C \end{pmatrix} \begin{pmatrix} I \\ C^T \end{pmatrix} x}{x^T \begin{pmatrix} I & C \end{pmatrix} \begin{pmatrix} I \\ C^T \end{pmatrix} x}$$

$$= \frac{x^T(I + CC^T)A(I + CC^T)x}{x^T(I + CC^T)x} = \frac{z^T(I + CC^T)^{1/2}A(I + CC^T)^{1/2}z}{z^Tz},$$

we see that the nonzero part of the spectrum of $A^{(2)}$ is given by the eigenvalues of

$$(I + CC^T)A, \tag{1.2}$$

and furthermore the nonzero eigenvalues of $A^{(2)}$ are related to the eigenvalues of A by $\lambda(A^{(2)}) = \lambda(A)(1 + \epsilon)$ with $0 \le \epsilon \le \lambda_{max}(CC^T)$.

Now we can think of the prolongation C as a preconditioner of the form $I + CC^T$ applied to the original matrix A (see also [1, 15]). A good preconditioner would be one that enlarges only the small eigenvalues of A without changing $\lambda_{max}(A)$. Hence the main task is to find a sparse matrix C such that its range contains the span of the eigenvectors associated with the small eigenvalues of the matrix A. Similar problems are considered and solved in [5]; but to obtain the exact solution would be too expensive and cannot be used here.

Special case: C is an $n \times 1$ matrix of rank 1. In this case the matrix C is reduced to one column vector. Then an optimal preconditioner should enlarge

$\lambda_1 = \lambda_{min}$ without changing $\lambda_n = \lambda_{max}$. Based on the eigensystem for A of the form $A = Q^T \Lambda Q$ the problem can be written as follows: Find a vector w such that the matrix $\tilde{\Lambda} = (\, I \quad w \,)^T \, \Lambda \, (\, I \quad w \,)$ has minimum condition number (neglecting the zero eigenvalues.) In view of the interlace property (see, e.g., [16]) we get

$$0 = \tilde{\lambda}_1 \le \lambda_1 \le \tilde{\lambda}_2 \le \lambda_2, \quad \tilde{\lambda}_n \le \lambda_n \le \tilde{\lambda}_{n+1},$$

and the new condition number is bounded by

$$\mathrm{cond}\big((I + ww^T)\Lambda\big) = \frac{\tilde{\lambda}_{n+1}}{\tilde{\lambda}_2} \ge \frac{\lambda_n}{\lambda_2}.$$

An optimal solution is therefore given by $C = \rho u_{\min}$, where u_{min} is an eigenvector related to the smallest eigenvalue λ_1. Then ρ has to be chosen such that $\lambda_2 \le (1 + \rho^2)\lambda_1 \le \lambda_n$, and the condition number is improved by a factor λ_1/λ_2.

In general we are interested in prolongations C with larger rank. This is also necessary to lead to a notable improvement of the condition number for ill-conditioned matrices. Therefore we now write C in the form

$$C = B\,P = B(\,:\,, J) \tag{1.3}$$

with an $n \times n$ matrix B and an elementary projection P. Now multiplying with P from the right yields a reduction to the columns given by the index set J.

Special case: C is an $n \times n$ matrix. To make things easier we first describe the case that $P = I$ and $C = B$. Now we can choose $B = \beta(\alpha I - A)$ with $\alpha = \lambda_{max}(A)$. This matrix has the desired property: λ_{min} becomes large in (1.2) and λ_{max} remains the same. For this special case we can fully analyse the resulting preconditioned system in order to find an optimal value for β.

Let u be any eigenvector of A with length 1 related to an eigenvalue λ, $\lambda_1 \le \lambda \le \alpha$. Then β has to be chosen as large as possible with

$$u^T \left(I + \beta^2(\alpha I - A)(\alpha I - A)^T\right) Au = \lambda\Big(1 + \beta^2(\alpha - \lambda)^2\Big) \le \alpha. \tag{1.4}$$

Hence, $\beta^2 \le \frac{1}{\lambda(\alpha - \lambda)}$. The function on the right hand side takes its minimum value for $\lambda = \alpha/2$, which leads to the optimal value $\beta = 2/\alpha$. The change of the eigenvalues of A under the transformation (1.2) is described by

$$\lambda \;\longrightarrow\; f(\lambda) = \lambda\left(1 + \frac{4}{\alpha^2}(\alpha - \lambda)^2\right).$$

In the interval $[\lambda_1, \alpha]$ the function f has a relative maximum at $\lambda = \alpha/2$ of size $f(\alpha/2) = \alpha$, a relative minimum for $\lambda = 5\alpha/6$ with $f(5\alpha/6) = 50\alpha/54$, a global maximum for $\lambda = \alpha$ with $f(\alpha) = \alpha$, and a global minimum for $\lambda = \lambda_1$ with $f(\lambda_1) = \lambda_1(1 + \frac{4}{\alpha^2}(\alpha - \lambda_1)^2) \approx 5\lambda_1$. Hence, by applying $I + BB^T$ as a preconditioner the condition number is approximately improved by a factor of 5. Note that not only the smallest eigenvalue is enlarged, but the whole spectrum is compressed.

For example, all eigenvalues of A in the interval $[\alpha/8, \alpha]$ are mapped into the interval $[65\alpha/128, \alpha] \approx [\alpha/2, \alpha]$, and in the interval $[\alpha/3, \alpha]$ into $[25/27\alpha, \alpha]$.

Now let us return to the two-level approach with P and index set J. Then the above relation (1.4) translates into

$$u^T(I + \beta^2(\alpha I - A)PP^T(\alpha I - A)^T)Au \;=\; \lambda(1 + \beta^2(\alpha - \lambda)^2\|P^T u\|^2) \leq \alpha\,. \tag{1.5}$$

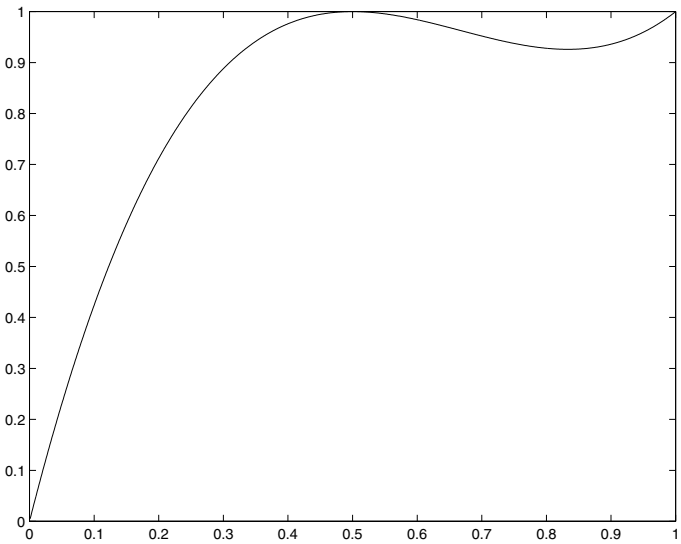

Figure 1.1: Function $f(\lambda)$ for $\alpha = 1$.

Therefore P has to be chosen carefully in such a way that for every small eigenvalue $P^T u$ does not become too small. If the eigenvectors are continuous in the sense that they can be seen as values $g(j/n)$ for a continuous function g, then e.g. $P = I(:, 2:2:n)$ gives $\|P^T u\|^2 \approx 1/2$ for all eigenvectors. This leads to an optimal value of $\beta^2 = 8/\alpha^2$ and we can expect that the smallest eigenvalue is approximately improved by a factor of 5; but now the related mapping C has only half the number of entries. Note, that for $A = (1/4) * \text{tridiag}(-1, 2, -1)$ this optimal factor 8 also appears by diagonal (Jacobi) scaling of the extended system (1.1). In this case the spectrum of $A^{(1)}$ is no longer contained in the interval $[5\lambda_{min}, \alpha]$, but all eigenvalues of A_2 are again smaller than α.

Note that the eigenvalues of $A_2 = C^T AC$ are closely related to the function $g(\lambda) = \beta^2(\alpha - \lambda)^2\lambda$, and therefore the spectrum of A_2 is contained in the interval

$$[\beta^2(\alpha - \lambda_{min})^2\lambda_{min}(A), (\beta\alpha)^2(4/27)\alpha]\,.$$

For $P = I(:, 2:2:n)$, an eigenvector u corresponding to a small eigenvalue of A leads to a small value of the Rayleigh Quotient related to the matrix A_2 and the

vector $P^T u$. Therefore we may expect that $P^T u$ is mainly contained in a subspace spanned by eigenvectors of A_2 that belong to small eigenvalues.

We have seen that the matrix B should reverse the order of the eigenvalues of A. For many examples there is another easy way to derive such a matrix B by defining $B = |A|$ the matrix with the entries $B_{i,j} = |A_{i,j}|$ (see [7, 13] for such matrix dependent prolongation/restriction operators). For $A = \text{tridiag}(-1, 2, -1)$ this leads to the standard prolongation $B = \text{tridiag}(1, 2, 1)$. Hence, for many matrices A we can expect a similar behaviour of the Matrix Multilevel approach related to $B = |A|$ and to $B = \alpha I - A$ after diagonal scaling, because the diagonal scaling transforms A nearly to $\text{tridiag}(-1, 2, -1)$.

We finally remark that we will not have the optimal factor β in any practical algorithm. We shall see in the following section how this role will more or less be taken over by diagonal scalings.

2 The multilevel method: additive and multiplicative variants.

2.1 Additive multilevel algorithms.

The analysis of the previous section describes an additive two-level method where the smoothing is reduced to a scaling factor. Now we have to generalize the approach in order to make it work in a multilevel fashion. So far we have arrived at the representation $A = A_1 = A^{(1)}$, $A_2 = C_1^T A_1 C_1$,

$$A^{(2)} = \begin{pmatrix} A_1 & A_1 C_1 \\ C_1^T A_1 & C_1^T A_1 C_1 \end{pmatrix} = \begin{pmatrix} I & C_1 \end{pmatrix}^T A_1 \begin{pmatrix} I & C_1 \end{pmatrix},$$

or in preconditioned form

$$(I + C_1 C_1^T) A_1.$$

Let us assume that the prolongation C_1 is chosen properly such that its range contains the eigenvectors associated with the small eigenvalues of A. Then, on the next level we can restrict ourselves to operators of the form $C = C_1 C_2$. Now we can apply (1.2) a second time and arrive at a preconditioner

$$(I + C_1 C_2 C_2^T C_1^T)(I + C_1 C_1^T) A_1 = (I + C_1 C_1^T + C_1 C_2 C_2^T C_1^T + C_1 C_2 C_2^T C_1^T C_1 C_1^T) A_1.$$

To make the preconditioner symmetric positive definite we delete the last nonsymmetric term and use only

$$I + C_1 C_1^T + C_1 C_2 C_2^T C_1^T = I + C_1 (I + C_2 C_2^T) C_1^T.$$

Then we have different formulas for the extended system:

$$\begin{aligned} A^{(3)} &= \begin{pmatrix} A & AC_1 & AC_1C_2 \\ C_1^T A & C_1^T AC_1 & C_1^T AC_1C_2 \\ C_2^T C_1^T A & C_2^T C_1^T AC_1 & C_2^T C_1^T AC_1C_2 \end{pmatrix} \\ &= \begin{pmatrix} I \\ C_1^T \\ C_2^T C_1^T \end{pmatrix} A \begin{pmatrix} I & C_1 & C_1C_2 \end{pmatrix} \end{aligned}$$

$$
\begin{aligned}
&= (\,I \quad C_1\,(\,I \quad C_2\,)\,)^T\, A\,(\,I \quad C_1\,(\,I \quad C_2\,)\,)\\
&= \begin{pmatrix} I & 0 & 0 \\ 0 & I & C_2 \end{pmatrix}^T \begin{pmatrix} A & AC_1 \\ C_1^T A & C_1^T A C_1 \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ 0 & I & C_2 \end{pmatrix}\\
&= \begin{pmatrix} I & 0 & 0 \\ 0 & I & C_2 \end{pmatrix}^T (\,I \quad C_1\,)^T\, A\,(\,I \quad C_1\,) \begin{pmatrix} I & 0 & 0 \\ 0 & I & C_2 \end{pmatrix}, \qquad (2.1)
\end{aligned}
$$

and in preconditioned form

$$(I + C_1C_1^T + C_1C_2C_2^TC_1^T)A = (I + C_1(I + C_2C_2^T)C_1^T)A$$

or

$$\begin{pmatrix} I & 0 \\ 0 & I + C_2C_2^T \end{pmatrix} \begin{pmatrix} A & AC_1 \\ C_1^TA & A_2 \end{pmatrix} = \begin{pmatrix} A & AC_1 \\ (I + C_2C_2^T)C_1^TA & (I + C_2C_2^T)A_2 \end{pmatrix}. \qquad (2.2)$$

This leads to different heuristics for choosing C_2. In view of (2.2) we can think of C_2 as a second preconditioning step related to $A_2 = C_1^TAC_1$ and therefore we can set

$$C_2 = \beta_2(\alpha_2 I - A_2)\,P_2. \qquad (2.3)$$

We can also derive (2.3) based on another approach. The new prolongation C_2 defines the preconditioned system

$$(I + C_1C_1^T + C_1C_2C_2^TC_1^T)A$$

and thus in the sense of (1.4) and (1.5) we get

$$
\begin{aligned}
u^TC_1C_2C_2^TC_1^Tu &= \beta_1^2u^T(\alpha I - A)PC_2C_2^TP^T(\alpha I - A)^Tu\\
&= \beta_1^2(\alpha - \lambda_{min})^2(u_{min}^TP)C_2C_2^T(P^Tu_{min})\\
&= \beta_1^2(\alpha - \lambda_{min})^2(u_{min}^TP_1)B_2P_2P_2^TB_2(P_1^Tu_{min}).
\end{aligned}
$$

Now B_2 should be chosen in such a way that it gets large for the vectors P_1^Tu related to small eigenvalues of A. In view of the previous remark at the end of Section 1 we can expect that the vectors P_1^Tu are related to small eigenvalues of A_2 which again suggests to define C_2 via (2.3).

We could also formulate another way for choosing C_2. Note that the eigenvalues of A_2 are closely related to the function $g(\lambda) = \beta^2(\alpha - \lambda)^2\lambda$. This shows that the large eigenvalues of A are also translated into very small eigenvalues of A_2. If we define C_2 with (2.3), then in this second step we try to enlarge these originally large eigenvalues together with the small eigenvalues of A. This suggests another way to define C_2, namely again as a projection of the first level-matrix $\alpha I - A$. For example, if A is a Toeplitz matrix then we can consider the Toeplitz matrix $\alpha I - A = T$ and choose C_j as a submatrix $T(1 : 2^l, 2 : 2 : 2^l)$ (for similar multigrid methods for Toeplitz matrices; see [4] or [6]).

In order to obtain similar improvements on the condition number on every level it is necessary that all the derived smaller systems have similar properties as the original matrix A. For example, if A_2 is well conditioned then obviously going

to a coarser grid will lead to no improvement of the spectrum. Hence we have to choose C and P in such a way that the matrix $\hat{A} = P^T B^T ABP$ inherits important properties of A. In many cases the behaviour of A on the vector $e_n = (1, \ldots, 1)^T$ is very important—this is related to the property that the rowsum of entries is often zero. Hence we may ask that

$$e_{n/2}^T \hat{A} e_{n/2} = e_n(J)^T B^T ABe_n(J) \approx e_n^T Ae_n/2.$$

We obtain this property by choosing B such that $Be_n(J) = e_{n/2}/\sqrt{2}$. For $B = \sqrt{2}*$ tridiag$(1/4, 1/2, 1/4)$ and $J = (2, 4, 6, \ldots., n)$ (the usual multigrid prolongation) this is obviously fulfilled. After diagonal scaling this is also nearly satisfied for both mappings $B = \lambda_{max} I - A$ and $B = |A|$ in many cases.

Now we have defined a multilevel method based only on the original matrix A and the maximum eigenvalues of the resulting systems A_j. It is necessary to include also some kind of smoothing operation on every level in order to get fast convergence. Here we will mainly consider the Jacobi method for smoothing. In (1.1) or (2.1) the Jacobi smoothing is nothing else than diagonal preconditioning. Note that in the same way one can use Gauss–Seidel or any other level-wise method.

To include Jacobi smoothing on every level let us consider the enlarged problem

$$A^{(k)} = \begin{pmatrix} A & AC_1 & AC_1C_2 & \cdots & AC_1..C_k \\ C_1^T A & C_1^T AC_1 & C_1^T AC_1C_2 & \cdots & C_1^T AC_1..C_k \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ C_k^T..C_1^T A & \cdots & \cdots & & C_k^T..C_1^T AC_1..C_k \end{pmatrix}$$

$$= (\, I \quad C_1 \quad C_1C_2 \quad \cdots \quad C_1..C_k \,)^T A \,(\, I \quad C_1 \quad C_1C_2 \quad \cdots \quad C_1..C_k \,)$$

$$= (\, I \quad C_1(I\, C_2(I \,\cdots\, (I\, C_k)\cdots)) \,)^T A \,(\, I \quad C_1(I\, C_2(I \,\cdots\, (I\, C_k)\cdots)) \,), \tag{2.4}$$

and in preconditioned form

$$(I + C_1C_1^T + C_1C_2C_2^T C_1^T + \,\cdots\, + C_1 \cdots C_k C_k^T \cdots C_1^T)A$$

$$= (I + C_1(I + C_2(I + \,\cdots\, (I + C_kC_k^T)\cdots)C_2^T)C_1^T)A = M^{(k)}A\,. \tag{2.5}$$

In the form (2.4) we can comprise any preconditioner on the matrix $A^{(k)}$, for example Jacobi, Gauss–Seidel or ILU preconditioners, and employ the conjugate gradient method with zero starting vector. But usually we want to compute only the small matrices A_j and the prolongations C_j on every level, but neither the whole system $A^{(k)}$ nor the—nearly dense—preconditioner $M^{(k)}$. Therefore, we will use only level-wise block-diagonal preconditioners based on the level matrices A_j.

From (2.4) we can translate preconditioners very easily to the form (2.5). In the Jacobi case for example we have $D_j = \text{diag}(A_j) = \text{diag}(C_j^T \cdots C_1^T AC_1 \cdots C_j)$ and for every matrix A_j we can use $D_j^{-1/2}$ as left and right preconditioner. Then, with

$$D = \text{diag}(D_1^{-1/2}, \ldots, D_k^{-1/2}),$$

(2.4) translates into

$$D \begin{pmatrix} A & AC_1 & AC_1C_2 & \cdots & AC_1..C_k \\ C_1^T A & C_1^T AC_1 & C_1^T AC_1C_2 & \cdots & C_1^T AC_1..C_k \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ C_k^T \cdots C_1^T A & \cdots & \cdots & & C_k^T \cdots C_1^T AC_1 \cdots C_k \end{pmatrix} D$$

$$= \begin{pmatrix} I & D_1^{1/2} C_1 D_2^{-1/2} & D_2^{1/2} C_2 D_3^{-1/2} & \cdot \end{pmatrix}^T D_1^{-1/2} A D_1^{-1/2} \begin{pmatrix} I & D_1^{1/2} C_1 D_2^{-1/2} & \cdot \end{pmatrix}.$$

Hence, we only have to replace A by $\tilde{A} = D_1^{-1/2} A D_1^{-1/2}$, and each C_j by $\tilde{C}_j = D_j^{1/2} C_j D_{j+1}^{-1/2}$. This leads to the new preconditioned form

$$(I + \tilde{C}_1(I + \tilde{C}_2(I + \cdots (I + \tilde{C}_k \tilde{C}_k^T) \cdots) \tilde{C}_2^T) \tilde{C}_1^T) \tilde{A}. \tag{2.6}$$

It turns out that we can improve the preconditioner (2.6) a lot by using the diagonally scaled matrix $\tilde{A}_j$ on every level for constructing the next prolongation/restriction: Note that the Jacobi scaling is necessary to ensure that the matrices $\tilde{A}_j$ share the same properties—in the sense that the rowsums are nearly zero. (Somehow this is the same idea that describes the difference between Jacobi and Gauss–Seidel smoothing: Try to use always the most recent information about the linear system that is available on every level.) This means that if we have arrived at the matrix A_j we should use the (already Jacobi scaled) matrix $\tilde{A}_j$ for the next step and therefore define the next prolongation via $B = \lambda I - \tilde{A}$ and apply it to the diagonally scaled $A^{(j)}$ like in (2.4). It is possible to include the Jacobi scaling directly in equation (2.6) in the form

$$(D_1^{-1} + \tilde{C}_1(D_2^{-1} + \tilde{C}_2(D_3^{-1} + \cdots (D_k^{-1} + \tilde{C}_k \tilde{C}_k^T) \cdots) \tilde{C}_2^T) \tilde{C}_1^T) A \tag{2.7}$$

The equation (2.7) leads to a first Matrix Multilevel Algorithm:

ALGORITHM 2.1. We start with the scaled matrix $\tilde{A}_1$.

- On every level compute $B_j = \alpha I - \tilde{A}_j$ and use $C_j = B_j(:, 2 : 2 : n)$ for prolongation and restriction.
- After computation of $A_{j+1} = C_j^T \tilde{A}_j C_j$ we step forward with the scaled matrix $\tilde{A}_{j+1}$.
- Based on the matrices C_j and $D_{j+1} = \mathrm{diag}(C_j^T \tilde{A}_j C_j)$, $j = 0, \ldots, l$, we can recursively implement the multiplication of the matrix (2.7) with a given vector and thus use the preconditioned CG method.

Similarly to (2.6), we can apply any level-wise direct preconditioner M_j on the matrix $\tilde{A}_j$ and get

$$(M_1 + \tilde{C}_1(M_2 + \tilde{C}_2(M_3 + \cdots (M_k + \tilde{C}_k \tilde{C}_k^T) \cdots) \tilde{C}_2^T) \tilde{C}_1^T) \tilde{A}.$$

Furthermore we can include, e.g. two smoothing steps on every level by replacing M_k by $2M_k - M_k A_k M_k$ in order to imitate a two-fold application of the preconditioner M_k. If M_k is not symmetric, we can use instead the symmetrized matrix $M_k + M_k^T - M_k A_k M_k^T$.

For the Jacobi or Gauss–Seidel iteration we often introduce damping factors. The same is possible here if we replace the preconditioner in (2.6) by

$$(I + \omega_1 \tilde{C}_1 (I + \omega_2 \tilde{C}_2 (I + \cdots) \tilde{C}_2^T) \tilde{C}_1^T) = (I + \omega_1 \tilde{C}_1 \tilde{C}_1^T + \omega_2 \tilde{C}_1 \tilde{C}_2 \tilde{C}_2^T \tilde{C}_1^T + \cdots).$$

In view of (1.1) and the analysis of Section 1 a factor $\omega < 1$ may be necessary to reduce the maximum eigenvalue to be $\leq \alpha$. A factor $\omega > 1$ can be helpful for faster convergence if it is possible to enlarge the small eigenvalues without changing λ_{max}.

2.2 Towards multiplicative multilevel algorithms.

So far, we have used generating systems in order to devise new prolongations and restrictions and we have presented our approach in a purely additive setting: Our algorithms are of the same type as BPX [1] or MDS [19]. From the work by Griebel [8, 9] we know that the MDS-preconditioner is nothing but the Block Jacobi method applied on the generating system and that multiplicative multigrid algorithms can simply be viewed as iterative methods of Gauss–Seidel type on the generating system. Furthermore, in their abstract convergence framework Griebel and Oswald [10] point out why good preconditioning via the additive multilevel algorithms implies fast convergence of corresponding multiplicative multilevel algorithms. Their theoretical reasoning justifies using the new prolongations/restrictions also in multiplicative variants. In the following we very briefly describe how the ideas of Algorithm 2.1 can be used in a multigrid V-cycle. (We do not give the whole V-cycle here, but refer to [12], pp. 193, for the details of the algorithm we programmed.)

ALGORITHM 2.2. We start with the scaled matrix $\tilde{A}_1$.

- On every level use $B_j = \alpha I - \tilde{A}_j$ and $C_j = B_j(:, 2:2:n)$ for prolongation and restriction.
- After computation of $A_{j+1} = C_j^T \tilde{A}_j C_j$ we step forward with the scaled matrix $\tilde{A}_{j+1}$ as the coarse grid matrix
- When going down we restrict the residuals using $R_j = D_{j+1}^{-1/2} C_j^T$ and when going up we use R_j^T for prolongation.

Now we can either employ a multigrid V-cycle as a solver or as a preconditioner within the CG method (by performing one V-cycle with the residual r and initial guess zero). In general, we would recommend to use V-cycles as preconditioners for CG guarantees convergence. Nevertheless we shall in the following numerical tests employ our V-cycles as solvers, because that way it will become much more clearly visible if one has picked prolongations/ restrictions suitable for the problem at hand—or not.

3 Numerical examples.

For practical implementations of the method we will consider different variations:

- We can use the Matrix Multilevel transfer operators in Algorithm 2.1 or 2.2 and compare them with the standard transfer operators.
- We can derive estimates for $\alpha = \lambda_{max}(A_j)$ on every level in different ways:
  Choose the exact eigenvalue ,
  run the Lanczos method (see e.g [12, Chapter 2]) up to $|m|$ steps and
  set $\alpha = \tilde{\lambda}_{max}$ (denoted by $m > 0$ in our tables) or
  set $\alpha = \tilde{\lambda}_{max} + \tilde{\lambda}_{min}$ (denoted by $m < 0$ in our tables) or
  we can set $B_j = abs(A_j)$ without any need of additional computation of eigenvalues.
- For the additive variants of the MML we apply the conjugate gradient method method with preconditioner of the form (2.7) and zero starting vector.
- We also use V-cycles with one pre-smoothing and one post-smoothing step as solvers; our smoother within these cycles is the symmetric Gauss–Seidel method and we start with zero initial guess.
- As stopping criterion we employ $\|r_k\|_2/\|r_0\|_2 \leq rex$, where rex is a small positive constant and r_k denotes the residual after k iterations.

As numerical examples we consider uniform finite difference discretizations of the 1D elliptic PDE

$$(a(x)u(x)_x)_x = f(x)$$

with homogenous Dirichlet boundary conditions and right hand side $f(x) \equiv 1$ on the unit interval for

(1) $a(x)$ constant,

(2) $a(x) = 1 + \sin(32\pi x)^2$,

(3) $a(x) = 1 + \exp(2\pi x)\sin(2\pi x)^2$,

(4) $a(x) = 1 + \exp(\pi x)\sin(8\pi x)^2$,

(5) $a(x) = 1 + \exp(2\pi x)\sin(8\pi x)^2$,

(6) $a(x) = 1 + \exp(8\pi x)\sin(8\pi x)^2$,

(7) $a(x)$ is piecewise constant with ten different values between 0.1 and 2.1,

(8) $a(x)$ is piecewise constant with ten different values between 0.1 and $1.5E + 4$.

Table 3.1: Condition number Example (1), $a(x) = const.$

n	MML	MML0	$\|A\|$	MDS	$D^{-1}A$
2^5	4.55	6.32	5.46	4.60	414.3
2^6	5.43	7.26	6.35	5.12	1.7E3
2^7	6.34	8.19	7.27	5.62	6.6E3
2^8	7.26	9.14	8.20	6.11	2.7E4

Table 3.2: Condition number Example (2), $a(x) = 1 + \sin(32\pi x)^2$.

n	MML	MML0	$\|A\|$	MDS	$D^{-1}A$
2^5	4.55	6.32	5.46	4.60	414
2^6	5.191	7.98	6.10	6.30	1.9E3
2^7	6.12	7.93	7.03	98.8	7.1E3
2^8	7.15	9.00	8.07	80.6	2.8E4

Table 3.3: Condition number Example (4), $a(x) = 1 + \exp(\pi x)\sin(8\pi x)^2$.

n	MML	MML0	$\|A\|$	MDS	$D^{-1}A$
2^5	4.37	6.08	5.28	36.4	605.4
2^6	5.37	7.16	6.28	32.0	2.2E3
2^7	6.32	8.16	7.25	29.9	8.8E3
2^8	7.26	9.13	8.20	31.2	3.6E4

Table 3.4: Condition number Example (5), $a(x) = 1 + \exp(2\pi x)\sin(8\pi x)^2$.

n	MML	MML0	$\|A\|$	MDS	$D^{-1}A$
2^5	4.37	6.10	5.26	250.5	2.6E3
2^6	5.37	7.17	6.28	149.4	6.2E3
2^7	6.32	8.18	7.25	97.5	1.7E4
2^8	7.26	9.14	8.20	86.3	6.2E4

Table 3.5: Condition number Example (6), $a(x) = 1 + \exp(8\pi x)\sin(8\pi x)^2$.

n	MML	MML0	$\|A\|$	MDS	$D^{-1}A$
2^5	4.25	5.94	5.13	8.5E8	6.3E9
2^6	5.47	7.17	6.31	5.7E8	1.3E10
2^7	6.42	8.23	7.31	3.7E8	2.7E10
2^8	7.32	9.19	8.25	2.2E8	5.5E10

In the first examples we always choose α as the exact eigenvalue of A. The next tables display the condition numbers for the additive Matrix Multilevel Method based on Algorithm 2.1 and computing α_j on every level (MML), only on the finest level (MML0), using $|A|$ for prolongation/restriction, for the standard MDS-method (MDS), and for the Jacobi-preconditioned original problem.

For the above examples we get nearly the same condition number for the different additive matrix multilevel preconditioners. It is very robust with respect to the function $a(x)$—whereas the standard MDS preconditioner is not—and the

condition numbers only grow slightly with the problem size n.

In the next table we compare the influence of the choice of α for the convergence of the Matrix Multilevel Method. It is obviously far too expensive to use the exact maximum eigenvalue on every level. Therefore, we compute an estimate by the Lanczos method. It turns out, that with only one or two iterations in the Lanczos process we get estimates for the maximum eigenvalue that lead to the same iteration number in the Matrix Multilevel Method.

Table 3.6: CG iteration numbers for additive preconditioners for Example (6) with $rex = 10^{-4}$. The final seven columns compare the different choices for α within the MML. (A $*$ denotes that the computation would have been too expensive.)

n	MDS	$\|A\|$	λ_{max}	m =2	m =3	m =4	m =-1	m =-2	m =-3
2^6	37	10	10	10	10	10	11	11	11
2^7	45	12	12	12	12	12	12	12	12
2^8	53	14	14	15	15	14	14	14	14
2^9	61	17	16	28	21	17	17	17	17
2^{10}	72	18	18	42	34	28	18	18	18
2^{11}	80	20	*	61	49	43	20	20	20
2^{12}	98	22	*	92	73	60	23	22	22

In general, we recommend to set $\alpha = \tilde{\lambda}_{max} + \tilde{\lambda}_{min}$ where $\tilde{\lambda}_{max}$ and $\tilde{\lambda}_{min}$ are estimates for largest and smallest eigenvalue of A. Table 3.6 shows clearly that very few Lanczos steps are sufficient in order to compute these estimates. Note that Lanczos yields inner approximations to the extremal eigenvalues and that α should be greater or equal λ_{max}, because otherwise the projected matrix can get indefinite.

We also observe that for most of our examples the MML is independent of how to compute α as long as we get a good upper bound for the largest eigenvalue: The quality of the Lanczos estimates $\tilde{\lambda}_{max}$ and $\tilde{\lambda}_{min}$ hardly depends on the starting vector used. All the results given in this section are based on the starting vector $(1, 0, \ldots, 0)$ with which we usually obtain the best estimates. Anyway, we also tested Lanczos estimates based on the starting vector $(1, -1, 1, -1, \ldots)$ and on random vectors—and all these estimates were very reasonable.

We finally remark that sometimes the projection with $abs(A)$ gives better results and sometimes the projections based on estimating the maximum eigenvalue does better. Note that the costs for estimating α with the Lanczos method correspond roughly to the costs for one Matrix Multilevel iteration step.

The following four tables, Tables 3.7–3.10, clearly demonstrate the superiority of the new MML transfer operators in comparison to standard prolongations/restrictions—the first two deal with additive preconditioners, the other two with V-cycle solvers. Other than the standard approach the MML approach is very robust with respect to the function $a(x)$ and we observe optimal computational behaviour in many cases:

The final example $T = \text{tridiag}(1, 2, 1)$ (see Table 3.11) shows that the Matrix Multilevel approach can be applied to more general problems than the usual

Table 3.7: CG iteration numbers for Example (7), $rex = 10^{-4}$.

n	MDS	$abs(A)$	MML: Alg. 2.1	$m = -2$	$m = -3$
2^7	32	13		13	13
2^8	43	14		14	14
2^9	51	16		16	16
2^{10}	57	17		17	17
2^{11}	66	18		18	18
2^{12}	80	19		19	19

Table 3.8: CG iteration numbers for Example (8), $rex = 10^{-4}$.

n	MDS	$abs(A)$	MML: Alg. 2.1	$m = -2$	$m = -3$
2^7	60	16		17	16
2^8	81	18		20	18
2^9	97	19		21	20
2^{10}	110	20		23	22
2^{11}	125	22		27	25
2^{12}	167	23		31	28

Table 3.9: Numbers of V-cycle iterations for Example (6), $rex = 10^{-6}$.

n	Standard MG	$abs(A)$	MML: Alg. 2.2	$m = -2$	$m = -3$
2^7	646	6		6	6
2^8	1381	6		6	6
2^9	> 2000	7		6	6
2^{10}	> 2000	7		6	6
2^{11}	> 2000	7		6	5
2^{12}	> 2000	7		7	5
2^{13}	> 2000	7		10	8

Table 3.10: Numbers of V-cycle iterations for Example (7), $rex = 10^{-6}$.

n	Standard MG	$abs(A)$	MML: Alg. 2.2	$m = -2$	$m = -3$
2^7	641	7		7	7
2^8	1223	7		7	7
2^9	> 2000	8		8	7
2^{10}	> 2000	7		6	6
2^{11}	> 2000	7		6	6
2^{12}	> 2000	7		6	6
2^{13}	> 2000	7		7	6

multigrid method (and, by the way, it also clearly points out that the choice $B = \alpha I - A$ for prolongation/restriction is more generally applicable than the choice $B = abs(A)$).

Table 3.11: CG iteration numbers for $T = \text{tridiag}(1, 2, 1)$, $rex = 10^{-4}$.

n	MDS	MML: Alg. 2.1	$m = -1$	$m = -2$	$m = -3$
2^6	53		7	7	7
2^7	113		7	7	7
2^8	247		7	7	7
2^9	330		7	7	7

4 The two-dimensional case.

The aim of this section is to generalize the derived method to the higher dimensional case. Therefore we have to decompose the original ill-conditioned matrix into a sum of one-dimensional problems, e.g. $A = A_1 + A_2$. Here, A_1 and A_2 should be ill-conditioned, too, and such that the 1D Matrix Multilevel approach works well. Furthermore, both matrices should be independent in a certain sense, otherwise the projections relative to A_1 lead to a strong change of A_2 and vice versa. Such a decomposition is given in a natural way by a PDE in terms of the parts relative to the derivatives in x-direction and in y-direction. In the following we mainly consider uniform finite difference discretizations of the 2D elliptic PDE

$$(a(x,y)u(x,y)_x)_x + (b(x,y)u(x,y)_y)_y = f(x,y). \tag{4.1}$$

Let us first focus on a separable PDE, i.e. $a(x,y) \equiv a(x)$ and $b(x,y) \equiv b(y)$. Then the matrix A can be written as a Kronecker sum (see e.g. [15])

$$A = (A_{K1} \otimes I) + (I \otimes A_{K2}).$$

The matrix B that is applied in the standard multigrid approach is given not by a Kronecker sum but by a Kronecker product $B_1 \otimes B_2$ with $B_i = \text{tridiag}(1, 2, 1)$. Hence, we will also choose a matrix B as a Kronecker product $B = B_1 \otimes B_2$ and then use B to define the matrix C in the form $C = BP$.

Now the eigenvalues and eigenvectors of A are given by the sum of the eigenvalues of A_{K1} and A_{K2}, and the Kronecker product of the eigenvectors. Hence, every eigenvector of A is of the form $u = u_1 \otimes u_2$. In view of the product rules for the Kronecker product we get

$$\begin{aligned} Au &= ((A_{K1} \otimes I) + (I \otimes A_{K2}))(u_1 \otimes u_2) \\ &= (A_{K1}u_1 \otimes u_2) + (u_1 \otimes A_{K2}u_2) = (\lambda_1 + \lambda_2)u. \end{aligned}$$

To find an efficient matrix B we again want to enlarge the small eigenvalues of A without changing the large ones. Hence, with

$$Bu = (B_1 \otimes B_2)(u_1 \otimes u_2) = (B_1u_1 \otimes B_2u_2)$$

we can choose

$$B_1 := (\alpha_1 I - A_{K1})\beta_1 \quad \text{and} \quad B_2 := (\alpha_2 I - A_{K2})\beta_2$$

with α_1 and α_2 the maximum eigenvalues of A_{K1} and A_{K2}, respectively. Then the minimum eigenvalues are approximately enlarged by a factor $1+(\beta_1\beta_2\alpha_1\alpha_2)^2$. Furthermore, the elementary projection matrix P can be chosen as $P = P_1 \otimes P_2$, and then on the second level we get

$$C^T AC = (C_1^T A_{K1} C_1 \otimes C_2^T C_2) + (C_1^T C_1 \otimes C_2^T A_{K2} C_2).$$

In the next step we can again define the restriction matrix via the Kronecker product where the first factor is designed to improve on $C_1^T A_{K1} C_1$, and the second factor is related to $C_2^T A_{K2} C_2$. This is the direct generalization of the 1D approach to higher dimensions via the Kronecker product. The 2D projection is derived as the Kronecker product of the 1D projections for the separated problems in A_{K1} and A_{K2}.

In the previous sections we showed that the Matrix Multilevel approach has to be applied in connection with diagonal scaling. This leads to difficulties in the higher dimensional case: Then the 1D and the 2D diagonal scaling do not correspond anymore. But the 2D diagonal scaling is also used for smoothing.

ALGORITHM 4.1. Additive Matrix Multilevel Method for separable problems with Kronecker structure $A = A_{K1} \otimes I + I \otimes A_{K2}$:

First set $A_1^{(1)} = A_{K1}$, $A_2^{(1)} = A_{K2}$ and $A^{(1)} = A$:

- On every level j scale the matrices $A_1^{(j)}$ and $A_2^{(j)}$ with their diagonals $D_1^{(j)}$ and $D_2^{(j)}$, respectively. We also scale $A^{(j)}$ on every level with $\mathrm{kron}(D_1^{(j)}, D_2^{(j)})$.
- Like in Algorithm 2.1 compute $B_i^{(j)} = \lambda_i I - A_i^{(j)}$ and $C_i^{(j)} = B_i^{(j)}(:, 2:2:n)$ for $i = 1, 2$ on every level j; we get the prolongation by $C = C_1^{(j)} \otimes C_2^{(j)}$.
- Step forward with the matrices $A_i^{(j+1)} = (C_i^{(j)})^T A_i^{(j)} C_i^{(j)}$ for $i = 1, 2$ and $A^{(j+1)} = (C^{(j)})^T A^{(j)} C^{(j)}$.
- For smoothing we include the diagonal parts of the resulting matrices $A^{(j)}$ on every level in the form (2.7).

Hence we use the Jacobi scaled 1D matrices $A_1^{(j)}$ and $A_2^{(j)}$ with diagonal matrices $D_1^{(j)}$ and $D_2^{(j)}$ for constructing the prolongation matrices on every level j; in the same way the matrix $A^{(j)}$ is scaled by $D_1^{(j)} \otimes D_2^{(j)}$. Furthermore, the diagonal of the matrix $A^{(j)}$ can be used for smoothing in the form (2.7).

We can expect very good results particularly for the case that for the original problem the 1D and 2D diagonal scalings coincide, e.g. if $\mathrm{diag}(A_1) = const * I$ and $\mathrm{diag}(A_2) = const * I$, or if the partial problems A_{K1} and A_{K2} are already diagonally scaled.

Now let us consider general matrices of the form $A = A_1 + A_2$ where A_1 and A_2 can be solved efficiently by the 1D Matrix Multilevel approach. We have to replace the Kronecker products by usual multiplications of matrices. To this aim we use a kind of semi-coarsening and think of the prolongations in the form

$$\begin{aligned} C = BP &= (B_1 \otimes B_2)(P_1 \otimes P_2) = (B_1 P_1 \otimes B_2 P_2) \\ &= (B_1 P_1 \otimes I)\,(I_r \otimes B_2 P_2) = F_1 * F_2\ , \end{aligned}$$

where I_r is the identity matrix of half size and each matrix F_i reduces the size by a factor 1/2. Then we write the prolongation

$$F = C_1 \otimes C_2 = (B_1 P_1) \otimes (B_2 P_2) = (B_1 \otimes I)(P_1 \otimes I) * (I_r \otimes B_2 P_2)$$
$$= \hat{B}_1 (P_1 \otimes I) * (I_r \otimes B_2)(I_r \otimes P_2) = \hat{B}_1 (P_1 \otimes I) * \hat{B}_2 (I_r \otimes P_2) = F_1 * F_2,$$

where $\hat{B}_1 = \lambda_{max}(A_1) I - A_1$ and $\hat{B}_2 = \lambda_{max}(\hat{A}_2) I_r - \hat{A}_2$ with the matrix $\hat{A}_2 = A_2(2:2:n, 2:2:n)$ chosen to be the result of a trivial projection of A_2 in x-direction. Hence, we get the whole 2D prolongation by $F = F_1 * F_2$ with F_1 an $n \times (n/2)$ matrix, and F_2 an $(n/2) \times (n/4)$ matrix. Then FAF^T is an $(n/4) \times (n/4)$ matrix. Thereby we do not need any Kronecker structure of the given matrix A, but only two independent ill-conditioned parts A_1 and A_2. The numerical realization is analogous to Algorithm 4.1:

ALGORITHM 4.2. Additive Matrix Multilevel Method for general problems with $A = A_1 + A_2$:
Again, we first set $A_1^{(1)} = A_1$, $A_2^{(1)} = A_2$ and $A^{(1)} = A$: The prolongations are defined as products $C^{(j)} = F_1^{(j)} * F_2^{(j)}$ where $F_1^{(j)}$ is derived by the diagonally scaled version of $A_1^{(j)}$ on every level j, e.g. by $\lambda_{max} I - (D_1^{(j)})^{-1} A_1^{(j)}$, and F_2 by the projected matrix $\hat{A}_2^{(j)}$. The matrices $A_i^{(j)}$, $i = 1, 2$, are transfered onto lower levels by applying the related transfer operators in the corresponding direction, and an elementary projection in the other direction. Again Jacobi smoothing is included in the form (2.7). Furthermore, according to the diagonal scalings $D_i^{(j)}$ of $A_i^{(j)}$ we also scale $A^{(j)}$ on every level by $D_1^{(j)} * D_2^{(j)}$.

In general we can write A in the form $A = A_1 + \cdots + A_d$, where each term is related to the i-th direction. Then for A_1 we can define a matrix $B_1 = \lambda_{max} I - A_1$ and apply the related prolongation F_1 to A_1, and trivial projections in x-direction on $A_2, \ldots, A_d$. Then, F_2 is given by the projected A_2, and elementary projections are again applied on $A_3, \ldots, A_d$ and so on. In the end we define $F = F_1 F_2 \cdots F_d$ with F_j of decreasing dimension.

The multiplicative versions of Algorithms 4.1 and 4.2 are almost identical to the additive algorithms: prolongations and restrictions are obtained in the same way and the identical coarse grid matrices are used. The residuals are restricted on the level j by multiplication with $(\mathrm{kron}(D_1^{(j)}, D_2^{(j)}))^{-1/2} * (C^{(j)})^T$ in the Kronecker case and with $(D_1^{(j)} D_2^{(j)})^{-1/2} * (C^{(j)})^T$ in the general case.

The following numerical examples compare Algorithm 4.1 (Kronecker) and Algorithm 4.2 (semi-coarsening) to the standard approach for 2D problems with Kronecker structure: We demonstrate the superiority of the MML transfer operators in comparison to standard prolongations/restrictions—it becomes particularly obvious when using V-cycles as solvers. Within the MML $\alpha = \lambda_{max} + \lambda_{min}$ was always estimated by two Lanczos steps in our tests; the starting vector was $(1, 0, \ldots, 0)$ in the Kronecker case and $(1, 0, \ldots, 0) \otimes (1, 1, \ldots, 1)$ and $(1, 1 \ldots, 1) \otimes (1, 0, \ldots, 0)$ in case of Algorithm 4.2. Tables 4.1 and 4.2 deal with additive preconditioners for CG, tables 4.3 and 4.4 with V-cycle solvers (with one pre-smoothing and one post-smoothing step by the symmetric Gauss–Seidel method on every level) :

Table 4.1: CG iteration numbers for $A = A_{K1} \otimes I + I \otimes A_{K2}$, $rex = 10^{-4}$, with A_{K1} from example 1 and A_{K2} from Example (7).

n	MDS	MML	Alg. 4.1—Kronecker	Alg. 4.2—Semi-coarsening
$2^4 * 2^4$	23		18	18
$2^5 * 2^5$	28		26	26
$2^6 * 2^6$	37		31	31
$2^7 * 2^7$	44		38	38

Table 4.2: CG iteration numbers for $A = A_{K1} \otimes I + I \otimes A_{K2}$, $rex = 10^{-4}$, with A_{K1} the diagonally scaled matrix from example 3 and A_{K2} the diagonally scaled matrix from Example (7). Note that this problem is different in style from Table 4.1, because the matrices A_{K1} and A_{K2} were diagonally scaled in advance.

n	MDS	MML	Alg. 4.1—Kronecker	Alg 4.2—Semi-coarsening
$2^4 * 2^4$	28		13	13
$2^5 * 2^5$	38		15	15
$2^6 * 2^6$	57		17	17
$2^7 * 2^7$	82		19	19

Table 4.3: Number of V-cycle iterations for A from Table 4.1, $rex = 10^{-6}$.

n	MG	MML	Alg. 4.1—Kronecker	Alg. 4.2—Semi-coarsening
$2^5 * 2^5$	30		16	16
$2^6 * 2^6$	44		16	16
$2^7 * 2^7$	51		16	17
$2^8 * 2^8$	115		17	17
$2^9 * 2^9$	168		17	17

Table 4.4: Number of V-cycle iterations for $A = B \otimes I + I \otimes B$, $rex = 10^{-6}$, with B from Example (7).

n	MG	MML	Alg. 4.1—Kronecker	Alg. 4.2—Semi-coarsening
$2^5 * 2^5$	76		25	25
$2^6 * 2^6$	145		28	28
$2^7 * 2^7$	238		28	28
$2^8 * 2^8$	> 250		28	29
$2^9 * 2^9$	> 250		28	28

At this point we would like to emphasize that for the problems in Tables 4.3 and 4.4 the MML shows optimal computational behaviour—in the sense that the numbers of iterations needed are independent of the problem size.

Finally, we would like to investigate general problems $A = A_1 + A_2$ without Kronecker structure. Therefore we consider uniform finite difference discretizations of 2D elliptic PDEs of the form (4.1) with right hand side $f(x, y) \equiv 1$ and homogenous Dirichlet boundary conditions on the unit square.

We expect a particularly good performance of the MML in the case $\mathrm{diag}(A_1) = \mathrm{diag}(A_2)$—then we can scale A and solve a system with a constant diagonal; see Table 4.5.

Table 4.5: Number of V-cycle iterations for a problem of the form (4.1) with $a(x,y) = b(x,y) = 1 + \exp(8 * (x+y)) * \sin(2 * (x+y))^2$, $rex = 10^{-6}$.

n	Standard MG	MML	Alg. 4.2—Semi-coarsening
$2^5 * 2^5$	16		8
$2^6 * 2^6$	18		8
$2^7 * 2^7$	22		8
$2^8 * 2^8$	23		8
$2^9 * 2^9$	23		8

However, in general we can definitely not expect Algorithm 4.2 to do significantly better than the standard approach. But if all the diagonal entries of A_1 and A_2 are of the same order, the MML may still lead to an improvement; see Table 4.6.

Table 4.6: Number of V-cycle iterations for a problem of the form (4.1) with $a(x,y) = 1+\exp(16*(x+y))*\sin(2*(x+y))^2$, $b(x,y) = 1+\exp(17*(x+y))*\sin(2*(x+y))^2$, and $rex = 10^{-6}$.

n	Standard MG	MML	Alg. 4.2—Semi-coarsening
$2^5 * 2^5$	26		13
$2^6 * 2^6$	33		13
$2^7 * 2^7$	62		19
$2^8 * 2^8$	66		19
$2^9 * 2^9$	68		20

In our final example (see Table 4.7) we investigate a matrix M that demonstrates the broader applicability of the MML compared to standard multigrid. This matrix M was obtained by first setting up the block tridiagonal matrix $A = T \otimes I + I \otimes T$ with $T = \mathrm{tridiag}(1, 2, 1)$. Afterwards we only left the first and last off-diagonal blocks as well as the second and last but one diagonal block untouched, but we made all the other off-diagonal entries jump between 1 and -1 by flipping signs.

Table 4.7: Numbers of V-cycle iterations for 'jump matrix' M, $rex = 10^{-6}$.

n	Standard MG	MML	Alg. 4.2—Semi-coarsening
$2^5 * 2^5$	79		5
$2^6 * 2^6$	221		5
$2^7 * 2^7$	> 250		5
$2^8 * 2^8$	> 250		5
$2^9 * 2^9$	> 250		5

5 Outlook and conclusions.

We have developed a purely matrix-dependent multilevel method for solving linear equations. The prolongation/restriction operator is defined by a shift $B = \lambda_{max}(A)I - A$, $B(:, 2:2:n)$, of the given matrix in order to enlarge $\lambda_{min}(A)$. This idea can be applied on every level of the method. The numerical results show a significant improvement over the standard multigrid approach—both in additive and multiplicative algorithms—not only in the 1D case, but also in two dimensions for problems with Kronecker structure and for problems of the form $A = A_1 + A_2$ with coinciding diagonals. Furthermore, the Matrix Multilevel approach is more generally applicable than the standard approach and it represents a promising idea to derive further new matrix-dependent multigrid algorithms.

One direction for future research will be to generalize the Matrix Multilevel Method for problems that are not symmetric positive definite: For A *symmetric indefinite* the eigenvalues are negative and positive. Hence, to find a polynomial that enlarges the small eigenvalues without changing the large ones, we can choose $B = \alpha^2 I - A^2$. Then small eigenvalues λ are replaced by $\lambda(1 + \rho)$ for a factor $\rho > 0$. The main disadvantage of this approach is that the subproblems A_j are losing their sparsity. Hence, we have to stop at a level j, or apply the full multilevel method, but with total costs of at least $O(n\,log(n))$.

If A is *nonsymmetric* then for the mapping B we can use a low degree polynomial $p(A)$ with $p(0) = 1$ and $|p(x)|$ is small for the extreme eigenvalues of A. In the normal case, $p(A, A^T) = |\lambda_{max}|^2 I - AA^T$ should be a good choice. To preserve the sparsity of the restricted linear systems A_j, one can choose the identity for the prolongation and $p(A, A^T)$ for the restriction or vice versa. In the symmetric and general nonsymmetric case a possible choice for α with prolongation $B = \alpha I - A$ is also $\alpha := \lambda_{max}((A + A^T)/2)$.

There also remain plenty of other questions for future investigations: One can try to improve our approach for special cases, e.g. for the case $A = A_1 + A_2$ one could try to make use of block structure and build prolongations block-wise. It will also be interesting to test and analyse to what extent it makes sense to apply the MML projections only on the finest level and use standard transfer operators on the coarser levels. Finally, the relation of the two choices $B = \lambda_{max}(A)I - A$ and $B = \text{abs}(A)$ calls for more analysis.

REFERENCES

1. J. Bramble, J. Pasciak, and J. Xu, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), pp. 1–22.
2. A. Brandt, *Multi-level adaptive solutions to boundary value problems*, Math. Comp., 31 (1977), pp. 333–390.
3. W. Briggs, *A Multigrid Tutorial*, SIAM, Philadelphia, PA, 1987.
4. R. H. Chan, Q. S. Chang, and H. W. Sun, *Multigrid method for ill-conditioned symmetric Toeplitz systems*, SIAM J. Sci. Comput., 19 (1998), pp. 516–529.
5. L. Elsner, C. He, and V. Mehrmann, *Minimizing the condition number of a positive definite matrix by completion*, Numer. Math., 69 (1994), pp. 17–23.
6. G. Fiorentino and S. Serra, *Multigrid methods for Toeplitz matrices*, Calcolo, 28 (1992), pp. 283–305.

7. M. Griebel, *Zur Lösung von Finite-Differenzen- und Finite-Element-Gleichungen mittels der Hierarchischen-Transformations-Mehrgitter-Methode*, TUM-I9007: SFB-Report 342/4/90 A, Institut für Informatik, TU München, 1990.
8. M. Griebel, *Multilevel algorithms considered as iterative methods on semidefinite systems*, SIAM J. Sci. Comput., 15 (1994), pp. 547–565.
9. M. Griebel, *Multilevelmethoden als Iterationsverfahren über Erzeugendensystemen*, Teubner, Stuttgart, 1994.
10. M. Griebel and P. Oswald, *On the abstract theory of additive and multiplicative Schwarz algorithms*, Numer. Math., 70 (1995), pp. 163–180.
11. A. Greenbaum, *Analysis of a multigrid method as an iterative technique for solving linear systems*, SIAM J. Numer. Anal., 21 (1984), pp. 473–485.
12. A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, PA, 1997.
13. W. Hackbusch, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.
14. W. Hackbusch and U. Trottenberg, *Multigrid Methods*, Springer-Verlag, Berlin, 1982.
15. R. Horn and C. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1989.
16. B. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
17. J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., SIAM, Philadelphia, PA, 1987, pp. 73–130.
18. H. Yserentant, *Old and new convergence proofs for multigrid methods*, Acta Numerica, 2 (1993), pp. 285–326.
19. X. Zhang, *Multilevel Schwarz methods*, Numer. Math., 63 (1992), pp. 521–539.