T U M

I N S T I T U T F Ü R I N F O R M A T I K

Efficient Computation of Sparse Approximate Inverses

Thomas K. Huckle

TUM-I9608
Februar 96

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-02-96-I9608-200/1.-FI

Druck: Fakultät für Mathematik und
Institut für Informatik der
Technischen Universität München

Efficient Computation of Sparse Approximate Inverses

by

Thomas K. Huckle

Institut für Informatik

TU München

Arcisstr. 21

D-80290 München

Abstract

We investigate different methods for computing a sparse approximate inverse M for a given sparse matrix A by minimizing $\|AM - E\|$ in the Frobenius norm. Such methods are very useful for deriving preconditioners in iterative solvers, especially in a parallel environment. We compare different strategies for choosing the sparsity structure of M and different ways for solving the small least-squares problem that are related to the computation of each column of M. Especially we show how we can take full advantage of the sparsity of A. Furthermore, we give assistance how to design and apply an algorithm for computing sparse approximate inverses for a general sparse matrix.

Efficient Computation of Sparse Approximate Inverses

by Thomas K. Huckle

1. Introduction

Sparse approximate inverses $M \approx A^{-1}$ for preconditioning linear systems

$$Ax = b \quad , \quad A \text{ a sparse } n \times n \text{ matrix}, \tag{1}$$

are very interesting especially in a parallel environment. To solve (1) we can apply an iterative method like BCG, CGS, BiCGSTAB, QMR, or GMRES (see e.g. [2] for an overview over these algorithms), on the preconditioned equations

$$AMy = b \quad , \quad x = My \ .$$

Here, M should be a good approximate right-inverse matrix for A. Similarly, one can consider the left-preconditioned system $NAx = Nb$ with an approximate left-inverse matrix N.

One possible method to derive such a matrix M is to compute an incomplete LU-factorization $A \approx LU$ with sparse lower and upper triangular matrices L and U. Then, we can define $M = U^{-1}L^{-1}$. In many applications this is a very satisfactorily approach. But solving sparse triangular equations with L and U is strongly sequential. Therefore, in a parallel environment, other methods should be considered.

In recent papers it is shown that the direct computation of sparse approximate inverses leads to suitable preconditioners in a parallel environment [3,6,13,14,5,11,12,10,1]. The main approach is the minimization problem

$$\min \|AM - E\|_F^2 \ = \ \sum_{k=1}^{n} \min \|AM_k - e_k\|_2^2 \ , \tag{2}$$

for a given sparsity pattern of M. This minimization problem is embarrassingly parallel. Here and in the following E denotes the identity matrix. If we allow only a few nonzero entries in the k-th column M_k of M, then (2) reduces to the solution of n small least-squares problems

$$\min \|AM_k - e_k\| \quad , \quad k = 1, ..., n \ . \tag{3}$$

Let us denote by J the set of indices with nonzero entries in M_k, and I the corresponding shadow of J in A [10], that means the set of indices of nonzero rows in the submatrix $A(:,J)$. Hence, (3) can be reduced to

$$\min \|A(I,J)M_k(J) - e_k(I)\| = \|\hat{A}\hat{M}_k - \hat{e}_k\| \ , \quad k = 1, ..., n \ , \qquad (3')$$

with $\hat{A} = A(I,J)$, $\hat{M}_k = M_k(J)$, and $\hat{e}_k = e_k(I)$.

First, let us consider the least-squares problems (3'). To reduce the costs for computing M we want to take full advantage of the sparsity of A. This can be acchieved with the Givens method, with sparse formulations of the Householder- or Gram-Schmidt-QR-decomposition [4,9], or by solving the normal equations iteratively in sparse mode. We will compare these different approaches to determine the fastest numerical stable methods for solving (3').

If we want to solve (3') based on a QR-decomposition, then we consider

$$\|Q \begin{pmatrix} R \\ 0 \end{pmatrix} \hat{M}_k - \hat{e}_k\| \ = \ \| \begin{pmatrix} Y & Z \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix} \hat{M}_k - \hat{e}_k\| \ , \qquad (4)$$

which is equivalent to

$$\| \begin{pmatrix} R\hat{M}_k \\ 0 \end{pmatrix} - \begin{pmatrix} Y^T\hat{e}_k \\ Z^T\hat{e}_k \end{pmatrix} \| \ . \qquad (5)$$

Hence, the solution M_k of (3) is given by

$$\hat{M}_k \ = \ R^{-1}(Y^T\hat{e}_k) \ , \qquad (6)$$

and the remaining residual norm is

$$\|AM_k - e_k\| = \|\hat{A}\hat{M}_k - \hat{e}_k\| \ = \ \|Z^T\hat{e}_k\| \ . \qquad (7)$$

In the Givens- and Householder approach we compute elementary Givens- or Householder matrices that eliminate the subdiagonal entries in $\hat{A}$. Hence, we get

$$G_l ... G_1 \hat{A} = \begin{pmatrix} R \\ 0 \end{pmatrix}$$

or

$$H_m ... H_1 \hat{A} = \begin{pmatrix} R \\ 0 \end{pmatrix}$$

with Givens matrices G_i or Householder matrices $H_i = E - 2q_i q_i^T$, q_i a vector of length 1. Hence, to solve (3'), we have to compute also $G_l ... G_1 \hat{e}_k$ or $H_m ... H_1 \hat{e}_k$, to get $Y^T \hat{e}_k$ and apply (6). To reduce the number of operations, the matrix Q is never computed explicitly.

If we use a Gram-Schmidt-type method, we orthogonalize each new column of $\hat{A}$ against the previous columns which leads to an explicit representation of Y and we get $\hat{A} = YR$. We subtract from one column $A(I, j)$ the projections on all the previous orthogonalized columns q_i, and normalize the result. The Modified Gram-Schmidt approach is numerically more stable. Here, in the j-th step we subtract the orthogonalized parts immediately from $A(I, j)$, as soon as they are available, while in the original Gram-Schmidt method the orthogonalization is done against the original $\hat{A}\hat{e}_j$. The matrix Y and the right hand side in (6) are always given explicitly.

We can also consider the normal equations, and apply the conjugate gradient algorithm with Jacobi preconditioner on $\hat{A}^T \hat{A} \hat{M}_k = \hat{A}^T \hat{e}_k$. Then, the multiplication with $\hat{A}$ and $\hat{A}^T$ can be done in sparse mode. In this case, we have no further information about the least-squares problem (3').

In implicit QR-decompositions like the Givens- or Householder approach, where we do not compute Q explicitly, the factors Q and R, and the right hand side $Y^T \hat{e}_k$ in (6) have to be updated if we add new indices to J. Updates are unnecassary in the iterative solution, and appear naturally in the Gram-Schmidt method without additional costs.

Now let us give an overview over different strategies for choosing indices in M_k. We assume that we have already computed an optimal solution $M_k(J)$, resp. the sparse n-vector M_k with residual r_k, by solving (3') for an given index set J. Now, we want to define dynamically new entries in M_k and solve (3') for this enlarged index set $\tilde{J}$ such that we derive a reduction in the norm of the new residual $\tilde{r}_k = A(\tilde{I}, \tilde{J}) M_k(\tilde{J}) - e_k(\tilde{I})$. We consider only indices j which appear in rows of A that are connected to the nonzero entries in the old residual r_k, otherwise they lead to no reduction of the residual norm. This is equivalent to determine the column indices j, that satisfy $r_k^T A e_j \neq 0$ with the old residual r_k (see the following equations (9)). Let us denote the index set of nonzero entries in r_k by L. Until the very first step, L will be always equal to I, the shadow of J in A. As $\tilde{J}_i$ we define the set of new indices that are related to the nonzero elements in the i-th row of A, and the set of all possible new indices by $J_n = \cup_{i \in L} \tilde{J}_i$. As set of possible index candidates J_c we can take all indices in J_n or we can consider only the indices

$\tilde{J}_i$ that correspond to the maximal entries in r_k.

Now, there are two different ways to determine new profitable indices from the index set J_c. We want to test one possible new index $j \in J_c$. Therefore, we can consider the reduced 1D problem [6,11]

$$\min_{\lambda} \|A(M_k + \lambda e_j) - e_k\| \ , \tag{8a}$$

or with $J^j = J \cup \{j\}$ the more expensive, but also more accurate problem [10]

$$\min_{M_k(J^j)} \|A(:, J^j)M_k(J^j) - e_k\| \ . \tag{8b}$$

Then, we add one or more new indices to J which are optimal in the sense of (8a) or (8b). If we want more than one new index in J in one step, it is necessary to first eliminate "bad" indices that may lead to an unsatisfactorily improvement. This can be done by deleting all indices with new residual norm larger than the mean value of all the computed residual norms (8a), resp. (8b), for all index candidates out of J_c.

The solution of (8a) for J with optimal solution M_k and corresponding residual r_k, and a new index $j \in J_c$ is given by [6,11]

$$\rho_j^2 = \|r_k\|^2 - \frac{(r_k^T A e_j)^2}{\|Ae_j\|^2} \ . \tag{9a}$$

The solution of (8b) can be derived by [10]

$$\sigma_j^2 = \|r_k\|^2 - \frac{(r_k^T A e_j)^2}{\|Ae_j\|^2 - \|Y^T \hat{A}\hat{e}_j\|^2} \ . \tag{9b}$$

The access on the matrix A is necessary in two steps. First, in (9) we need inner products with the j-th column of A. Hence, A should be stored columnwise. Next, for finding the index sets $\tilde{J}_i$ of possible candidates, for every i-th row of A we have to know the corresponding column indices. Hence, we need for example an array of size n that contains for every row i of A the list of corresponding nonzero column indices.

We have seen, that there is a whole variety of different methods for finding enlarged index sets J and solving (3') for this J. We add new profitable indices as long as $\|r_k\| > \epsilon$ or $|J| \leq q$, where $0 < \epsilon < 1$ and q is an upper bound for the number of nonzero entries in M_k. Especially, if we set $\epsilon = 1$,

then we get $M = M_0$ the optimal solution corresponding to the start sparsity, and if we set ϵ very small and $q = n$ then we would get $M = A^{-1}$ in exact arithmetic.

In the following, we will compare these different algorithms for computing sparse approximate inverses. We state the following general guidelines that are justified theoretically and/or by numerical experiments:

(i) We compute M_k as fast and good as possible, but not necessarily optimal. The objective is not to satisfy exactly $\|r_k\| \leq \epsilon$ for all k or for very small ϵ, but to reach a small residual for many columns M_k with still sparse M. If M_k has too many entries then solving the least-squares problem (3') gets too expensive (in the extreme case we would have to solve a least-squares problem with the full matrix A).

(ii) On a sequential computer the costs for solving all the least-squares problems should be not much more expensive than the explicit solution of (1).

(iii) The number of nonzero entries in M_k, $k = 1, ..., n$, should differ not too widely. Note, that in a parallel environment the total costs are ruled by the most expensive M_k.

(iv) It is better to add more indices in one step. Sometimes, if we add only one optimal solution of (8a) or (8b) to J, then we do not reach an profitable index set. The criteria (8) do not guarantee to find optimal index sets, they are only related to an optimal enlargement of the previous J with one new entry. If we add three or more indices per step we can avoid this difficulty and raise the possibiltiy of finding profitable index sets. Furthermore, we save also a lot of comparisons in (8) and (9), because it is not necessary to determine new index candidates for every new entry j. Note, that if in one step we add exactly one superfluous index that leads to no reduction in r_k, then also in the following steps we will find no profitable extensions and $\|r_k\|$ will remain unchanged while J is getting larger and larger.

(v) We apply (8) not on all possible indices J_n, but only on the indices corresponding to large entries in r_k. Hence, we start with the maximum entry in r_k, say the i-th entry, and define as the set of possible new index

candidates J_c all the indices in $\tilde{J}_i$ related to the corresponding i-th row of A. Next, we consider the second large entry in r_k, add the related indices to J_c, and continue, as long as a prescribed number mi for the size of J_c is not reached. With this method we don't have to compute (9) for every index candidate. But, we have to allow enough possible indices such that we are able to include "good" entries in J.

(vi) If we have computed ρ_j or σ_j with (9) for all possible new indices J_c, we eliminate indices with values near the old residual $\|r_k\|$. To this aim, we delete all indices with values larger than the mean value of all possible candidates in J_c. Then, we add not more than a prescribed number s of new indices to J. Here, we use the arithmetic mean value $\sum \rho_j/|J_c|$. The mean value lies between $\|r_k\|$ and $\rho_{optimal}$, and we can hope that it seperates promising indices from indices with values near $\|r_k\|$. The same idea can be applied on σ_j and (8b).

(vii) Other preconditioners may be much cheaper (or even free) to compute compared to constructing sparse approximate inverses. But the main measure for the effectivity of a preconditioner is the number of iterations in the unpreconditioned iterative scheme that takes the same number of flops or CPU-time. This is also interesting because some important preconditioners can be implemented in such a way that the preconditioned step has the same complexity as the unpreconditioned step (e.g. Eisenstat-trick). The costs for computing a preconditioner should be not much more than the costs of a "few" steps of the iterative solver for (1). Then, in the case of fast convergence the total costs for solving (1) are of the order of a "few" iteration steps.

Hence, in view of (i)-(iii) we have to reduce q, the allowed size of J, to be of the order of p, where p denotes the average number of nonzero entries in columns of A. We can choose ϵ for example in [0.2 , 0.6], but the stronger limitation will be given by q.

In the following sections we will analyse the different approaches for solving (3') especially with regard to utilizing the sparsity of $A(I, J)$. In Section 5. we present numerical results that allow to compare these different methods for computing a sparse approximate inverse. In the conclusions we will give

guidelines, what kind of method and what parameters should be used in constructing sparse approximate inverses with (3).

2. Iterative Solution

If we want to use an iterative solver for (3'), we can apply the conjugate gradient method on the normal equations $\hat{A}^T\hat{A}\hat{M}_k = \hat{A}^T\hat{e}_k$. No further evaluation is necessary, if we do not multiply both matrices. As preconditioner we can define $diag(\hat{A}^T\hat{A})$. The matrix$\times$vector-multiplications in the cg-method can be done in sparse mode in two steps with $y = \hat{A}x$ and $z = \hat{A}^T y$, where x, y, and z are small dense vectors. Note, that the matrix $\hat{A}$ is stored columnwise in sparse mode. Hence, the sparse multiplication $\hat{A}x$ is carried out by $|J|$ SAXPY's of sparse column vectors, while $\hat{A}^T y$ is computed as $|J|$ inner products of a sparse column of $\hat{A}$ with the full vector y. Hence, the total costs for solving (3') are mainly given by $2|J|p \times \#iterations \leq 2|J|^2 p$.

The advantage of this approach is, that we need no additional memory, no updates, and we need no old information on (3') if we want to solve (3') for an enlarged index set J. Furthermore, sometimes we want to find factorized sparse approximate inverses that minimize for example

$$\|AM_1...M_l - E\|_F \ .$$

Hence, in every step for given $M_1, ..., M_{l-1}$ we have to compute the new approximate sparse inverse to $AM_1...M_{l-1}$. If we solve the resulting least-squares problems iteratively we can avoid the explicit product $AM_1...M_{l-1}$, that will be much denser than the original A. For computing a column of M_l with index set J_l, we need the index set $I_l = J_{l-1}$, defined by the shadow of J_l in M_{l-1}, and similarly $J_{l-2} = I_{l-1}, ..., J_0 = I_1$.

The disadvantage of this iterative method is that it takes more arithmetic operations to solve every enlarged least-squares problem, especially if we add only one new index in every step. Hence, with this iterative solution method, one should add more indices in one step, because then there are fewer systems (3') to solve.

3. Implicit QR-Decomposition

First, let us assume that we add only one new index in J per step. Then we begin with index sets $J_1 = \{j_1\}$ and I_1, and we have to compute the QR-decomposition of $\hat{A}_1 = A(I_1, j_1)$. In the Householder approach, we use one elemantary Householder matrix $H_1 = E - 2q_1 q_1^T$, that transforms the matrix $\hat{A}$ via $H_1 \hat{A}_1 = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$ in upper triangular form.

In the second step we add one profitable new index j_2 to J, which leads to the new matrix

$$\hat{A}_2 = \begin{pmatrix} \hat{A}_1 & \hat{B}_1 \\ 0 & \hat{B}_2 \end{pmatrix} ,$$

where $\hat{B}_1$ is the part of the new column that is related to indices in I_1, while $\hat{B}_2$ is related to new indices that are only induced by the shadow of j_2. Now, we have to update the QR-decomposition. Therefore, we have to compute the QR-decomposition of

$$\left(\begin{array}{c|c} R_1 & H_1 \hat{B}_1 \\ \hline 0 & \hat{B}_2 \end{array}\right) = \begin{pmatrix} R_1 & \tilde{B}_1 \\ 0 & \tilde{B}_2 \end{pmatrix} .$$

We compute the new Householder vector q_2 related to the matrix $\tilde{B}_2$ with $H_2 \tilde{B}_2 = \begin{pmatrix} R_2 \\ 0 \end{pmatrix}$. This leads to the equation

$$\begin{pmatrix} 1 & 0 \\ 0 & H_2 \end{pmatrix} \begin{pmatrix} H_1 & 0 \\ 0 & E \end{pmatrix} \hat{A}_2 = \begin{pmatrix} R_1 & \tilde{B}_2 \\ 0 & R_2 \\ 0 & 0 \end{pmatrix} .$$

We can write this equation in a more convenient form by adding zeros in the vectors q_1 and q_2, to extend these vectors to the row length of $\hat{A}_2$. Then, we get

$$\tilde{H}_2 \tilde{H}_1 \hat{A}_2 = \begin{pmatrix} \tilde{R}_2 \\ 0 \end{pmatrix}$$

with

$$(\tilde{q}_1 \quad \tilde{q}_2) = \left(\begin{array}{c|c} q_1 & 0 \\ \mathbf{0}_{i_1} & q_2 \end{array}\right) .$$

If we continue to add new indices to J, and to extend the vectors q_i and $\tilde{q}_i$ to the corresponding row length, then we get the matrix

$$\hat{A}_m = \begin{pmatrix} \hat{A}_1 & * & \cdots & \cdots & * \\ \mathbf{0} & B_2 & * & & \vdots \\ \mathbf{0} & \mathbf{0} & B_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & * \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & B_m \end{pmatrix} \tag{10}$$

and Householder vectors of the form

$$\tilde{H}_m \cdots \tilde{H}_1 \hat{A}_k = \begin{pmatrix} \tilde{R}_m \\ \mathbf{0} \end{pmatrix}$$

with

$$(\tilde{q}_1 \quad \tilde{q}_2 \quad \cdots \quad \tilde{q}_m) = \left(\begin{array}{c|c|c|c|c} q_1 & 0 & \cdots & 0 & 0 \\ \mathbf{0}_{i_1} & q_2 & \ddots & \vdots & \vdots \\ \mathbf{0}_{i_2} & \mathbf{0}_{i_2} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & q_{m-1} & 0 \\ \mathbf{0}_{i_m} & \mathbf{0}_{i_m} & \cdots & \mathbf{0}_{i_m} & q_m \end{array} \right) .$$

This matrix is a lower triangular matrix where additionally the last entries in every column are zero. Hence, we have to store only the short nonzero kernels q_l and the related lengths of these vectors. Then, every multiplication with H_l can be reduced to nontrivial arithmetic operations. In this way we can use the sparsity of A in the Householder matrices $\tilde{H}_l$.

To solve the least-squares problem (3'), we have to compute

$$Q^T \hat{e}_k = \tilde{H}_m \cdots \tilde{H}_1 \hat{e}_k \ .$$

We can update this vector for every new index j_{m+1}, hence in every step there is only one product with the new Householder matrix $\tilde{H}_{m+1}$ necessary. For computing the new optimal $\hat{M}_k$, we solve a linear equation in the triangular matrix $\tilde{R}_m$ and the upper part of the vector $Q^T \hat{e}_k$ (6). Note also, that with (7) we can read of the new residual norm $\|AM_k - e_k\|$ by the norm of the lower part of $Q^T \hat{e}_k$.

If we add more than one new index per step we can derive the same sparsity structure in $\hat{A}_m$ and the Householder matrices $\tilde{H}_l$ if we partition the

index set I in such a form that $I = I_1 \cup I_2 \cup \cdots I_m$ where I_l is the set of new row indices induced by the column index j_l.

If we want to choose new profitable indices with the criterion (8b), then for indices $j \in J_c$ we have to evaluate (9b). Especially, we need $Y^T \hat{A}\hat{e}_j$, the upper part of the vector $Q^T \hat{A}\hat{e}_j$. To this aim, for every j we have to compute the longer vector $\tilde{H}_m \cdots \tilde{H}_1(\hat{A}\hat{e}_j)$. Hence, for (8b) this implicit representation of Q is not suited very well. But for (8a) it is numerically very stable and fast.

Similarly, instead of Householder matrices we can apply Givens or fast Givens transformations [9]. Then, the sparsity structure of A is also preserved, but with a loss of speed and/or accuracy (see the numerical results in Section 5).

4. Gram-Schmidt Orthogonalization

Here, in the original form, we orthogonalize the new column Ae_j against all the previous orthogonalized columns. Again, let us assume that the index set I is of the form $I = I_1 \cup \cdots \cup I_m$ with I_l related to a column index j_l. The QR-decomposition is of the form

$$\hat{A} = Q \begin{pmatrix} R \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} Y & Z \end{pmatrix} \begin{pmatrix} R \\ \mathbf{0} \end{pmatrix} = YR \ .$$

The orthogonalization of the new column is evaluated by

$$r_j = Y^T(\hat{A}\hat{e}_j) \ \ , \ \ q_j = \hat{A}\hat{e}_j - Yr_j \ \ , \ \ r_{jj} = \|q_j\| \ \ , \ \ q_j = q_j/r_{jj} \ .$$

Then, the new matrix R is given by

$$\left(\begin{array}{c|c} R_{j-1} & r_j \\ \hline 0 & r_{jj} \end{array} \right)$$

and the matrix Y is built up by the vectors q_j, and of the form (10) with the same sparsity structure as $\hat{A}$.

Unfortunately, the Gram-Schmidt process is numerically unstable, and in many examples it will be necessary to find a stable generalization. A common more robust variant is the Modified Gram-Schmidt algorithm. In the k-th step, the k-th column of Q and the k-th row of R are determined.

But, in contrast to the classical Gram-Schmidt procedure the inner products are computed not with $\hat{A}\hat{e}_j$ but with $\hat{A}\hat{e}_j - \tilde{Y}\tilde{R}\hat{e}_j$, where $\tilde{Y}$ and $\tilde{R}$ are all the previous evaluated columns and rows of Q and R. The increase of numerical stability causes a loss of sparsity: $(\hat{A}-\tilde{Y}\tilde{R})\hat{e}_j$ will be denser then $\hat{A}\hat{e}_j$. Hence, the evaluation of $Y^T(\hat{A}-\tilde{Y}\tilde{R})\hat{e}_j)$ will be more expensive than in the classical Gram-Schmidt algorithm with sparse $\hat{A}\hat{e}_j$.

For some classes of ill-conditioned matrices also the Modified Gram-Schmidt procedure gets numerically unstable. This occurs mainly for matrices with small, but very ill-conditioned submatrices $A(I, J)$. In this case, we have to use the Householder approach or some iterative refinement of the Gram-Schmidt algorithm, for example the method, introduced by Daniel, Gragg, Kaufmann, and Stewart [7]. Here, the columns q_k are refined iteratively to ensure the orthogonality against the previous columns. The main step in every iteration is like in the Gram-Schmidt algorithm. This method is described in [10], in our numerical tests we did not use it.

Note, that with these Gram-Schmidt-like approaches Y will have the special sparsity pattern (10) if we order the index set I relative to the new columns of $\hat{A}$. For (3'), we have to solve a linear equation with R and the right hand side $Y^T\hat{e}_j$, the j-th column of Q that is given explicitly. For the multiplication with Y^T and Y we can use the sparsity structure of Y, if we store one vector that contains the number of nonzero entries in every column of Y, resp. $\hat{A}$. In the same way, for evaluating (9b) we need one matrix-vector product $Y^T(\hat{A}\hat{e}_j)$, and can take advantage of the structure of Y and the sparsity of $\hat{A}\hat{e}_j$. Hence, for (8b) this approach is very favourable, but to be numerically stable we need robust generalizations of the Gram-Schmidt orthogonalization.

5. Numerical results

We used MATLAB programs

- with different parameters for choosing enlarged index sets,
- different orthogonalization methods,
- with and without using the sparsity of $\hat{A}$ and Y,

- based on (8a) or (8b).

The programs have the following input parameters:

- $1 > \epsilon > 0$, with M_k is accepted as soon as $\|r_k\| \leq \epsilon$;
- m, the maximum number of steps, where new entries are added;
- s, the maximum number of new entries in column M_k in one step;
- q, the maximum allowed number of nonzero entries per column of M;
- mi, the maximum number of indices for that we evaluate (9a) or (9b) (in the order of the magnitude of the entries of the actual residual); if $mi = 0$, then all entries of r_k are used without ordering.

We always start with $J = \emptyset$ and use $\epsilon = 0.4$. In the iterative method of Section 2, we stop the cg-method after $|J|$ steps or if the relative residual is smaller than 10^{-8}.

As iterative solver we consider only BiCGSTAB. We stop, if the relative residual is smaller than 10^{-8}. Other iterative methods like BCG, CGS, or GMRES show a similar convergence behaviour. As examples we present the matrices ORSIRR2, PORES2 transposed, and SHERMAN2 from the Harwell-Boeing collection [8], that are of increasing difficulty. The right hand side are random vectors, or the provided vector for SHERMAN2.

First, we consider the matrix ORSIRR2 with $n = 886, nnz(A) = 5970$ (nnz denotes the number of nonzero entries). In the following we consider the condition (8a) for choosing new indices and compare different methods for solving (3'). In Table 1 with (m, s, q) and $(10, 5, 15)$ we define $m = 10, s = 5, q = 15$.

mi=15, (8a), (m,s,q):	(10,5,15)	(5,5,10)	(20,1,20)
$nnz(M)$	3944	3739	3744
$\|AM - E\|_F$	8.799	8.953	9.869
# k with $\|r_k\| > 0.4$	0	40	2
$max\|r_k\|$	< 0.4	0.416	0.493
maximum # of entries in M_k	15	10	20
average # of steps	1.87	1.85	4.23
maximum # of steps	5	5	20
average # of new entries/step	2.38	2.28	1

Table 1. Sparse approximate inverses for various parameters, ORSIRR2

mi=15, (8a), (m,s,q):	(10,5,15)	(5,5,10)	(20,1,20)
GS	1.94E6	1.57E6	2.68E6
sparse GS	1.49E6	1.26E6	2.35E6
Modified GS	2.50E6	1.93E6	3.10E6
sparse Mod. GS	2.01E6	1.62E6	2.77E6
Householder	2.71E6	2.13E6	3.13E6
sparse Househ.	1.96E6	1.59E6	2.53E6
Givens	3.02E6	2.40E6	3.33E6
Fast Givens	2.35E6	1.93E6	2.95E6
Iteratively	2.76E6	2.35E6	5.52E6

Table 2. Number of flops for computing sparse approximate inverses, ORSIRR2

Table 1 shows that it is better to add more than one new index per step. The error $\|AM - E\|$ in the Frobenius norm and in the 1-norm and the computational costs are smaller for $s = 5$ than for $s = 1$. Hence, $s > 1$ helps to find profitable index sets. From Table 2. we see, that the sparse Gram-Schmidt-Orthogonalization needs the smallest number of flops, followed by the sparse Householder factorization.

In the next table we concentrate on the fastest QR-method, the sparse Gram-Schmidt-method, and display the sparse approximate inverses that are related to different choices of parameters.

m=10, s=5, q=15, (8a):	mi=0	mi=15	mi=10
$nnz(M)$	4139	3944	4100
$\|AM - E\|_F$	8.78	8.80	8.81
# k with $\|r_k\| > 0.4$	0	0	0
flops	2.0E6	1.5E6	1.7E6

Table 3. Sparse approximate inverses for various parameters mi, ORSIRR2

m=20, s=1, q=20, (8a):	mi=0	mi=15	mi=10	mi=5
$nnz(M)$	4162	3744	3862	4587
$\|AM - E\|_F$	9.94	9.87	9.86	9.82
# k with $\|r_k\| > 0.4$	9	2	2	2
$max\|r_k\|$	0.54	0.49	0.54	0.54
flops	5.1E6	2.4E6	2.3E6	3.2E6

Table 4. Sparse approximate inverses for various parameters mi, ORSIRR2

m=q=15, (8a):	s=6,mi=15	s=5,mi=15	s=4,mi=15	s=3,mi=9
flops	1.494E6	1.493E6	1.492E6	1.846E6

Table 5. Sparse approximate inverses for various parameters s and mi, ORSIRR2

These tables show that it is better to consider a limited but not too small number of indices mi related to the larger entries of r_k, and to add more than one element per step.

Next, we consider sparse approximate inverses by choosing new indices using (8b).

m=q=15, (8b):	s=1,mi=0	s=1,mi=10	s=3,mi=15
$nnz(M)$	3543	3967	3748
$\|AM - E\|_F$	9.36	9.34	8.80
# k with $\|r_k\| > 0.4$	0	0	0
maximum # of entries in M_k	10	13	11
average # of steps	4.00	4.48	2.05
maximum # of steps	10	13	6
average # of new entries/step	1	1	2.06

Table 6. Sparse approximate inverses for various parameters, ORSIRR2

m=q=15, (8b):	s=1,mi=0	s=1,mi=10	s=3,mi=15
GS	6.6E6	3.9E6	2.2E6
sparse GS	5.3E6	3.1E6	1.7E6
sparse Mod. GS	5.5E6	3.5E6	1.9E6
sparse Househ.	27.3E6	10.3E6	5.1E6

Table 7. Number of flops for computing sparse approximate inverses, ORSIRR2

The fastest method is again the sparse Gram-Schmidt algorithm. Hence, in the following we compare various sparse approximate inverses computed by the sparse Gram-Schmidt orthogonalization. Again we see, that appropiate chosen $mi > 0$ and $s > 1$ give the best results.

m=10, s=3, q=15, (8b):	mi=0	mi=15	mi=10
$nnz(M)$	3851	3748	4016
$\|AM - E\|_F$	8.79	8.80	8.81
# k with $\|r_k\| > 0.4$	0	0	0
flops	2.7E6	1.7E6	2.1E6

Table 8. Sparse approximate inverses for various parameters mi, ORSIRR2

m=q=mi=15,(8b):	s=5	s=4	s=3	s=2	s=1
$nnz(M)$	3999	3869	3748	3723	3566
$\|AM - E\|_F$	8.793	8.796	8.802	8.804	9.357
# k with $\|r_k\| > 0.4$	0	0	0	0	0
flops	1.85E6	1.75E6	1.72E6	2.01E6	2.92E6

Table 9. Sparse approximate inverses for various parameters s, ORSIRR2

Next, we display the total costs for solving a linear equation (1) with BiCGSTAB and computing the sparse approximate inverse with the sparse Gram-Schmidt method. The examples in Table 10 are related to optimal choices of the parameters with minimal total costs. The number of flops for computing M is more than a factor 2 smaller than the number of flops for the BiCGSTAB iteration in this sequentiell computation. Hence, also in a parallel environment the computation of M will be not expensive compared with the actual iteration for solving (1).

(m,s,q,8a or b):	(10,5,15,a)	(5,5,10,a)	(10,3,15,b)	(10,3,10,b)
flops for M	1.49E6	1.26E6	1.72E6	1.67E6
flops BiCGSTAB	3.72E6	3.84E6	3.67E6	3.67E6
Iter. BiCGSTAB	43	45	43	43
total flops	5.22E6	5.10E6	5.40E6	5.34E6

Table 10. Total costs for solving $Ax = b$, $mi = 15$, using (8a) or (8b), ORSIRR2

Next, we consider the transpose of the matrix PORES2 ($n = 1224, nnz = 9613$) from the Harwell-Boeing collection. Here, the Gram-Schmidt process was not stable for some columns, especially for the parameter $s = 1$. Hence, we use the Modified Gram-Schmidt method in the sparse version.

(m,s,8a or b):	(10,5,a)	(10,8,a)	(10,5,b)	(20,1,b)	(10,3,b)
$nnz(M)$	9376	9413	9845	6955	9623
$\|AM - E\|_F$	9.20	9.16	9.08	9.60	10.23
#k $\|r_k\| > 0.4$	57	50	40	57	104
$max\|r_k\|$	0.62	0.57	0.57	0.79	0.68
max.#entr.M_k	20	20	20	20	20
aver. # steps	2.4	2.3	2.8	5.7	3.4
max. # steps	8	7	8	20	9
av.#new en./st.	3.2	3.3	2.9	1	2.3
flops M	6.24E6	6.13E6	8.41E6	13.1E6	10.3E6
fl. BiCGSTAB	14.4E6	12.8E6	12.8E6	15.3E6	12.8E6
Iter. BiGSTAB	84	85	84	113	84
total flops	20.6E6	19.0E6	21.3E6	28.4E6	23.1E6

Table 11. PORES2 transposed with $q = 20$ and $mi = 24$.

For the last example, the matrix SHERMAN2 ($n = 1080, nnz = 23094$), only the Householder-QR-algorithm gave acceptable solutions of the least-squares problems. Gram-Schmidt, Modified Gram-Schmidt and the Givens-method failed. We present here two examples that gave satifactorily results based on the sparse Householder approach.

m=30, s=60, mi=60:	q=200, (8a)	q=100, (8b)
$nnz(M)$	54936	42139
$\|AM - E\|_F$	8.27	8.65
#k with $\|r_k\| > 0.4$	38	73
$max\|r_k\|$	0.873	0.996
maximum # of entries in M_k	200	100
average # of steps	6.2	3.7
maximum # of steps	25	14
average # of new entries/step	8.2	10.5
flops M	1.9E9	2.2E9
flops BiCGSTAB	1.5E7	2.6E7
Iterat. BiCGSTAB	30	61

Table 12. Sparse approximate inverses for SHERMAN2

Here, the total costs are ruled by the costs for computing M. But, if we try to use an approximate inverse with less entries, then we loose also the convergence in the iterative method. If we use the criterion (8b), then like described in [10], we need some iterative refined (and more expensive) Gram-Schmidt algorithm, e.g. the Daniel-Gragg-Kaufman-Stewart-method, or we apply also the Householder factorization like in Table 12. But this is not so effective in connection with (8b) in view of the implicit representation of Q.

6. Conclusions

Let us state some results of our theoretical and numerical considerations.

- Adding more than one new index per step by choosing $s > 1$ leads to slightly overshooting but helps to find profitable index sets and not to be caught with unprofitable indices. A good choice can be $s \approx p$ in connection with (8a) or $s \approx p/2$ with (8b) ($p = nnz(A)/n$). The mean value criterion helps to delete unprofitable indices.

- The total number q of allowed entries per column should be of the order of p. If this can not be satisfied the costs for computing M may be much higher then the costs for the iterative solver (compare Table 12).

- Considering not all possible new indices but only the indices related to

larger entries in r_k, saves costs. A good choice is $mi \approx 3p$, then there are enough candidates, and we can find profitable indices.

- The best choice for the QR-decomposition is the implicit Householder method for (8a), or the modified or the iteratively refined GramSchmidt method for (8b). The classical Gram-Schmidt algorithm would always be the fastest method, but is numerically unstable. In all QR-methods it is possible to reduce the costs by using the sparsity of A and Q.

- For problems with ill-conditioned submatrices $\hat{A}$ the exact minimization (8b) needs less entries in columns of M. Hence, for better load balancing between the columns M_k, (8b) may be favourable in this case. But in general our numerical results show that (8a) is peferable.

- In some examples also the iterative solution of the least-squares problem may be advisable. But then s should be large (≥ 5) and m should be small (≤ 3), in order to reduce the number of least-squares problems that have to be solved.

References

[1] Axelsson,O.: Iterative Solution Methods, *Cambridge University Press, Cambridge*, 1994.

[2] Barrett,R., Berry,M., Chan,T., Demmel,J., Donato,J., Dongarra,J., Eijkhout,V., Pozo,R., Romine,C., van der Vorst,H.: *Templates for the solution of linear systems: building blocks for iterative methods, SIAM, Philadelphia*, 1994.

[3] Benson,M.W., Frederickson,P.O.: Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems, *Utilitas Mathematica*, 22, pp. 127-140, 1982.

[4] Björck,A.: Least squares methods, in *P.G.Ciarlet and J.L.Lions, eds., Handbook of Numerical Analysis 1, Elsevier/North Holland, Amsterdam*, 1989.

[5] Chow,E., Saad,Y.: Approximate Inverse Preconditioners for general sparse matrices, *Research Report UMSI 94/101, University of Minnesota Supercomputing Institute, Minneapolis, Minnesota*, 1994.

[6] Cosgrove,J.D.F., Diaz,J.C., and Griewank,A.: Approximate inverse preconditioning for sparse linear systems, *Intl. J. Comp. Math. 44*, pp. 91-110, 1992.

[7] Daniel,J.W., Gragg,W.B., Kaufman,L.C., and Stewart,G.W.: Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR-factorization, *Mathematics of Computation, 30*, pp. 772-795, 1976.

[8] Duff,I.S., Grimes,R.G., and Lewis,J.G.: User's guide for the Harwell-Boeing sparse matrix collection (Release 1), *Technical Report RAL-92-086, Rutherford Appleton Laboratory, Chilton, England*, 1992.

[9] Golub,G., van Loan,Ch.: Matrix Computations, *The Johns Hopkins University Press, Baltimore, second ed.*, 1989.

[10] Gould,N.I.M., Scott,J.A.: On approximate-inverse preconditioners, *Technical Report RAL 95-026, Rutherford Appleton Laboratory, Chilton, England*, 1995.

[11] Grote,M., Huckle,T.: Parallel preconditioning with sparse approximate inverses,
SIAM J. Sci. Comput., (to appear).

[12] Grote,M., Huckle,T.: Effective parallel preconditioning with sparse approximate inverses, in *D.H.Bailey, ed., Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, SIAM, Philadelphia, pp. 466-471, 1995.

[13] Grote,M., Simon,H.D.: Parallel preconditioning and approximate inverses on the Connection machine, in *Parallel Processing for Scientific Computing, R.F. Sincovec, D.E. Keyes, L.R.Petzold, D.A.Reed,eds.*, SIAM, Philadelphia, PA, vol. 2, pp. 519-523, 1993.

[14] Kolotilina,L.Yu., Yeremin,A.Yu.: Factorized sparse approximate inverse preconditionings I. Theory, *SIAM J. Mat. Anal. 14*, pp. 45-58, 1993.

SFB 342: Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

bisher erschienen :

Reihe A

342/1/90 A Robert Gold, Walter Vogler: Quality Criteria for Partial Order Semantics of Place/Transition-Nets, Januar 1990

342/2/90 A Reinhard Fößmeier: Die Rolle der Lastverteilung bei der numerischen Parallelprogrammierung, Februar 1990

342/3/90 A Klaus-Jörn Lange, Peter Rossmanith: Two Results on Unambiguous Circuits, Februar 1990

342/4/90 A Michael Griebel: Zur Lösung von Finite-Differenzen- und Finite-Element-Gleichungen mittels der Hierarchischen Transformations-Mehrgitter-Methode

342/5/90 A Reinhold Letz, Johann Schumann, Stephan Bayerl, Wolfgang Bibel: SETHEO: A High-Performance Theorem Prover

342/6/90 A Johann Schumann, Reinhold Letz: PARTHEO: A High Performance Parallel Theorem Prover

342/7/90 A Johann Schumann, Norbert Trapp, Martin van der Koelen: SETHEO/PARTHEO Users Manual

342/8/90 A Christian Suttner, Wolfgang Ertel: Using Connectionist Networks for Guiding the Search of a Theorem Prover

342/9/90 A Hans-Jörg Beier, Thomas Bemmerl, Arndt Bode, Hubert Ertl, Olav Hansen, Josef Haunerdinger, Paul Hofstetter, Jaroslav Kremenek, Robert Lindhof, Thomas Ludwig, Peter Luksch, Thomas Treml: TOPSYS, Tools for Parallel Systems (Artikelsammlung)

342/10/90 A Walter Vogler: Bisimulation and Action Refinement

342/11/90 A Jörg Desel, Javier Esparza: Reachability in Reversible Free- Choice Systems

342/12/90 A Rob van Glabbeek, Ursula Goltz: Equivalences and Refinement

342/13/90 A Rob van Glabbeek: The Linear Time - Branching Time Spectrum

342/14/90 A Johannes Bauer, Thomas Bemmerl, Thomas Treml: Leistungsanalyse von verteilten Beobachtungs- und Bewertungswerkzeugen

342/15/90 A Peter Rossmanith: The Owner Concept for PRAMs

342/16/90 A G. Böckle, S. Trosch: A Simulator for VLIW-Architectures

342/17/90 A P. Slavkovsky, U. Rüde: Schnellere Berechnung klassischer Matrix-Multiplikationen

342/18/90 A Christoph Zenger: SPARSE GRIDS

Reihe A

342/19/90 A Michael Griebel, Michael Schneider, Christoph Zenger: A combination technique for the solution of sparse grid problems

342/20/90 A Michael Griebel: A Parallelizable and Vectorizable Multi- Level-Algorithm on Sparse Grids

342/21/90 A V. Diekert, E. Ochmanski, K. Reinhardt: On confluent semi-commutations-decidability and complexity results

342/22/90 A Manfred Broy, Claus Dendorfer: Functional Modelling of Operating System Structures by Timed Higher Order Stream Processing Functions

342/23/90 A Rob van Glabbeek, Ursula Goltz: A Deadlock-sensitive Congruence for Action Refinement

342/24/90 A Manfred Broy: On the Design and Verification of a Simple Distributed Spanning Tree Algorithm

342/25/90 A Thomas Bemmerl, Arndt Bode, Peter Braun, Olav Hansen, Peter Luksch, Roland Wismüller: TOPSYS - Tools for Parallel Systems (User's Overview and User's Manuals)

342/26/90 A Thomas Bemmerl, Arndt Bode, Thomas Ludwig, Stefan Tritscher: MMK - Multiprocessor Multitasking Kernel (User's Guide and User's Reference Manual)

342/27/90 A Wolfgang Ertel: Random Competition: A Simple, but Efficient Method for Parallelizing Inference Systems

342/28/90 A Rob van Glabbeek, Frits Vaandrager: Modular Specification of Process Algebras

342/29/90 A Rob van Glabbeek, Peter Weijland: Branching Time and Abstraction in Bisimulation Semantics

342/30/90 A Michael Griebel: Parallel Multigrid Methods on Sparse Grids

342/31/90 A Rolf Niedermeier, Peter Rossmanith: Unambiguous Simulations of Auxiliary Pushdown Automata and Circuits

342/32/90 A Inga Niepel, Peter Rossmanith: Uniform Circuits and Exclusive Read PRAMs

342/33/90 A Dr. Hermann Hellwagner: A Survey of Virtually Shared Memory Schemes

342/1/91 A Walter Vogler: Is Partial Order Semantics Necessary for Action Refinement?

342/2/91 A Manfred Broy, Frank Dederichs, Claus Dendorfer, Rainer Weber: Characterizing the Behaviour of Reactive Systems by Trace Sets

342/3/91 A Ulrich Furbach, Christian Suttner, Bertram Fronhöfer: Massively Parallel Inference Systems

342/4/91 A Rudolf Bayer: Non-deterministic Computing, Transactions and Recursive Atomicity

342/5/91 A Robert Gold: Dataflow semantics for Petri nets

Reihe A

342/6/91 A A. Heise; C. Dimitrovici: Transformation und Komposition von P/T-Netzen unter Erhaltung wesentlicher Eigenschaften

342/7/91 A Walter Vogler: Asynchronous Communication of Petri Nets and the Refinement of Transitions

342/8/91 A Walter Vogler: Generalized OM-Bisimulation

342/9/91 A Christoph Zenger, Klaus Hallatschek: Fouriertransformation auf dünnen Gittern mit hierarchischen Basen

342/10/91 A Erwin Loibl, Hans Obermaier, Markus Pawlowski: Towards Parallelism in a Relational Database System

342/11/91 A Michael Werner: Implementierung von Algorithmen zur Kompaktifizierung von Programmen für VLIW-Architekturen

342/12/91 A Reiner Müller: Implementierung von Algorithmen zur Optimierung von Schleifen mit Hilfe von Software-Pipelining Techniken

342/13/91 A Sally Baker, Hans-Jörg Beier, Thomas Bemmerl, Arndt Bode, Hubert Ertl, Udo Graf, Olav Hansen, Josef Haunerdinger, Paul Hofstetter, Rainer Knödlseder, Jaroslav Kremenek, Siegfried Langenbuch, Robert Lindhof, Thomas Ludwig, Peter Luksch, Roy Milner, Bernhard Ries, Thomas Treml: TOPSYS - Tools for Parallel Systems (Artikelsammlung); 2., erweiterte Auflage

342/14/91 A Michael Griebel: The combination technique for the sparse grid solution of PDE's on multiprocessor machines

342/15/91 A Thomas F. Gritzner, Manfred Broy: A Link Between Process Algebras and Abstract Relation Algebras?

342/16/91 A Thomas Bemmerl, Arndt Bode, Peter Braun, Olav Hansen, Thomas Treml, Roland Wismüller: The Design and Implementation of TOPSYS

342/17/91 A Ulrich Furbach: Answers for disjunctive logic programs

342/18/91 A Ulrich Furbach: Splitting as a source of parallelism in disjunctive logic programs

342/19/91 A Gerhard W. Zumbusch: Adaptive parallele Multilevel-Methoden zur Lösung elliptischer Randwertprobleme

342/20/91 A M. Jobmann, J. Schumann: Modelling and Performance Analysis of a Parallel Theorem Prover

342/21/91 A Hans-Joachim Bungartz: An Adaptive Poisson Solver Using Hierarchical Bases and Sparse Grids

342/22/91 A Wolfgang Ertel, Theodor Gemenis, Johann M. Ph. Schumann, Christian B. Suttner, Rainer Weber, Zongyan Qiu: Formalisms and Languages for Specifying Parallel Inference Systems

342/23/91 A Astrid Kiehn: Local and Global Causes

342/24/91 A Johann M.Ph. Schumann: Parallelization of Inference Systems by using an Abstract Machine

Reihe A

342/25/91 A	Eike Jessen: Speedup Analysis by Hierarchical Load Decomposition
342/26/91 A	Thomas F. Gritzner: A Simple Toy Example of a Distributed System: On the Design of a Connecting Switch
342/27/91 A	Thomas Schnekenburger, Andreas Weininger, Michael Friedrich: Introduction to the Parallel and Distributed Programming Language ParMod-C
342/28/91 A	Claus Dendorfer: Funktionale Modellierung eines Postsystems
342/29/91 A	Michael Griebel: Multilevel algorithms considered as iterative methods on indefinite systems
342/30/91 A	W. Reisig: Parallel Composition of Liveness
342/31/91 A	Thomas Bemmerl, Christian Kasperbauer, Martin Mairandres, Bernhard Ries: Programming Tools for Distributed Multiprocessor Computing Environments
342/32/91 A	Frank Leßke: On constructive specifications of abstract data types using temporal logic
342/1/92 A	L. Kanal, C.B. Suttner (Editors): Informal Proceedings of the Workshop on Parallel Processing for AI
342/2/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: The Design of Distributed Systems - An Introduction to FOCUS
342/2-2/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: The Design of Distributed Systems - An Introduction to FOCUS - Revised Version (erschienen im Januar 1993)
342/3/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: Summary of Case Studies in FOCUS - a Design Method for Distributed Systems
342/4/92 A	Claus Dendorfer, Rainer Weber: Development and Implementation of a Communication Protocol - An Exercise in FOCUS
342/5/92 A	Michael Friedrich: Sprachmittel und Werkzeuge zur Unterstüt- zung paralleler und verteilter Programmierung
342/6/92 A	Thomas F. Gritzner: The Action Graph Model as a Link between Abstract Relation Algebras and Process-Algebraic Specifications
342/7/92 A	Sergei Gorlatch: Parallel Program Development for a Recursive Numerical Algorithm: a Case Study
342/8/92 A	Henning Spruth, Georg Sigl, Frank Johannes: Parallel Algorithms for Slicing Based Final Placement
342/9/92 A	Herbert Bauer, Christian Sporrer, Thomas Krodel: On Distributed Logic Simulation Using Time Warp
342/10/92 A	H. Bungartz, M. Griebel, U. Rüde: Extrapolation, Combination and Sparse Grid Techniques for Elliptic Boundary Value Problems

Reihe A

342/11/92 A M. Griebel, W. Huber, U. Rüde, T. Störtkuhl: The Combination Technique for Parallel Sparse-Grid-Preconditioning and -Solution of PDEs on Multiprocessor Machines and Workstation Networks

342/12/92 A Rolf Niedermeier, Peter Rossmanith: Optimal Parallel Algorithms for Computing Recursively Defined Functions

342/13/92 A Rainer Weber: Eine Methodik für die formale Anforderungsspezifkation verteilter Systeme

342/14/92 A Michael Griebel: Grid– and point–oriented multilevel algorithms

342/15/92 A M. Griebel, C. Zenger, S. Zimmer: Improved multilevel algorithms for full and sparse grid problems

342/16/92 A J. Desel, D. Gomm, E. Kindler, B. Paech, R. Walter: Bausteine eines kompositionalen Beweiskalküls für netzmodellierte Systeme

342/17/92 A Frank Dederichs: Transformation verteilter Systeme: Von applikativen zu prozeduralen Darstellungen

342/18/92 A Andreas Listl, Markus Pawlowski: Parallel Cache Management of a RDBMS

342/19/92 A Erwin Loibl, Markus Pawlowski, Christian Roth: PART: A Parallel Relational Toolbox as Basis for the Optimization and Interpretation of Parallel Queries

342/20/92 A Jörg Desel, Wolfgang Reisig: The Synthesis Problem of Petri Nets

342/21/92 A Robert Balder, Christoph Zenger: The d-dimensional Helmholtz equation on sparse Grids

342/22/92 A Ilko Michler: Neuronale Netzwerk-Paradigmen zum Erlernen von Heuristiken

342/23/92 A Wolfgang Reisig: Elements of a Temporal Logic. Coping with Concurrency

342/24/92 A T. Störtkuhl, Chr. Zenger, S. Zimmer: An asymptotic solution for the singularity at the angular point of the lid driven cavity

342/25/92 A Ekkart Kindler: Invariants, Compositionality and Substitution

342/26/92 A Thomas Bonk, Ulrich Rüde: Performance Analysis and Optimization of Numerically Intensive Programs

342/1/93 A M. Griebel, V. Thurner: The Efficient Solution of Fluid Dynamics Problems by the Combination Technique

342/2/93 A Ketil Stølen, Frank Dederichs, Rainer Weber: Assumption / Commitment Rules for Networks of Asynchronously Communicating Agents

342/3/93 A Thomas Schnekenburger: A Definition of Efficiency of Parallel Programs in Multi-Tasking Environments

342/4/93 A Hans-Joachim Bungartz, Michael Griebel, Dierk Röschke, Christoph Zenger: A Proof of Convergence for the Combination Technique for the Laplace Equation Using Tools of Symbolic Computation

Reihe A

342/5/93 A Manfred Kunde, Rolf Niedermeier, Peter Rossmanith: Faster Sorting and Routing on Grids with Diagonals

342/6/93 A Michael Griebel, Peter Oswald: Remarks on the Abstract Theory of Additive and Multiplicative Schwarz Algorithms

342/7/93 A Christian Sporrer, Herbert Bauer: Corolla Partitioning for Distributed Logic Simulation of VLSI Circuits

342/8/93 A Herbert Bauer, Christian Sporrer: Reducing Rollback Overhead in Time-Warp Based Distributed Simulation with Optimized Incremental State Saving

342/9/93 A Peter Slavkovsky: The Visibility Problem for Single-Valued Surface (z = f(x,y)): The Analysis and the Parallelization of Algorithms

342/10/93 A Ulrich Rüde: Multilevel, Extrapolation, and Sparse Grid Methods

342/11/93 A Hans Regler, Ulrich Rüde: Layout Optimization with Algebraic Multigrid Methods

342/12/93 A Dieter Barnard, Angelika Mader: Model Checking for the Modal Mu-Calculus using Gauß Elimination

342/13/93 A Christoph Pflaum, Ulrich Rüde: Gauß' Adaptive Relaxation for the Multilevel Solution of Partial Differential Equations on Sparse Grids

342/14/93 A Christoph Pflaum: Convergence of the Combination Technique for the Finite Element Solution of Poisson's Equation

342/15/93 A Michael Luby, Wolfgang Ertel: Optimal Parallelization of Las Vegas Algorithms

342/16/93 A Hans-Joachim Bungartz, Michael Griebel, Dierk Röschke, Christoph Zenger: Pointwise Convergence of the Combination Technique for Laplace's Equation

342/17/93 A Georg Stellner, Matthias Schumann, Stefan Lamberts, Thomas Ludwig, Arndt Bode, Martin Kiehl und Rainer Mehlhorn: Developing Multicomputer Applications on Networks of Workstations Using NXLib

342/18/93 A Max Fuchs, Ketil Stølen: Development of a Distributed Min/Max Component

342/19/93 A Johann K. Obermaier: Recovery and Transaction Management in Write-optimized Database Systems

342/20/93 A Sergej Gorlatch: Deriving Efficient Parallel Programs by Systemating Coarsing Specification Parallelism

342/01/94 A Reiner Hüttl, Michael Schneider: Parallel Adaptive Numerical Simulation

342/02/94 A Henning Spruth, Frank Johannes: Parallel Routing of VLSI Circuits Based on Net Independency

342/03/94 A Henning Spruth, Frank Johannes, Kurt Antreich: PHIroute: A Parallel Hierarchical Sea-of-Gates Router

Reihe A

342/04/94 A Martin Kiehl, Rainer Mehlhorn, Matthias Schumann: Parallel Multiple Shooting for Optimal Control Problems Under NX/2

342/05/94 A Christian Suttner, Christoph Goller, Peter Krauss, Klaus-Jörn Lange, Ludwig Thomas, Thomas Schnekenburger: Heuristic Optimization of Parallel Computations

342/06/94 A Andreas Listl: Using Subpages for Cache Coherency Control in Parallel Database Systems

342/07/94 A Manfred Broy, Ketil Stølen: Specification and Refinement of Finite Dataflow Networks - a Relational Approach

342/08/94 A Katharina Spies: Funktionale Spezifikation eines Kommunikationsprotokolls

342/09/94 A Peter A. Krauss: Applying a New Search Space Partitioning Method to Parallel Test Generation for Sequential Circuits

342/10/94 A Manfred Broy: A Functional Rephrasing of the Assumption/Commitment Specification Style

342/11/94 A Eckhardt Holz, Ketil Stølen: An Attempt to Embed a Restricted Version of SDL as a Target Language in Focus

342/12/94 A Christoph Pflaum: A Multi-Level-Algorithm for the Finite-Element-Solution of General Second Order Elliptic Differential Equations on Adaptive Sparse Grids

342/13/94 A Manfred Broy, Max Fuchs, Thomas F. Gritzner, Bernhard Schätz, Katharina Spies, Ketil Stølen: Summary of Case Studies in FOCUS - a Design Method for Distributed Systems

342/14/94 A Maximilian Fuchs: Technologieabhängigkeit von Spezifikationen digitaler Hardware

342/15/94 A M. Griebel, P. Oswald: Tensor Product Type Subspace Splittings And Multilevel Iterative Methods For Anisotropic Problems

342/16/94 A Gheorghe Ştefănescu: Algebra of Flownomials

342/17/94 A Ketil Stølen: A Refinement Relation Supporting the Transition from Unbounded to Bounded Communication Buffers

342/18/94 A Michael Griebel, Tilman Neuhoeffer: A Domain-Oriented Multilevel Algorithm-Implementation and Parallelization

342/19/94 A Michael Griebel, Walter Huber: Turbulence Simulation on Sparse Grids Using the Combination Method

342/20/94 A Johann Schumann: Using the Theorem Prover SETHEO for verifying the development of a Communication Protocol in FOCUS - A Case Study -

342/01/95 A Hans-Joachim Bungartz: Higher Order Finite Elements on Sparse Grids

342/02/95 A Tao Zhang, Seonglim Kang, Lester R. Lipsky: The Performance of Parallel Computers: Order Statistics and Amdahl's Law

Reihe A

342/03/95 A Lester R. Lipsky, Appie van de Liefvoort: Transformation of the Kronecker Product of Identical Servers to a Reduced Product Space

342/04/95 A Pierre Fiorini, Lester R. Lipsky, Wen-Jung Hsin, Appie van de Liefvoort: Auto-Correlation of Lag-k For Customers Departing From Semi-Markov Processes

342/05/95 A Sascha Hilgenfeldt, Robert Balder, Christoph Zenger: Sparse Grids: Applications to Multi-dimensional Schrödinger Problems

342/06/95 A Maximilian Fuchs: Formal Design of a Model-N Counter

342/07/95 A Hans-Joachim Bungartz, Stefan Schulte: Coupled Problems in Microsystem Technology

342/08/95 A Alexander Pfaffinger: Parallel Communication on Workstation Networks with Complex Topologies

342/09/95 A Ketil Stølen: Assumption/Commitment Rules for Data-flow Networks - with an Emphasis on Completeness

342/10/95 A Ketil Stølen, Max Fuchs: A Formal Method for Hardware/Software Co-Design

342/11/95 A Thomas Schnekenburger: The ALDY Load Distribution System

342/12/95 A Javier Esparza, Stefan Römer, Walter Vogler: An Improvement of McMillan's Unfolding Algorithm

342/13/95 A Stephan Melzer, Javier Esparza: Checking System Properties via Integer Programming

342/14/95 A Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Point-to-Point Dataflow Networks

342/15/95 A Andrei Kovalyov, Javier Esparza: A Polynomial Algorithm to Compute the Concurrency Relation of Free-Choice Signal Transition Graphs

342/16/95 A Bernhard Schätz, Katharina Spies: Formale Syntax zur logischen Kernsprache der Focus-Entwicklungsmethodik

342/17/95 A Georg Stellner: Using CoCheck on a Network of Workstations

342/18/95 A Arndt Bode, Thomas Ludwig, Vaidy Sunderam, Roland Wismüller: Workshop on PVM, MPI, Tools and Applications

342/19/95 A Thomas Schnekenburger: Integration of Load Distribution into ParMod-C

342/20/95 A Ketil Stølen: Refinement Principles Supporting the Transition from Asynchronous to Synchronous Communication

342/21/95 A Andreas Listl, Giannis Bozas: Performance Gains Using Subpages for Cache Coherency Control

342/22/95 A Volker Heun, Ernst W. Mayr: Embedding Graphs with Bounded Treewidth into Optimal Hypercubes

342/23/95 A Petr Jančar, Javier Esparza: Deciding Finiteness of Petri Nets up to Bisimulation

Reihe A

342/24/95 A M. Jung, U. Rüde: Implicit Extrapolation Methods for Variable Coefficient Problems

342/01/96 A Michael Griebel, Tilman Neunhoeffer, Hans Regler: Algebraic Multigrid Methods for the Solution of the Navier-Stokes Equations in Complicated Geometries

342/02/96 A Thomas Grauschopf, Michael Griebel, Hans Regler: Additive Multilevel-Preconditioners based on Bilinear Interpolation, Matrix Dependent Geometric Coarsening and Algebraic-Multigrid Coarsening for Second Order Elliptic PDEs

342/03/96 A Volker Heun, Ernst W. Mayr: Optimal Dynamic Edge-Disjoint Embeddings of Complete Binary Trees into Hypercubes

342/04/96 A Thomas Huckle: Efficient Computation of Sparse Approximate Inverses

SFB 342: Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

bisher erschienen :

Reihe B

342/1/90 B Wolfgang Reisig: Petri Nets and Algebraic Specifications

342/2/90 B Jörg Desel: On Abstraction of Nets

342/3/90 B Jörg Desel: Reduction and Design of Well-behaved Free-choice Systems

342/4/90 B Franz Abstreiter, Michael Friedrich, Hans-Jürgen Plewan: Das Werkzeug runtime zur Beobachtung verteilter und paralleler Programme

342/1/91 B Barbara Paech1: Concurrency as a Modality

342/2/91 B Birgit Kandler, Markus Pawlowski: SAM: Eine Sortier- Toolbox -Anwenderbeschreibung

342/3/91 B Erwin Loibl, Hans Obermaier, Markus Pawlowski: 2. Workshop über Parallelisierung von Datenbanksystemen

342/4/91 B Werner Pohlmann: A Limitation of Distributed Simulation Methods

342/5/91 B Dominik Gomm, Ekkart Kindler: A Weakly Coherent Virtually Shared Memory Scheme: Formal Specification and Analysis

342/6/91 B Dominik Gomm, Ekkart Kindler: Causality Based Specification and Correctness Proof of a Virtually Shared Memory Scheme

342/7/91 B W. Reisig: Concurrent Temporal Logic

342/1/92 B Malte Grosse, Christian B. Suttner: A Parallel Algorithm for Set-of-Support
Christian B. Suttner: Parallel Computation of Multiple Sets-of-Support

342/2/92 B Arndt Bode, Hartmut Wedekind: Parallelrechner: Theorie, Hardware, Software, Anwendungen

342/1/93 B Max Fuchs: Funktionale Spezifikation einer Geschwindigkeitsregelung

342/2/93 B Ekkart Kindler: Sicherheits- und Lebendigkeitseigenschaften: Ein Literaturüberblick

342/1/94 B Andreas Listl; Thomas Schnekenburger; Michael Friedrich: Zum Entwurf eines Prototypen für MIDAS