On the Use of Block Stretching for Solving Unassembled Linear Systems.

Michel J. Daydé[1] — Jérôme P. Décamps[1] — Nicholas I. M. Gould[2]

[1]*ENSEEIHT-IRIT, 2 rue Camichel, 31071 Toulouse CEDEX, France,*
[2]*CCD, Rutherford Appleton Laboratory, Oxfordshire, OX11 0QX, England.*

ABSTRACT. *We consider so-called "matrix stretching" technique that make structured unassembled linear systems larger, but sparser. Our solution technique combines a direct factorization of the leading block diagonal submatrix of the stretched system, with a preconditioned conjugate gradient solution of the Schur complement system which results from the factorization of the diagonal blocks. We show that matrix stretching is an effective technique, particularly for ill-conditioned systems. The Schur complement is often considerably better conditioned than the whole system.*

RÉSUMÉ. *Nous nous intéressons à une technique d'étirement de matrices qui trasnforme des systèmes linéaires structurés en des systèmes plus grande taille et plus creux. Notre méthode de ésolution est composée d'une factorisation de la première sous-matrice du système étiré qui est bloc diagonale, et un gradient conjgué préconditioné sur le complément de Schur résultant de la factorisation des blocs diagonaux. Nous montrons que cette technique est particulièrement efficace sur des systèmes mal conditionnés, essentiellement parce que le complément de SChur est mieux conditionné que le système initial.*

1. Introduction

Stretching, a sparse matrix preprocessing technique that makes matrices sparser but, at the same time, larger was first introduced in [10]. [1] showed that such techniques are an effective way of treating matrices with dense rows or columns before forming their LDU or QR factorizations. Similar ideas have been developed by [14] and [2] for solving the Schur complements arising from interior point methods in the context of linear programming.

In this paper, we apply matrix stretching to the solution of a linear systems of "finite-element" type without assembling the coefficient matrices from its "elements". This requires that we introduce extra variables to recouple the "element" blocks of the matrix. The resulting enlarged system is a sparse "augmented system", which can be solved using a range of direct and iterative methods.

To be precise, we consider the solution of the symmetric system of linear equations of the form

$$\mathbf{B}\mathbf{x} = (\sum_{i=1}^{n_e} \mathbf{B}_i)\mathbf{x} = \mathbf{b}, \qquad [1.1]$$

where each symmetric elementary matrix $\mathbf{B}_i$ only involves a small subset of the variables $\mathbf{x}$. We let $\mathbf{B}^i$ be the matrix obtained from by $\mathbf{B}_i$ by removing its zero rows and columns. Note that we do not assume that each $\mathbf{B}_i$ is positive definite, merely that it is non-singular. Linear systems of the form (1.1) arise when solving the Newton equations for the unconstrained minimization of a partially separable function ([11]), and from finite-element methods for the solution of partial-differential equations.

The stretched system we obtain is an augmented system of the form

$$\begin{pmatrix} \mathbf{B}^{\mathrm{s}} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x}^{\mathrm{s}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{b}^{\mathrm{s}} \\ \mathbf{0} \end{pmatrix} \qquad [1.2]$$

where $\mathbf{B}^{\mathrm{s}}$ is block diagonal and $\mathbf{A}$ is very sparse. The diagonal blocks of $\mathbf{B}^{\mathrm{s}}$ are the $\mathbf{B}^i$. The matrix $\mathbf{A}$ is used to recouple the $\mathbf{B}_i$ when they share variables. The required solution $\mathbf{x}$ may easily recovered from $\mathbf{x}^{\mathrm{s}}$. The system (1.2) may be solved in many ways and we choose the well-known Schur complement approach.

2. Description of the stretching algorithm

The general algorithm for building the required stretched system is simple. Suppose $\mathbf{B}$ is of order n. We construct a system of the form

$$\begin{pmatrix} {\mathbf{B}_1}^{\mathrm{s}} & & & \mathbf{A}_1 \\ & \ddots & & \vdots \\ & & \mathbf{B}^{\mathrm{s}}_{n_e} & \mathbf{A}_{n_e} \\ \mathbf{A}_1^T & \ldots & \mathbf{A}_{n_e}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x}^{\mathrm{s}}_1 \\ \vdots \\ \mathbf{x}^{\mathrm{s}}_{n_e} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{b}^{\mathrm{s}}_1 \\ \vdots \\ \mathbf{b}^{\mathrm{s}}_{n_e} \\ \mathbf{0} \end{pmatrix} \qquad [2.3]$$

using the algorithm described in Algorithm 2.1.

Having solved the stretched system, we simply recover $\mathbf{x}$ from $\mathbf{x}^{\mathrm{s}}$, by setting $\mathbf{x}_i$ to its value in one of the blocks involving it. Note that we start the stretching process for variable i with the first element in which it is involved. Other strategies may prove more effective, and further experiments are required.

We now illustrate this on a simple example. We consider the solution of $\mathbf{Bx} = \mathbf{b}$ where $\mathbf{B}$ is structured as

$$\begin{pmatrix} 8 & 1 & \\ 1 & 8 & 1 \\ & 1 & 8 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}. \qquad [2.4]$$

Additionally, we assume that $\mathbf{B}$ is composed of two elements $\mathbf{B}_1$ and $\mathbf{B}_2$ — $\mathbf{B}_1$ involves variables $x_1,\ x_2$, while $\mathbf{B}_2$ involves $x_2,\ x_3$. Thus

$$\mathbf{B}^1 = \begin{pmatrix} 8 & 1 \\ 1 & 4 \end{pmatrix} \quad \text{and} \quad \mathbf{B}^2 = \begin{pmatrix} 4 & 1 \\ 1 & 8 \end{pmatrix}$$

The "overlap" in this simple example only involves the variable x_3. Our stretching algorithm then yields the symmetric augmented system

$$\begin{pmatrix} 8 & 1 & & & \\ 1 & 4 & & & +1 \\ & & 4 & 1 & -1 \\ & & 1 & 8 & \\ & +1 & -1 & & \end{pmatrix} \begin{pmatrix} x_1{}^{\mathrm{s}} \\ x_2{}^{\mathrm{s}} \\ x_3{}^{\mathrm{s}} \\ x_4{}^{\mathrm{s}} \\ \lambda_1 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ 0 \\ b_3 \\ 0 \end{pmatrix}. \qquad [2.5]$$

3. The solution of stretched linear systems

3.1. *The Schur complement method*

The Schur complement method (see [4]) takes advantage of the structure of the system (2.3). The method is formalized as Algorithm 3.1. This approach offers great opportunities for parallelisation since Steps 1 and 3 can be computed element-wise in parallel. Moreover, one hopes, in Step 2, to exploit parallelism within the PCG iteration — conjugate gradients (CG) are chosen since we may not want to form $\mathbf{S}$. We use the LDL^T factorization of the elements formed during the step 1 to compute the Schur complement. We then obtain

$$\mathbf{S} = \sum_{i=1}^{n_e} \mathbf{A}_i^T (\mathbf{B}_i{}^{\mathrm{s}})^{-1} \mathbf{A}_i = \sum_{i=1}^{n_b} \mathbf{s}_i \mathbf{s}_i^T, \qquad [3.6]$$

where n_b is the dimension of $\mathbf{B}^{\mathrm{s}}$ in (1.2). This decomposition proves to be useful during the construction of the preconditioners we consider.

Algorithm 2.1: The stretching algorithm

```
Initialization of $\mathbf{B}^{s}$
  For each element $i = 1, \cdots, n_e$
    $\mathbf{B}^{s}_{i}$ is the matrix $\mathbf{B}^{i}$
Initialization of $\mathbf{A}$ and $\mathbf{b}^{s}$
  For each variable $i = 1, \cdots, n$
    Compute $list$, the list of elements possessing $i$.
    If $i$ is shared by deg($i$) elements,
    create (deg($i$)-1) columns for each $\mathbf{A}_j$ for $j = 1, n_e$.
    $list(1)$ is the first element possessing $i$
    Let $k$ be the row of $\mathbf{B}^{s}_{list(1)}$ associated with variable $i$.
    Set the $(deg(i) - 1)$ columns of row $k$ of $\mathbf{A}_{list(1)}$ to 1.
    Set the corresponding value in $\mathbf{b}^{s}$ to $b_i$.
    For $j = 2, \cdots, deg(i)$
      Let $k$ be the row of $\mathbf{B}^{s}_{list(j)}$ associated with variable $i$.
      Set column $j - 1$ of row $k$ of $A_{list(j)}$ to $-1$
      Set the corresponding value in $\mathbf{b}^{s}$ to 0.
```

Algorithm 3.1: The Schur complement algorithm on the stretched system

1. $\forall i \in 1, .., n_e$, solve $\mathbf{B}_i{}^{s}\mathbf{z}_i = \mathbf{b}_i{}^{s}$ (using the LDL^T factorization from LAPACK, see [3])

2. Use the PCG method to solve $\mathbf{S}\boldsymbol{\lambda} = \mathbf{s}$, where the n_s by n_s Schur complement $\mathbf{S} = \sum_{i=1}^{n_e} \mathbf{A}_i^T (\mathbf{B}_i{}^{s})^{-1}\mathbf{A}_i$ and $\mathbf{s} = \sum_{i=1}^{n_e} \mathbf{A}_i^T\ \mathbf{z}_i$

3. Solve $\mathbf{B}_i{}^{s}\mathbf{x}_i{}^{s} = \mathbf{b}^{s} + \mathbf{A}_i\boldsymbol{\lambda}$ (using LAPACK)

4. $\forall k \in 1, .., n$, recover x_k from its value in $\mathbf{x}_i{}^{s}$, where (e.g.) $\mathbf{B}_i{}^{s}$ is the last element involving x_k.

3.2. *Preconditioning the Schur complement*

We have used a number of different preconditioners. These include the Chan diagonal preconditioner, a traditional band preconditioner, and the Element-by-Element preconditioners EBE. We consider each in some detail. Other preconditioners have also been considered in [7].

3.2.1. *The Chan diagonal preconditioner*

The aim here is to estimate values of the diagonal of $\mathbf{S}$ merely by forming products of $\mathbf{S}$ with appropriate vectors. [5] propose approximating this diagonal by the product of the Schur complement with p ($p < n_s$) so-called *probing* vectors. The i^{th} probing vector, $\mathbf{v}_i$, has ones in positions $i + kp$, $k = 0, 1, \cdots$, and zeros elsewhere. Any negative diagonal elements so formed are subsequently replaced by $\epsilon_m^{1/3}$ — ϵ_m is the of the machine precision — which appears, empirically, to be a satisfactory value.

3.2.2. *Traditional band preconditioner*

We might equally assemble a band submatrix of the Schur complement, and factorize it using the LANCELOT band factorization ([6]). Unlike the probing vector preconditioners, the band submatrix computed in this way is the exact band restriction of the Schur complement $\mathbf{S}$.

3.2.3. *Element-by-Element preconditioners*

Element-By-Element (EBE) preconditioners were introduced in [12] and [13] and have been successfully applied in a number of applications in engineering and physics and in the solution of partially separable linear systems (see [8]).

We can construct the elements of the Schur complement as the sum $\sum_{i=1}^{n_e} \mathbf{S}_i$ where $\mathbf{S}_i = \mathbf{A}_i^T \mathbf{B}_i^{-1} \mathbf{A}_i$ and then form an EBE preconditioner based this unassembled structure. Let $\Delta(\mathbf{S})$ and $\Delta(\mathbf{S}_i)$ be the diagonal parts of $\mathbf{S}$ and $\mathbf{S}_i$ respectively. Then we obtain the EBE preconditioner

$$\mathbf{P}_{EBE} \quad = \quad \Delta(\mathbf{S})^{1/2} \left\{ \prod_{i=1}^{n_e} \mathbf{L}_i \prod_{i=1}^{n_e} \mathbf{D}_i \prod_{i=n_e}^{1} \mathbf{L}_i^T \right\} \Delta(\mathbf{S})^{1/2},$$

where $\mathbf{L}_i$ and $\mathbf{D}_i$ are the LDL^T factors of the Winget terms

$$\mathbf{L}_i \mathbf{D}_i \mathbf{L}_i^T = \mathbf{W}_i \equiv \mathbf{I} + \Delta(\mathbf{S})^{-1/2} (\mathbf{S}_i - \Delta(\mathbf{S}_i)) \Delta(\mathbf{S})^{-1/2}$$

4. Experiments with matrix stretching

We have experimented matrix stretching on some of the unassembled matrices from the Harwell-Boeing collection (see [9]). Since only the structure of these problems is available, the numerical values of the elemental matrices have been set to random values. We have generated, for the two structures `CEGB2802` and `MAN5976`, four test problems with increasing condition numbers. Further experiments are available in [7].

In order to have an efficient Schur complement method, the size of the Schur complement should ideally be much smaller than the size of the initial problem. For this reason, we have used element amalgamation in our test problems. This is achieved with the merge algorithm introduced by [8]) where elements are amalgamated until a certain threshold is reached. The lower the threshold, the smaller is the Schur complement. For our experiments, the threshold used by the amalgamation algorithm is -10^5. Some of the characteristics of the problems used in our tests are given in Table 4.1.

We compare the following solution techniques:

— Solution of the initial system $\mathbf{Bx} = \mathbf{b}$:

> **CG Diag** — PCG with a diagonal preconditioner, and
>
> **CG EBE** — PCG with the Element-by-Element preconditioner EBE.

— Solution of the stretched system (1.2):

> **NONE** — unpreconditioned CG applied to $\mathbf{S}\boldsymbol{\lambda} = \mathbf{s}$,
>
> **CHANd** — PCG with the Chan diagonal preconditioner applied to $\mathbf{S}\boldsymbol{\lambda} = \mathbf{s}$,
>
> **EBE** — PCG with an EBE preconditioner applied to $\mathbf{S}\boldsymbol{\lambda} = \mathbf{s}$,
>
> **Band** — PCG with a band preconditioner applied to $\mathbf{S}\boldsymbol{\lambda} = \mathbf{s}$, and
>
> **Dir** — a direct solver applied to $\mathbf{S}\boldsymbol{\lambda} = \mathbf{s}$.

When solving the initial system (1.1) by an iterative method, we stop as soon as $||\mathbf{Bx}-\mathbf{b}|| \leq 10^{-9}\ ||\mathbf{b}||$; when we solve the stretched system (1.2) using Algorithm 3.1, Step 2 is terminated as soon as $||\mathbf{S}\boldsymbol{\lambda} - \mathbf{s}|| \leq 10^{-10}\ ||\mathbf{s}||$. In our experiments, the number of probing vectors used for the Chan diagonal preconditioner is $0.1n_s$. For the band preconditioner, we report results using a matrix of semi-bandwidth of $0.2n_s$. Other values were tried for the various preconditioners but those reported here achieved the best compromise over all the problems considered in our experiments.

In the Table 4.1, we report the results of the tests performed on our test set. *Ttotal* gives the total time required to solve the linear system and *Iter* gives the number of iterations required to solve the Schur complement system. The experiments were performed on a SUN workstation with a 125 MHZ HyperSPARC processor.

We make the following observations. Firstly, solving the systems using CG without preconditioning is the least effective method of all. Unfortunately, the Chan diagonal preconditioner also appears to be both rather costly and ineffective for these general matrices. Secondly, the EBE preconditioner appears to be reliable and efficient with all our test problems, for both the initial and stretched systems. However, the cost of constructing the preconditioner may be high, the method only pays off overall for the more ill conditioned problems. [8] observed a similar behaviour under more general circumstances. The main drawback when using this preconditioner on the Schur complement is that its memory requirements are sometimes prohibitive. Thirdly, it is clear here that the direct factorization of the Schur complement is a very appealing alternative when it is feasible. In our experiments it is often the best option, but EBE proves to be almost as competitive. We would not expect that, for larger problems, the method will not be as attractive, as the Schur complement, while having considerable hidden structure, is quite often dense. Finally, from a numerical point of view, the matrix stretching approach is highly interesting because of the observation we made in Table 4.1 that the Schur complement is, for the examples we considered, always better conditioned than the initial matrix. We noted that by amalgamating more and more of the elements from the problem `CEGB2802` the condition number of the Schur complement also improved. Once the condition number of the initial problem is large enough, matrix stretching appears to be more efficient than directly attacking the initial system.

CEGB2802

Threshold = -1e-5, n = 2694, n_e = 12, min = 267, max = 432, $aver$ = 290.3, n_s = 789, $\%T_s$ = 29.3

	Number		1	2	3	4
		κ	2.1×10^2	1.1×10^4	5.2×10^5	2.3×10^7
System	Preconditioner	κ_S	9.4×10^1	2.7×10^3	8.6×10^4	3.4×10^6
Stretched	NONE	Ttotal	17.5	66.4	282.4	609.6
		Iter	92	408	1822	3945*
	CHANd	Ttotal	23.0	40.5	91.7	169.2
		Iter	59	169	446	969
	EBE	Ttotal	**10.9**	**13.4**	17.5	**21.0**
		Iter	14	24	38	62
	Band	Ttotal	13.2	**13.4**	**14.8**	107.6
	20%	Iter	4	6	7	451
	Dir	Ttotal	24.7	24.9	24.6	23.7
		Iter	1	1	1	1
Initial	CG	Ttotal	**5.5**	25.3	132.9	668.3
	Diag	Iter	47	234	1245	6150
	CG	Ttotal	6.6	**15.7**	**58.1**	**241.7**
	EBE	Iter	13	48	211	915

MAN5976

Threshold = -1e-5, n = 5882, n_e = 22, min = 226, max = 445, $aver$ = 331.7, n_s = 1416, $\%T_s$ = 24.1

	Number		1	2	3	4
		κ	1.7×10^2	8.8×10^3	4.6×10^5	1.8×10^7
System	Preconditioner	κ_S	8.0×10^1	1.9×10^3	5.7×10^4	1.9×10^6
Stretched	NONE	Ttotal	38.2	135.6	501.6	1695.8
		Iter	84	337	1287	4368
	CHANd	Ttotal	72.6	121.7	264.5	1177.3
		Iter	59	181	543	2913
	EBE	Ttotal	**25.4**	**33.3**	**49.4**	80.6
		Iter	17	36	73	136
	Band	Ttotal	40.0	42.6	45.2	368.9
	20%	Iter	6	10	17	573
	Dir	Ttotal	67.6	67.6	68.2	**71.6**
		Iter	1	1	1	1
Initial	CG	Ttotal	**12.2**	51.0	196.4	632.0
	Diag	Iter	44	187	720	2482
	CG	Ttotal	14.3	**28.1**	**59.8**	**124.6**
	EBE	Iter	13	36	86	190

Table 4.1. *A comparison of the preconditioners on variants of* CEGB2802 *and* MAN5976. *Key: n is the number of variables in the initial problem; n_e is the number of blocks in the initial system; min, max and $aver$ are respectivly the minimum, maximum and average size of the blocks in the initial system; n_s is the number of variables in the Schur complement; and %T_s is the percentage 100 n_s / n; κ and κ_S are the condition numbers of, respectively, the problem and the Schur complement.*

5. Conclusions

In this paper, we have demonstrated the advantages of using a block stretching method in combination with a Schur complement solution approach. Such a method can be applied on the symmetric unassembled matrices arising from finite element problems or from partially separable optimization. The main benefit of matrix stretching appears to be that on ill-conditioned systems, the condition number of the Schur complement is lower than that of the initial system, and thus less sophisticated preconditioned may be required. The other main benefit of matrix stretching is obviously its potential for parallelisation, since the diagonal blocks within the stretched system can be factorized independently. The challenge is to make sure that the iterative solution step on the Schur complement is also effectively parallelised. Naturally, we do not expect such an approach to be efficient on all kinds of partially separable linear systems, but believe that we have demonstrated that, at least in some cases, there is considerable benefit to be gained from stretching. Amalgamation algorithms such as the ones described in [8] are of great importance since they may provide a decrease in size of the Schur complement. The most difficult problem is still to identify efficient preconditioners for the Schur complement.

6. References

[1] F. L. Alvarado. Matrix enlarging methods and their application. *BIT*, 37(3):473–505, 1997.

[2] K. D. Andersen. A modified schur-complement method for handling dense columns in interior-point methods for linear programming. *ACM TOMS*, 22(3):348–356, 1996.

[3] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. Lapack : A portable linear algebra library for high-performance computers. Technical Report Report CS-90-105, Computer Science Departement, University of Tennessee, 1990. Technical Report LAPACK Working Note 20.

[4] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, Cambridge, USA, 1996.

[5] T. Chan and D. Goovaerts. Domain decomposition benefical even sequentially. Technical Report Technical report CAM 88–18, Departement of Mathematic, UCLA, Los Angeles CA 90024–1555, 1988.

[6] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT*: a Fortran package for large-scale nonlinear optimization (Release A)*. Number 17 in Springer Series in Computational Mathematics. Springer Verlag, Heidelberg, Berlin, New York, 1992.

[7] M. Daydé, J. P. Décamps, and N. I. M. Gould. Subspace-by-subspace preconditioners for structured linear systems. Technical report, ENSEEIHT-IRIT, Toulouse, France, 1997. to appear.

[8] M. J. Daydé, J. Y. L'Excellent, and N. I. M. Gould. On the use of element-by-element preconditioners to solve large scale partially separable optimization problems. *SIAM Journal on Scientific Computing*, 18(6):1767–1787, 1997.

[9] I. S. Duff, R. G. Grimes, and J. G. Lewis. Users' guide for the Harwell-Boeing sparse matrix collection. Technical Report TR/PA/92/86, CERFACS, Toulouse, France, 1992.

[10] J. F. Grcar. Matrix stretching for linear equations. Technical Report SAND90-8723, Sandia National Laboratories, 1990.

[11] A. Griewank and Ph. L. Toint. Local convergence analysis for partitioned quasi-Newton updates. *Numerische Mathematik*, 39:119–137, 1982.

[12] T. J. R Hughes, I. Levit, and J. Winget. An element-by-element solution algorithm for problems of structural and solid mechanics. *Compututational Methods in Applied Mechanics and Engineering*, 36:241–254, 1983.

[13] M. Ortiz, P. M. Pinsky, and R. L. Taylor. Unconditionally stable element-by-element algorithms for dynamic problems. *Compututational Methods in Applied Mechanics and Engineering*, 36:223–239, 1983.

[14] R. J. Vanderbei. Splitting dense columns in sparse linear systems. *Linear Algebra Appl.*, 152:107–117, 1991.