NULL SPACE ALGORITHM AND SPANNING TREES IN SOLVING DARCY'S EQUATION

Mario Arioli[1] and Gianmarco Manzini[2]

ABSTRACT

A Null Space algorithm is considered to solve the augmented system produced by the mixed finite element approximation of Darcy's Law. The method is based on the combination of a Gaussian factorisation technique for sparse matrices with an iterative Krylov solver. The computational efficiency of the method relies on the use of spanning trees to compute the Gaussian factorization without fill-in and on a suitable stopping criterion for the iterative solver. We experimentally investigate its performance on a realistic set of selected application problems.

Keywords: Augmented systems, sparse matrices, mixed finite elements.

AMS(MOS) subject classifications: 65F05, 65F10, 65F50.

Current reports available by anonymous ftp to ftp.numerical.rl.ac.uk in directory pub/reports.
[1] M.Arioli@rl.ac.uk, Rutherford Appleton Laboratory,
[2] Gianmarco.Manzini@ian.pv.cnr.it, IMATI - CNR, via Ferrata 1, 27100 Pavia, Italy
The work of second author was supported by EPSRC grant GR/R46427/01.

Computational Science and Engineering Department
Atlas Centre
Rutherford Appleton Laboratory
Oxon OX11 0QX

September 25, 2001

Contents

1 Introduction

In this paper we present a null space method to solve the algebraic problem that arise from the Mixed Finite Element approximation of Darcy's law in dual formulation (Brezzi and Fortin 1991). Our strategy combines a spanning tree based direct LU factorisation of the divergence matrix with a preconditioned conjugate gradient iterative resolution of the linear problem for the projected Hessian matrix. For a detailed review of the bibliography on this topic the interested reader is referred to Arioli and Baldini (2001), where the theoretical properties concerning the backward stability when using finite-precision arithmetic has been investigated and proved, and to Arioli and Manzini (2001). We outline that the iterative solver stopping criterion has a strong impact on the computational efficiency of our approach. The stopping criterion adopted in this paper is based on an estimate of the approximation error in the energy norm of the problem. See Arioli (2000) for a more detailed theoretical presentation. The performance of the final algorithm is experimentally investigated on a representative set of problems. In the following, we will denote by E_1 and E_2 the $n \times m$ and $n \times (n-m)$ (with $m \leq n$) matrices

$$\mathsf{E}_1 = \begin{bmatrix} I_m \\ 0_{n-m,m} \end{bmatrix} \qquad \text{and} \qquad \mathsf{E}_2 = \begin{bmatrix} 0_{m,n-m} \\ I_{n-m} \end{bmatrix}. \tag{1}$$

2 Problem Formulation

The Darcy's law describes the relationship between the hydraulic head $p(\mathbf{x})$ and the velocity field $\mathbf{u}(\mathbf{x})$ in ground-water flow. The governing equations take the form

$$\begin{aligned} \mathbf{u}(\mathbf{x}) &= -K(\mathbf{x}) \operatorname{grad} p(\mathbf{x}), & \mathbf{x} \in \Omega, \\ \operatorname{div} \mathbf{u}(\mathbf{x}) &= f(\mathbf{x}), & \mathbf{x} \in \Omega, \end{aligned} \tag{2}$$

where Ω is a connected, bounded, polygonal domain in $\mathbb{R}^2$ defined by the closed boundary curve Γ. The first equation in (2) relates the vector field $\mathbf{u}$ to the scalar field p via the permeability tensor K, which accounts for the soil characteristics. The second equation relates the divergence of $\mathbf{u}$ to the source-sink term $f(\mathbf{x})$.

Equations (2) are completed by the following set of boundary conditions for both $\mathbf{u}$ and p

$$\begin{aligned} p(\mathbf{x}) &= g_D(\mathbf{x}), & \mathbf{x} \in \Gamma_D, \\ \mathbf{u} \cdot \mathbf{n} &= g_N(\mathbf{x}), & \mathbf{x} \in \Gamma_N, \end{aligned} \tag{3}$$

where $\mathbf{n}$ denotes the external normal to Γ, and Γ_D and Γ_N are two distinct and non-overlapping sub-sets of the domain boundary such that $\Gamma = \Gamma_D \cup \Gamma_N$. Dirichlet and Neumann-type boundary conditions are respectively imposed on Γ_D and Γ_N using two regular functions g_D and g_N. In the following, we will assume that $g_N = 0$.

Let us consider the mixed finite element method on the dual weak formulation of system (2) in the lowest-order Raviart-Thomas space. This method is defined on a family of triangulations $\mathcal{T}_h$, where the parameter h is the mesh diameter. Any mesh of $\mathcal{T}_h$ is a set of disjoint triangles τ covering Ω and such that no triangle has one vertex on g_D and another vertex on g_N and more

than one edge lying on Γ. We also assume that $\mathcal{T}_h$ is regular and conforming in the sense of Ciarlet (1978), which means that triangles do not degenerate in the approximation process for $h \to 0$. We refer to the book by Brezzi and Fortin (1991) for a detailed analysis.

Let us denote by n the number of edges and by m the number of triangles in the current mesh of $\mathcal{T}_h$. Using the $\mathcal{RT}_0$ discrete functional space for the approximation of the velocity field and the piece-wise constant $\mathcal{P}_0$ elements for the pressure field leads to the solution of the linear system

$$\begin{bmatrix} \mathsf{M} & \mathsf{A} \\ \mathsf{A}^T & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} q \\ b \end{bmatrix}, \tag{4}$$

where M is a $n \times n$ symmetric and positive definite matrix, and A^T the $n \times m$ divergence matrix. The augmented system (4) is non-singular because the *inf-sup* condition satisfied by the mixed finite element formulation implies that $\mathrm{Ker}\left(\mathsf{A}^T\right) \cap \mathrm{Ker}(\mathsf{M}) = 0$.

3 Graph properties of the matrix A

The matrix A in (4) shows very strong structural properties related to the edge-triangle graph connectivity of the mesh. This latter one, indicated by the symbol $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$, is associated to the 2-D triangulation as follows:

- $\mathcal{N} = \{\tau_1, \tau_2, \ldots \tau_m\} \cup \{\tau_R\}$ is the set of graph nodes corresponding to the mesh triangles $\tau_i \leftarrow T_i$, for $i = 1, m$; the root node τ_R is identified with the region external to Ω;

- $\mathcal{A} = \{(\tau_i, \tau_j)\}$ is the set of graph arcs where (τ_i, τ_j) is an arc if and only if the triangles T_i and T_j shares the edge e_{ij} or e_{ij} is a Dirichlet boundary edge.

Notice that $\mathcal{G}$ is binary graph when $\mathcal{T}_h$ is a 2-D mesh and a ternary one in the 3-D case.

The vector $\mathbf{n}_{e_i}$, which is orthogonal to the mesh edge e_i, is defined in order to point like the edge flux, the direction of this latter being uniquely determined by sign (u_i). The non-zero entries of A are thus equal to ± 1, and $\mathsf{A}_{e_i T'_{e_i}} + \mathsf{A}_{e_i T''_{e_i}} = 0$ when $e_i \cap \Gamma_D \neq e_i$. Let us define the column vector r whose non-zero entries corresponds to the edge indices i such that $e_i \subset \Gamma_D$ and whose values is such that $r_{e_i} + \mathsf{A}_{e_i T_{e_i}} = 0$. The $(m+1) \times n$ matrix $\mathsf{A}' = [\mathsf{A}, \mathsf{r}]$, obtained by augmenting A with the column vector r, is the incidence matrix of $\mathcal{N}$. The matrix A' is *totally unimodular* and has rank m (Alotto and Perugia 1999, Arioli and Manzini 2001). This implies that every sub-matrix obtained removing a column of A' has full rank (Tarjan 1983). Therefore, A is also a full rank matrix and its entries are equal to either 1, -1, or 0.

Every spanning tree of $\mathcal{G}$ with root in τ_R induces a partition of the graph arcs in *in-tree* arcs and *out-of-tree* arcs. In accord with a given spanning tree, the rows of the matrix A can be permuted by renumbering firstly the ones corresponding to in-tree arcs, and then the others. In-tree arcs (matrix rows) and graph nodes (matrix columns) are then renumbered using the spanning tree from root to leaves in a depth-search strategy. Let the symbol P and Q denote the two permutation matrices determined by this renumbering process. It turns out that the permuted matrix takes the form

$$\mathsf{PAQ} = \begin{bmatrix} \mathsf{L}_1 \\ \mathsf{L}_2 \end{bmatrix} = \begin{bmatrix} \mathsf{L}_1 & 0 \\ \mathsf{L}_2 & \mathsf{I}_{(n-m)} \end{bmatrix} \begin{bmatrix} \mathsf{I}_m \\ 0 \end{bmatrix} = \mathsf{LE}_1. \tag{5}$$

where L_1 is a non-singular lower triangular matrix.

The *optimal choice* for the rooted spanning tree is the one that minimizes the number of non-zero entries in the kernel matrix of A^T, that is $\mathsf{Z} = \mathsf{L}^{-T}\mathsf{E}_2$, as follows from the definition of the Null Space algorithm in the next section. This is equivalent to find the tree for which the sum of all the lengths of the fundamental circuits is minimal, which is an $\mathcal{NP}$-complete problem. Thus, we considered the two heuristic algorithms discussed below. The first one consists of the heap version of the *Shortest Path Tree* (SPT) algorithm, while the second one is the *Minimum Cost Tree* (MCT) algorithm in the Prim's implementation version (Tarjan 1983). Both ones use the cost function $\mathrm{cost}_{SPT} : \mathcal{A} \to \mathbb{R}_+$

$$\forall \alpha \in \mathcal{A} \begin{cases} \mathrm{cost}_{SPT}(\alpha) = 0 & \text{if } \exists i,\ \alpha = (\tau_R, \tau_i) \\ \mathrm{cost}_{SPT}(\alpha) = \mathsf{M}_{\alpha\alpha} & \text{otherwise.} \end{cases} \tag{6}$$

The set of paths produced by the SPT algorithm can be given an interesting physical interpretation, because it identifies the more significant flux lines in the following sense. A graph node (mesh triangle) lying in a low permeability zone — that is, where K^{-1} is very large — is connected to the root by a path of nodes mainly lying outside the low permeability zones. Thus, the paths built by the SPT algorithm try to avoid islands where the lowest values of permeability are attained and the flux is expected to be minimum.

4 The Null Space Algorithm

In this section, we shortly describe the classical null space algorithm, for the minimisation of linearly constrained quadratic forms, a complete presentation can be found in the book by Gill, Murray and Wright (1981). For the sake of simplicity, we assume that M, A and A^T indicate the matrices permuted using the edge/triangle numbering system built on a spanning tree as discussed in the previous section. Let $\mathsf{Y} \in \mathbb{R}^{n\times m}$ be a right pseudo-inverse of A^T and $\mathsf{Z} \in \mathbb{R}^{n\times(n-m)}$ be a basis of the kernel of A^T. That is,

$$\mathsf{Y}^T\mathsf{A} = \mathsf{I}_m \qquad \text{and} \qquad \mathsf{Z}^T\mathsf{A} = 0_{n-m,m}. \tag{7}$$

The Null Space algorithm can be formulated as follows:

Null Space Algorithm:

$$\begin{aligned} &1.\quad u_0 = \mathsf{Y}b, \\ &2.\quad \mathsf{Z}^T\mathsf{M}\mathsf{Z}\,w = \mathsf{Z}^Tq - \mathsf{Z}^T\mathsf{M}u_0 = s, \\ &3.\quad u = u_0 + \mathsf{Z}w, \\ &4.\quad p = \mathsf{Y}^Tq - \mathsf{Y}^T\mathsf{M}u. \end{aligned} \tag{8}$$

Arioli and Manzini (2002) presented a version of this algorithm that uses the matrices Y and Z built by an orthogonal factorisation of the matrix A. In the present work we instead consider the LU Gaussian factorisation of A that is obtainable *without floating-point operations* by exploiting the strong graph properties of the previous section. The sparsity of the Gaussian factors deriving from this approach make possible to compute implicitly and efficiently all the

matrix-vector products required by the algorithm. Step 2 requires to solve a linear system of the form $\mathsf{Z}^T\mathsf{M}\mathsf{Z}\,w = s$. This latter one can be solved by a preconditioned conjugate gradient method because $\mathsf{Z}^T\mathsf{M}\mathsf{Z}$ is a symmetric positive definite matrix, thus requiring only the specification of a matrix-vector product.

From (5), the matrices Y and Z can be implicitly computed as

$$\mathsf{Y} = \mathsf{L}^{-T}\mathsf{E}_1 \qquad \text{and} \qquad \mathsf{Z} = \mathsf{L}^{-T}\mathsf{E}_2. \tag{9}$$

The solution of the linear systems involving L or L^{-1} can be computed by visiting the tree without storing explicitly the matrices. Let $\tilde{\mathsf{M}} = \mathsf{L}^{-1}\mathsf{M}\mathsf{L}^{-T}$, then the matrix-vector products needed by the conjugate gradient method are computed by the following algorithm:

$$\tilde{\mathsf{M}}_{22}y = \mathsf{E}_2^T\mathsf{L}^{-1}(\mathsf{M}(\mathsf{L}^{-T}\mathsf{E}_2 y)). \tag{10}$$

This approach is particularly suitable when the projected Hessian matrix cannot be calculated because either the complexity would be too high – $\mathcal{O}((n-m)^3)$ – or the resulting matrix would be fairly dense, despite the sparsity of M. We point out that the dimension of $\tilde{\mathsf{M}}_{22}$ is the number of nodes of the mesh plus one $(n-m)$.

5 Stopping Criterion

As we use the conjugate gradient method, it is quite natural to have a stopping criterion which takes advantage of the minimisation property of this method. At each step j the conjugate gradient minimises the energy norm of the error $\delta w = w - w^{(j)}$ on a Krylov space. The space $\mathbb{R}^{n-m}$ with the norm

$$||\, y \,||_{\mathsf{Z}^T\mathsf{M}\mathsf{Z}} = \left(y^T\mathsf{Z}^T\mathsf{M}\mathsf{Z}y\right)^{\frac{1}{2}} \tag{11}$$

induces on its dual space the dual norm

$$||\, f \,||_{(\mathsf{Z}^T\mathsf{M}\mathsf{Z})^{-1}} = \left(f^T\left(\mathsf{Z}^T\mathsf{M}\mathsf{Z}\right)^{-1}f\right)^{\frac{1}{2}} \tag{12}$$

A stopping criterion such as

$$\textbf{if} \qquad ||\, \mathsf{Z}^T\mathsf{M}\mathsf{Z}\,w^{(j)} - s \,||_{(\mathsf{Z}^T\mathsf{M}\mathsf{Z})^{-1}} \leq \eta||\, s \,||_{(\mathsf{Z}^T\mathsf{M}\mathsf{Z})^{-1}} \qquad \textbf{then STOP}, \tag{13}$$

will guarantee that the computed solution $w^{(j)}$ satisfies the perturbed linear system

$$\begin{aligned} &\mathsf{Z}^T\mathsf{M}\mathsf{Z}\,w^{(j)} = s + f, \\ &||\, f \,||_{(\mathsf{Z}^T\mathsf{M}\mathsf{Z})^{-1}} \leq \eta||\, s \,||_{(\mathsf{Z}^T\mathsf{M}\mathsf{Z})^{-1}}. \end{aligned} \tag{14}$$

The choice of η will depend on the properties of the problem that we want to solve, and, in the practical cases $\eta >> \epsilon$, where ϵ is the rounding unit. Therefore, it is appropriate to analyse the influence of the perturbations on the error between the exact solutions u and p and the computed u^* and p^* neglecting the part depending on ϵ. Using the results of Arioli and Baldini (2001), we can prove that

$$\begin{aligned} ||\, u - u^* \,||_{\mathsf{M}} &\leq \eta||\, u - u_0 \,||_{\mathsf{M}} \\ ||\, p - p^* \,||_{\mathsf{A}^T\mathsf{M}^{-1}\mathsf{A}} &\leq \eta\sqrt{\zeta}||\, u - u_0 \,||_{\mathsf{M}} \end{aligned} \tag{15}$$

where ζ is the spectral radius of $\mathsf{Z}^T\mathsf{M}\mathsf{Z}\mathsf{E}_2\mathsf{M}^{-1}\mathsf{E}_2$.

Furthermore, we need to add some tool within the conjugate gradient algorithm for estimating the values in (13). The estimate ξ_j of $|| \mathsf{Z}^T\mathsf{M}\mathsf{Z}\, w^{(j)} - s \,||_{(\mathsf{Z}^T\mathsf{M}\mathsf{Z})^{-1}}$ can be computed using the algorithm proposed by Hestenes and Stiefel (1952) which is equivalent to the Gauss quadrature rule proposed by Golub and Meurant (1997) and tested by Arioli (2000) and Meurant (1999). At step j of the conjugate gradient, ξ_j estimates the true value of the error at step $j-d$, where d is an a priori selected integer value. Strakos and Tichy (2002) proved that the Hestenes and Stiefel rule is numerically stable in finite-precision arithmetic.

Finally, we must estimate $|| \, s \, ||_{(\mathsf{Z}^T\mathsf{M}\mathsf{Z})^{-1}}$. Let $r^{(0)T}$ be the residual $\mathsf{Z}^T\mathsf{M}\mathsf{Z}\, w^{(0)} - s$ computed at step 0. Taking into account that

$$\|s\|^2_{(\mathsf{Z}^T\mathsf{M}\mathsf{Z})^{-1}} = \|w\|^2_{\mathsf{Z}^T\mathsf{M}\mathsf{Z}} \geq s^T w^{(0)} - r^{(0)T} w^{(j)},$$

and that the lower bound converges to $\|w\|^2_{\mathsf{Z}^T\mathsf{M}\mathsf{Z}}$ (Arioli 2000), we replace $\|s\|_{(\mathsf{Z}^T\mathsf{M}\mathsf{Z})^{-1}}$ with its lower bound at the step j of the conjugate gradient. Therefore, (13) can be replaced by:

$$\text{IF} \quad \xi_j^2 \leq \eta^2 (s^T w^{(0)} - r^{(0)T} w^{(j)}) \quad \text{THEN STOP} \, . \tag{16}$$

Moreover, (16) needs only a scalar product at each step.

6 Preconditioners

Arioli and Manzini (2001) have shown how, using the spanning tree and the properties of the mesh, it is possible to compute *a priori* the structure of $\tilde{\mathsf{M}}_{22}$. This allows us to decide what kind of preconditioner we can afford and to identify the diagonal blocks of $\tilde{\mathsf{M}}_{22}$ that can be used to compute a block Jacobi preconditioner.

Both the SPT and MCT algorithms compute a tree that places on the diagonal of $\mathsf{M}_{22} = \mathsf{E}_2^T\mathsf{M}\mathsf{E}_2$ the biggest entries of the diagonal of M. Therefore, the possibility of using the simplest choice of the diagonal of M_{22} is sensible. Moreover, with this choice we can avoid the additional cost of computing either the diagonal (Jacobi) or the diagonal blocks (block Jacobi) of $\tilde{\mathsf{M}}_{22}$.

In Section 7, we will give numerical evidence that this choice is very efficient for several test problems.

7 Numerical Experiments

All our experiments were performed on a PC workstation comparing our free-matrix implementation of the Null Space algorithm and the sparse code **MA47** of the HSL (2000).

We compare the exact solution u and p of (4) with the values u^* and p^* computed by the null space algorithm where, in step 2, we used the conjugate gradient method with (16), we chose $w^{(0)} = 0$, and $d = 5$ in our stopping criterion. We assume that the values computed by the HSL routine **MA47** which implements a direct sparse Gaussian factorisation applied to (4) are exact.

We generated two test problems using a square unit box. In the first test problem, for a given triangulation, the values of $K(\mathbf{x})$ are constant within each triangle and this value is computed following the law: $K_{\tau_i} = 10^{-12 r_i^3}$, $i = 1, \ldots, m$ where $r_i \in [0,1]$ are numbers computed using a random uniform distribution. In the second test problem, we have four rectangular regions

where the tensor $K(\mathbf{x})$ assumes different values. In Figure 1, we illustrate the geometry of the domain and the boundary conditions. The values of the tensor $K(\mathbf{x})$ are chosen as follow: $K(\mathbf{x}) = 0.5$ in Ω_1, $K(\mathbf{x}) = 10^{-4}$ in Ω_2, $K(\mathbf{x}) = 10^{-4}$ in Ω_3, $K(\mathbf{x}) = 10^{-4}$ in Ω_4, and $K(\mathbf{x}) = 1$ in $\Omega \setminus \{\Omega_1 \cup \Omega_2 \cup \Omega_3 \cup \Omega_4\}$. In (2), we take the right-hand side $f(\mathbf{x}) = 0$ in all our test problems. We generated two regular meshes on Ω with an increasing number m of triangles. In Table 1,

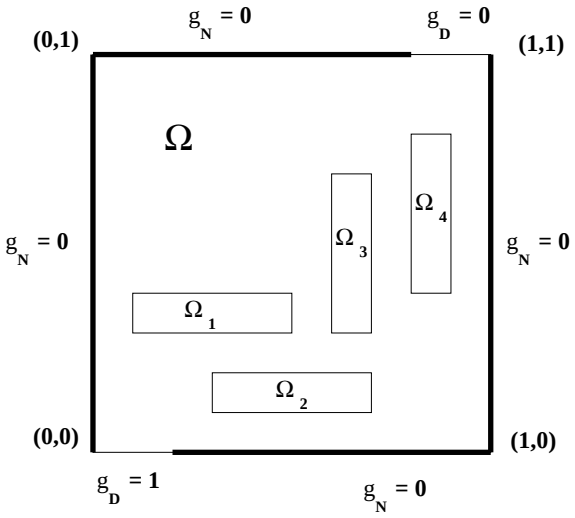

Figure 1: Domain Ω.

we report on the values n, m (number of degrees of freedom of the problem), N_V (number of nodes in the mesh) and the values of $nnz(M)$ and $nnz(A)$, the number of nonzero entries in M and A.

Table 1: Parameters of the runs.

Mesh	m	n	N_V	$nnz(M)$	$nnz(A)$	h
1	15472	23381	7908	116213	46107	0.02159
2	155746	234128	78381	1168604	466319	0.00687

In Tables 2-3, we summarise the numerical performance of the SPT versions of the algorithm for each test case and each mesh versus the performance of **MA47**. In Tables 4-5, we summarise the numerical performance of the SPT and MCT versions of the algorithm. All the results are relative to the use of $diag(\mathsf{M}_{22})$ as the preconditioner for the conjugate gradient method.

Finally, in Table 6, we report the performances of the conjugate gradient algorithm using the preconditoners $diag(\mathsf{M}_{22})$, Jacobi, and block Jacobi on the test problems relative to mesh 1.

The use of the stopping criteria (16) did not downgraded the numerical quality of the computed solution: the energy norm of the error is always $\mathcal{O}(h)$ (see Table 7).

Table 2: MA47 versus the null-space algorithm. Boxed domain with random permeabilities. CPU times (in seconds) and storage (in MBytes).

Mesh	**MA47**				null space algorithm (SPT)		
	Symb.	Fact.	Sol.	Stge	Symb.	CG(#It)	Sol.
1	0.51	0.75	0.03	9.03	0.04	0.81 (42)	0.04
2	5.53	27.96	0.25	129.72	0.64	35.12 (174)	0.32

Table 3: MA47 versus the null-space algorithm. Boxed domain with four isles of different permeabilities. CPU times (in seconds) and storage (in MBytes).

Mesh	**MA47**				null space algorithm (SPT)		
	Symb.	Fact.	Sol.	Stge	Symb.	CG(#It)	Sol.
1	0.41	0.33	0.02	6.09	0.04	1.07 (90)	0.04
2	5.42	9.87	0.20	82.08	0.51	68.01 (345)	0.40

Table 4: SPT versus MCT null-space algorithm. Boxed domain with random permeabilities. CPU times (in seconds) and storage (in MBytes).

Mesh	SPT null space algorithm			MCT null space algorithm		
	Symb.	CG(#It)	Sol.	Symb.	CG(#It)	Sol.
1	0.04	0.81 (42)	0.04	0.05	0.58 (30)	0.05
2	0.64	35.12 (174)	0.32	0.60	34.74 (175)	0.40

Table 5: SPT versus MCT null-space algorithm. Boxed domain with four isles of different permeabilities. CPU times (in seconds) and storage (in MBytes).

Mesh	SPT null space algorithm			MCT null space algorithm		
	Symb.	CG(#It)	Sol.	Symb.	CG(#It)	Sol.
1	0.04	1.70 (90)	0.04	0.05	1.84 (94)	0.03
2	0.51	68.01 (345)	0.40	0.58	75.10 (377)	0.41

Table 6: Comparison of the preconditioners on mesh 1.

	K	Preconditioner	#CG It.	CPU Time (in sec.)	
				Building	CG solve
SPT	Random	$diag(M_{22})$	42	0.00	0.81
	Random	$diag(\tilde{\mathsf{M}}_{22})$	16	6.03	0.32
	Random	$blockdiag(\tilde{\mathsf{M}}_{22})$	13	5.55	0.35
	isles	$diag(M_{22})$	90	0.01	1.70
	isles	$diag(\tilde{\mathsf{M}}_{22})$	69	5.65	1.36
	isles	$blockdiag(\tilde{\mathsf{M}}_{22})$	53	4.60	1.35
MCT	Random	$diag(M_{22})$	30	0.01	0.58
	Random	$diag(\tilde{\mathsf{M}}_{22})$	16	6.15	0.33
	Random	$blockdiag(\tilde{\mathsf{M}}_{22})$	14	5.53	0.38
	isles	$diag(M_{22})$	94	0.01	1.84
	isles	$diag(\tilde{\mathsf{M}}_{22})$	93	6.23	1.80
	isles	$blockdiag(\tilde{\mathsf{M}}_{22})$	68	5.47	1.71

Table 7: Velocity and pressure errors (SPT and $diag(M_{22})$ preconditioner)

K	h	$\|\|u-u^*\|\|_M/\|\|u\|\|_M$	$\|\|u-u^*\|\|_2/\|\|u\|\|_2$	$\|\|p-p^*\|\|_2/\|\|p\|\|_2$
Random	0.02159	0.01853	0.01313	0.00235
Random	0.00687	0.01775	0.01152	0.00145
isles	0.02159	0.03000	0.02869	0.00669
isles	0.00687	0.02025	0.01964	0.00322

8 Conclusions

The numerical results give evidence of a slightly better performance for the SPT choice. Even if **MA47** can be 6 time faster than the best version of the null space algorithm (see Table 4), we want to highlight that the absence of fill-in is promising when we need to solve Darcy's equations in 3D domains.

The cost of computing the Jacobi or the block Jacobi preconditioner makes the overall cost of the null space algorithm not competitive respect to its cost when we choose $diag(\mathsf{M}_{22})$ as preconditioner.

Nevertheless, in the presence of strong discontinuities and of anisotropies in the permeability function K, we are obliged to use either the diagonal Jacobi preconditioner or a block diagonal Jacobi to obtain convergence.

References

Alotto, P. and Perugia, I. (1999), 'Mixed finite element methods and tree-cotree implicit condensation', *CALCOLO* **36**, 233–248.

Arioli, M. (2000), A stopping criterion for the conjugate gradient algorithm in a finite element method framework, Technical Report Tech. Report IAN-1179, IAN.

Arioli, M. and Baldini, L. (2001), 'A backward error analysis of a null space algorithm in sparse quadratic programming', *SIAM J. Matrix Anal. and Applics.* **23**, 425–442.

Arioli, M. and Manzini, G. (2001), A network programming approach in solving Darcy's equations by mixed finite-element methods, Technical Report RAL-TR-2001-037, RAL-CLRC, Rutherford Appleton Laboratory, Oxon OX11 0QX, UK.

Arioli, M. and Manzini, G. (2002), 'A null space algorithm for mixed finite-element approximations of Darcy's equation', *Commun. Numer. Meth. Engng* **18**, 645–657. Also RAL-TR-2001-006.

Brezzi, F. and Fortin, M. (1991), *Mixed and Hybrid Finite Element Methods*, Vol. 15, Springer-Verlag, Berlin.

Ciarlet, P. (1978), *The Finite Element Method for Elliptic Problems*, North-Holland, Amsterdam, The Netherlands.

Gill, P. E., Murray, W. and Wright, M. H. (1981), *Practical Optimization*, Academic Press, London.

Golub, G. and Meurant, G. (1997), 'Matrices, moments and quadrature II; how to compute the norm of the error in iterative methods', *BIT* **37**, 687–705.

Hestenes, M. and Stiefel, E. (1952), 'Methods of conjugate gradients for solving linear systems', *J. Res. Nat. Bur. Standards* **49**, 409–436.

HSL (2000), 'Harwell Software Library. A collection of Fortran codes for large scale scientific computation', http://www.cse.clrc.ac.uk/Activity/HSL.

Meurant, G. (1999), 'Numerical experiments in computing bounds for the norm of the error in the preconditioned conjugate gradient algorithm', *Numerical Algorithms* **22**, 353–365.

Strakos, Z. and Tichy, P. (2002), 'On error estimation by conjugate gradient method and why it works in finite precision computations', *Electronic Transactions on Numerical Analysis*.

Tarjan, R. E. (1983), *Data Structures and Network Algorithms*, SIAM, Philadelphia, PA.