INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2005; **63**:1797–1823
Published online 18 April 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme.1330

A dual mesh multigrid preconditioner for the efficient solution of hydraulically driven fracture problems

A. P. Peirce[1,*,†] and E. Siebrits[2]

[1]*Department of Mathematics, University of British Columbia, Vancouver, BC, Canada V6T 1Z2*
[2]*Schlumberger Integrated Productivity and Conveyance Center, Sugar Land, TX 77478, U.S.A.*

SUMMARY

We present a novel multigrid (MG) procedure for the efficient solution of the large non-symmetric system of algebraic equations used to model the evolution of a hydraulically driven fracture in a multi-layered elastic medium. The governing equations involve a highly non-linear coupled system of integro-partial differential equations along with the fracture front free boundary problem. The conditioning of the algebraic equations typically degrades as $O(N^3)$. A number of characteristics of this problem present significant new challenges for designing an effective MG strategy. Large changes in the coefficients of the PDE are dealt with by taking the appropriate harmonic averages of the discrete coefficients. Coarse level Green's functions for multiple elastic layers are constructed using a single dual mesh and superposition. Coarse grids that are sub-sets of the finest grid are used to treat mixed variable problems associated with 'pinch points.' Localized approximations to the Jacobian at each MG level are used to devise efficient Gauss–Seidel smoothers and preferential line iterations are used to eliminate grid anisotropy caused by large aspect ratio elements. The performance of the MG preconditioner is demonstrated in a number of numerical experiments. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: multigrid preconditioning; fluid-driven fractures; hydraulic fracture; BEM

1. INTRODUCTION

Hydraulic fracturing is a process by which a fracture is propagated in a brittle material by forcing a fluid into the fracture. Such a physical process occurs both naturally and by human intervention. Examples of natural hydraulic fractures occur when pressurized magma forces the formation of fractures in the earth's crust (see Reference [1]) or the mechanism of 'seismic pumping' in which high fluid pressures that develop in deep impermeable rock formations lead

*Correspondence to: A. P. Peirce, Department of Mathematics, UBC, Vancouver, BC, Canada V6T 1Z2.
† E-mail: peirce@math.ubc.ca

Received 6 February 2004
Revised 2 July 2004
Accepted 22 December 2004

to the formation of fluid-driven fractures that propagate until the pressure is released. This cyclic process is believed to be one of the primary mechanisms for the formation of veins of mineral deposits [2]. The deliberate generation of hydraulic fractures is typically achieved by injecting water (or some other fluid) into a small section of a bore-hole under a sufficiently high pressure to overcome the tensile strength of the rock as well as the far-field minimum principal geological stress. As a result, a fracture surface, which is typically assumed to be planar, develops in a direction perpendicular to the far-field minimum principal geological stress. Examples of practical applications of hydraulic fracturing abound in the Geosciences. For example, hydraulic fracturing is routinely used by field engineers in the oil and gas recovery industry to induce fractures in reservoirs in order to substantially enhance the flow of hydrocarbons. Hydraulic fracturing has more recently been used in the mining industry to introduce large fractures in brittle rock to achieve more predictable and stable 'caving' of the ore-body—a procedure by which large volumes of rock in the roof of a mining excavation are induced to fall into the excavation for later processing. Environmental engineers have also used hydraulic fracturing to isolate toxic substances by injecting impermeable materials into fractures. In all these processes, it is desirable to be able to predict the evolution of the fracture surface under known stress and geological conditions in order to avoid undesirable fracture penetration of environmentally sensitive regions. Thus robust, efficient, and accurate numerical modelling of hydraulically driven fractures is of considerable interest.

The governing equations for the evolution of a fluid-driven fracture couple the Reynold's lubrication equation, which expresses the conservation of fluid volume, with a boundary integral equation, which expresses the balance of forces between the fluid pressure within the fracture, the far-field minimum principal geological stress in the rock, and the stresses induced in the layered elastic rockmass by the opening of the fracture. Alternatively full 3D finite element (FE) or finite difference discretizations could be used to model the semi-infinite layered elastic medium. However, the computational costs of these approaches would be considerable compared to the boundary integral formulation, which only involves discretization of that part of the plane that is actually fractured (see for example References [3, 4]). The footprint of the fracture at any time is not known *a priori*, which means that the fracture's perimeter needs to be determined as part of the problem. In order to determine the location of the fracture region as the fracture evolves, an additional propagation condition is required. This condition, from linear elastic fracture mechanics, requires that the stress intensity at each point on the tip of a propagating fracture is equal to the so-called fracture toughness of the material in which that portion of the fracture tip finds itself.

A number of discretization strategies have been used to approximate the governing equations. For example, Lagrangian FE methods have been used (see References [5, 6]) in which the fracture surface is re-meshed at each time-step. Unfortunately, the re-meshing requires an expensive re-generation of the fully populated Green's function influence matrix at each time-step of the calculation. The presence of elastic layers requires the solution of N three-dimensional boundary value problems to generate the Green's function matrix (where N is the number of degrees of freedom in the crack model). The juxtapositioning of thin and thick layers as well as large changes in elastic moduli between layers precludes the reliable use of various approximate solutions to these boundary value problems. Thus to reduce the computational burden of these influence calculations, we adopt an Eulerian approach in which the fracture is considered to grow into a pre-defined rectangular 'parent mesh' which conforms to the structure of the parallel elastic layers. The Green's influence matrix for this regular

problem can be determined *a priori* thus eliminating the need for expensive re-calculation of the influence coefficients at each time-step that an unstructured mesh would entail (see References [3, 4]). The structured regular mesh also enables the shift invariance of the governing integral equation parallel to the layers to be exploited. This yields considerable savings in terms of computational costs and memory resources, which makes it possible to contemplate models with a large number of degrees of freedom.

In this paper, we discuss the development of a MG algorithm to accelerate the solution of the large, fully populated, non-symmetric system of linear algebraic equations that need to be solved in the modelling of a fluid-driven fracture in an elastic medium. The Jacobian for these problems typically involves the product of an operator A, which expresses the conservation of fluid volume within the fracture, and an operator C, which characterizes the linear elastic pressure response of the medium to the fracture opening. If finite difference, finite volume, or finite element methods are used to discretize the fluid flow equations, then A is sparse. However, the Jacobian is fully populated since the non-local operator C is always fully populated. The condition number of the Jacobian increases with the cube of the number of degrees of freedom in the problem (see the analysis presented in Appendix A). This stiffness, which results in a dramatic increase in the number of iterations required to invert the Jacobian, combined with the expense of performing the matrix–vector products involved with each iteration, is a strong incentive to reduce the number of required iterations. This is achieved by developing a MG algorithm that can be used as a preconditioner for these algebraic equations.

The presence of the product AC in the Jacobian leads to a non-standard MG problem that poses a number of challenges. For example, what is the appropriate procedure to define coarse-grid C operators for piecewise homogeneous elastic media comprising many parallel layers with very different moduli? For a Newtonian fluid, the coefficients of the degenerate elliptic fluid flow operator A have a cubic dependence on the width w of the fracture. Moreover, it is not uncommon for the width in an element to differ by an order of magnitude from that of its neighbours. In this case, defining the coefficients of the A operator on coarser grids needs to be performed with due care. Because of the product AC in the Jacobian, it is not appropriate to simply use the 'black-box' MG approach of Dendy [7, 8] for defining the coarse mesh versions of the A operator, or other techniques that rely on operator-based interpolation combined with Galerkin coarsening [9]. In addition, depending on the far-field minimum principal stress that prevails in the elastic layers or on the relative stiffnesses of neighbouring elastic layers, sub-regions of the fracture footprint can develop in which the width of the fracture becomes vanishingly small or may even try to become negative. Such situations occur at so-called 'pinch points' at which the net pressure (i.e. the fluid pressure minus the far-field minimum principal stress) can be negative. In these elements it is necessary to introduce a minimum width constraint, which requires special treatment when developing a MG algorithm.

In Section 2 we describe the continuous and discrete coupled integro-partial differential equations that govern the evolution of a fluid-driven fracture. In Section 3, we describe components of the novel MG algorithm, including: the interpolation and restriction operators; the complementary field variable formulation for pinch-point problems; the coarse-grid fluid-flow operators; a novel dual mesh approach to define the coarse-grid Green's functions for layered elastic media; efficient Gauss–Seidel smoothers based on localized Jacobian operators; and line iteration strategies to mitigate the effect of mesh-induced anisotropy due to large aspect ratio elements. In Section 4, we present results on the performance of the MG preconditioner

for three distinct numerical examples. In Section 5, we make some concluding remarks. In Appendix A, the asymptotic conditioning of the coupled equations is demonstrated for a simple one-dimensional example.

2. GOVERNING EQUATIONS AND DISCRETIZATION

In a 3D layered elastic medium (see Figure 13(a) in which a schematic view of a fracture perimeter has been plotted) the integral equation governing the width profile for a planar crack can be written in the form

$$Cw = \int_{\Omega(t)} C(x, y; \xi, \eta) w(\xi, \eta, t)\, \mathrm{d}\xi\, \mathrm{d}\eta = p(x, y, t) - \sigma_c(x, y) \tag{1}$$

where the fracture is subjected to a fluid pressure $p(x, y, t)$ that works against the far-field minimum principal stress field $\sigma_c(x, y)$ within the elastic medium. If p is sufficiently large to overcome σ_c then the fracture opens by an amount $w(\xi, \eta, t)$. The Green's function $C(x, y; \xi, \eta)$ contains all the information about the stiffness of the layered elastic medium. The fracture at time t is assumed to occupy the region denoted by $\Omega(t)$, which has a boundary that we denote by $\partial\Omega$. The procedure for constructing the Green's functions for layered elastic materials both in plane strain and in three dimensions is described in Reference [3].

In the case of a planar fracture that grows in a 3D elastic medium, the fluid flow equations are well approximated by the 2D Reynold's lubrication equation:

$$\frac{\partial w}{\partial t} = \nabla \bullet (D(w) \nabla p) + \delta(x, y) Q \tag{2}$$

where $Q = Q(x, y, t)$ is the volume of fluid pumped into the fracture at the well bore. For simplicity, we have assumed no leakoff of fluid to the elastic medium and that the fluid is Newtonian so that $D(w) = w^3/12\mu$, where μ is the fluid viscosity. For a non-Newtonian power-law fluid $D(w)$ is replaced by $D(w, |\nabla p|)$ in which the dependence of D on w is larger than a cubic power while D has a power-law dependence on $|\nabla p|$ in which the exponent is positive.

We assume that the fracture region $\Omega(t)$ evolves in a window that has been divided into rectangular elements. For the discretization of the elastic integral equation (1) we assume that the width is piece-wise constant over each element and collocate at the element centres. In this case the discrete elasticity equations assume the form

$$Cw = p - \sigma_c \tag{3}$$

In the case of an inhomogeneous elastic medium the discretized collocation operator C is not symmetric.

By integrating both sides of (2) over the control element $\Delta x_C \Delta y_C$ centred at C (see Figure 1) and assuming that w and p are piecewise constant over each cell, the resulting set of difference equations can be expressed in the form

$$\begin{aligned} \frac{\Delta w}{\Delta t} &= A_w p_W + A_e p_E + A_s p_S + A_n p_N + A_c p_C + F_C \\ &= A(w) \bullet p + F \end{aligned} \tag{4}$$

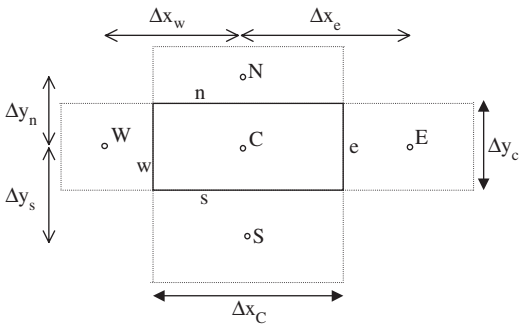


Figure 1. Centre, west, east, south and north elements and notation of parent cell system.

where for example $A_w = (D_w/\Delta x_w)(1/\Delta x_C)$, $A_e = (D_e/\Delta x_e)(1/\Delta x_C), \ldots, A_c = -(A_w + A_e + A_s + A_n)$, and $F_C = Q/(\Delta x_C \Delta y_C)$. In the event that $\Delta x_C = \Delta x_e = \Delta x_w$ and $\Delta y_C = \Delta y_s = \Delta y_n$ we obtain the well known coefficients of the form $A_w = D_w/(\Delta x)^2$ or $A_s = D_s/(\Delta y)^2$. Due to the stiffness of the coupled system matrix AC (see Appendix A), the backward Euler scheme has been used for time discretization.

We use Newton's method to solve the coupled equations (3) and (4) for the width and the pressure fields w and p within the crack surface, which can be expressed in the form

$$J(AC)\delta w_k = -r_k \tag{5}$$

where $r_k = \Delta w_k - \Delta t A p_k - \Delta t F$ is the kth residual. Since C is not a symmetric matrix, the Jacobian $J(AC)$ is also not symmetric.

For pinch points, at which the net pressure within the fracture is negative, it is mathematically possible for the width to become negative (since the kernel of the elastic integral equation is positive). However, this represents the physically impossible situation in which the two sides of the fracture inter-penetrate. Thus to avoid this aphysical situation from occurring, a minimum width constraint $w \geqslant w_c$ needs to be imposed within these regions of the fracture. In this case, the linearized system of equations that needs to be solved involves a mixture of the primary variables. In particular, within the region where the width constraint is active, the fluid pressure becomes the primary variable, while, within the region where the width constraint is not active, the width of the fracture is the primary unknown.

3. MG PRECONDITIONER FOR HYDRAULIC FRACTURE PROBLEMS

3.1. MG iteration procedure

In this paper, we discuss the efficient solution of the non-symmetric system of linear equations (5) using the BiCGSTAB algorithm (see References [10, 11]) with MG preconditioning. This approach adds considerable robustness to the solution strategy and results in an algorithm with substantially reduced iteration counts—relative to the standard BiCGSTAB algorithm.

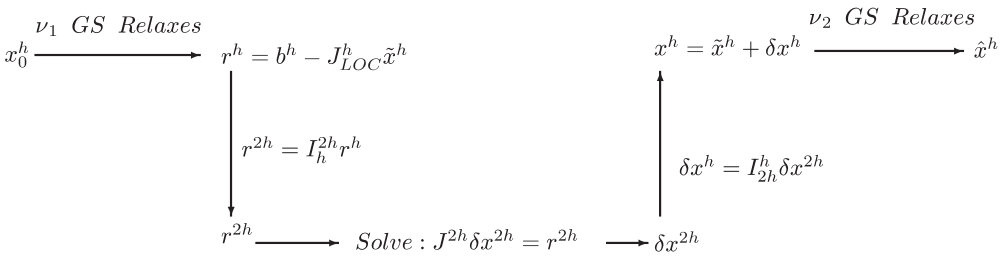


Figure 2. Single MG V cycle.

Since there are a number of excellent monographs on MG methods (see for example Reference [12] and references therein) we will restrict our discussion to a brief description of MG emphasizing the novel features of the MG algorithm proposed in this paper. The basic idea behind a successful MG algorithm is to use a complementary strategy that involves fine-grid relaxation to smooth the high-frequency errors in a trial solution and coarse-grid correction to eliminate the low-frequency errors. Typically these operations are sequenced in such a way that the high-frequency errors on the finest mesh are removed by a small number of smoothing relaxations to enable the trial solution to be transferred to a coarser mesh for further correction. The transfer of the fine-grid problem to a coarser grid involves the application of so-called restriction operators. This coarsening process is repeated recursively until a discrete model with very few degrees of freedom is obtained. The coarsest grid problem is typically solved to high precision—often by LU decomposition. This coarsest grid correction yields the lowest frequency information about the solution, which typically eludes most iterative smoothers when applied to the fine mesh problem. The coarse-grid correction is then transferred back to the finest grid by interpolating these corrections back onto the same sequence of finer grids taken in reverse order and applying a small number of smoothing relaxations to eliminate spurious errors introduced by the interpolation. This process describes the elements of a single MG V cycle. There are a number of variations on this theme. For example, MG W cycles: when the coarse-grid corrections are interpolated to the finest grid, each of the coarser grids may be re-visited several times before proceeding to the finest level.

In this paper, we will consider preconditioners based on MG $V(\nu_1, \nu_2)$ cycles represented schematically in Figure 2 for the case of just two grids for the problem $J^h x^h = b^h$. We observe that the ν_k Gauss–Seidel relaxations make use of a localized Jacobian J^h_{LOC} (which will be described later in this section) while the coarse-grid solve is performed using an LU decomposition of the full coarse Jacobian. We will compare the performance when $\nu_1 = 0$ to that when $\nu_1 \neq 0$.

3.2. Grid coarsening and width constraints

The MG procedure makes use of a sequence of coarser meshes that double in size with each coarsening. Due care is required if width constraints are active on the finest mesh. Within the regions in which a width constraint is active, the fluid pressure becomes the primary variable, while, within the regions where the width constraint is not active, the width of the fracture is the primary unknown. In order to be able to unambiguously represent constrained and unconstrained field variables on the coarser meshes, we adopt a grid coarsening procedure in which the nodes

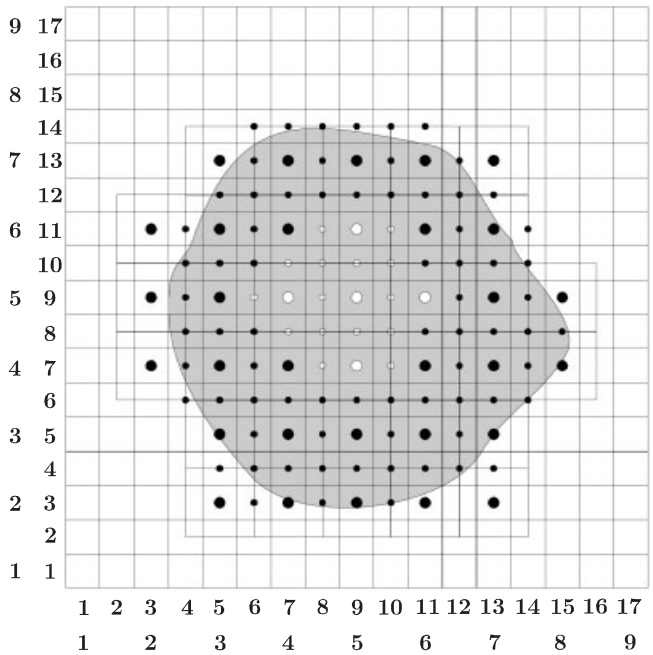


Figure 3. A fracture surface (shaded) is modelled on a fine grid by elements whose centres are marked with (○ and ●) symbols. The (○) symbols are used to denote those elements for which there is a width constraint active, while the (●) symbols are used to indicate those elements that are free to open. In the MG process the fracture is also modelled by a grid with twice the dimensions that encapsulates the fracture surface. The centres of these coarse-grid elements are represented by larger circles. Observe that the coarser mesh introduces some elements whose nodes were not on the fine mesh.

for the coarse mesh elements are a subset of those on the fine mesh. In Figure 3 we indicate the coarse mesh model which results from coarsening the fine level representation of the given fracture. The centres of the coarse (fine) mesh elements are denoted by the larger (smaller) circular nodes. The coarse and fine mesh elements at which width constraints are active are denoted by the empty circles, while those elements at which the width is free to vary are denoted by solid circles. Note that this coarsening process may introduce coarse mesh elements that were not active nodes in the fine mesh representation of the fracture. In this illustration the Eulerian approach to the fracture modelling can be seen. The fracture is arranged to evolve within a 17×17 fine mesh computational window which is coarsened to a 9×9 coarse mesh computational window.

3.3. Interpolation and restriction operators

We use the standard bilinear interpolation operator I_{2h}^{h} (see Reference [12]) to transfer coarse-grid field variables to finer grids, i.e.

$$w^h = I_{2h}^{h} w^{2h}$$

and the complementary restriction operator $I_h^{2h} = \frac{1}{4}[I_{2h}^h]^{\mathrm{T}}$ is used to transfer fine mesh to coarse mesh field variables.

In a mixed problem, in which a width constraint is active, the primary unknowns in the problem are the incremental fluid pressures δp_c at points at which width constraints are active and the incremental fracture widths δw_f at the remaining points. For the MG algorithm it is necessary to interpolate both these field variables from coarse grids to fine grids. Let $[\delta p_c \ \delta w_f]^{\mathrm{T}}$ be the vector comprising these mixed unknowns. The corresponding perturbation to the width field for all points within the fracture is $[0 \ \delta w_f]^{\mathrm{T}}$—since the constrained widths do not vary. The corresponding pressure field over the fracture is given by $[\delta p_c \ C_{ff}\delta w_f]^{\mathrm{T}}$, where $C_{ff}\delta w_f$ represents the elastic stresses on the unconstrained elements due to the width perturbations δw_f. In order to perform the interpolation to a finer mesh, it is necessary to first form the pressure and width fields at all points within the fracture and then to apply the interpolation operators.

3.4. Coarse-grid operators

The following Galerkin procedure (see References [8, 9]) is commonly used for defining coarse-grid elliptic operators

$$C^{2h} = I_h^{2h} C^h I_{2h}^h \quad \text{and} \quad A^{2h} = I_h^{2h} A^h I_{2h}^h \tag{6}$$

where I_{2h}^h and I_h^{2h} are the interpolation and restriction operators such as those defined in the previous sub-section. For problems in which there are large changes in the coefficients of A for example, it is necessary to define the weights of the interpolation operator in terms of the coefficients of A (see for example References [7–9]) in order to conserve flux in the corresponding coarse-grid operator. We need to construct coarse-grid matrix product AC operators, thus an interpolation procedure based on the coefficients of A is inappropriate for interpolating w and for defining the coarse-grid C operators. Using a Galerkin procedure with different interpolation operators for A and C leads to a scheme in which the interpolation operators for field variables w and restriction operators for residuals are incompatible. In this sub-section, we describe the procedure used to define the appropriate coarse-grid operators C^{2h} and A^{2h} that are used to define the combined operator $(A^{2h}C^{2h})$.

3.4.1. Coarse-grid elasticity operators C^{2h}: a dual mesh approach. Since the calculation of the elements of the elasticity matrix is a computationally intensive process, which is too costly to calculate for each fracture footprint $\Omega(t)$, we precalculate and store the complete set of numerical Green's function influence coefficients that would be required to determine all possible element-to-element influences within the rectangular computational window within which the fracture is assumed to evolve. In this section, we describe how the complete set of coarse-grid elasticity influences can be constructed by extending the computational window to an encompassing dual computational window and calculating a set of influences for elements that have dimensions $\Delta x/2 \times \Delta y/2$, where Δx and Δy are the mesh dimensions on the finest MG level. Combining these influences on the dual computational window with those in the fine-grid computational window, it is possible to assemble the full set of coarse-grid Green's function coefficients for each of the computational windows for each of the MG levels.

In Figure 4, we illustrate this procedure for a fine mesh comprising 5×5 rectangular elements which is coarsened to a 3×3 mesh. Of the nine coarse mesh elements, the one centred on

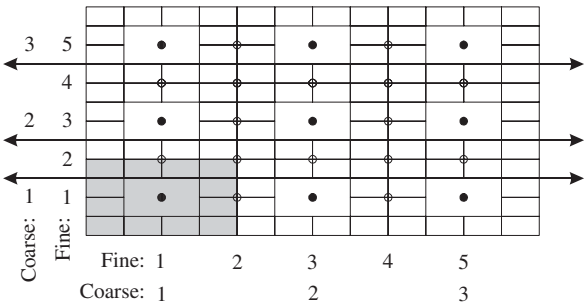


Figure 4. Fine and coarse mesh elements as well dual mesh sub-elements that are used to construct the coarse-grid influences. The centroids of the fine mesh elements include those marked with ○ and • symbols, while the • symbols denote those fine mesh nodes that are also on the coarse mesh. The arrows indicate elastic layer interfaces.

the co-ordinates $(1, 1)$ is shaded so that it can be easily identified. We also depict elements having half the dimensions of the elements in the original 5×5 fine mesh, which form part of the so-called dual mesh. The elements of this dual mesh form part of a 12×12 mesh that encompasses the elements of the coarse mesh. We have only depicted the subset of the 12×12 dual mesh elements that are required to calculate the coarse mesh influences. The dual mesh elements that are required to assemble the coarse mesh influences can be clearly seen in the case of the shaded coarse mesh element. For a homogeneous medium the introduction of the dual mesh is an unnecessarily complicated way to calculate the influences of the coarse mesh since the coarse mesh influences can be calculated directly. However, for a layered elastic medium the construction of the matrix coefficients for sending influences from coarse-grid elements that straddle two or more layers needs to be done by the superposition of component influences either side of the layer interfaces. To illustrate this point, the mesh shown in Figure 4 is intersected by three layer interfaces (shown by the heavy lines with arrows) which define four distinct elastic layers. Observe that the elements of the 5×5 fine mesh are contiguous with the layer interfaces, while those of the coarse mesh straddle the layers (see for example the shaded coarse-grid element $(1, 1)$). The influence due to the coarse-grid sending element at $(1, 1)$ is constructed by superposition of the influence due to the fine-grid element located at $(1, 1)$ and those of the 12 dual grid neighbours that make up the coarse-grid element. If the coarse-grid is coarsened further we observe that the influences of these coarse-grid elements can be constructed by enlarging the encompassing dual mesh to include 16×16 elements and by using the coarse-grid influences of the first level of coarsening. Thus the dual mesh with elements having half the dimensions of those on the fine mesh, i.e. $(\Delta x/2) \times (\Delta y/2)$, can be used to construct a full range of coarse-grid influences for multiple levels of coarsening.

For a given sending element numbered j let N_j denote the set of neighbouring dual sending elements that constitute the coarse mesh sending element centred at node j. For example, the dual mesh elements shown in Figure 4 that form the set $N_{(1,1)}$ of neighbouring elements for the element centred at node $(1, 1)$ are shaded. Rather than refer to elements in terms of ordered pairs, we will refer to an element according to some global element numbering sequence. Let C_{ij}^h denote the fine mesh elastic influence of the jth fine mesh sending element on the

ith node and let $C_{ik}^{h/2}$ denote the dual mesh elastic influence of the kth dual mesh neighbour on the ith node. Then the desired coarse mesh influence $C_{ij}^{2h}$is given by

$$C_{ij}^{2h} = C_{ij}^{h} + \sum_{k \in N_j} C_{ik}^{h/2}$$

3.4.2. Coarse-grid fluid-flow operators: *a harmonic mean approach.* We make use of the flux matching approach introduced by Alcouffe *et al*. [13] to obtain coarse-grid N, S, E and W coefficients for a five node stencil by taking a weighted average of the harmonic means of the fine-grid coefficients. For example to calculate the E coarse-grid coefficient that governs the flow between the coarse-grid nodes (i, j) and $(i+2, j)$ we use the formula:

$$D_{i,j}^{2h,E} = \frac{1}{2}\,\delta\left(D_{i,j}^{h,E}, D_{i+1,j}^{h,E}\right) + \frac{1}{4}\left(\delta\left(D_{i,j+1}^{h,E}, D_{i+1,j+1}^{h,E}\right) + \delta\left(D_{i,j-1}^{h,E}, D_{i+1,j-1}^{h,E}\right)\right) \tag{7}$$

where $\delta(a, b)$ is the harmonic mean operator defined by $\delta(a, b) = 2ab/(a+b)$ (Figure 5).

3.5. Smoothing operators

Another essential ingredient for a successful MG strategy is the appropriate smoothing operator to damp the high-frequency errors before and after the restriction or interpolation operators are applied. Because the elasticity matrix is fully populated, multiplication by C soon dominates the cost of the solution process as the number of degrees of freedom in the model increases. Indeed, evaluating and storing the elements of C can involve considerable computational expense and memory resources. Thus it is typical to take advantage of as many symmetries in the problem as possible. For example, in a layered elastic medium, exploiting the shift invariance of the problem parallel to the layers can lead to considerable savings in memory requirements and

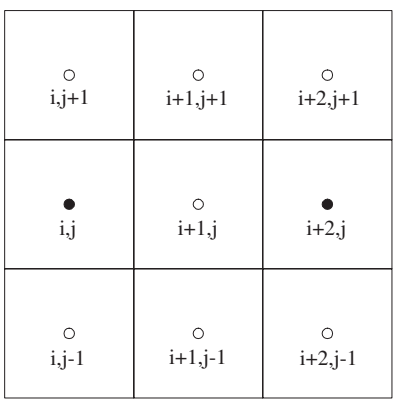


Figure 5. Two coarse-grid nodes (marked by (•) symbols) are surrounded by a number of neighbouring fine-grid nodes that are marked by (○) symbols. In order to determine the east coefficient between the coarse-grid node located at (i, j) and that located at $(i+2, j)$, we take a weighted average of the harmonic means of the three rows of fine-grid east–west coefficients that lie between points with first indices i and $i+2$.

computational costs, by deploying the fast Fourier transform (FFT). In this case the discrete elasticity equation (3) can be written in the x-convolution form

$$\sum_{l=1}^{n}\sum_{s=1}^{m} C_{k-s;rl}w_{sl} = p_{kr}$$

The FFT can be used to evaluate this convolution sum efficiently by exploiting the FFT convolution theorem $\mathscr{F}^{-1}(F_q G_q)_k = \sum_{s=0}^{m-1} f_{k-s}g_s$, where $F_q = \mathscr{F}(f_k)$ and $G_q = \mathscr{F}(g_k)$ denote the FFTs of f_k and g_k, and the symbol $\mathscr{F}^{-1}$ is used to denote the inverse FFT. However, exploiting these symmetries in evaluating the matrix vector products Cw restricts one to vector updates (such as those performed in damped Jacobi iteration) rather than being able to incorporate the influence of point updates immediately (as is done in Gauss–Seidel iteration) (see References [12, 14, 15]). Because of the expense of computing the matrix vector products, the use of conventional smoothers is only practicable for those that perform vector updates. In this section we introduce a novel Gauss–Seidel scheme which uses a localized approximation to the Jacobian. The localized Gauss–Seidel scheme is adapted to deal with the mesh-induced anisotropy associated with large aspect ratio rectangular elements.

3.5.1. Gauss–Seidel iteration with a localized Jacobian. We describe a localized approximation to the Jacobian that can be used along with Gauss–Seidel iteration to achieve efficient damping of the high-frequency components. Because the localized Jacobian is constructed from neighbouring components of the Jacobian, it faithfully approximates the high-frequency components of the spectrum of J. The construction of the localized Jacobian is depicted in Figure 6(a). Consider the receiving point (i, j) at the centre of a 5×5 patch of elements. The crucial

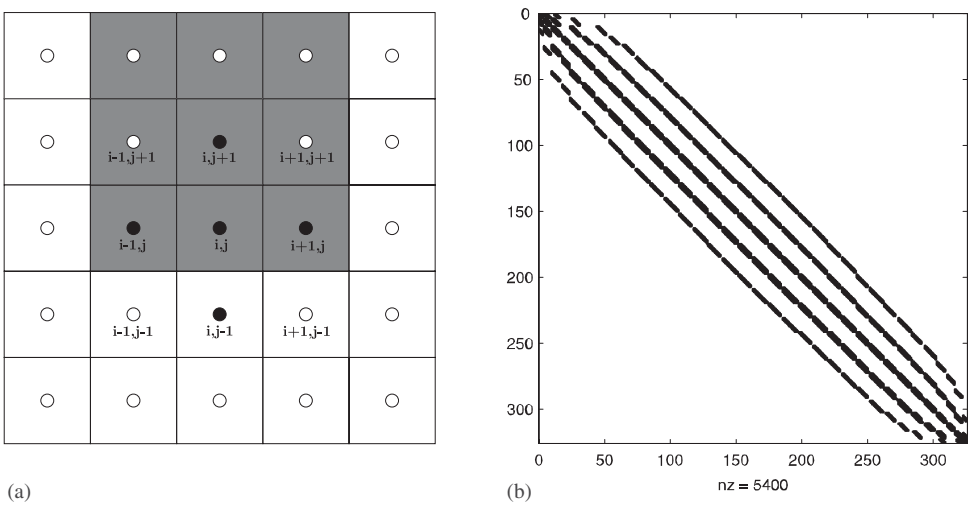


Figure 6. (a) Geometric view of the procedure used to construct the elements of the localized Jacobian associated with node (i, j); and (b) the sparsity pattern for the localized Jacobian obtained by discarding all but the nearest-neighbour elasticity influences.

component in this process is the approximation of the matrix product AC. We observe that, due to the local support of the matrix A, the influence on the receiving point (i, j) only involves the pressures at the neighbouring points $(i, j+1)$, $(i, j-1)$, $(i-1, j)$, $(i+1, j)$ as well as the pressure at the point (i, j). These points are indicated by the solid circles in Figure 6(a). Since C is fully populated, the pressures at each of these neighbouring points depends on the widths throughout the current fracture surface. However, because the coefficients of the C matrix decay at a rate of $O(1/r^3)$ as the distance r from the sending element increases, we adopt an approximation that only includes the elastic influences due to the widths in each of the neighbouring elements to pressure points marked by the solid circles (•). In Figure 6(a) we, depict by grey shading the span of influence of the set of 3×3 nearest neighbour width elements that are used to approximate the pressure at the point $(i, j+1)$, while the C matrix influences due to the widths at all remaining elements within the fracture are ignored. Similar patches of 3×3 elements are constructed around each of the five elements of the difference stencil for A that are denoted by the solid circles. The matrix vector product can be expressed in the form

$$(AC)_{MN} = \sum_{K \in \mathcal{N}_M^5} A_{MK} C_{KN} \approx \begin{cases} \displaystyle\sum_{K \in \mathcal{N}_M^5} A_{MK} C_{KN} & \text{when } N \in \mathcal{N}_K^9 \\ 0 & \text{when } N \notin \mathcal{N}_K^9 \end{cases}$$

Here we have used capital indices K, M, and N to denote a global scheme to number the elements as opposed to the co-ordinate-based scheme depicted in Figure 6(a). The receiving element (i, j) in this case corresponds to element M. Here $\mathcal{N}_M^5$ represents the set of five neighbouring elements associated with the difference stencil of A centred at element M, while $\mathcal{N}_K^9$ refers to the Kth stencil element as well as its eight neighbours—such as those shaded elements in Figure 6(a) that are associated with the stencil element $(i, j+1)$. Following the procedure described above, the fully populated Jacobian is approximated by a sparse localized Jacobian that has at most 21 non-zero elements in each row. In Figure 6(b), the non-zero elements in the localized Jacobian are plotted. We observe that there are 5400 non-zero elements as opposed to the $325^2 = 105\,625$ elements that are associated with the fully populated Jacobian for this case.

In the MG algorithm we are interested in a smoother that focuses on damping the high-frequency components, therefore this local approximation to the Jacobian is appropriate since it provides an accurate representation of the high-frequency local interactions while it avoids the computational burden associated with matrix vector products involving the fully populated Jacobian. The low-frequency components are dealt with by solving the coarsest grid problem with the coarse-grid Jacobian that includes both long and short-range interactions associated with the coarse grid level. In Figure 7, we compare the damping characteristics of the localized Gauss–Seidel scheme (i.e. in which the full Jacobian has been replaced by the localized Jacobian) with those of the full Gauss–Seidel scheme and a two-step damped Jacobi scheme. We observe that the Gauss–Seidel scheme utilizing the localized Jacobian enjoys superior damping properties for the high-frequency components.

3.5.2. Line iteration for large aspect ratio elements. For the efficient numerical modelling of finger-like fracture geometries within layers, it is desirable to make use of rectangular elements in which the dimension parallel to the layering is larger than the dimension perpendicular to

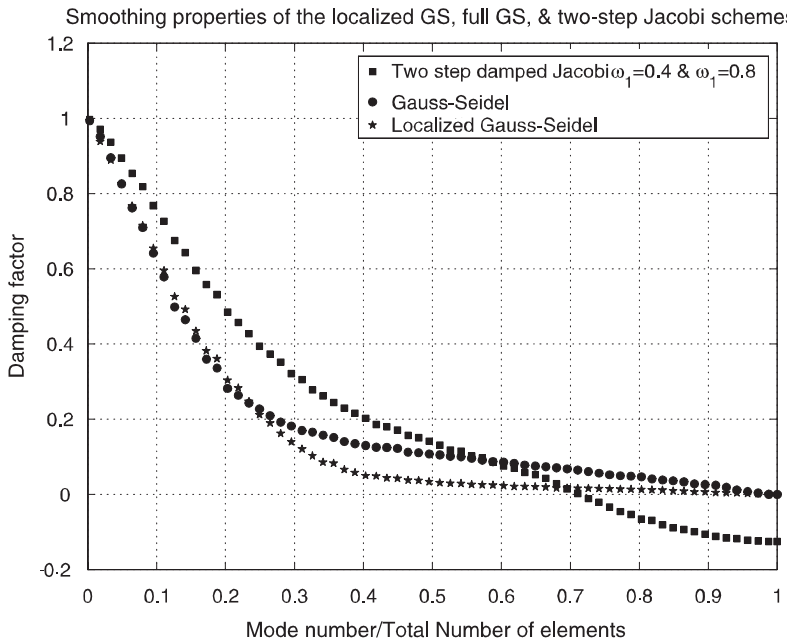


Figure 7. A comparison of the smoothing characteristics of the localized Gauss–Seidel scheme, the full Gauss–Seidel scheme, and the two-step damped Jacobi scheme with damping parameters $\omega_1 = 0.4$ and $\omega_2 = 0.8$.

the layering. Thus for horizontal layering it is not uncommon to require elements in which the so-called element aspect ratio $\Delta x/\Delta y$ can be as large as 5:1. In this case the discrete equations have anisotropic coefficients that result from the large aspect ratio of the elements. In particular, as can be seen from (4), elements with aspect ratios of 5 will have associated coefficients A_n and A_s that are 25 times larger than the corresponding coefficients A_e and A_w. This grid-induced anisotropy results in a discrete system of equations that is much stiffer in the north–south direction than in the east–west direction. As a result, due care is required so that the high-frequency modes in the stiffer direction are damped appropriately.

Similar problems with mesh anisotropy have been reported in the MG solution of elliptic and hyperbolic problems (see for example References [16–20]). Two strategies have been found to be successful in dealing with anisotropic coefficients. These include directional coarsening (semi-coarsening in our case) in which the grid is coarsened preferentially in the direction normal to the direction of maximal grid stretching and employing so-called line or block smoothers that are implicit in the direction normal to the direction of maximal stretching. The first strategy of directional coarsening has the disadvantage of not fully exploiting the reduction in complexity introduced by grid coarsening. Indeed, the semi-coarsening approach appropriate for the rectangular grid discretizations described in this paper would result in coarser meshes that have only half as many degrees of freedom as the next finest mesh, while full coarsening would result in a four-fold reduction in the number of degrees of freedom. The second approach, involving line implicit smoothers, are in fact not useful if we are making use

of the localized Jacobian since it does not accurately reproduce the low-frequency modes in the problem. Therefore, a line inversion in the direction normal to the grid stretching involving a tridiagonal system introduces undesirable spurious corrections to the low-frequency error components, which degrades the performance of the MG algorithm.

In order to preferentially reduce the high-frequency modes associated with the stiffer directions without introducing spurious low-frequency errors, we perform more iterations in the stiffer direction than in the direction of the grid stretching. For each line of vertical elements we perform a number of Gauss–Seidel sweeps before proceeding to the next vertical line of elements. The number of vertical sweeps for each line is taken to be equal to the aspect ratio $\Delta x/\Delta y$.

4. NUMERICAL RESULTS

In this section, we demonstrate the performance of the MG preconditioner for a number of different test problems. All the CPU times provided in this section were measured on a 2 GHz Pentium IV computer under Windows 2000 Professional using the version 6.1 Compaq Visual Fortran Compiler.

4.1. A radial fracture

The first problem comprises a hydraulic fracture injected with a viscous fluid having a viscosity $\mu = 0.2\,\text{Pa s}$ that is injected at a constant rate $Q = 2.5 \times 10^{-2}\,\text{m}^3/\text{s}$ into a homogeneous elastic medium having a Young's modulus $E = 5\,\text{GPa}$, a Poisson's ratio of $v = 0.2$, in which the far-field minimum principal stress $\sigma_c(x, y) = 6.6\,\text{MPa}$ is constant, and for which the toughness is assumed to be zero. The solution involves a growing fracture that is radially symmetric (see Reference [21]). In the first test we compare the performance of the MG preconditioner involving a single $V(0, 2)$ cycle with that involving a single $V(2, 1)$ cycle for a problem in which square elements are used. In the second test we demonstrate the efficacy of vertical line iterations by assessing the performance of the MG preconditioner for two problems in which the aspect ratios are: $\Delta x/\Delta y = 1$ and 3.91. Since the $V(0, 2)$ MG preconditioner proved to be more effective in the previous test we restrict this test to $V(0, 2)$ MG preconditioners.

4.1.1. Single $V(0, 2)$ cycle preconditioner versus single $V(2, 1)$ cycle preconditioner. In the runs performed in this section we use square elements $\Delta x = \Delta y = 5m$ to model the radial fracture described above. In Figure 8(a) we plot as a function of N the cumulative solution times using the BiCGSTAB (- - -), the BiCGSTABMG preconditioned by a single $V(0, 2)$ cycle (solid), and the BiCGSTABMG preconditioned by a single $V(2, 1)$ cycle (. . .). The increasing number of active elements is obtained by considering the solution times associated with the increasing fracture footprints measured during the evolution of the fracture. For any particular N, the data represent the cumulative CPU times of many consecutive solutions of the linear equations (5) associated with the implementation of Newton's method. In order to compare the solution times of the three algorithms we have solved the same linear system with BiCGSTAB followed by the two BiCGSTABMG algorithms and measured each of the solution times separately. The total number of BiCGSTAB iterations involved with this run over the entire evolution of the fracture were about 3.8×10^5 whereas the BiCSTABMG algorithms involved roughly 2×10^4 iterations. We observe that the solution times with the MG preconditioners (0.27 h

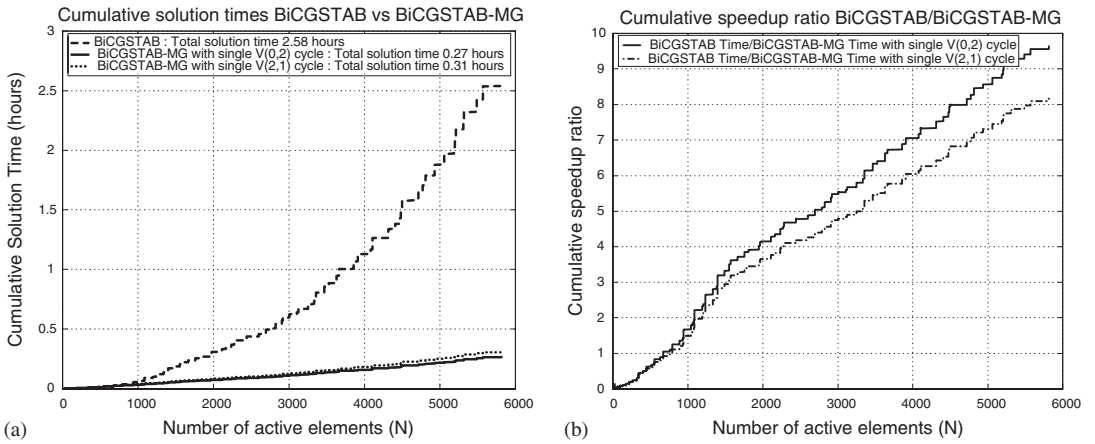


Figure 8. Performance results obtained for a fluid-driven radial fracture evolving in a homogeneous elastic medium under constant minimum principal stress conditions σ_c and using square elements $\Delta x = \Delta y = 5m$: (a) the cumulative solution times using the BiCGSTAB solver (- - -), the BiCGSTABMG solver with preconditioner involving a single $V(0,2)$ (solid), and the BiCGSTABMG solver with preconditioner involving a single $V(2,1)$ cycle (. . .); and (b) the cumulative speedup ratio as measured by $T_{\text{BiCGSTAB}}/T_{\text{BiCGSTABMG}}$ for the $V(0,2)$ and the $V(2,1)$ cycle preconditioners.

for $V(0,2)$ and 0.31 h for $V(2,1)$ cycles) are considerably shorter than the solution time of 2.58 h required by the BiCGSTAB algorithm with diagonal preconditioning. We observe that the $V(0,2)$ preconditioner is slightly more efficient than the $V(2,1)$ preconditioner. In Figure 8(b) we plot the cumulative speedup ratios for the two MG preconditioners against the number of active elements N as the fracture evolves. We observe that the point at which the MG solvers will overtake the BiCGSTAB algorithm in this case occurs when approximately 800 elements are active. In fact, the break-even point occurs at 350 elements but it takes some time for the MG algorithms to make up the deficit. This indicates that a simple switch between the diagonal preconditioner at the break-even point would lead to a more efficient algorithm. We also observe that the cumulative speedup achieved by the MG solvers reaches a factor of nine by the end of the simulation.

In Figure 9, we provide a $\log_{10}$–$\log_{10}$ plot of the average number of iterations required by the three algorithms against the number of active elements N. We observe that for the MG algorithms the number of iterations is roughly 10 for each solution of the linear system (5) over the history of the evolution of the fracture. The iteration count for the BiCGSTAB algorithm with diagonal preconditioning increases at approximately $N^{0.72}$ as can be seen from the slope of the linear regression line shown in the plot. While the number of iterations for the MG $V(2,1)$ cycle preconditioner is very similar if not slightly less than that of the $V(0,2)$ preconditioner, this has not been matched in the time trial performance shown in Figure 8(b) due to the additional computational cost of the MG $V(2,1)$ cycle compared to the $V(0,2)$ cycle.

If the FFT algorithm is used to evaluate the convolution products Cw, then for large N the relative cost of performing a $V(0,2)$ BiCGSTABMG iteration is about twice that required to

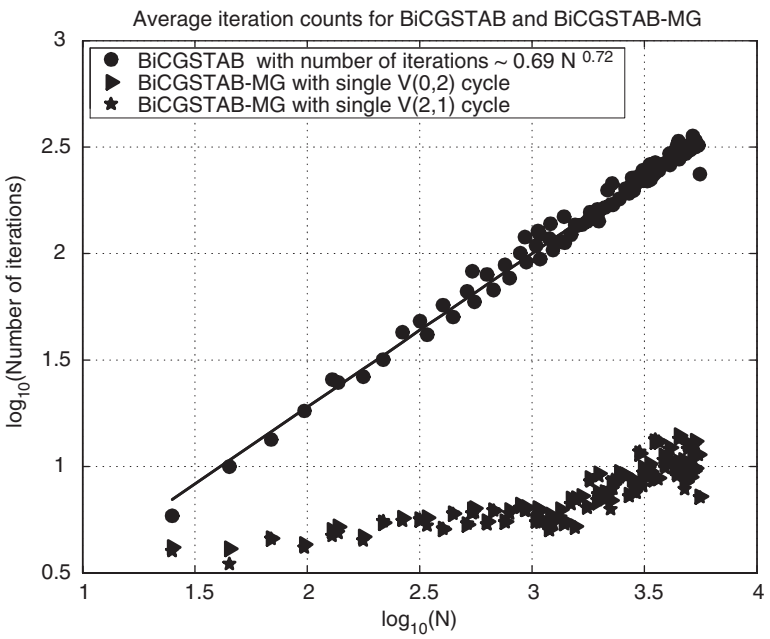


Figure 9. A $\log_{10}$–$\log_{10}$ plot of the average number of iterations against the number of active elements for the BiCGSTAB solver with diagonal preconditioning (•), the BiCGSTAB solver preconditioned by a single leg $V(0,2)$ (▶), the BiCGSTAB solver preconditioned by a single MG $V(2,1)$ (⋆). These results were obtained for a fluid-driven radial fracture evolving in a homogeneous elastic medium under constant minimum principal stress conditions σ_c and using square elements $\Delta x = \Delta y = 5m$.

perform a BiCGSTAB iteration. This takes into account the overhead involved in setting up the required coarse-grid operators, as well as the cost of the interpolation and restriction operators. If the convolution products are evaluated directly, there is asymptotically no difference between the cost of performing a single $V(0,2)$ BiCGSTABMG and a BiCGSTAB iteration because the convolution cost dominates the whole iteration process. The break-even point for the MG algorithm occurs at roughly 400 active elements beyond which the performance increases with N. Speedup factors of between 10 and 18 are achieved for problems involving more than 3000 active elements.

In Table I we provide the average iteration counts and CPU times that are required to solve the linear system (5) for an evolving radial fracture using BiCGSTAB with Diagonal, $V(0,2)$ and $V(2,1)$ cycle preconditioning. Only a few sample data points that have been displayed in the preceding graphs are provided. We observe that the MG iterations asymptote to approximately 12 iterations for $N>3000$. We notice that, due to the convolution product Cw, the CPU time continues to increase superlinearly with N, even though the number of iterations remains constant. To illustrate this, consider a fracture that fills a rectangular window comprising m horizontal and n vertical elements with $N = m \times n$. For an elastic medium that comprises horizontal layers, the horizontal translational invariance of the associated elastic influence matrix C can be exploited to perform Cw products very efficiently using the FFT.

Table I. Comparative average iteration counts and CPU times for linear solves for a radial fracture using BiCGSTAB with Diagonal, $V(0,2)$, and $V(2,1)$ cycle preconditioning.

	Iterations			CPU Time (s)		
N	Diagonal	$V(0,2)$	$V(2,1)$	Diagonal	$V(0,2)$	$V(2,1)$
97	18	4	4	0.01	0.06	0.06
513	66	5	5	0.17	0.10	0.12
1041	109	6	5	0.90	0.16	0.18
2109	164	8	7	2.42	0.32	0.37
3005	225	12	12	4.57	0.65	0.79
4081	294	11	11	12.11	0.92	1.12
5061	306	9	9	14.66	0.93	1.15

The asymptotic operation count for the FFT algorithm to perform a Cw matrix vector product for such a layered medium is given by $OP_{\mathrm{FFT}} = 2n^2m + 5 \times 4nm \log_2 2m = O(nN)$ which explains the superlinear increase in CPU times with increasing N.

4.1.2. Performance of the vertical line iteration strategy for various aspect ratios. In this subsection, we demonstrate the efficacy of vertical line iterations by comparing the performance of the $V(0,2)$ preconditioner using vertical line iterations for a problem having an aspect ratio $\Delta x/\Delta y = 3.91$ with the standard $V(0,2)$ preconditioner for a problem having an aspect ratio $\Delta x/\Delta y = 1$. The 3.91 aspect ratio model will require roughly one quarter the number of elements than that required by the square element model. It is useful to compare the performance of the MG algorithm for two different aspect ratio models that are attempting to approximate the same fracture footprint that occur at the same elapsed time t since the initiation of the fracture. Since the single $V(0,2)$ preconditioner proved to be more effective in the previous test, we restrict this test to MG preconditioners involving a single $V(0,2)$ cycle.

In Figure 10(a) we plot the cumulative run times for the BiCGSTAB algorithm and the BiCGSTABMG algorithm for a radial hydraulic fracture modelled by square elements and rectangular elements with an aspect ratio of 3.91. For the model with square elements, standard localized Gauss–Seidel iterations are performed, while for the $V(0,2)$ model with an aspect ratio of 3.91 the vertical line iterations described in Section 3.5.2 are used. In spite of the reduced numbers of active elements for the larger aspect ratio problem it is interesting to note that the BiCGSTAB solver requires roughly the same solution time (2.58 versus 2.68 h) for the two different aspect ratio problems. This is due to the degradation in the conditioning of the Jacobian for the problem with large aspect ratio elements. We observe that with the use of the vertical line iterations, the performance of the $V(0,2)$ preconditioner is maintained even though the aspect ratio changes. If vertical line iterations are not used for problems with large aspect ratio elements, then the performance of the $V(0,2)$ algorithm degrades (see the results presented in Table II). In Figure 10(b) the cumulative speedup ratio versus the number of active elements N is plotted for the $V(0,2)$ algorithm for the two different aspect ratio problems. For the larger aspect ratio problem there is a slight degradation in performance of the $V(0,2)$ but the speedup ratios are roughly similar for the same size of problem. This is

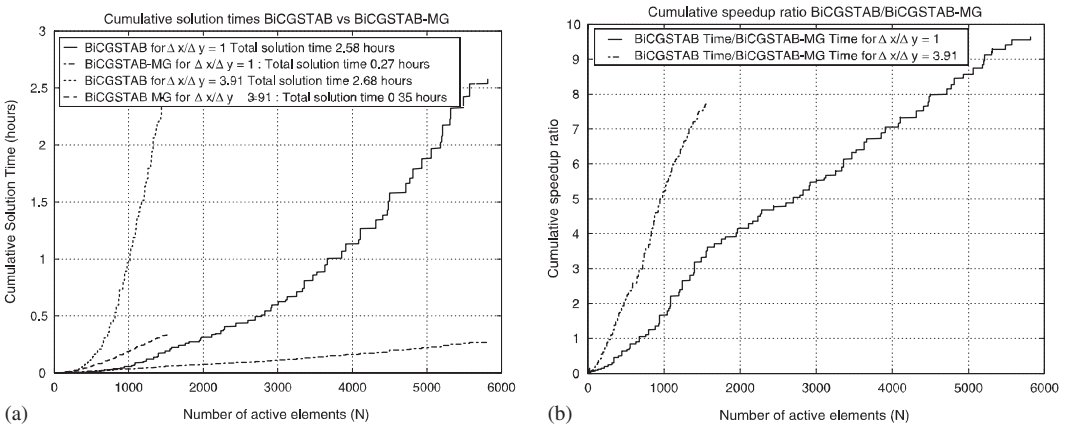


Figure 10. These results were obtained for a fluid-driven radial fracture evolving in a homogeneous elastic medium under constant minimum principal stress conditions σ_c and using rectangular elements for which $\Delta y = 5m$: (a) the cumulative solution times using the BiCGSTAB solver with 1×1 elements (solid), the BiCGSTABMG solver with 1×1 elements (— -), the BiCGSTAB solver with 3.91×1 elements ($\cdots$), and BiCGSTABMG solver with 3.91×1 elements (- - -) with vertical iterations; and (b) the cumulative speedup ratio as measured by $T_{\mathrm{BiCGSTAB}}/T_{\mathrm{BiCGSTABMG}}$ for the $V(0,2)$ algorithm using different aspect ratio elements 1×1 and 3.91×1. Specialized vertical line iterations are performed for the large aspect ratio elements.

Table II. Comparative average iteration counts and CPU times for linear solves for a radial fracture using BiCGSTAB iteration with diagonal and FMG preconditioning. The results for two models with different aspect ratios 1×1 and 3.91×1 are compared. The degradation in FMG performance can be seen when vertical line iteration is not used.

	Diagonal 1×1		$V(0,2)$ 1×1			Diagonal 4×1		$V(0,2)$ 4×1		$V(0,2)$ 4×1+Vert Its	
N	Iters	CPU (s)	Iters	CPU (s)	N	Iters	CPU (s)	Iters	CPU (s)	Iters	CPU (s)
97	18	0.01	4	0.06	25	12	< 0.01	6	0.04	5	0.04
513	66	0.17	5	0.10	127	40	0.04	15	0.10	8	0.07
1041	109	0.90	6	0.16	247	84	0.16	18	0.14	9	0.10
2109	164	2.42	8	0.32	503	144	0.58	21	0.23	11	0.15
3005	225	4.57	12	0.65	757	195	1.72	23	0.39	12	0.24
4081	294	12.11	11	0.92	1009	241	2.83	25	0.52	12	0.31
5061	306	14.66	9	0.93	1247	281	4.07	25	0.61	13	0.39

because less elements are active in the high aspect ratio model which reduces the potential for speedup due to the lower cost of performing the convolution products Cw.

A $\log_{10}$–$\log_{10}$ plot of the average number of required iterations against the number of active elements N is very similar to the results presented in Figure 9. The number of BiCGSTAB iterations exhibits a similar power relationship with N in which the exponent is roughly the same for the square and large aspect ratio cases, while the constant factor is quite different for

each. For the higher aspect ratio problem the BiCGSTAB algorithm with $V(0, 2)$ preconditioning using vertical line iterations to compensate for grid-induced anisotropy performs similarly to the $V(0, 2)$ preconditioner with standard Gauss–Seidel preconditioning for the problem in which the aspect ratio is 1. The break-even point for the MG algorithm occurs at roughly $N = 200$ active elements in the large aspect ratio case compared to 400 in the square element case and in both cases the performance increases with N. Speedup factors of between 10 and 12 are achieved for problems involving large aspect ratio active element numbers between 1000 and 1500, while speedup factors of between 10 and 18 are achieved for square active element numbers between 3000 and 5000.

In Table II we provide samples of the average iteration counts and CPU times that are required to solve the linear system (5) for an evolving radial fracture using BiCGSTAB with Diagonal and $V(0, 2)$ preconditioning for a model with square elements as well as for a problem in which the elements have an aspect ratio of 3.91. In this table, we have selected some sample data points that have been displayed in the preceding graphs as well as a corresponding sample of the performance for the $V(0, 2)$ algorithm if the vertical line iterations are not used (see columns 9 and 10). The degradation of the $V(0, 2)$ performance without vertical line iterations can be observed by comparing the results in columns 9 and 10 with those in columns 11 and 12. Since the number of elements for the high aspect ratio model should be roughly one quarter that of the square element model when representing the same fracture footprint, we have chosen to compare numbers for the two problems that represent roughly the same size radial fracture. By comparing the iteration counts in columns 4 and 11 we observe that the vertical line iterations allow the $V(0, 2)$ algorithm to maintain its performance even if the aspect ratio of the problem is increased. We observe that, although the $V(0, 2)$ iteration counts are roughly similar, the actual CPU times for the high aspect ratio $V(0, 2)$ algorithm are roughly one third those for the corresponding square element model. This occurs because the smaller number of active elements requires less convolution time. This reduction in CPU time with reduced numbers of active elements can also be observed by comparing the CPU times for the diagonal preconditioner in columns 3 and 8.

4.2. A fracture passing through a stress jump

The second problem comprises a hydraulic fracture injected with a viscous fluid having a viscosity $\mu = 0.2\,\text{Pa s}$ at a constant rate $Q = 2.5 \times 10^{-2}\,\text{m}^3/\text{s}$ from a single point source located at $(x, y) = (0, 177\,\frac{1}{2})$. The fracture propagates in a homogeneous elastic medium having a Young's modulus $E = 5\,\text{GPa}$, a Poisson's ratio of $\nu = 0.2$, and zero toughness. In this problem the far-field minimum principal stress is assumed to be piecewise constant with a stress jump across an interface located at $y = 155\,\text{m}$, defined by

$$\sigma_c(x, y) = \begin{cases} 6.6\,\text{MPa} & \text{if } y > 155\,\text{m} \\ 0\,\text{MPa} & \text{if } y < 155\,\text{m} \end{cases}$$

A snapshot of the crack opening displacement profile associated with this problem at time $t = 14355\,\text{s}$ is plotted in Figure 11. This extreme situation has been chosen as it presents some considerable challenges for the MG algorithm since there is a zone just above the interface at which there is a width constraint $w \geqslant w_c$ active. Near this zone there is a large difference between the widths in neighbouring elements. In particular, the width in the constrained zone is two orders of magnitude smaller than the maximum width in the unconstrained regions.

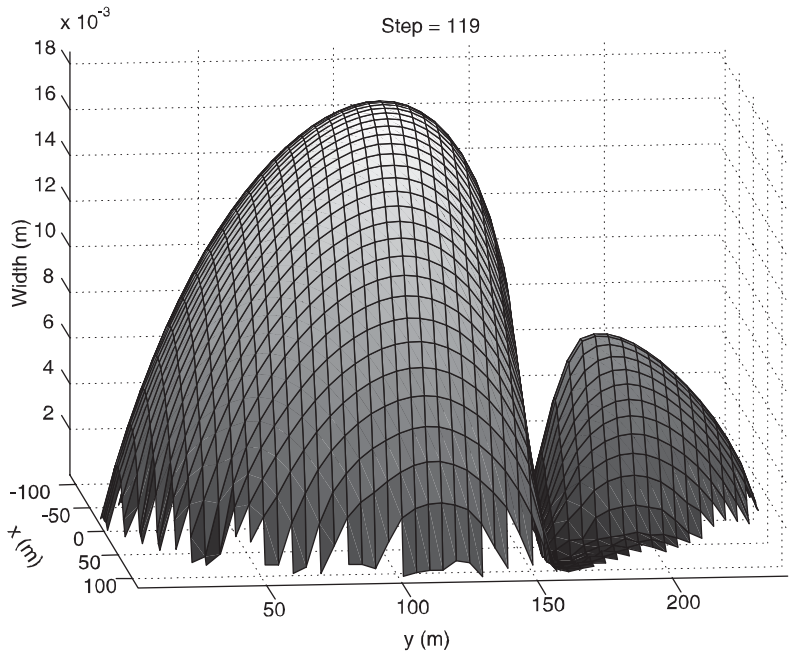


Figure 11. The crack opening displacement for a fluid-driven fracture propagating through an interface across which there is a jump in the minimum principal stress σ_c.

Because of the cubic dependence of the coefficients of the fluid conservation equation (2) on the width, this leads to rapid changes in these coefficients between neighbouring elements and is therefore a stringent test on the procedure used to define the coarse-grid A matrix coefficients described in Section 3.4. This problem also serves to test the interpolation scheme described in Section 3.3 for problems in which there is a mixture of constrained and unconstrained elements. We also observe that due to the difference in confinement in the two regions, the width in the region close to the source is approximately one third of the maximum width in the region below the interface. This is due to a 'herniation' process caused by the lower resistance to fracturing in the low confinement region $y<155\,\mathrm{m}$.

In Figure 12(a) we plot the cumulative CPU times spent by the BiCGSTAB algorithm compared to that of the BiCGSTABMG algorithm versus the number of active elements N in the fracture. The variation in the number of active elements is obtained by considering the solution times associated with the increasing fracture footprints measured over the evolution the fracture. For any particular N the data represent the cumulative CPU times of many consecutive solutions of the linear equations (5) associated with the implementation of Newton's method. In order to compare the solution times of the two algorithms, we have solved the same linear system with BiCGSTAB followed by BiCGSTABMG and measured each of the solution times separately. The total number of BiCGSTAB iterations were about 1.22×10^6 whereas the BiCSTABMG algorithm involved 9.5×10^4 iterations. In this case we have used the $V(0,2)$ version of the MG preconditioner. We observe that the $V(0,2)$ preconditioning

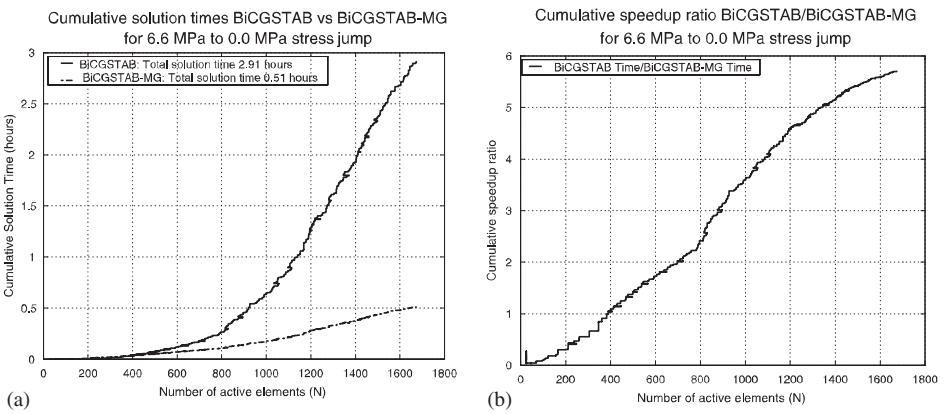


Figure 12. Results for a fracture growth problem in which the minimum principal stress σ_c jumps from 6.6 to 0 MPa across an interface located at $y = 157.5$ m: (a) cumulative CPU times spent on the solution process by the BiCGSTAB (solid) and the BiCGSTABMG algorithms (dashed); and (b) the cumulative speedup ratio as measured by $T_{\mathrm{BiCGSTAB}}/T_{\mathrm{BiCGSTABMG}}$.

leads to considerably shorter solution times in spite of the fact that there is a significant region within which there are active width constraints.

A detailed look at the BiCGSTAB curve plotted in Figure 12(a) shows horizontal ripples with small regions within which the function is not single valued. These are caused by a restart procedure associated with a poorly conditioned Jacobian, typically due to a trial time step Δt that is too large. In this case the algorithm to locate the free boundary associated with the fracture footprint abandons the current trial fracture footprint in favor of one with a smaller time step that involves fewer elements. The iteration and CPU results for these re-start solutions have been included in those presented below. In Figure 12(b) we plot the cumulative speedup ratio for the $V(0, 2)$ preconditioner. We observe that in spite of the extreme changes in the coefficients associated with the width constraint the $V(0, 2)$ algorithm still delivers more than five-fold speedup for the relatively moderate number of active elements $N = 1800$. In this example the plot of the average iteration counts shows similar trends to those shown in Figure 9 for the radial fracture example, but has been omitted for the sake of brevity.

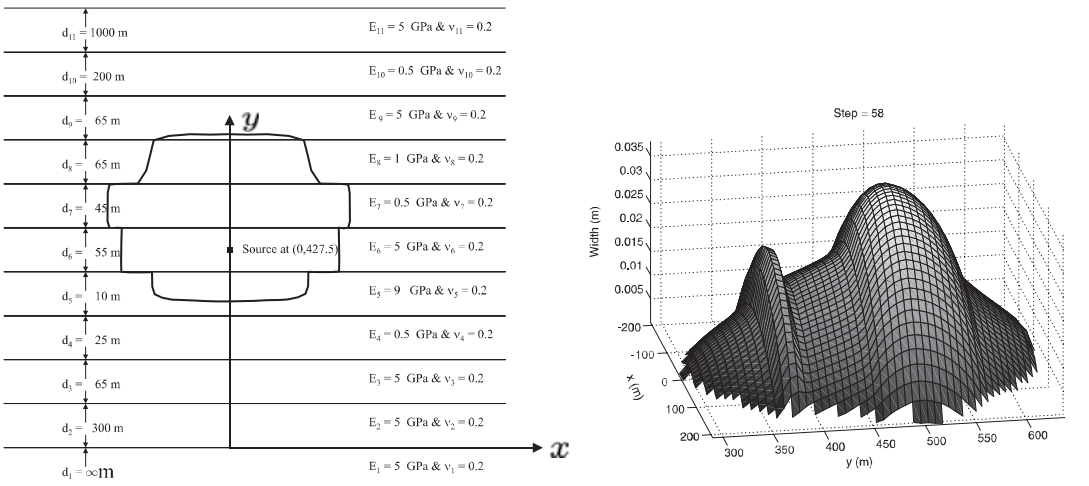
In Table III we provide a sample of iteration counts and CPU solution times for this stress jump problem. We observe that the iteration counts for the MG preconditioner remain relatively constant while those for the BiCGSTAB algorithm with diagonal preconditioning exhibit a power law increase in the number of iterations. The level of improvement in performance can also be observed by comparing the two columns containing the CPU times. The speedup factors for the same number of elements compare favourably with those of the previous examples.

4.3. A fracture evolving in a layered elastic medium with extreme modulus changes

In this section, we present results for a problem in which a fluid with a viscosity $\mu = 0.2$ Pa s is injected at a constant rate $Q = 2.5 \times 10^{-2}\,\mathrm{m}^3/\mathrm{s}$ to propagate a fracture in a layered elastic medium having layer thicknesses d_k, Young's moduli E_k, and Poisson's ratio's v_k as shown in Figure 13(a). A schematic snapshot of the fracture footprint is indicated in this figure. In this

Table III. Comparative average iteration counts and CPU times for Jacobian inversion for a fluid-driven fracture in a medium with a 6.6 to 0 MPa stress jump using BiCGSTAB with Diagonal and $V(0, 2)$ preconditioning.

	Diagonal		$V(0, 2)$ MG	
N	Iterations	CPU (s)	Iterations	CPU (s)
104	34	0.02	8	0.07
257	57	0.05	8	0.08
500	81	0.24	8	0.10
749	94	0.41	7	0.11
1001	120	1.05	8	0.16
1252	130	1.30	7	0.20
1595	148	1.92	8	0.25



(a) (b)

Figure 13. (a) Layer thicknesses d_k and layer moduli E_k and v_k for the test problem in which a fluid driven fracture evolves within a layered elastic medium. The fracture footprint is represented schematically in this figure; and (b) a snapshot of the crack opening displacement for a fluid driven fracture propagating in a layered elastic medium at time $t = 73126$ s.

problem the minimum principal stress $\sigma_c(x, y) = 6.6$ MPa is constant and fracture toughness is assumed to be zero. The fluid is assumed to emanate from a point source located at $(0, 427.5)$. In Figure 13(b) we plot the crack opening displacement for the fracture that has propagated in the layered elastic medium described above. The herniation effect is caused by the lower resistance of the moduli in layers 4 and 7 that straddle the source layer which are an order of magnitude softer than the modulus in the source layer.

In Figure 14(a) the cumulative solution times for the BiCGSTAB and BiCGSTAB algorithm with $V(0, 2)$ preconditioning are plotted against the number of active elements N. By comparing

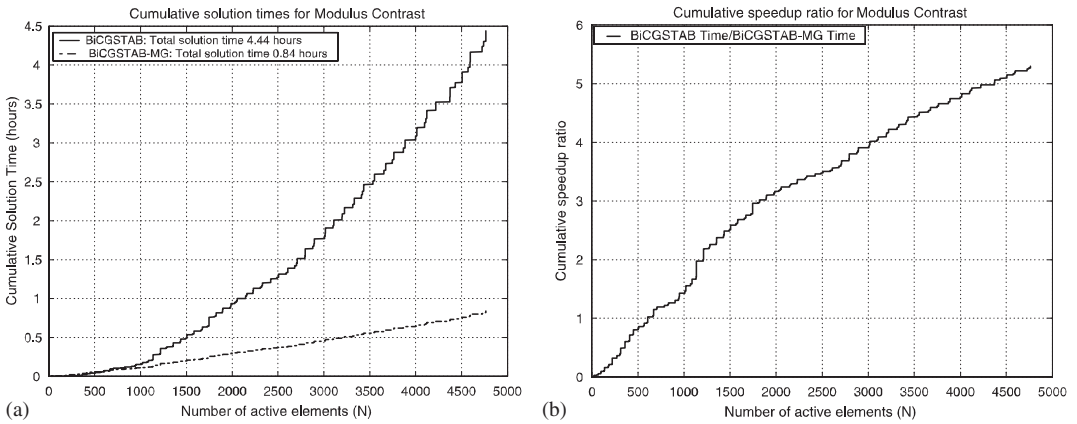


Figure 14. Simulation results for a fluid driven fracture evolving in a layered elastic medium with up to 10 fold changes in elastic moduli between the layers: (a) cumulative solution times for the BiCGSTAB algorithm with diagonal preconditioning compared to that of the BiCGSTAB algorithm with $V(0, 2)$ preconditioning; and (b) the cumulative speedup ratio as measured by $T_{\mathrm{BiCGSTAB}}/T_{\mathrm{BiCGSTABMG}}$ plotted as a function of the number of active elements N.

the results for the layered case in Figure 14(a) with those for the homogeneous case in 8(a) we observe that the contrast in material properties leads to a significant increase in the total solution time (4.44 versus 2.58 h). This difference can be ascribed partly to a difference in the conditioning of the linear systems (5) and partly to the increased number of Newton iterations that are required. A similar increase in the MG solution time can also be seen. However, the MG algorithm still leads to a five-fold reduction in the solution time. In Figure 14(b), we plot the cumulative speedup ratio for the BiCGSTAB and BiCGSTABMG algorithms. We observe that the MG scheme manages to achieve an accumulated speedup factor of 5. For this example, the plot of the iteration counts shows similar trends to that shown in Figure 9 for the radial fracture example and has been omitted for the sake of brevity.

In Table IV we provide sample iteration counts and solution CPU times for the diagonal and MG preconditioners. The iteration counts for both algorithms presented for this layer contrast case are noticeably higher than the corresponding results for the homogeneous case presented in Table I. We observe that the MG preconditioner leads to significant reductions in the number of iterations as well as concomitant reductions in the CPU times.

5. CONCLUSIONS

In this paper, we have presented a novel MG preconditioner for the numerical solution of the coupled elastic integral equation and the degenerate fluid-flow PDE that describe the evolution of hydraulically driven fractures. The solution of the coupled equations require the inversion of a Jacobian matrix that involves the product of the sparse fluid-flow matrix and the fully populated discrete Green's function matrix. The structure of the Jacobian for this coupled system requires a non-standard MG implementation.

Table IV. Comparative average iteration counts and CPU times for Jacobian inversion for a hydraulic fracture propagating in a layered elastic medium using BiCGSTAB with Diagonal and $V(0, 2)$ cycle preconditioning.

	Diagonal		$V(0, 2)$ MG	
N	Iterations	CPU (s)	Iterations	CPU (s)
101	35	0.01	6	0.08
273	70	0.27	9	0.13
501	96	0.27	11	0.20
786	138	0.59	13	0.29
1011	163	1.34	13	0.36
1506	214	2.40	17	0.59
2030	275	3.67	17	0.71
3008	309	8.23	20	1.28
3999	381	12.54	22	1.78
4507	457	16.75	25	2.19

If the fluid-driven fracture has pinch regions in which the width of the fracture can become vanishingly small, then the cubic dependence of the PDE coefficients on the fracture width results in Jacobian coefficients that can differ by three orders of magnitude between neighbouring elements. A scheme proposed by Alcouffe *et al.* [13] that uses harmonic averaging of the coefficients is shown to be effective. When pinch regions require a width constraint it is necessary to use a mixed variable formulation and interpolation based on the evaluation of the local pressure field. The performance of the harmonic average and mixed variable approach has been demonstrated for a hydraulic fracture straddling a stress jump in which a significant pinch regions are active.

Defining the coarse-grid Green's function matrix for a layered elastic material presents another significant challenge. We have introduced a novel dual mesh approach by which the required hierarchy of approximate coarse-grid Green's function matrices can be evaluated by superposition. This construction relies on the pre-calculation of a set of influence coefficients for elements that are half the dimensions of those on the finest mesh. The efficacy of the dual mesh approach has been demonstrated by numerical examples in which there is a large contrast in the elastic properties between neighbouring elements.

Because of the fully populated Green's function matrix C, for problems with large numbers of elements, it is necessary to use the FFT to exploit the horizontal shift invariance of the problem in order that these matrix vector products do not become prohibitively expensive. Therefore the use of conventional smoothers is only practicable for those that perform vector updates such as damped Jacobi iteration. To alleviate computational costs of multiplying by the fully populated C matrix we have introduced a novel Gauss–Seidel smoother which makes use of a localized Jacobian that is constructed to exploit the rapid decay of the elements of the Green's function matrix with distance from the source element. Because the localized Jacobian is constructed from the high-frequency nearest-neighbour influences it forms a very effective smoother for the high-frequency components of the problem. The localized Gauss–Seidel algorithm is adapted to define line iteration strategies necessary to mitigate the mesh-induced anisotropy introduced

by high aspect ratio elements. The performance of the localized Gauss–Seidel smoothers has been demonstrated in numerical examples for square as well as large aspect ratio elements.

In the numerical experiments presented in this paper the performance of the MG preconditioner has been assessed for an evolving fracture in which the number of active elements increases as the fracture evolves. The MG preconditioner typically becomes more competitive as the number of elements in the problem increases. For problems in which there are 1000 active elements, the MG algorithm leads to a five-fold speed-up in solution times, while for problems with as many as 3000 active elements the speed-up factor can be as high as 10. The MG preconditioner approximately doubles the cost of performing a BiCGSTAB iteration, but the reduced iteration count more than compensates for this overhead as can be seen from the overall reduction in solution times.

APPENDIX A: THE CONDITIONING OF THE COUPLED OPERATOR

In order to demonstrate the degradation in the conditioning of the coupled operator AC we consider a simple one-dimensional model problem comprising a crack propagating in a situation of plane strain. In this case the governing integral equation is given by

$$Cw = -\frac{E'}{4\pi}\int_{-l(t)}^{l(t)} \frac{w(s,t)}{(s-x)^2}\,\mathrm{d}s = p(x,t) - \sigma_c(x)$$

Assuming piecewise constant basis functions and collocation at element centres, we obtain the following discrete form of the integral equation:

$$\sum_{n=1}^{N} C_{mn} w_n = p_m - (\sigma_c)_m, \quad C_{mn} = -\frac{E'}{4\pi\Delta x}\left[\frac{1}{(m-n)^2 - \frac{1}{4}}\right]$$

Because of the convolution form of these discrete equations, the discrete Fourier transform of the kernel function yields the eigenvalues of the discrete operator:

$$\hat{C}_k = -\frac{E'}{4\pi\Delta x}\sum_{k=-\infty}^{\infty} \frac{\mathrm{e}^{\mathrm{i}km\Delta x}}{m^2 - \frac{1}{4}} = -\frac{E'}{4\pi\Delta x}\left[-2\pi\,\sin\left(\frac{|k|\Delta x}{2}\right)\right]$$

The applicable Reynold's equation, in the absence of leakoff, is given by

$$\frac{\partial w}{\partial t} = \frac{\partial}{\partial x}\left(D(w)\frac{\partial p}{\partial x}\right) + \delta(x)Q = Ap + \delta(x)Q$$

If for the purposes of this analysis, we assume that the width is slowly varying and frozen to some nominal value $\bar{w}$, the discrete form of the A operator and its eigenvalues are of the form

$$Ap_n = \frac{\bar{D}}{\Delta x^2}(p_{n+1} - 2p_n + p_{n-1}), \qquad \hat{A}_k = -\frac{4\bar{D}}{\Delta x^2}\,\sin^2\left(\frac{k\Delta x}{2}\right)$$

Thus the eigenvalues λ_k of the combined operator AC are given by

$$\lambda_k = \hat{A}_k \hat{C}_k = -\frac{2E'\bar{D}}{\Delta x^3} \sin^3\left(\frac{|k|\,\Delta x}{2}\right)$$

From this analysis and considering typical values for E', $\bar{D}$, and Δx, it is evident that the coupled equations are extremely stiff. Thus explicit time stepping will become prohibitively expensive. For example using Euler's method for time stepping we would expect a time step restriction of the form:

$$\Delta t < \frac{\Delta x^3}{E'\bar{D}}$$

This small step-size is exacerbated by the fact that each time-step will require the multiplication by the matrix C, which is fully populated. For planar fractures, unless the shift invariance of the fracture problem parallel to the layers is exploited to reduce the computational costs from $O(N^2)$ the time stepping will become prohibitively expensive. For this reason L-stable time integration methods such as the backward Euler scheme are required. From the expression for λ_k it can be seen that the asymptotic condition number of the coupling matrix AC that essentially needs to be inverted at each time step for these implicit methods degrades as $O(N^3)$.

ACKNOWLEDGEMENTS

The authors would like to thank Schlumberger for permission to publish this paper. The first author would like to acknowledge the support of NSERC for his research program.

REFERENCES

1. Rubin AM. Propagation of magma-filled cracks. *Annual Review of Earth and Planetary Sciences* 1995; **23**:287–336.
2. Sibson RH, Moore RM, Rankin AH. Seismic pumping—a hydrothermal transport mechanism. *Journal of the Geological Society*, *London* 1975; **131**:653–659.
3. Peirce AP, Siebrits E. Uniform asymptotic approximations for accurate modelling of cracks in layered elastic media. *International Journal of Fracture* 2001; **110**:205–239.
4. Peirce AP, Siebrits E. The scaled flexibility matrix method for the efficient solution of boundary value problems in 2D and 3D layered elastic media. *Computer Methods in Applied Mechanics and Engineering* 2001; **190**:5935–5956.
5. Clifton RJ, Abou-Sayed AS. On the computation of the three-dimensional geometry of hydraulic fractures. *SPE 7993*, *Presented at the SPE/DOE Symposium of Low-Permeability Gas Reservoirs*, Denver, CO, 1979.
6. Clifton JR. Three-dimensional hydraulic fracturing. *SPE Monograph*, 1989.
7. Dendy JE. Black box multigrid. *Journal of Computational Physics* 1982; **48**:366–386.
8. Moulton JD, Dendy JE, Hyman JM. The black box multigrid numerical homogenization algorithm. *Journal of Computational Physics* 1998; **142**:80–108.
9. Chan TF, Wan WL. Robust multigrid methods for nonsmooth coefficient elliptic linear systems. *Journal of Computational and Applied Mathematics* 2000; **123**(1–2):323–352.
10. van der Vorst H. BiCGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 1992; **13**:631–644.
11. Barrett R, Berry M, Chan TF, Demmel J, Donato J, Dongara 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, PA, 1994.
12. Briggs W, Henson VE, McCormick SF. *A Multigrid Tutorial* (2nd edn). SIAM: Philadelphia, PA, 2000.

13. Alcouffe RE, Brandt A, Dendy JE, Painter JW. The multi-grid method for the diffusion equation with strongly discontinuous coefficients. *SIAM Journal on Scientific and Statistical Computing* 1981; **2**(4).
14. Axelsson O. *Iterative Solution Methods*. Cambridge University Press: Cambridge, 1996.
15. Golub GH, Van Loan CF. *Matrix Computations* (2nd edn). The Johns Hopkins University Press: Baltimore, MD, 1989.
16. Brandt A. Multigrid techniques with applications to fluid dynamics. *VKI Lecture Series*, 1984; 1–176.
17. Mulder WA. A new multigrid approach to convection problems. *Journal of Computational Physics* 1989; **83**:303–323.
18. Mulder WA. An investigation of cell-centered and cell-vertex multigrid schemes for the Navier–Stokes equations. *A.I.A.A*, *Paper 89-053*, 1989.
19. Schaffer S. A semicoarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients. *SIAM Journal on Scientific Computing* 1998; **20**(1):228–242.
20. Mavriplis DJ. Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. *Journal of Computational Physics* 1998; **145**:141–165.
21. Savitski A, Detournay E. Propagation of a penny-shaped fluid-driven fracture in an impermeable rock: asymptotic solutions. *International Journal of Solids and Structures* 2002; **39**(26):6311–6337.

