

Efficient Solvers for Incompressible Fluid Flows in Geosciences

A. Amirbekyan ^{*} L. Gross [†]

June 2008

Abstract

Saddle point problems involving large systems of linear equations arise in a wide variety of applications in computational science and engineering. A variety of solvers have been developed for these type of problems typically with specific applications in mind. In this paper we will focus on saddle point problems as they arise from incompressible fluid flow problems in applications in geosciences. They are characterized through a spatially variable viscosity when modeling temperature dependencies (e.g. in Earth mantel convection models) or moving material interfaces (e.g. in subduction zones simulation and numerical volcano models). In this paper we will give an overview on some of the iterative techniques that can be used and discuss suitable preconditioning techniques. We will discuss the implementation of the schemes using the python module Escript and compare the efficiency of these schemes when applied to convection models on a parallel computer.

Contents

1	Introduction	2
2	Uzawa Solver	3
3	Coupled Solver	4

^{*}Earth System Sciences Computational Center, University of Queensland, AUSTRALIA.
<mailto:artak@uq.edu.au>

[†]Earth System Sciences Computational Center, University of Queensland, AUSTRALIA.
<mailto:l.gross@uq.edu.au>

4	Preconditioner for Schur Complement	5
5	Implementation	6
6	Experiments	8
7	Conclusion	10

1 Introduction

Finite-element and finite-difference discretizations of the Navier-Stokes equations for incompressible flow lead to equations of saddle point type, which can be formulated as a solution of the following operator problem for $u \in V$ and $p \in Q$ with suitable Hilbert spaces V and Q :

$$\begin{bmatrix} A & B \\ B^* & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \quad (1)$$

where A is coercive, self-adjoint linear operator in V , B is a linear operator from Q into V and B^* is the adjoint operator of B . f and g are given elements from V and Q respectively. In most case the equation (1) is given in the form

$$a(v, u) + b(v, p) = \langle f, v \rangle, \quad (2)$$

$$b(u, q) = \langle g, q \rangle. \quad (3)$$

for all $v \in V$ and $q \in Q$. $\langle \cdot, \cdot \rangle$ denotes the scalar product in V and Q . It is $b(u, q) = \langle u, Bq \rangle$ and $a(u, v) = \langle Au, v \rangle$. If b meets the LBB condition, the linear problem (1) has a unique solution (u, p) .

In this paper we are particularly looking for suitable methods to solve the Stokes problem with variable viscosity. In this case we have $V = H_0^1(\Omega)^d$, $Q = L_0^2(\Omega)$ and

$$a(v, u) = \int_{\Omega} \eta(v_{i,j}u_{i,j} + v_{i,j}u_{j,i})d\Omega \quad (4)$$

$$b(v, q) = \int_{\Omega} v_{i,i}qd\Omega \quad (5)$$

where η is the spatial dependent viscosity with $\eta \geq \eta_{min} > 0$. This type of problems plays a key role in geoscience applications. For instance, in models for convection in the Earth's mantel the viscosity becomes a function of the temperature T in the form

$$\eta = \eta_0 e^{a \left(\frac{1}{1+T} - \frac{2}{3} \right)} \quad (6)$$

where $\eta_0 = 1$ is the viscosity for $T = \frac{1}{2}$. The constant a is called the Arrhenius number. For the case of the Earth a takes the value of 22, see [5, 7] for more details. It is pointed out that this value for a produces steep viscosity gradients providing a big challenge for any solver for the saddle-point problem (1). Notice that $a = 0$ defines the case of constant viscosity. In this context the external force f is given as

$$\langle f, v \rangle = \int_{\Omega} RaTv_2 d\Omega \quad (7)$$

where Ra is the Rayleigh number. For the Earth one sets $Ra = 10^6$.

In the case that η is constant the operator A can be simplified to a multiple of the Laplacean operator. This case has been extensively studied, for instance see [9, 3, 2]. In this paper we will investigate how the results for the case of constant viscosity can be applied to the case of spatially variable viscosity.

2 Uzawa Solver

One possible approach for solving the saddle point problem 1 is to eliminate the velocity v from the problem, see [?]. This is possible if A is invertible. In this case one gets $v = A^{-1}(f - Bp)$ which can be inserted into the second equation which leads to the problem

$$Sp = B^*A^{-1}f. \quad (8)$$

where $S := B^*A^{-1}B$ is the Schur complement of A . As the Schur complement is symmetric and positive definite, the problem (8) can be solved iteratively using PCG [6] using the standard inner product in L^2 . The question of a suitable preconditioner P_S for the Schur complement will be discussed later. Once p has been calculated v can be recovered in a postprocessor step as $v = A^{-1}(f - Bp)$. The residual r_k in the k -th iteration step can be written as

$$r_k = Sp_k - B^*A^{-1}f = B^*A^{-1}(Bp_k - f) = -B^*v_k, \text{ with } v_k = A^{-1}(f - Bp_k) \quad (9)$$

So representing the residual as the pair $r_k = (v_k, -B^*v_k)$, and then inspecting the first component of the current residual when terminating the iteration through a stopping criteria, will save the execution of the postprocessing.

When implementing the PCG one needs to provide a function which returns for a given p as an increment to the residual. This is calculated in the

form

$$\begin{aligned} (a) \quad & \text{Solve} \quad Az = Bp, \\ (b) \quad & \text{Solve} \quad q = -B^*z, \end{aligned} \tag{10}$$

where the tuple (z, q) is returned.

3 Coupled Solver

A problem with the Uzawa approach is that the evaluation of the Schur complement requires a very accurate solution for problems $Az = f$. An approach that works directly on the saddle point problem can potentially avoid this however a suitable preconditioner for the coupled problem is needed.

In the literature [9, 3, 2], different preconditioning techniques are developed for saddle point type problems. Here we will review one of them [3], which we will use in our implementations. We apply a block preconditioner of the form

$$\begin{bmatrix} A^{-1} & 0 \\ R & -S^{-1} \end{bmatrix},$$

where $R = S^{-1}B^*A^{-1}$. The iteration matrix now takes the form

$$\begin{bmatrix} A^{-1} & 0 \\ R & -S^{-1} \end{bmatrix} \begin{bmatrix} A & B \\ B^* & 0 \end{bmatrix} = \begin{bmatrix} I & A^{-1}B \\ 0 & I \end{bmatrix}.$$

The spectrum of the iteration matrix consists of 1 which is optimal. The evaluation of the preconditioned block matrix for a given vector (v, p) takes the form

$$\begin{aligned} (a) \quad & \text{Solve} \quad Aw = Av + Bp \\ (b) \quad & \text{Solve} \quad Sq = B^*(v - w) \end{aligned} \tag{11}$$

to calculate the return value (w, q) . An approximation P_A of A^{-1} can easily be constructed by a few iterations of a Krylov subspace method or of a multi-grid scheme, Therefore solution for (a) can be provided easily to any given accuracy. As presented in the previous section one can use iterative techniques to solve step (b), to a given accuracy using a suitable preconditioner P_S to approximate S^{-1} . As we argue that using more accurate schemes in step (b) would in fact lead to an Uzawa type scheme we apply the preconditioner P_S in our tests without further corrections although this might not provide the best preconditioner for the coupled problem. The problem of constructing a suitable P_S will be discussed later.

Now the preconditioned system can be solved with a Krylov subspace methods such as GMRES [8]. As a norm we use

$$\|(u, p)\|^2 = \int_{\Omega} \frac{1}{\eta^2} p^2 d\Omega + \int_{\Omega} u_{i,j}^2 d\Omega \tag{12}$$

where $u \in V = H_0^1(\Omega)^d$, $p \in Q = L_0^2(\Omega)$. This choice corresponds to the stress norm

$$\sqrt{\sigma_{ij}\sigma_{ij}} = \sqrt{p^2 + 4\eta^2 D_{ij}D_{ij}}. \quad (13)$$

where σ_{ij} represents the stress tensor and $D_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i})$ denotes the stretching. which is commonly used in mantle convection simulations.

4 Preconditioner for Schur Complement

It has been shown, see [3], that in the case of the Stokes equations the Schur complement operator S that can be preconditioned by $\frac{1}{\eta}$, where η is the viscosity. In this investigation a constant viscosity is assumed which allows to simplify the operator A to the Laplacian operator using the incompressibility condition. The important difference we discuss in this paper is the fact that for convection models η is not a constant, but spatially dependent. Unfortunately this generalization does not allow to simplify the operator A to the Laplacian operator using the incompressibility condition which is a key in the theoretical investigations. The nature of the spatial dependency of η affects the overall performance of the model, in particular a steep gradient of η will lead to a large deviation from the case of constant viscosity in which $\frac{1}{\eta}$ provides a suitable preconditioner for S .

In our tests we will choose the preconditioner $p = P_S q$ of the Schur complement by solving

$$\frac{1}{\tilde{\eta}} p = q \quad (14)$$

where we consider two choices for $\tilde{\eta}$ namely

- spatially dependent viscosity: $\tilde{\eta} = \eta$
- constant average viscosity: $\tilde{\eta} = \frac{\min(\eta) + \max(\eta)}{2}$

In the case of constant viscosity both cases coincide. Notice that in this case we do not simplify the theoretical operator A to the Laplacian operator.

5 Implementation

We use the *escript* environment to implement the convection code. In this section we will briefly outline the basic idea of *escript* and show how to

implement the evaluation step for the Schur complement (10). For more details on *escript* we refer to [4].

The *escript* module is designed to implement PDE based models in *python*. It uses PDE based terminology providing an abstraction layer for spatial discretization methods. Its key component is a class used to define steady, linear partial differential equations which are solved using a C/C++ library. In our implementation we have used the finite element solver library *finley* [1]. The coefficients of the PDE are defined through expressions which are evaluated by the *escript* library. The *escript* library is parallelized for both OpenMP and MPI using the the data distribution used by the underlying PDE solver library.

The general form of the PDE in *escript* for an unknown vector-valued spatial function u_i is

$$-(A_{ijkl}u_{k,l} + B_{ijk}u_k)_{,j} + C_{ikl}u_{k,l} + D_{ik}u_k = -X_{ij,j} + Y_i . \quad (15)$$

The coefficients A , B , C , D , X and Y are functions of their location in the domain, in particular they may depend on solutions of other PDEs, previous time steps or non-linear iteration steps. Moreover, natural boundary conditions of the form

$$n_j (A_{ijkl}u_{k,l} + B_{ijk}u_k) + d_{ik}u_k = n_j X_{ij} + y_i , \quad (16)$$

can be defined where y and d are given functions. Notice that A , B and X are already used in the PDE (15). To set values of u_i to r_i on certain locations of the domain one can define constraints of the form

$$u_i = r_i \text{ where } q_i > 0 , \quad (17)$$

where q_i is a given function defining a positive value through the locations where the constraint is applied. For more details on *escript* including code examples refer to [4].

With these tools under the belt it is very straight forward to implement the evaluation of the Schur complement S as defined in (10) within the Uzawa scheme. The following *python* function implements this step:

```
from escript import *
from escript.linearPDEs import LinearPDE
def evalS(dom, eta, p):
    v_pde=LinearPDE(dom)
    id=identityTensor4(dom)
    v_pde.setValue(A=eta*(id+swap_axes(id,1,2)), \
                  X=-p*kroncker(dom))
```

```

p_pde=LinearPDE(dom)
p_pde.setReductionOn()
z=v_pde.getSolution()
p_pde.setValue(D=1, Y=-div(z))
q=p_pde.getSolution()
return q

```

This function returns the pressure increment q . The class `LinearPDE` provides an interface to the PDE defined by (15)–(17). The corresponding values of PDE coefficients are set via the `setValue` method call. The argument `dom` is an *escript* `Domain` object defining the domain of the PDE including information on the discretization to be used. `eta` is representing the viscosity η which may be constant or a spatial function represented as an *escript* data object. The call of the `setReductionOn` method of `p_pde` switches on the usage of a reduced polynomial order for the pressure approximation as required to meet the requirements of the LBB condition.

In a similar fashion one can implement the application of the preconditioner P_A from the Schur complement in (14):

```

from escript import *
from escript.linearPDEs import LinearPDE
def evalP_S(dom, eta, p):
    pde=LinearPDE(dom)
    pde.setReductionOn()
    pde.setValue(D=1/eta, Y=p)
    return pde.getSolution()

```

It is pointed out that it would be more efficient to keep a copy of the instances of `LinearPDE` class and to reuse them in the `evalS` and `evalP_S` calls. This allows for the potential reuse of information such a preconditioners. This can be implemented using a *python* class.

As discussed in Section 2 it is advantageous to represent the residual in PCG using the pair $(v, -B*v)$ where v is the current velocity approximation. In *python* this can be easily implemented using standard PCG code and overloading algebraic operations. In this case `evalS` returns the pair (z, q) .

Additional to the saddle point problem for pressure and velocity the convection model requires the solution of the time-dependent temperature advection-diffusion problem. In *escript* and *finley* a algebraic flux correction scheme is used which is not discussed here.

6 Experiments

In the first test we investigate the efficiency of the Uzawa scheme. As a measure we use the number of inner iteration steps needed to reach a fixed tolerance (see Figure 1). As a stopping criterion we use the L^2 of the divergence of the velocity relative to its H^1 norm. For solving the convection problems we proceed by the following way:

- Initialize all constant parameters.
- For every time step in a given range, using saddle point solver compute (v, p) .
- Update temperature.
- Go to the next time step.

In the following figures outer iterations will refer to the loop in time steps, whereas inner iterations will refer to the iterations of the saddle-point solver.

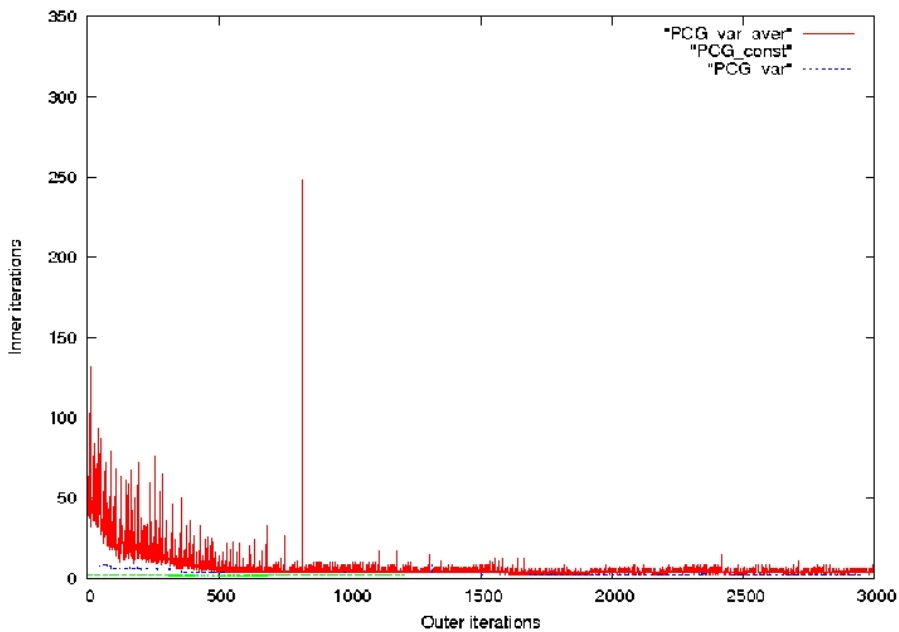
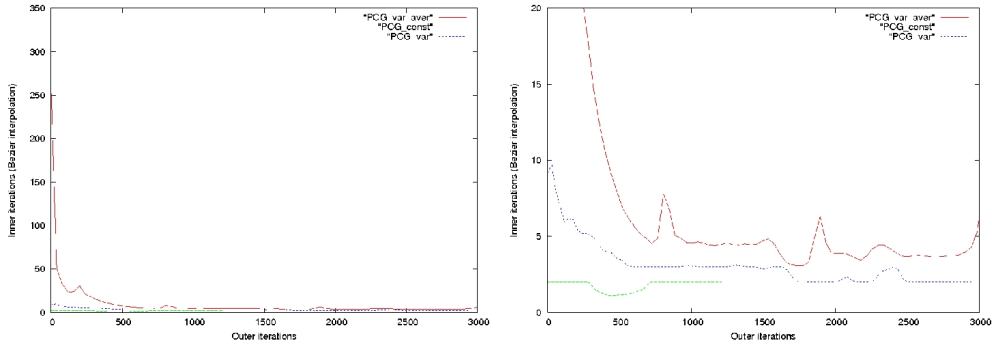


Figure 1: Comparison of inner iterations for the cases of constant viscosity (PCGR_const), variable viscosity with average viscosity preconditioning (PCG_var_aver) and variable viscosity (PCG_var).

In order to make Figure 1 more readable we present our data with Bezier curve interpolation (see Figure 2).



(a) Range for inner iterations is the same as in Figure 1. (b) Range for inner iterations is cut from 0 to 20 for showing difference more clearly.

Figure 2: Bezier curve interpolation for data presented in Figure 1.

Experiments clearly confirms that using $\frac{1}{\eta}$ as an approximation for the Schur complement is a very effective way to achieve fast convergence in iterative solvers for saddle point problems. Note that in both cases for constant and variable viscosities we have achieved very fast convergence. We tested this approach with other solvers as well such as GMRES and we got similar convergence. Similar results were also achieved with GMRES coupled solver for constant viscosity however for the case of variable viscosity one observed slow convergence for both cases of the P_S preconditioner.

In the next experiment we investigated how the Arrhenius number affects the convergence behavior of the solver. We have already presented the cases where Arrhenius number $a = 0$ (constant viscosity) and $a = 22$ (variable viscosity). The interesting question will be how $a = 11$ will affect the solver. In Figure 3 we present experimental results.

7 Conclusion

As we mentioned earlier, approximations and preconditioning techniques are one of the main techniques to speed up iterative solvers. For earth mantle convection problems where we observe variable viscosity, we were able to show empirically that $1/\eta$ can be used as a suitable approximation for the Schur complement. We also presented a way of implementation using the Python-based PDE modeling environment Escript.

Acknowledgements: This work is making use of infrastructure provided through the Auscope National Collaborative Research Infrastructure Strat-

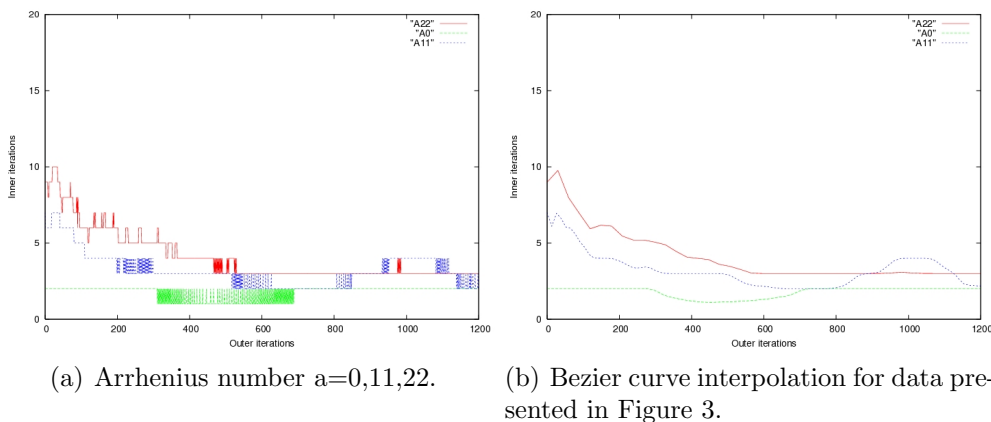


Figure 3: How Arrhenius number affects PCG solver.

egy (NCRIS) with funding from the Australian Commonwealth, the Queensland State Government and the University of Queensland.

References

- [1] K. Arrow, L. Hurwicz, and H. Uzawa. *Studies in linear and nonlinear programming*. Stanford University Press. Stanford, CA, 1958.
- [2] M. Davies, L. Gross, and H.-B. Muhlhaus. Scripting high performance earth systems simulations on the sgi altix 3700. In *High Performance Computing and Grid in Asia Pacific Region, 2004. Proceedings. Seventh International Conference on*, pages 244 – 251. IEEE, 2004.
- [3] A. de Niet and W. Wubs. Two preconditioners for saddle point problems in fluid flows. *International Journal for Numerical Methods in Fluids*, 54:355–377, 2007. Published online 19 December 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/flid.1401.
- [4] H. Elman and D. Silvester. Fast nonsymmetric iterations and preconditioning for Navier-Stokes equations. *SIAM Journal on Scientific Computing.*, 17(1):33–46, 1996.
- [5] L. Gross, L. Bourguin, A.J. Hale, and H.-B. Mhlhaus. Interface modeling in incompressible media using level sets in escript. *Physics of The Earth and Planetary Interiors*, 163(1-4):23–34, 2007.
- [6] M. Kameyama, A. Kageyama, and T. Sato. Multigrid iterative algorithm using pseudo-compressibility for three-dimensional mantle con-

- vection with strongly variable viscosity. *Journal Computational Physics*, 206(1):162–181, 2005.
- [7] A. V. Knyazev. A preconditioned conjugate gradient method for eigenvalue problems and its implementation in a subspace. In *International Ser. Numerical Mathematics, v. 96, Eigenwertaufgaben in Natur- und Ingenieurwissenschaften und ihre numerische Behandlung, Oberwolfach, 1990.*, pages 143–154. Birkhuser Basel, 1991.
- [8] L. Moresi, F. Dufour, and H.-B. Muehlhaus. Mantle convection modeling with viscoelastic/brittle lithosphere: Numerical methodology and plate tectonic modeling. *Journal on Pure and Applied Geophysics*, 159(10):2335–2356, 2002.
- [9] Y. Saad and M. Schultz. GMRES a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific Statistics and Computing.*, 7:856–869, 1986.
- [10] D. Silvester, H. Elman, D. Kay, and A. Wathen. Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow. *J. Comput. Appl. Math.*, 128(1-2):261–279, 2001. Numerical analysis 2000, Vol. VII, Partial differential equations.