A Convergent Algorithm for Time Parallelization Applied to Reservoir Simulation

Izaskun Garrido, Magne S. Espedal, and Gunnar E. Fladmark[1]

University of Bergen, LIM / CIPR, (http://www.mi.uib.no/~izaskun/)

Summary. Parallel methods are not usually applied to the time domain because the sequential nature of time is considered to be a handicap for the development of competitive algorithms. However, this sequential nature can also play to our advantage by ensuring convergence within a given number of iterations. The novel parallel algorithm presented in this paper acts as a predictor corrector improving both speed and accuracy with respect to the sequential solvers. Experiments using our in house fluid flow simulator in porous media, `Athena`, show that our parallel implementation exhibit an optimal speed up relative to the method.

1 Time parallel reservoir simulator: `Athena`

Sub-stepping is common practice in reservoir simulation, in this technique some unknowns of a given system are computed in time-steps smaller than the normal step size, such that the overall system resolution will be comparable to that obtained with the small time-step. We propose a novel modified algorithm where this sub-stepping is computed in parallel following a technique of type Parareal, as studied in Baffico et al. [2002], Lions et al. [2001], Bal and Maday [2002]. This technique is a predictor corrector, PC, where the corrector runs in parallel. Thus, the time domain is subject to the standard treatment of domain decomposition Briggs et al. [1990], Keyes [2002], Brenner and Sung [2000]; being separated into sub-domains where different numerical solvers and discretizations may be applied. At each predictor corrector iteration the coarse solver acts as a predictor handling the sub-domain interfaces by providing initial boundary conditions for the parallel-fine system of equations. Each parallel solution will be used as a corrector determining the modification of the coarse system for the next iteration. Due to the sequential nature of time, one of these solutions is independent from the rest. This particularity will be exploited to modify the algorithm so as to obtain convergence rendering it suitable to dynamic load balancing schemes Lan et al. [2002]. The motivation for this type of parallelization is not only to improve both computational speed and convergence properties but aims to implement multi-grid with nested parallelism.

The paper is organized as follows: The numerical model implemented in our reservoir simulator `Athena` is presented in Section 2. This model contains a sub-stepping technique, which will be modified to run in parallel according to the method described in Section 3. Numerical examples illustrate in Section 4 the performance of these algorithms. Finally, conclusions of this work are given in Section 5.

2 The numerical model and standard method in `Athena`

Athena is a multicomponent multi-phase flow simulator in porous media. This simulator is based on a mathematical model consisting of three coupled non-linear differential equations which solve for the primary variables: molar masses, temperature and water pressure. The equations are derived from the mass conservation, energy conservation and volume balance method. These systems will be decoupled and discretized using Finite Volume in space and a Backward Euler scheme in time. However, the energy equation has coefficients dominated by the rock temperature, which is almost constant, so that their values at time $t^{[n+1]}$, may be approximated by those at previous time, $t^{[n]}$, leading to an explicit equation of the form

$$\mathcal{J}^{[n]} \Delta \mathbf{T}^{[n]} = -\mathbf{f}^{[n]} , \tag{1}$$

where the temperature increment $\Delta \mathbf{T}^{[n]} = \mathbf{T}^{[n+1]} - \mathbf{T}^{[n]}$. Solving the pressure equation with the *Newton-Raphson* method, we get the expression

$$\mathcal{J}^{[n(k)]} \Delta \mathbf{p}^{[n(k)]} = -\mathbf{f}^{[n(k)]} , \tag{2}$$

where $n(k)$ denotes the k^{th} *Newton-Raphson* iteration at the n^{th} time level

$$\Delta \mathbf{p}^{[n(k)]} = \mathbf{p}^{[n(k+1)]} - \mathbf{p}^{[n(k)]} ,$$

$$\mathcal{J}^{[n(k)]} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{p}} \right)^{[n(k)]} \simeq \frac{\mathcal{D}^{[n(k)]}}{\Delta t^{[n]}} + \mathcal{A}^{[n]} ,$$

and the right hand side looks like

$$\mathbf{f}^{[n(k)]} = \mathcal{D}^{[n(k)]} \frac{\mathbf{p}^{[n(k)]} - \mathbf{p}^{[n]}}{\Delta t^{[n]}} + \mathcal{A}^{[n]} \mathbf{p}^{[n(k)]} - \mathbf{b}^{[n]} .$$

Note that at each Newton iteration we consider an approximation by updating only the diagonal, $\mathcal{D}$, of the Jacobian matrix $\mathcal{J}$. Finally, the molar mass equations are decoupled considering the cross-derivatives between different components to be negligible. This assumption enables to solve sequentially the following residual equations for the molar masses

$$\left(\frac{\mathcal{I}}{\Delta t^{[n]}} + \mathcal{A}_\nu^{[n(k)]}\right) \Delta \mathbf{N}_\nu^{[n(k)]} = \mathbf{b}_\nu^{[n(k)]} , \qquad \nu = 1, \ldots, n_c \tag{3}$$

where for a chemical system consisting of n_c components located on a domain decomposed into n_{cv} cells, $\mathcal{I}$ is the identity matrix and each matrix $\mathcal{A}_\nu$ has n_{cv}^2 entries $(\mathcal{A}_\nu)_{ij}^{[n(k)]} = \sum_l \alpha_{\nu_i,\nu_j}^{l\,[n(k)]}$, $i\,, j \in \{1, \ldots, n_{\text{cv}}\}$, which are derived from an analytical expression.

The set of equations (1), (2) and (3), results in a compact numerical model, since they allow to find all the primary and secondary variables for the new time step. As the molar masses change very fast in relation to either the temperature or pressure, computational stability requires small time-steps and a greater number of Newton iterations for the mass conservation equation. In order to mitigate the time step restriction that the molar mass equation, see (3), imposes on the overall system, it is common practice to use sub-stepping. This technique involves computing with a coarse time-step the implicit solution for the temperature and pressure, whilst the molar masses are calculated for the same overall time step with several smaller sub-steps. For further details about `Athena`, its numerical model and implementation we refer the interested reader to a number of previous publications Fladmark [1997], Øye and Reme [1999], Garrido et al. [2003]. The rest of this paper will be devoted to the development and implementation of a Parareal-type algorithm which will parallelize the sequential sub-stepping technique.

3 General algorithm

Denoting by $\bar{\Omega} = \Gamma \cup \Omega$ an arbitrary time-step where $\Gamma = t^n$ and $\Omega = \Omega^n = (t^n, t^{n+1}]$, equation (3) is commonly solved by sub-stepping over a number, N, of sub-domains $\Omega_i = (t^n + (i-1)\Delta t, t^n + i\Delta t]$, where $\Delta t = (t^{n+1} - t^n)/N$, have either artificial boundaries Γ_i, $i > 1$ or real one $\Gamma_1 = \Gamma$ and are discretized independently of one another. The convergence of the sub-stepping over Ω^n determines adaptively the size of the next time step Ω^{n+1}.

The algorithm to be presented is of the Parareal form as proposed by Maday and Lyons Baffico et al. [2002], Lions et al. [2001], Bal and Maday [2002] and uses a PC where the corrector runs in parallel. In what follows, the PC will be described on equation (3) for a general time-step domain, $\bar{\Omega}$, and a given component, ν. For simplicity of notation we rewrite (3) as a residual equation of the form

$$\begin{aligned} \mathcal{J} \Delta u &= f\ , \quad \bar{\Omega}\ , \\ u &= g\ , \quad \Gamma\ . \end{aligned} \tag{4}$$

Denoting the kth PC iteration with the superscript k, the method begins by predicting a solution of

$$\begin{aligned} \mathcal{J}^k \Delta u^k &= f^k\ , \quad \bar{\Omega}\ , \quad k = 1\ , \\ u^k_{|\Gamma} &= g\ , \quad \Gamma\ , \quad k = 1\ , \end{aligned} \tag{5}$$

sub-stepping a number G of times, with $G << N$ and $\Omega = \cup_{i=1}^{G} \Omega_i$, a coarsening of the original domain decomposition. Note that system (5) is solved over each Ω_i using the *Newton-Raphson* method so that $\mathcal{J}^k$ has to be updated at every Newton iteration. This sub-stepping gives an approximation to the intermediate values $u^k_{|\Gamma_i}, i = 2, \ldots, G$ which together with the initial boundary condition, $u^k_{|\Gamma} = u^k_{|\Gamma_1} = g$ serve as initial guesses for the boundary conditions of each independent system

$$\left.\begin{aligned} \mathcal{J}^k_{|\bar{\Omega}_i} \Delta u^k_i &= f^k_{|\bar{\Omega}_i} \\ u^k_{i|\Gamma_i} &= g + \textstyle\sum_{j=1}^{i-1} \Delta u^k_{|\Omega_j} \end{aligned}\right\} \qquad i = 1, \ldots, G\ . \tag{6}$$

This set of systems will be solved in parallel so that the ith processor solves the ith system by sub-stepping on Ω_i a number $F(i)$ of times $\Omega_i = \cup_{j=1}^{F(i)} \Omega_j$, giving an approximation to the values $u^k_{|\Gamma_i}, i = 2, \ldots, G$. If these approximations do not differ more than a given tolerance to those obtained previously from system (5) convergence for time step Ω has been achieved. Else, a new PC iteration, for systems (5) and (6) is computed, where the predictor equation (5) is corrected with data from the previous iteration as

$$\mathcal{J}^k_{|\bar{\Omega}_i} \Delta u^k = f^k_{|\bar{\Omega}_i} + \mathcal{J}^{k-1}_{|\bar{\Omega}_i} (u^{k-1}_{|\Gamma_{i+1}} - u^{k-1}_{i|\Gamma_{i+1}})\ . \tag{7}$$

Due to the non-linear nature of the PDE system under study, the correction term $\mathcal{J}^{k-1}_{|\bar{\Omega}_i} (u^{k-1}_{|\Gamma_{i+1}} - u^{k-1}_{i|\Gamma_{i+1}})$ is also non-linear and needs to be updated at every Newton iteration. This correction is equivalent to a Jacobi iteration for a linearized system with matching discretization along the artificial boundaries.

Another modification of this scheme can be obtained by adding in equation (5) all the corrections obtained from previous iterations to obtain schemes of the Parareal form

$$\mathcal{J}^k_{|\bar{\Omega}_i} \Delta u^k = f^k_{|\bar{\Omega}_i} + \sum_{j=1}^{k-1} \mathcal{J}^j_{|\bar{\Omega}_i} (u^j_{|\Gamma_{i+1}} - u^j_{i|\Gamma_{i+1}})\ . \tag{8}$$

Even more, after the first iteration the active domain is redefined to be the reduced

$$\Omega = \bar{\Omega} | \bar{\Omega}_1\ , \tag{9}$$

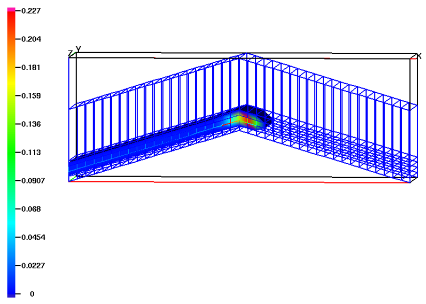
$u^k_{|\Gamma_1} = g$ and the initial boundary condition satisfies

$$u^k_{|\Gamma_2} = u^1_{1|\Gamma_2}\ . \tag{10}$$

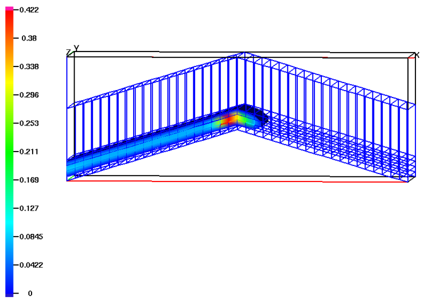
Assuming that after iteration $k = G - 1$ it is satisfied

$$u^k_{|\Gamma_1} = g, \ldots, u^k_{|\Gamma_G} = u^{G-1}_{G-1|\Gamma_G} \tag{11}$$

where $\Omega = \bar{\Omega} | \cup_{i=1}^{G-1} \bar{\Omega}_i$; at the Gth iteration both predictor and corrector are defined over the same active domain and share the same initial boundary



a. Gas saturation



b. Oil saturation

Fig. 1. Hydrocarbon migration simulated by `Athena` for $100\,y$ with $0.0047\,y$ as time step

condition, so that the best approximation is given by the solution to the fine-parallel system, $u^G_{G|\Gamma_{G+1}}$. Therefore, this algorithm converges in at most G iterations, where G denotes the amount of systems to be solved in parallel and its accuracy is determined by the corrector approximation.

4 Numerical examples

In this section we will illustrate with two different experiments the scalability and performance of the algorithm implemented for the molar mass equations within the `Athena` fluid flow migration simulator. We have carried out the experiments on a Linux cluster, with PIII processors to explore the behavior of the methods.

Before proceeding with the numerical experiments, the geological domain and boundary conditions shall be described. The three dimensional domain has 50 m depth on the ends, and a size of $1000\,\text{m} \times 100\,\text{m} \times 70\,\text{m}$. There are four different layers in the z direction: shale, sandstone, shale and sandstone again. The lithology for the sandstone has a porosity of $\phi = 0.5$ and a permeability of $K_x = 500\,\text{mD}$, $K_y = 500\,\text{mD}$ and $K_z = 500\,\text{mD}$ while the corresponding values for the shale are $\phi = 0.5$ and $K_x = 5 \cdot 10^{-6}\,\text{mD}$, $K_y = 5 \cdot 10^{-6}\,\text{mD}$ and $K_z = 5 \cdot 10^{-6}\,\text{mD}$. The domain is initially filled with water and the boundary conditions consist of an explicitly given flux of oil and gas with value $5 \cdot 10^{-5}\,\text{mol/m}^2\text{s}$ going inwards on the left hand side and an outwards water flux with value $6.5 \cdot 10^{-4}\,\text{mol/m}^2\text{s}$ in the right hand side. There are also temperature boundary conditions of 450°K at the top and 460°K at the bottom. The domain is uniformly subdivided in each direction as is shown in Fig. 1, which serve as an illustration of the `Athena` output for a simulated time of 100 years.

We consider the algorithm as described in Section 3 where, due to implementation issues in our particular simulator, the modification term, $S^{[n],k}_{|\bar{\Omega}_i}$, will be an approximation of that given in (7) by using the current matrix

$$\mathcal{J}^k_{|\bar{\Omega}_i} \Delta u^k = f^k_{|\bar{\Omega}_i} + \mathcal{J}^k_{|\bar{\Omega}_i} (u^{k-1}_{|\Gamma_{i+1}} - u^{k-1}_{i|\Gamma_{i+1}}) \, . \tag{12}$$

Computational results: Load balance

In this experiment we are mainly interested in the scalability of the implementation. By varying the number of processors used, we want to explore the speedup compared to the sequential program. When increasing the number of processors the wall clock time is expected to decrease, but there is a limit where adding more processors will not decrease the wall clock time. Besides this MPI implementation has a master-slave structure where the number of sub-domains equals the number of parallel (slave) processors used and the master deals with the non-parallelizable part of the algorithm.

The domain is partitioned considering a fixed underlying grid with a total of $s = \sum s_G = 2^4$ cells. Therefore, letting the number of sub-domains to double, $G = 2^i$, with $i = 0, \ldots, 4$ implies that the number of cells in each sub-domain halves, $s_G = 2^{4-i}$. Besides, since each sub-domain uses only one processor, increasing the number of sub-domains is equivalent to increase the overhead. In in the left of Fig. 2 the run time for the slaves is plotted against

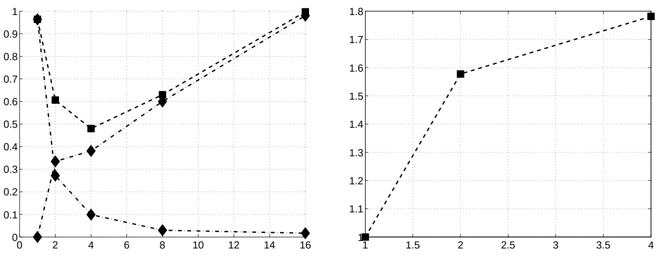


Fig. 2. Space domain with 200 cells. Run time vs. the number of sub-domains. In the left hand side, communication (.- increasing), calculation (.- decreasing) and addition of both times (- -) for the slaves. In the right, the parallel speedup for the addition of slave and master run times.

the number of sub-domains, considering the run time values to be the average of all parallel processors. It can be seen that as the number of sub-domains (or processors) doubles, the calculation time at each processor halves (see the monotonic decreasing curve), whilst the time for collective broadcasting increases (see the monotonic increasing curve). The relevant values correspond to the addition of both communication and calculation times; these are plotted by the curve marked with squares which indicates that the method is competitive up to a certain parallelization degree when the communication time overrides the computational time.

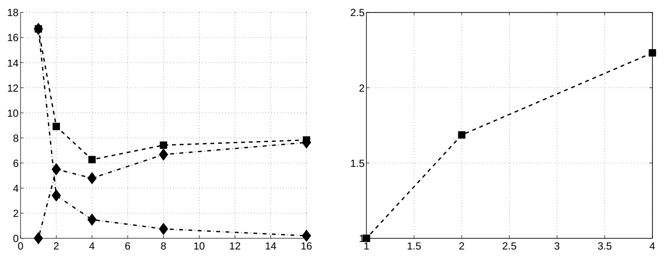


Fig. 3. Space domain with 800 cells. Run time vs. the number of sub-domains. In the left hand side, communication (.- increasing), calculation (.- decreasing) and addition of both times (- -) for the slaves. In the right, the parallel speedup for the addition of slave and master run times.

The speedup for the overall method, addition of master (coarse solver) and slave (parallel-fine solver) run times, is displayed in the right of Fig. 2 where the best time is clearly obtained when $G = s_G$, which can easily be proven to coincide with the case when optimal load balance occurs.

Computational Results: Overhead

Given a fixed spatial discretization, the previous section studies the scalability in time as the underlying time-mesh remains constant whilst the ratio coarse to fine cells varies. We aim to demonstrate that the scalability properties hold as the spatial domain increases, therefore decreasing the space overhead. Even when the communication is seen in the left of Fig. 3 to became negligible with respect to the time for calculations, the master-slave structure of the algorithm gives a fixed overhead due to the communication master-slave and it can be seen in the right of Fig. 3 that the total run time is best when the load balance reaches optimality.

5 Conclusions

In this paper a convergent parallel algorithm has been derived in a constructive manner. It acts as a predictor corrector and the numerical experiments indicate that even for highly non-linear problems this parallel formulation improves both speed and accuracy with respect to the standard sequential solvers. Besides, the sequential nature of time allows to ensure convergence within a given number of iterations. The implementation of this algorithm in our fluid flow simulator, `Athena`, has optimal speed-up.

A Galerkin-type of algorithm based on Additive Schwarz Bjørstad et al. [2002], Xu and Zikatanov [2002, 2003], Cai et al. [2002] is under current consideration for space-time parallelization.

Acknowledgement. This work was supported by the Norwegian Research Council.

References

L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah. Parallel in time molecular dynamics simulations. *Phys. Rev. E.*, 66, 2002.

G. Bal and Y. Maday. A parareal time discretization for non-linear pde's with application to the pricing of an american put. In *Proceedings of a Workshop on Domain Decomposition*, LNCSE. Springer Verlag Zurich, 2002.

P. E. Bjørstad, M. Dryja, and T. Rahman. Additive schwarz methods for elliptic mortar finite element problems. *Submitted to Numerische Mathematik*, 2002. http://www.ii.uib.no/p̃etter/reports/pbmdtr2002AddSchMort.ps.gz.

S. C. Brenner and L.-Y. Sung. Lower bounds for non-overlapping domain decomposition preconditioners in two dimensions. *Math. Comput.*, 69(232): 1319–1339, 2000.

W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial.* SIAM, 1990.

X.-C. Cai, D. E. Keyes, and L. Marcinkowski. Non-linear additive schwarz preconditioners and application in computational fluid dynamics. *Int. J. Numer. Meth. Fluids*, pages 1463–1470, 2002.

G. E. Fladmark. Secondary oil migration. mathematical and numerical modeling in som simulator. Technical Report R-077857, Norsk Hydro, Bergen, 1997.

I. Garrido, E. Øian, M. Chaib, G. E. Fladmark, and M. S. Espedal. Implicit treatment of compositional flow. *Computational Geosciences*, 2003. To appear.

D. E. Keyes. Domain decomposition in the mainstream of computational science. In *Proceedings of the 14 international conference on Domain Decomposition Methods*, 2002.

Z. Lan, V. E. Taylor, and G. Bryan. A novel dynamic load balancing scheme for parallel systems. *J. Parallel Distrib. Comput.*, 62(12):1763–1781, 2002.

J.-L. Lions, Y. Maday, and G. Turinici. Rèsolution d'edp par un schéma en temps pararéel. *C. R. Acad. Sci. Paris*, 332(1):1–6, 2001.

G. Å. Øye and H. Reme. Parallelization of a compositional simulator with a galerkin coarse/fine method. In P.Amestoy et al., editors, *Euro-Par'99*, volume 1685. Springer-Verlag, Berlin, 1999.

J. Xu and L. Zikatanov. The method of alternating projections and the method of subspace corrections in Hilbert space. *J. of AMS*, 15, 2002. Technical report, PennState, November 2000a.

J. Xu and L. Zikatanov. Some observations on Babuska and Brezzi theories. *Num. Math.*, 94, 2003. Technical report, PennState, September 2000b.