Dynamic Optimization of Linear Solver Parameters in Mathematical Modelling of Unsteady Processes

Dmitry Bagaev[1,2(✉)], Igor Konshin[1,3], and Kirill Nikitin[1]

[1] Institute of Numerical Mathematics of the Russian Academy of Sciences, Moscow 119333, Russia
bvdmitri@gmail.com, igor.konshin@gmail.com, nikitin.kira@gmail.com
[2] Lomonosov Moscow State University, Moscow 119991, Russia
[3] Dorodnicyn Computing Centre, FRC CSC RAS, Moscow 119333, Russia

Abstract. The optimization of linear solver parameters in unsteady multiphase groundflow modelling is considered. Two strategies of dynamic parameters setting for the linear solver are proposed when the linear systems properties are modified during simulation in the INMOST framework. It is shown that the considered algorithms for dynamic selection of linear solver parameters provide a more efficient solution than any prescribed set of parameters. The results of numerical experiments on the INM RAS cluster are presented.

Keywords: Parallel linear solver · Mathematical modelling · Unsteady process · Automatic performance tuning · Black-Oil Simulator

1 Introduction

The problem of software performance tuning is of great importance for efficient usage of the modern supercomputer facilities. It is very important for numerical modelling applications exploiting such a software.

One of the most famous examples of automatic software tuning is the ATLAS package [1] which carry out the performance optimization for several BLAS functions during the installation of the package.

Another important and very popular idea of software performance tuning is the usage of data mining techniques. For example, Self-Adapting Numerical Software (SANS) [2] and Self-Adapting Large-scale Solver Architecture (SALSA) [3] perform the analysis of the input data to select the linear solver from the set of available ones. The machine learning techniques is used for the same goal as well [4].

The genetic algorithms are used in [5] in a software system called Intelligent Performance Assistant (IPA) to improve the performance of ExxonMobil's proprietary reservoir simulator, EMpower^{TM}.

V. Voevodin and S. Sobolev (Eds.): RuSCDays 2017, CCIS 793, pp. 54–66, 2017.
https://doi.org/10.1007/978-3-319-71255-0_5

In the present paper we would like to return 'back to basics' of linear algebra and to knowledge on the mathematical properties of the preconditioned iterative algorithms considered. For this reason we consider the solution of unsteady problems that comes from multiphase black-oil reservoir simulation. The main difficulty of selecting the optimal parameters of the linear solvers at each simulation time step is the modification of the stiffness matrix properties. If the linear solver is already selected prior to the unsteady problem solution, then one has at least a possibility to select the input set of linear solver parameters. The main idea is to construct the procedure of automatic and dynamic selection of parameters that are close to the optimal ones, i.e. provide the minimum of the solution time. In the present paper we propose two different algorithms for this approach.

For our numerical experiments on dynamic parameters tuning we have exploited an INMOST software platform [6]. Besides the ability to operate with the distributed meshes of general form, this platform includes a convenient interface for solving large sparse linear systems. It allows user to forget about specific implementations of each particular linear solver and to focus only on parameters optimizations. INMOST provides a large variety of different linear solvers, some of them are implemented inside the platform, the others can be enabled as external libraries, such as PETSc [7] or Trilinos [8].

We consider the INMOST linear solver BIILU2 for our numerical experiments. This solver is the combination of the second order incomplete triangular factorization ILU2(τ) and the incomplete inverse LU factorization BIILU(q) (as a replacement of additive Schwarz preconditioning AS(q)) [9,10]. Here, τ is the factorization threshold and q is the number of overlap levels for blocks corresponding to each processor. The use of ILU2(τ) factorization is chosen due to it's robust and efficient preconditioning in comparison with the conventional structural incomplete factorization ILU(k) or the conventional incomplete threshold factorization ILU(τ).

As an example of simulation we use the multi-phase flow model based on the fully implicit time discretization and the nonlinear monotone two-point approximation for the Darcy fluxes in Jacobian matrix [11].

2 Algorithm's Description

2.1 The Choice of Appropriate Optimization Algorithm

The function to be optimized can be defined as

$$T_k = G(A_k, b_k, p, \varepsilon) \equiv F(A_k, b_k, p) \pm \varepsilon, \tag{1}$$

where A_k is the linear system matrix on kth time step, b_k is the right-hand side vector on kth time step, p is the parameter (or parameters) of some liner solver to be optimized, T_k is the return value of function which is equal to the time needed for solving the linear system $A_k x = b_k$.

A lot of difficulties associated with the real unsteady processes should be taken into account while choosing algorithms for parameter optimization of linear solvers:

1. Function G can behave differently from run to run, as we solve the problem in parallel mode using the MPI library. The time of messages delivery between processors is nondeterministic, so the function value may vary on some small but essential unknown value ε which is impossible to predict. Therefore the target function may have several local minima and maxima.
2. During unsteady process both A_k and b_k will be modified with each time step and as a result optimal parameters p will be changed as well. The optimization algorithm should be able to find these parameters (or close to it) regardless their modification in time.
3. The value of function G can be calculated only once for given A_k, b_k and for the selected parameters p. There is no reason to solve linear system $A_k x = b_k$ again even with more optimal parameters.
4. As A_k may vary with the simulation time, the minimum value of function F may increase. This is why it is really hard to use the previous values of $F(A_k, b_k, p)$.
5. The algorithm should not be computationally expensive and time spent on the parameters optimization should not affect the total time of solving the unsteady problem.

To deal with the above difficulties we should also use a number of assumptions on the function F:

1. For given A_k and b_k the function is continuous by parameters p and has the form close to a paraboloid, and since $F > 0$ the global minima exists and finite.
2. In a real simulation matrices A_k may differ, however they have about the same structure and properties, and as a result we expect that the optimal parameters based on the kth time step are moved in its small neighborhood and within this area the values of the T_k are roughly equal.
3. We also assume that at some time step k' the minimal solution time T_k is not increasing and depends only on parameters p.

Based on the above issues and assumptions we have proposed to use two optimization algorithms.

2.2 Very Fast Simulated Re-annealing

Annealing (simulated annealing, SA) is a probabilistic technique for approximating the global optimum of a given function. At each step, the SA heuristic considers some neighbouring state s' of the current state s, and probabilistically decides between moving the system to state s' or staying in state s. These probabilities ultimately lead the system to move to states of lower energy. Typically

this step is repeated until the system reaches a state that is good enough for the application, or until a given computation budget has been exhausted.

The probability of making the transition from the current state s to a candidate new state s' is specified by an *acceptance probability function* $h(e, e', T)$, that depends on the energies $e = E(s)$ and $e' = E(s')$ of the two states, and on a global time-varying parameter T called the *temperature*. States with a smaller energy are better than those with a greater energy. The probability function P must be positive even when e' is greater than e. This feature prevents the method from becoming stuck at a local minimum that is worse than the global one.

When T tends to zero, the probability $h(e, e', T)$ must tend to zero if $e' > e$ and to a positive value otherwise. For sufficiently small values of T, the system will then increasingly favor moves that go "downhill" (i.e., to lower energy values), and avoid those that go "uphill". With $T = 0$ the procedure reduces to the *greedy algorithm*, which makes only the downhill transitions [12].

The method of simulated annealing consists of three functional relationships:

g – probability density of state-space of D parameters $x = \{x^i, i = 1, D\}$;
h – probability density for acceptance of new cost-function given the just previous value;
$T(k)$ – schedule of annealing temperature T in annealing time steps k, i.e. of changing volatility or fluctuations of the two previous probability densities.

The acceptance probability is based on the chances of obtaining a new state s' relative to a previous state s,

$$h = \frac{\exp(-e'/T)}{\exp(-e'/T) + \exp(-e/T)} \approx \frac{1}{1 + \exp(\Delta E/T)}, \tag{2}$$

where ΔE represents the *energy* difference between the present and previous values of the cost-function appropriate to the physical problem, i.e. $\Delta E = e' - e$ (see [13]).

The algorithm itself can be described by the following steps:

1. Select a random state s. The energy values of the system is set to $E(s)$.
2. On kth step:
   (a) Compare the energy of the system $E(s)$ in the state s with the global minimum. If it is smaller then change the global minimum value.
   (b) Generate a new state s' and calculate $E(s')$.
   (c) Generate a random number α uniformly distributed over $[0, 1]$. If $\alpha < h(\Delta E, T(k))$ then set s' as the current state and go to the next iteration $k + 1$. Otherwise repeat the previous step until a suitable state s' will be found.

In the present paper we are using the "very fast annealing scheme" produced by Ingber [13]. In this scheme different parameters may have different finite ranges, fixed by physical considerations, and different annealing-time-dependent sensitivities, measured by the curvature of the cost-function at local minima.

Consider parameters x_k^i in ith dimension generated by an annealing step k with the following range

$$x_k^i \in [A_i, B_i] \tag{3}$$

calculated with the random variable ξ_i:

$$x_{k+1}^i = x_k^i + \xi_i(B_i - A_i), \qquad \xi_i \in [-1, 1]. \tag{4}$$

The above formula can be applied several times until $x_{k+1}^i \in [A_i, B_i]$.

Generating function defined as

$$g_T(\xi) = \prod_{i=1}^{n} \frac{1}{2(|\xi_i| + T_i)\ln(1 + 1/T_i)} \equiv \prod_{i=1}^{n} g_{(i;T)}(\xi_i), \qquad \xi_i \in [-1, 1],$$

$$\xi_i = \operatorname{sgn}\left(\alpha_i - \frac{1}{2}\right) T_i((1 + 1/T_i)^{|2\alpha_i - 1|} - 1), \tag{5}$$

where α_i are random numbers, uniformly distributed over segment $[0, 1]$.

Annealing schedule will be defined as

$$T_i(k) = T_{(i;0)} \exp(-c_i k^{1/D}), \qquad c_i > 0. \tag{6}$$

It is proven [13], that the very fast annealing algorithm are one of the most effective method of random search of optimal solutions for a wide class of problems.

2.3 Alternating Parameters Probe Based Tuning

Another idea for constructing the algorithm for dynamic parameters tuning for unsteady problem is the attempt to stay at a local minimum probing a nearby area. If the current parameters set is near to the minimum or the minimum is moving not too fast then the algorithm may track the minimum.

The algorithm (1U) for unsteady problem can be formulated as follows:

Specify initial values for τ, q, and probe direction dir from $\{\delta_{\tau+}, \delta_{q+}, \delta_{\tau-}, \delta_{q-}\}$
while simulation stopping criterion **do**
  Make time step
  Solve linear system
  if new minimum found **then**
    Update minimum set (τ, q)
  end if
  if dir$= \delta_{\tau+}$ **then**
    ind$(\tau) + +$
  else if dir$= \delta_{q+}$ **then**
    ind$(q) + +$
  else if dir$= \delta_{\tau-}$ **then**
    ind$(\tau) - -$

```
    else if dir= δ_{q−} then
      ind(q) − −
    else
      Stay with no change of (τ, q)
    end if
  end while
```

2.4 Linear Brute-Force Searching

Linear brute-force search is the simplest algorithm, which can find the global minimum of the given function $F(\overline{x})$, where $\overline{x} = (x_1, x_2, \ldots x_n)$ on an arbitrary grid D.

Linear brute-force search algorithm implies optimizing each variable x_i independently and was implemented in the following way:

- The set of runs for different values of τ from $\tau_{\min}$ to $\tau_{\max}$ for a fixed value of overlap size parameter $q = 3$ was performed, and a quasi-optimal value of τ^* was found.
- The set of runs for different values of q from $q_{\min}$ to $q_{\max}$ for a fixed value of quasi-optimal τ^* was performed, and a quasi-optimal pair of parameters (τ^*, q^*) was found.

This method are very computationally expensive and therefore can't be recommended for solving real problems. However it can be used to find the almost precise global minimum on quite dense grids and enable us to verify the other parameter tuning approaches.

3 Numerical Experiments

3.1 INM Cluster Configuration

All numerical experiments was performed on INM RAS cluster. The configuration of the cluster computational nodes, used for numerical experiments [14]:

- Compute Node Arbyte Alkazar+ R2Q50;
- 16 cores (two 8-core processors Intel Xeon E5-2665@2.40 GHz);
- 64 Gb RAM;
- SUSE Linux Enterprise Server 11 SP1 (x86_64).

3.2 Dependance on Parameters for a Sample Problem

As a sample linear system we have used the system (called below N14) obtained from the INM RAS Black-Oil Simulator for Scholars (BOSS) for the well-known SPE-10 problem [15]. The size of the model mesh is $60 \times 220 \times 85$ cells ($1.122 \cdot 10^6$ cells). The top 35 layers of the model is a Tarbert formation, and is a representation of a prograding near shore environment, while the bottom 50

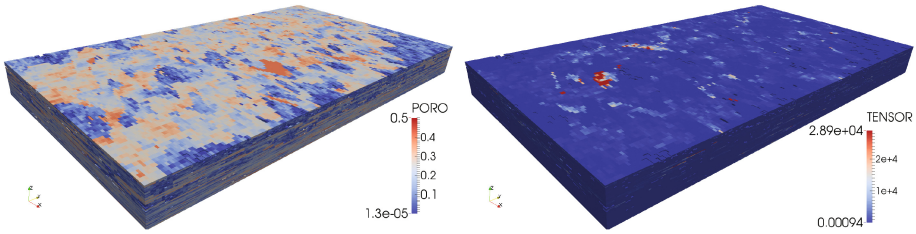
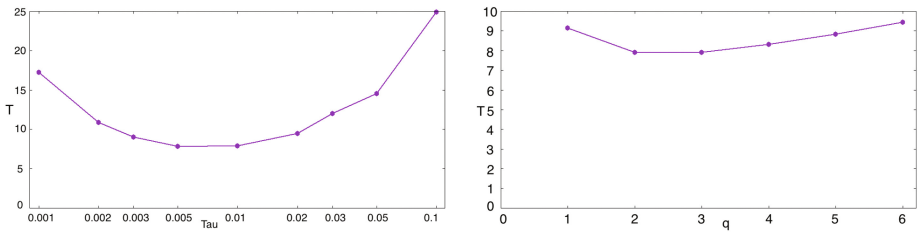


Fig. 1. The porosity and permeability distributions for SPE-10 problem

layers represents Upper Ness which is fluvial. The coefficients of the media are very contrast. The porosity varies from $1.3 \cdot 10^{-5}$ to 0.5 (see Fig. 1, left) and the permeability varies from 10^{-3} to $3 \cdot 10^4$ (see Fig. 1, right). The model has 5 vertical wells completed throughout formation. The central well is an injector and the other 4 wells in the corners are producers.

The dimension of the obtained linear system N14 is 3 896 013 unknowns. The dependences of solution time T (in seconds) on parameters τ and q is demonstrated in Fig. 2 for 16 cores and it is quite smooth with the minimum pronounced.

Figure 3 shows the 2D surface of the solution time T in variables τ and q for the same problem N14 solved on 16 cores. The obtained surface is of paraboloid type.



(a) $T = f(q = 3, \tau)$ (b) $T = f(q, \tau = 0.003)$

Fig. 2. Total solution time T in s. for N14 depending on τ and q for $p = 16$

3.3 Dynamic Function Simulation

We consider the following two-parameter function for the research purposes:

$$f(\tau, q) = \left(\frac{16}{25}(\lg(\tau/\tau_0))^2 + 1\right)\left(\frac{1}{25}\left(\frac{17.5(q - q_0)}{7.5 + q - q_0}\right)^2 + 1\right), \tag{7}$$

$$\tau_0 = 0.003, \quad q_0 = 3. \tag{8}$$

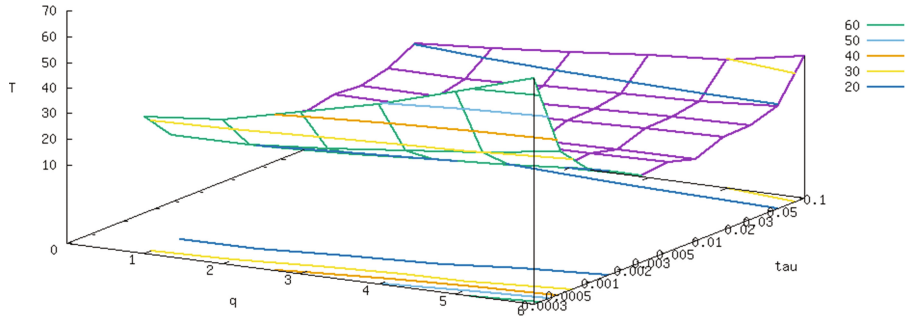


Fig. 3. Total solution time T in s. for N14 in variables τ and q for $p = 16$

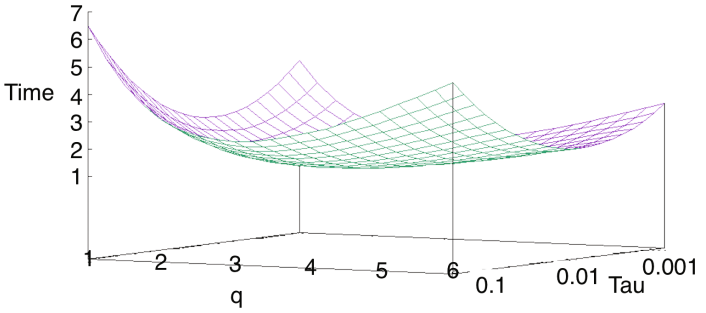
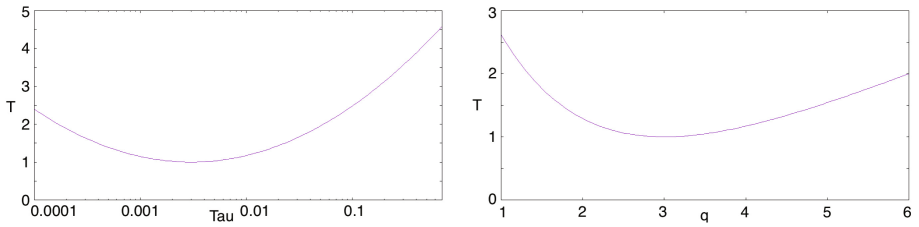


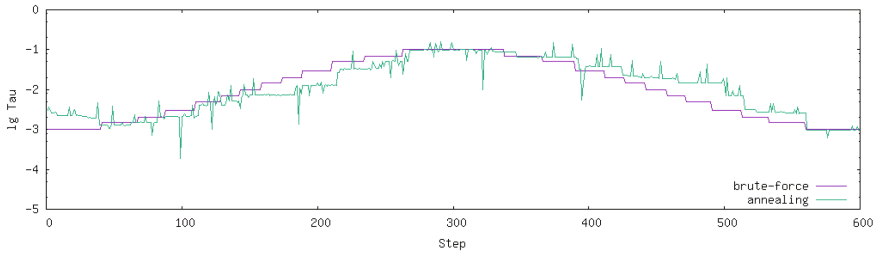
Fig. 4. Two-parameter function (7) and (8)



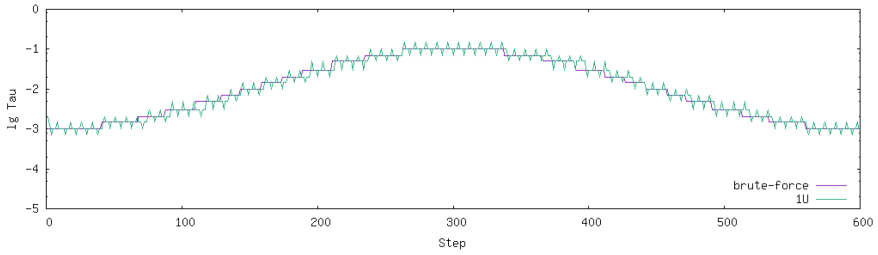
(a) $T = f(q = 3, \tau)$ (b) $T = f(q, \tau = 0.003)$

Fig. 5. Cross-sections for $q = q_{\text{opt}} = 3$ and $\tau = \tau_{\text{opt}} = 0 : 003$

This function can be used as a solution time T measured in seconds instead of that for real black-oil simulation process and demonstrates more strong dependence on $\lg(\tau)$ as well as more weak one on overlap parameter q. Figure 4 demonstrates the respective paraboloid for the above function, which is qualitatively similar to the paraboloid on Fig. 3. The minimum of this two-parameter function is in $(\tau = 0.003, q = 3)$ in accordance with (8). Figures 5a and b show the cross-sections for $q = q_{\text{opt}} = 3$ and $\tau = \tau_{\text{opt}} = 0.003$, respectively. With this simple



(a) Brute-force search and SA algorithm values of τ



(b) Brute-force search and 1U algorithm values of τ

Fig. 6. τ_{opt} depending on the time step k for function (7), (9)

function we can easily examine the proposed parameter tuning approaches as well as provide the complete repeatability of our numerical experiments.

The most interesting is the behavior of proposed algorithms in the unsteady case. We can modify the above steady state function (7) in the following way:

$$\tau_0 = 10^{-2-\cos(2\pi t/t_0)}, \quad q_0 = 2 + \cos(2\pi t/t_0), \quad t_0 = 100 \tag{9}$$

where we have the local optimal values $\lg \tau \in [-3; -1]$ and $q \in [1; 3]$ for time moment $t \in [0; t_0]$.

Figures 6a and b plot τ_{opt} depending on the time step for above unsteady-state function (7), (9). This figures show that proposed algorithms SA and 1U are able to track the optimal parameters even if they change in time.

3.4 Unsteady Black-Oil Simulation

We consider the two-phase flow model of the INM RAS BOSS simulator for the real unsteady problem. We simulate 6000 days of the quarter five spot problem with one injector and one producer wells. The initial water saturation is equal to residual saturation which results in rather sharp front. The starting time step is 0.0001 days, which increases to 25 days later in the simulation. An incremental time step leads to an increase in the complexity of linear systems, so does the water breakthrough which results in higher flow velocities. In our simulation the water breakthrough occurs at about time 1400 or at about 65th time steps.

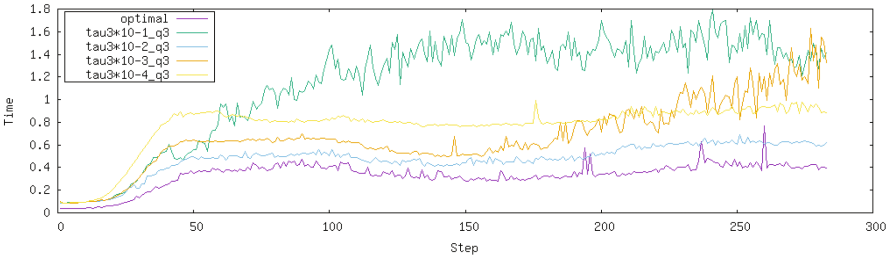


Fig. 7. Unsteady black-oil simulation times with fixed parameters and the dynamic optimal ones depending on the simulation time step k

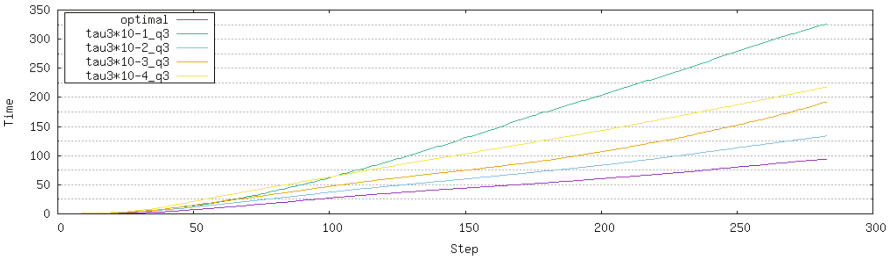


Fig. 8. Unsteady black-oil simulation cumulative times with the fixed parameters and the dynamic optimal ones depending on the simulation time step k

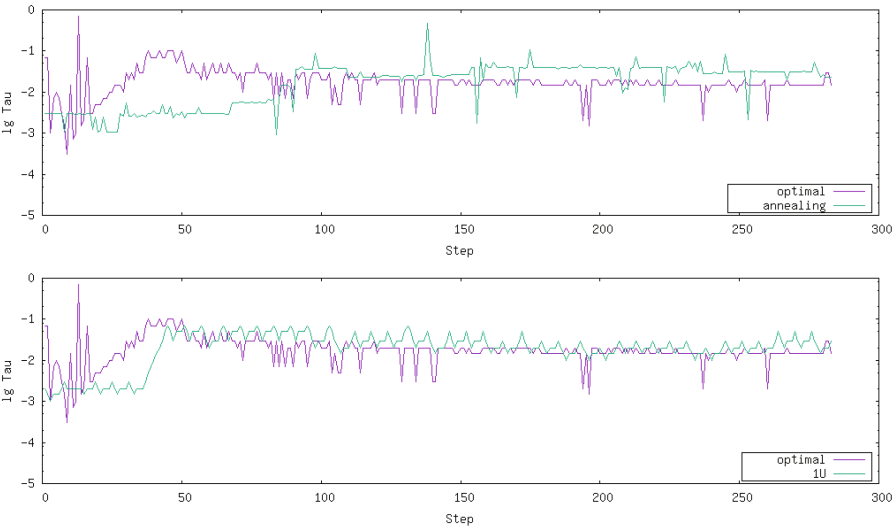


Fig. 9. Optimizing τ for black-oil simulator

First, in Figs. 7 and 8 we plot solution and cumulative times depending on the simulation time step for several fixed sets of parameters: $(\tau = 0.3, q = 2)$, $(\tau = 0.03, q = 3)$, $(\tau = 0.003, q = 3)$, $(\tau = 0.0003, q = 3)$ and compare it with the optimal one, which was found using the linear brute-force search algorithm. One can see that any fixed set of parameters produce the result which is far from the optimal solution time.

The same experiment was performed for the two proposed algorithms very fast simulated re-annealing (SA) and 1U. Figures 9 and 10 present the plots for the estimated value τ_{opt}, local and cumulative solution time T and T_{Σ}, respectively, depending on the simulation time step k. One can observe that the results of the proposed algorithms are very close to that for the optimal set of parameters. The cumulative solution time for all the proposed algorithms is less than that for any observed fixed set of parameters (τ, q) (Fig. 11).

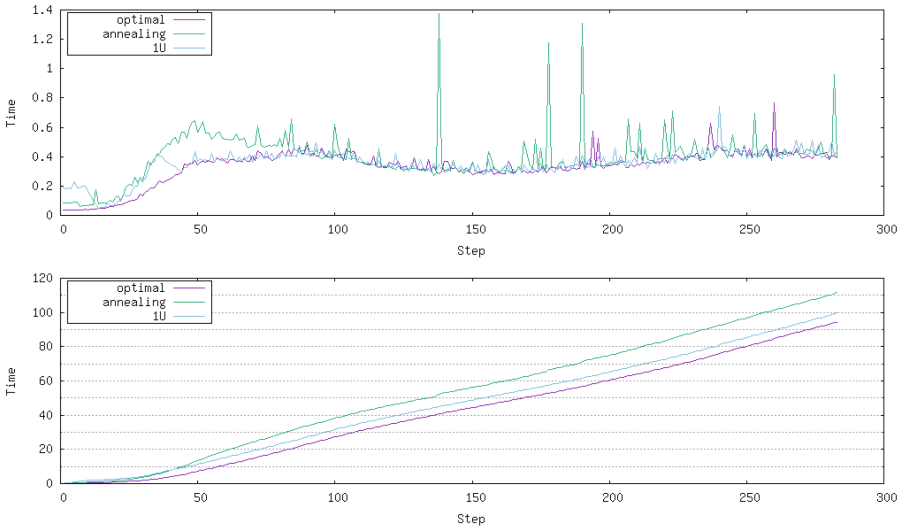


Fig. 10. Local and cumulative times depending on the simulation time step k

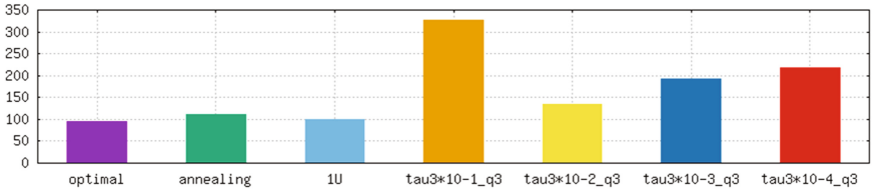


Fig. 11. Cumulative times bar chart for default sets of parameters and for proposed algorithms compared with the optimal one

4 Conclusion

The proposed linear solver parameters tuning algorithms were implemented in the INMOST framework as an optional toolkit named Trace and Tuning Software Platform (TTSP). In conclusion of the present paper, we formulate the most important issues of the progress in this area:

- The influence of the linear solver parameters on the real black-oil simulation performance was examined;
- The set of optimization algorithms for linear solver parameters tuning were proposed;
- The TTSP toolkit for parallel linear solver parameters tuning was developed and verified for the INM RAS black-oil reservoir simulator for scholars (BOSS);
- It was shown that proposed algorithms essentially increase the performance of the real unsteady black-oil simulation in comparison with even the best fixed set of parameters.

Acknowledgements. This work has been supported in part by RFBR grant 17-01-00886, Russian Federation President Grant MK-2951.2017.1, and ExxonMobil Upstream Research Company.

References

1. Automatically Tuned Linear Algebra Software (ATLAS). http://math-atlas.sourceforge.net/. Accessed 15 Apr 2017
2. Eijkhout, V., Fuentes, E., Eidson, T., Dongarra, J.: The component structure of a self-adapting numerical software system. Int. J. Parallel Program. **33**(2), 14–18 (2005)
3. Self-Adapting Large-scale Solver Architecture (SALSA). http://icl.cs.utk.edu/salsa/index.html. Accessed 15 Apr 2017
4. Bhowmick, S., Eijkhout, V., Freund, Y., Fuentes, E., Keyes, D.: Application of machine learning to the selection of sparse linear solvers. Int. J. High Perform. Comput. Appl. 1–24 (2006)
5. Mishev, I.D., Beckner, B.L., Terekhov, S.A., Fedorova, N.: Linear solver performance optimization in reservoir simulation studies. In: Society of Petroleum Engineers. SPE Reservoir Simulation Symposium, pp. 1–9. The Woodlands, Texas (2009)
6. INMOST - a toolkit for distributed mathematical modelling. http://www.inmost.org. Accessed 15 Apr 2017
7. PETSc - Portable, Extensible Toolkit for Scientific computation. https://www.mcs.anl.gov/petsc/. Accessed 15 Apr 2017
8. The Trilinos Project. https://trilinos.org. Accessed 15 Apr 2017
9. Kaporin, I.E., Konshin, I.N.: Parallel solution of large sparse SPD linear systems based on overlapping domain decomposition. In: Malyshkin, V. (ed.) PaCT 1999. LNCS, vol. 1662, pp. 436–446. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48387-X_45

10. Kaporin, I.E., Konshin, I.N.: A parallel block overlap preconditioning with inexact submatrix inversion for linear elasticity problems. Numer. Linear Algebra Appl. **9**(2), 141–162 (2002)
11. Nikitin, K.D., Terekhov, K.M., Vassilevski, Y.V.: A monotone nonlinear finite volume method for diusion equations and multiphase OWS. Comp. Geosci. **18**(3), 311–324 (2014)
12. Simulated annealing - Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Simulated_annealing. Accessed 15 Apr 2017
13. Ingber, L.: Very fast simulated re-annealing. Math. Comput. Model. **12**, 967–973 (1989)
14. INM RAS cluster. http://cluster2.inm.ras.ru. Accessed 15 Apr 2017. (in Russian)
15. SPE Comparative Solution Project. http://www.spe.org/web/csp. Accessed 15 Apr 2017