Partially Updated Restricted Additive Schwarz Preconditioner

Laurent Berenguer[1] and Damien Tromeur-Dervout[1]

Introduction

The solution of differential equations with implicit methods requires the solution of a nonlinear problem at each time step. We consider Newton-Krylov [9, Chapter 3] methods to solve these nonlinear problems: the linearized system of each Newton iteration of each time step is solved by a Krylov method. Generally speaking, the most time-consuming part of the numerical simulation is the solution of the sequence of linear systems by the Krylov method. Then, providing a good preconditioner is a critical point: a balance must be found between the ability of the preconditioner to reduce the number of Krylov iterations, and its computational cost. The method that combines a Newton-Krylov method with a Schwarz domain decomposition preconditioner is called Newton-Krylov-Schwarz (NKS) [4]. In this paper, we deal with the Restricted Additive Schwarz (RAS) preconditioner [5]. We propose to freeze this preconditioner for a few time steps, and to partially update it. Here, the partial update of the preconditioner consists in recomputing some parts of the preconditioner associated to certain subdomains, keeping the other ones frozen. These partial updates improve the efficiency and the longevity of the frozen preconditioner. Furthermore, they can be computed asynchronously in order to improve the parallelism.

The remainder of this paper is organized as follows: Section 1 presents the partial update of the Restricted Additive Schwarz (RAS) preconditioner. In Section 2, we propose to compute this partial update asynchronously on additional devices in order to achieve a parallel algorithm. The third section is devoted to numerical experiments on a reaction-diffusion problem. They

Université de Lyon, Université Lyon 1, CNRS, Institut Camille Jordan (umr5208), 69622 Villeurbanne, France `laurent.berenguer@univ-lyon1.fr` · `damien.tromeur-dervout@univ-lyon1.fr`

show that the partially updated preconditioner is more robust than the frozen preconditioner, and that a superlinear speed-up can be achieved.

1 The partial update of the RAS preconditioner

We consider ordinary differential equations of the form $\dot{x} = f(x,t)$ where $x \in \mathbb{R}^n$ is the solution and the function f from $\mathbb{R}^{n+1}$ to $\mathbb{R}^n$ is nonlinear. The problem $\dot{x} = f(x,t)$ is solved for a given initial condition $x(0) = x_0$ and suitable boundary conditions. If an implicit method is used for the time integration, then a nonlinear problem of the form $F(x^l, t^l) = 0$ must be solved in x^l at each time step t^l. This nonlinear problem is generally solved by Newton-like methods that require the solution of linear systems of the form

$$J(x_k^l, t^l)\delta x_k^l = -F(x_k^l, t^l) \tag{1}$$

where the subscript k stands for the number of the Newton iteration, $J(x_k^l, t^l) \in \mathbb{R}^{n\times n}$ is the Jacobian matrix of $F(\cdot, t^l)$ at the solution x_k^l. The Newton-Krylov method can be viewed as an inexact Newton method if Eq. (1) is solved by a Krylov method. A good preconditioning method is needed to accelerate the convergence of Krylov methods. The preconditioning matrix M_k^l should approximate $J(x_k^l, t^l)$ and its inverse must be computed easily. Preconditioners based on domain decomposition methods are often used because their application to vectors requires only the solution of subdomain problems. The domain of n unknowns is split in N overlapping subdomains. Each subdomain i has n_i unknowns if we include the overlap, and $\tilde{n}_i$ if we exclude the overlap (i.e. $n = \sum_i^N \tilde{n}_i$). Let $R_i \in \mathbb{R}^{n_i \times n}$ denote the operator that restricts a vector to the ith subdomain, including the overlap. We also denote by $\tilde{R}_i \in \mathbb{R}^{n_i\times n}$ the restriction operator to the ith subdomain that excludes the overlap by setting to zero the lines corresponding to the overlap. For simplicity of notation, we write J instead of $J(x,t)$ when no confusion can arise. Thus, the RAS preconditioner of the matrix J is given by Eq. (2).

$$M_{RAS}^{-1} = \sum_{i=1}^{N} \tilde{R}_i^T \left(R_i J R_i^T\right)^{-1} R_i \tag{2}$$

In Eq. (2), we assumed that the local Jacobian matrices $J_i = R_i J R_i^T$ are invertible. This is not necessary to compute explicitly the matrices J_i^{-1} because the Krylov method requires only the application of the preconditioner to vectors. This application is computed by the parallel solution of N local linear systems. In the following, we solve this local linear system using the LU factorizations of the matrices J_i.

Several methods have been proposed to optimize the solution of a sequence of slightly changing linear systems, and all of them consist in reusing some computations done at the previous linear systems. Then, several ways to reuse the Krylov subspace have been considered. An overview of these techniques is given in [10] but it is worth mentioning that the information provided by the Krylov subspace can be used to update a preconditioner. Hence, in [7] a preconditioner based on deflation is updated at each restart of GMRES. In order to save computational time, we consider the reuse of the same RAS preconditioner for a few successive linear systems: this frozen preconditioner is called Lagged RAS (LRAS) in the following. In this case, it may be relevant to update the preconditioner from one linear system to another, instead of recomputing it. Several ways to update a preconditioning matrix have already been considered. It has been proposed in [3] to update the preconditioner, adding a low rank matrix that corresponds to the quasi-Newton update. The update of a factorized preconditioner has also been considered: the update AINV preconditioner has been studied in [2] for a sequence of diagonally shifted matrices. In [13], an algebraic formula is derived to update the ILU preconditioner from the difference of two successive linear operators. This idea has been extended to Jacobian-free methods in [6]. The frozen preconditioner is expected to become less and less efficient from one linear system to another. Then, a recomputation of the preconditioner may be needed to prevent convergence failures of the Krylov method. It is a difficult task to decide when a frozen preconditioner needs recomputed, but two heuristic criteria are often used:

- The preconditioner can be recomputed if the previous linear system has needed more than K_{max} Krylov iterations.
- The preconditioner can also be updated every L linear systems.

In this paper, we choose the first approach that seems more flexible: it can allow to save numerous of unnecessary global updates. On the other hand, the number of needed Krylov iterations can vary during the simulation because it does not only depend on the age of the preconditioner. Then, to be optimal, K_{max} should be adapted during the simulation. However, this topic exceeds the scope of this paper.

When LRAS is used, the update of the preconditioner is global: all the local LU factorizations are computed simultaneously. We can extend this idea to a partial update: only some parts of the preconditioner are updated. Hence, the preconditioner can be written as in Eq. (3), where *AsRAS* stands for *Asynchronous Restricted Additive Schwarz*. The preconditioner is now compounded of local Jacobian matrices evaluated at different Newton iterations or time steps. It is worth pointing out that if $t_i = t$ and $k_i = k$ for $i = 1 \ldots N$ then $M^{-1}_{AsRAS} = M^{-1}_{LRAS}$.

$$M^{-1}_{AsRAS} = \sum_{i=1}^{N} \tilde{R}_i^T J_i(x^{l_i}_{k_i}, t^{l_i})^{-1} R_i \tag{3}$$

In order to avoid idle time, asynchronous solvers have been studied for linear and nonlinear problems [8, 12]. The disadvantage of asynchronous solvers lies in the fact that one needs to make extra assumptions on the problem and its splitting to ensure the convergence. Here, the updates of the local parts of the preconditioner are asynchronous, but the communications between subdomains are synchronous. Then, the theoretical framework of Newton-Krylov solvers applies directly: the exact solution of preconditioned linear systems is the same regardless of the preconditioning matrix. That being said, Krylov methods approximate the solution to a given tolerance, the digits of the solution beyond this tolerance might differ from one preconditioner to another.

One can expect that the partial update allows to save Krylov iterations during the simulation. Numerical results will confirm this idea from a global point of view, but we do not assume that every single partial update improves the condition number of the linear systems.

The sequential implementation of the AsRAS preconditioner is straightforward. The implementation on a parallel computer is a much more difficult task because idle time may arise when only some processors compute the LU factorization. To circumvent this difficulty, in the next section we propose to dedicate additional processes to the LU factorizations.

2 Parallel implementation of the asynchronous RAS preconditioner

The method presented in the previous section does not seem suitable for parallel computing because the load is not balanced: some of the processors will have to wait while the other ones compute the LU factorizations since the Krylov method entails synchronizations. In the following, we present an efficient algorithm where the LU factorizations are computed by processors that are not in charge of a subdomain.

The key point of the asynchronous partial update of the RAS preconditioner is to define two kinds of tasks that communicate: the first one is the solution of a subdomain problem (i.e. the classical Newton-Krylov method). The second one is the LU factorizations of local Jacobian matrices. Then, in order to solve the physical problem, one should assign most of the CPU cores to the first task.

Algorithm 1 describes how to implement the method in a client-server approach. The client processes are those assigned to subdomains, while the server processes are devoted to the computation of LU factorizations. The client processes must be able to continue the computation between the sending of the local Jacobian matrix and the reception of the factorized matrix. The reception of the factorized matrix is the partial update since the new LU factorization is received in the memory space of the previous one. In our MPI implementation, the communication pattern is the following: client processes

Algorithm 1 Asynchronous update of the preconditioner

```
1: // Client process
2: for each time step do
3:     // Newton iterations:
4:     repeat
5:         if a LU factorization is available,
           partially update $M_{AsRAS}^{-1}$
6:         if global update then
7:             $M_{AsRAS}^{-1} = \sum_{i=1}^{N} \tilde{R}_i^T J_i^{-1} R_i$
8:         end if
9:         Krylov method to solve
           $J\delta x = -F(x)$ preconditioned by
           $M_{AsRAS}^{-1}$
10:        $x \leftarrow x + \delta x$
11:        if needed, send $J$
12:    until convergence
13: end for
```

```
1: // Server process
2: repeat
3:     receive the matrix
4:     compute the LU factorization
5:     send the factorization
6: until the end of the integration
```

check if the server is ready to receive the local Jacobian matrix before they send it. This checking is implemented using `MPI_Test`. Likewise, client processes check if the LU factorization is ready before they start the reception. The reception can be done between two Newton iterations or even between two Krylov iterations if the Krylov method allows variable preconditioners [11]. Finally, Algorithm 1 can be viewed as an improvement of LRAS: both algorithms are equivalent if no processes are assigned to the partial update of the preconditioner. As stated above, there are more client processes than server processes. As a consequence, only few partial updates can be computed simultaneously. Then, one needs to decide in which order the requests will be treated. In the remainder of this paper, we limit ourselves to a cyclic update: we first update the first subdomains, then the second ones and so on. This approach is not optimal, because it does not update more frequently the subdomains where highly nonlinear phenomena appear. Since we proposed an asynchronous implementation of AsRAS in Algorithm 1, we cannot assume that the partial updates will be received it time to prevent convergence failure of the Krylov method. That is the reason why, there is a global restart in Algorithm 1, step 7, as in classical LRAS implementations.

3 Numerical tests

This section presents some tests that highlight the behavior of the methods. The computer cluster used for the numerical experiments is an SGI Altix XE 1300, with two six-cores Intel Xeon 5650 per node. The PETSc library [1] was used for the implementation. Generally speaking, one should avoid the exchange of factorized matrix through the network, then a MPI library that performs efficient intranode communications is required. The tasks must be distributed in such a way that client processes and their server share some memory. For example, one core per processor hosts the server process and the other cores host its client processes. We compare the behavior of AsRAS to LRAS for a reaction-diffusion problem given in Eq. (4). The domain is the unit square with periodic boundary conditions.

$$\begin{cases} \dot{u} - \alpha_1 \Delta u = A + u^2 v - (B+1)u \\ \dot{v} - \alpha_2 \Delta v = Bu - u^2 v \end{cases} \tag{4}$$

We consider the following parameters:

- The domain is discretized in 100×100 points, using a five-point stencil and decomposed in 16 subdomains.
- The problem is solved for $t \in [0, 10]$ using the backward Euler scheme with a variable time step. The solution at $t = 0$ is $u(x, y) = A + 10^{-2} \times r(x, y)$ and $v(x, y) = A/B + 10^{-2} \times r(x, y)$ where $r(x, y) \in [-0.5, 0.5]$ are random numbers.
- The coefficients of the reaction are $A = 3.5$, $B = 12$, $\alpha_1 = 1 \times h^2$ and $\alpha_2 = 2.6 \times h^2$, where h is the spatial length scale.
- The nonlinear solver is a Newton method with line search where Jacobian matrices are approximated by finite differences. The linear solver is a right-preconditioned BiCGSTAB [14].
- The RAS preconditioner, with an overlap of one, is recomputed if the number of Krylov iterations of the previous linear system has exceed K_{max}.
- LRAS has run on 16 cores, and AsRAS on 20 cores (16 cores associated to subdomains, and 4 cores dedicated to the partial update of the preconditioner).

Let us remark that, because of the asynchronous behaviour of our implementation of AsRAS, the number of Krylov iterations and the last digits of the solution may vary from one run to another. That is the reason why the execution time and the number of Krylov iterations given in Figure 1 and 2 are averages over five runs. The cumulate number of Krylov iterations is given in Fig. 1. In both cases, the number of total Krylov iterations increases with respect to K_{max}, that is to say when the number of global updates decreases. However, the partial update limits this increase. The total execution times are plotted in Fig. 2. The minimum wall time is given by the best balance between the Krylov iterations and the computational cost of the preconditioner. The minimum wall time is 2.46 for LRAS and 2.15 for AsRAS. If we compare these minimum wall times, the speedup is 1.14 which is lower than the theoretical linear speedup 1.25. On the other hand, if we consider all the tests, the average speedup is about 1.5 because AsRAS is less sensitive to K_{max} than LRAS. In practice, one will solve the problem only once with a poor approximation of the best K_{max}. In that case a superlinear speedup can be obtained.

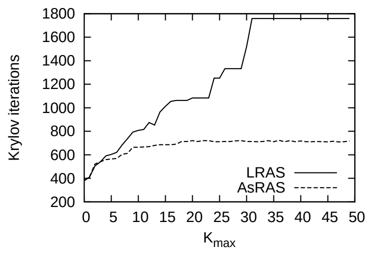

Fig. 1 Total number of Krylov iterations (averages over five runs)

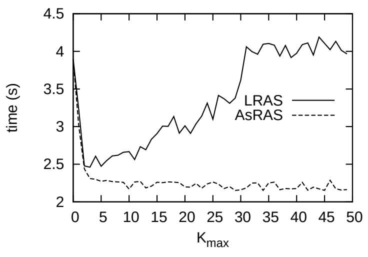

Fig. 2 Wall times in seconds (averages over five runs)

4 Conclusions

The utilization of domain decomposition preconditioners allows us to update only certain parts of the preconditioner, keeping the other ones constant. In the context of parallel computing, this partial update can be computed asynchronously, that is to say that the time-stepper computations are not stopped if the update is not available. Finally, numerical results showed that superlinear speedups can be obtained by adding processes dedicated to the LU factorizations. Furthermore, the preconditioner is continually updated which makes the results less sensitive to the frequency of global update. In this paper, all subdomain parts of the preconditioner are successively updated, but it would be relevant to update more often the LU factorizations associated to subdomains with high local nonlinearities. Then, the AsRAS preconditioner should benefit from a numerical criterion that helps to choose which subdomains need the more an update.

Acknowledgements

This work has been supported by the French National Agency of Research (project ANR-MONU12-0012 H2MNO4), and the région Rhône-Alpes. Authors also thank the Center for the Development of Parallel Scientific Computing (CDCSP) of the University of Lyon 1 for providing us with computing resources.

References

[1] Satish Balay, Jed Brown, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes,

Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.3, Argonne National Laboratory, 2012.

[2] Michele Benzi and Daniele Bertaccini. Approximate inverse preconditioning for shifted linear systems. *BIT Numerical Mathematics*, 43(2):231–244, 2003.

[3] L Bergamaschi, R Bru, A Martínez, and M Putti. Quasi-Newton preconditioners for the inexact Newton method. *Electronic Transactions on Numerical Analysis*, 23:76–87 (electronic), 2006.

[4] Xiao-Chuan Cai, William D Gropp, David E Keyes, and Moulay D Tidriri. Newton-Krylov-Schwarz methods in CFD. In *Proceedings of the International Workshop on Numerical Methods for the Navier-Stokes Equations*, pages 17–30, Braunschweig, 1995. Vieweg.

[5] Xiao-Chuan Cai and Marcus Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM J. Sci. Comput.*, 21(2):792–797 (electronic), 1999.

[6] Jurjen Duintjer Tebbens and Mirosla Tuma. Preconditioner updates for solving sequences of linear systems in matrix-free environment. *Numer. Linear Algebra Appl.*, 17:997–1019, 2010.

[7] Jocelyne Erhel, Kevin Burrage, and Bert Pohl. Restarted gmres preconditioned by deflation. *Journal of computational and applied mathematics*, 69(2):303–318, 1996.

[8] Andreas Frommer and Daniel B. Szyld. On asynchronous iterations. *J. Comput. Appl. Math.*, 123(1-2):201–216, 2000. Numerical analysis 2000, Vol. III. Linear algebra.

[9] C. T. Kelley. *Solving nonlinear equations with Newton's method*, volume 1 of *Fundamentals of Algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2003.

[10] Michael L Parks, Eric De Sturler, Greg Mackey, Duane D Johnson, and Spanda Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM Journal on Scientific Computing*, 28(5):1651–1674, 2006.

[11] Youcef Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, 1993.

[12] Pierre Spiteri, Jean-Claude Miellou, and Didier El Baz. Parallel asynchronous Schwarz and multisplitting methods for a nonlinear diffusion problem. *Numer. Algorithms*, 33(1-4):461–474, 2003. International Conference on Numerical Algorithms, Vol. I (Marrakesh, 2001).

[13] Jurjen Duintjer Tebbens and Miroslav Tuma. Efficient preconditioning of sequences of nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, 29(5):1918–1941, 2007.

[14] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644, 1992.