Authors' response
EGUSPHERE-2023-2690

Daniel Moreno-Parada, Alexander Robinson,
Marisa Montoya, and Jorge Alvarez-Solas.

March 18, 2025

Contents

1 Editor's remarks

The authors are thankful to the editor (Ludovic Räss) for his comments, thus resulting on a more detailed discussion section (Section 7 in the manuscript). We herein clarify some technical aspects raised by the editor. Editor's remarks are given in italic, whereas the authors' answers read in regular font.

- *Regarding the technical aspects, could you clarify the versioning of the software?*

Thank you for the suggestion. Nix GitHub repository did not have any tags whatsoever. The authors have now tagged the current version as 1.0, where potential future changes will follow SemVer. We have updated the repository as well as the default main branch when cloning into Nix.

- *Moreover, you may want to enhance slightly your README(s) as to better convey the information and important steps the users need to follow in order to get started. Being in Markdown format, please add code snippets and/or blocks where needed, and make use of (sub-)section separators to enhance readability - Thanks!*

The README file has been updated accordingly. Readability was also enhanced exploiting the Markdown format. Nix GitHub repository also reflects these last changes.

- *It would be interesting to provide some directions or hint on why there is only a marginal performance increase when using threads > 1. As you explain, the heavy lifting is done using the Eigen library. May it be that this library calls e.g. into BLAS under the hood and utilises already more than one thread by default? Any addition on this topic would be beneficial to the reader.*

There are a wide range of reasons that may explain a marginal performance when parallelization is considered. Before diving into the details, we must note that Eigen library does not call BLAS. Instead, Eigen's native multithreading uses OpenMP (de thafault) or Intel TBB.

Given that the velocity solution carries the heaviest computations and it relies on the BiCGSTAB method. Moreover, we have additionally tested the scalability performance on a slightly simpler approach: the Conjugate Gradient method (CG). Results are consistent for both algorithms and suggest that the bottleneck is given by the limitations of sparse matrix-vector multiplications. We now discuss potential limitations concerning the particular linear solvers employed in Nix:

- First, it is important to note that the CG and BiCGSTAB approaches are memory-bound, and not compute-bound. As a result, performance is limited by memory access speed rather than computational power. Unlike dense matrix-matrix multiplication, CG and BiCGSTAB involve two main types of computations:
  - Sparse matrix-vector products (SpMV).
  - Vector updates (i.e., dot products and norms).

  These operations do not parallelize as effectively since they involve multiple memory accesses and low arithmetic intensity.

- Difficulties with parallel sparse Matrix-Vector Multiplication (SpMV). In both CG and BiCGSTAB, the dominant operation is SpMV and it is hard to parallelize for a number of reasons:
  - It involves random memory access patterns that do not benefit from CPU cache locality (recall that sparse matrices are not stored in continuous block of memory). As a result, each CPU frequently loads new, non-contiguous data into the cache.
  - It is bandwidth-limited: CPUs are often limited by how fast they can fetch data from RAM rather than by raw FLOPS.
  - Load balancing issues: different rows of a sparse matrix have different numbers of nonzeros entries, leading to workload imbalance among threads.

  Eigen method remains simple and centred on a data-parallel approach (i.e., partitioning the sparse matrix). Namely, rows are divided among the available threads leveraged by OpenMP and then, each thread processes a contiguous block of rows to minimize the synchronization overhead. Since the number of non-zero entries per row remains constant, the partitioning approach seemed justified. Nevertheless, Eigen does not allow for more complex partitioning strategies such as graph-based approaches (Çatalyürek and Aykanat, 1999), corner partitioning (Wolf et al., 2008) or block-cyclic partitioning (Aboelaze et al., 1991; Petitet and Dongarra, 1999) to minimize the total communication volume while keeping the computation balanced across compute nodes.

- Potential overhead. Since the BiCGSTAB method computes two matrix-vector operations per iteration, there is an increased communication overhead in distributed settings. For small to medium-sized problems, the overhead of creating/managing threads may negate some potential speedup. This is of particular relevance for the problem at hand: a 2D ice-sheet model. The Blatter-Pattyn stress balance implies a two-dimensional velocity field, considerably less computationally expensive than the three-dimensional counterpart. On the other hand, CG only computes one matrix-vector multiplication (Shewchuk, 1994), thus decreasing the communication. Even so, both methods show a similar efficiency when parallelization is enabled, suggesting that overhead is not the main cause of marginal speed-up.

Overall, the potential causes for a marginal speed-up in parallel runs mainly concern the solver employed in the linear problem. Both the size of the physical problem and the extremely well optimized memory allocation (as shown in serial runs) yield a rapid decrease in efficiency. For completeness, a similar scalability analysis was also performed for the CG algorithm and results are consistent. The maximum speed-up remains marginal, reaching values of ~ 2.5. This further supports the hypothesis that SpMV is the limiting component when enabling parallelization. We suggest that Eigen simplicity of a row-wise partitioning of the sparse matrix is not enough to yield optimal parallelization. Future work is needed to explore more advanced approaches of the partitioning strategy.

As a conclusion, the discussion herein presented emphasizes that Nix ice sheet model is designed to study a complex system with minimum resources: within a few hours, a 100-metres-resolution Blatter-Pattyn stress balance coupled with full thermodynamics is feasible even on a regular laptop. The discussion section of the manuscript version has been updated to reflect the causes explored by the authors.

References

Petitet, A., and Dongarra, J.J. (1999). Algorithmic Redistribution Methods for Block-Cyclic Decompositions. IEEE Trans. Parallel Distributed Syst., 10, 1201-1216.

M. Aboelaze, N. Chriso choides, and E. Houstis. The Parallelization of Level 2 and 3 BLAS Op erations on Distributed Memory Machines. Technical Rep ort CSD-TR-91-007, Purdue University, West Lafayette, IN, 1991.

U. Çatalyürek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. IEEE Trans. Parallel Dist. Systems, 10(7):673–693, 1999.

Shewchuk, J. R. (1994). An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Technical Report. Carnegie Mellon University, USA.

Schoof, C. (2007), Ice sheet grounding line dynamics: Steady states, stability, and hysteresis, J. Geophys. Res., 112, F03S28, `https://doi.org/10.1029/2006JF000664`, 2007.

Fischler, Y., Rückamp, M., Bischof, C., Aizinger, V., Morlighem, M., and Humbert, A.: A scalability study of the Ice-sheet and Sea-level System Model (ISSM, version 4.18), Geosci. Model Dev., 15, 3753–3771, `https://doi.org/10.5194/gmd-15-3753-2022`, 2022.

Wolf, Michael M., Bruce Hendrickson and Erik G. Boman.: Optimizing parallel sparse matrix-vector multiplication by partitioning, 2008.