Micromechanics Simulations Coupling the deal.II Software Library With a Parallel FETI-DP Solver

S. Köhler, O. Rheinbach, and S. Sandfeld

1 Introduction

We consider adaptive finite elements, using the open source finite element library deal.II [1], and an implementation [11] of the FETI-DP (Finite Element Tearing and Interconnecting Dual–Primal) method based on PETSc, for the solution of problems from dislocation micromechanics. The library deal.II is well known for its adaptive finite element approach based on hanging node constraints. The parallel data structures in deal.II are meant to be used with global parallel matrices, which are assembled across the interface. However, in FETI-DP [6] or BDDC [4] methods, access to the Neumann matrices for each subdomain is needed. Here, we show that the deal.II infrastructure can still be used to efficiently construct the FETI-DP preconditioner. We have reported on first computational results of our approach in [9]; different improvements, including the construction of the coarse space, have been made since. A related implementation of a BDDC method, using adaptive mesh refinement not based on deal.II, has obtained good scalability to up to 2048 cores in [10].

Stephan Köhler · Oliver Rheinbach
Institut für Numerische Mathematik und Optimierung, Fakultät für Mathematik und Informatik, Technische Universität Bergakademie Freiberg, Akademiestr. 6, 09596 Freiberg e-mail: oliver.rheinbach@math.tu-freiberg.de, stephan.koehler@math.tu-freiberg.de, url: http://www.mathe.tu-freiberg.de/nmo/mitarbeiter/oliver-rheinbach
Stefan Sandfeld
Forschungszentrum Jülich, Institute for Advanced Simulation – Materials Data Science and Informatics (IAS-9), Wilhelm-Johnen-Straße, 52428 Jülich and RWTH Aachen University, Chair of Materials Data Science and Materials Informatics, Faculty 5, 52072 Aachen

2 Micromechanical Model Problem

To compute the stresses associated with dislocations within a specimen for the characterization of the microstructure [12, 13], we start by considering a linear elastic model described by

$$\operatorname{div}\sigma = 0, \quad \sigma = \sigma^T, \quad \sigma = C : \varepsilon^{\mathrm{el}}, \quad \text{and} \quad \varepsilon^{\mathrm{el}} = \frac{1}{2}\left(\nabla u + (\nabla u)^T\right)$$

to be solved for the displacements u. Here, σ is the stress tensor, $\varepsilon^{\mathrm{el}}$ the elastic strain tensor, and C the stiffness tensor. Dislocations are one-dimensional defects present in crystalline materials. They are the boundary of a planar area over which two subdomains of a crystal have been displaced relative to each other with the directions given by the Burgers vector $\boldsymbol{b}$.

In the linear elastic context, dislocations may be modeled using an eigenstrain approach [5] by expressing the total strain by $\varepsilon^{\mathrm{tot}} = \varepsilon^{\mathrm{el}} + \varepsilon^{\mathrm{eig}}$, where $\varepsilon^{\mathrm{eig}}$ is the eigenstrain contribution caused by the dislocation microstructure. The area enclosed by a dislocation is described by an orthogonal vector $\boldsymbol{A}$. The eigenstrain contributions $\mathrm{d}\varepsilon^{\mathrm{eig}} = \frac{1}{2}(\boldsymbol{b} \otimes \mathrm{d}\boldsymbol{A} + \mathrm{d}\boldsymbol{A} \otimes \boldsymbol{b})$, where $\otimes$ denotes the outer product, are regularized using the non-singular formulation proposed in [3], similarly to [7]. The eigenstrain of a dislocation is a contribution to the body force term occurring in the elasticity problem.

As a benchmark problem, we chose an artificial dislocation structure which, however, reflects already many details of realistic microstructures that can be found in dislocation simulations. First of all, the considered sample is a cubic box with edge lengths of 1 µm; see also section 5. Single crystalline copper was used as a material, which has the anisotropic elastic constants $C_{11} = 168.4\,\mathrm{GPa}$, $C_{12} = 121.4\,\mathrm{GPa}$, and $C_{44} = 75.4\,\mathrm{GPa}$. Copper is a material that has a "face centered cubic" crystallographic structure with 12 possible slip systems on which dislocations can nucleate and move. In this artificial dislocation microstructure, all 28 dislocations are considered to be closed, circular loops; their center points and radii have been chosen randomly.

3 Parallel mesh handling in deal.II

For simplicity, let us first consider a domain $\Omega \subset \mathbb{R}^2$ decomposed into two subdomains $\Omega_1 \subset \Omega$ and $\Omega_2 \subset \Omega$; see Figure 1. In deal.II, each cell is owned by exactly one MPI rank, the *locally owned cells*. Each MPI rank has information about its locally owned cells and one additional layer of *ghost cells* of the neighboring subdomains; see Figure 2.

The degrees of freedom (dofs) have a global numbering. Each dof belongs to exactly one MPI rank; all dofs belonging to an MPI rank form the *locally owned dofs* of this rank. Each locally owned dof belongs to a locally owned cell, but some dofs of a locally owned cell may belong to the locally owned dofs of a different rank; see

Figure 3. The union of all dofs of all locally owned cells is called *locally active dofs*. The union of the locally active dofs and the degrees of freedom of the cells of the ghost layer is called *locally relevant dofs*; for details, see, e.g., [2] and Figure 3.

4 Subdomain Neumann matrices in deal.II

In deal.II, the global stiffness matrix K can be assembled by an instance of the class `AffineConstraints`, which also handles the hanging node constraints and the Dirichlet boundary values.

For nonoverlapping domain decomposition methods, such as FETI-DP and BDDC methods, we have to assemble the local subdomain stiffness matrices $K^{(i)}$ for each subdomain Ω_i, $i = 1, \ldots, N$. These local subdomain matrices are not assembled across the interface. There is currently no built-in support in the deal.II library for this operation. We have therefore added a layer on top of deal.II to implement the necessary functionality.

Computing the local subdomain Neumann matrices

To assemble a local stiffness matrix, $K^{(i)}$, we need a local sparsity pattern, the local constraints and a local numbering $1, \ldots, n_i$ of all locally relevant dofs of this subdomain. We construct a local sparsity pattern and the local constraints from the global ones by copying the entries and the values with respect to the local numbering. The Dirichlet boundary needs some special care; see section 4.

Computing the interface, and faces, edges, and vertices

In FETI-DP and related methods, the interface and its decomposition into faces, edges, and vertices are needed. The dofs of the interface can be computed as the intersection of the locally active dofs and the locally relevant dofs. But, due to the hanging node constraints, such an index set is not always appropriate for FETI-DP methods, where we need to introduce Lagrange multipliers on the interface. For hanging nodes, we therefore replace the hanging node dofs by those non-hanging node dofs which constrain them; see, e.g., Figure 4.

We denote the interface dofs, as outlined above, of Ω_i by Γ_i and name them as *locally active interface* dofs. Let us remark that not all dofs on the geometric interface belong to Γ_i and, vice versa, see, e.g., Figure 5.

Fig. 1: Left: Domain Ω. **Right:** Decomposition into Ω_1 and Ω_2; | Interface.

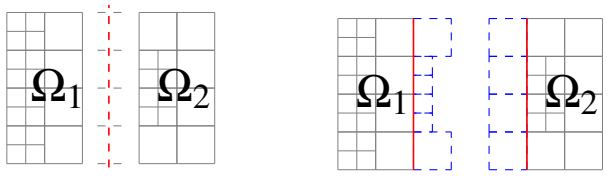

Fig. 2: Left: Locally owned cells of Ω_1, Ω_2. **Right:** Locally owned cells; - - ghost cells.

Fig. 3: An example of the classification of the degrees of freedom with Q_1 elements. **Left:** Subdomain Ω_1: • locally owned dofs; ∘ locally active dofs. Subdomain Ω_2: • locally owned dofs; ∘ locally active dofs. **Right:** Subdomain Ω_1: • locally relevant dofs; Subdomain Ω_2: ∘ locally relevant dofs.

Fig. 4: Partition into locally inner and locally active interface dofs. **Left:** Subdomain Ω_1: • locally inner dofs. Subdomain Ω_2: • locally inner dofs. **Right:** Subdomain Ω_1: • locally active interface dofs. Subdomain Ω_2: • locally active interface dofs.

For $\Omega \subset \mathbb{R}^3$, we compute vertices, edges, and faces as follows: The basic idea is to compute the faces of all subdomains, after that, we build edges as intersection of faces and vertices as intersection of edges. Let us remark that, although the computation of faces is completely local to all MPI ranks, and, therefore, also the computation of edges and vertices, we need to communicate all computed edges and vertices to the neighboring subdomains, see, Figure 5; here, "neighboring" means subdomains which have a non-empty intersection of the locally relevant dofs; due to hanging node constraints, the result can be counterintuitive; see Figure 5 for the case of two dimensions.

The use of p4est (based on space filling curves), which is standard in deal.II's parallel distributed mesh class, does not guarantee that a subdomain is connected, and it may only be connected through vertices or edges of cells. This can be dealt with but it will typically increase the coarse problem size.

Fig. 5: Domain Ω partitioned into Ω_1, Ω_2 and Ω_3 with edge dofs between the subdomains. Only subdomain Ω_2 computes the vertices as intersection of the edges. These vertices are not seen by Ω_1 and Ω_3. **Left:** Domain Ω | interface. **Right:** • edge dofs between Ω_1 and Ω_2. ◇ edge dofs between Ω_2 and Ω_3.

Denote the locally active interface of Ω_i by Γ_i and the locally owned dofs by I_i^o.

1. Decompose Γ_i into $\Gamma_i^o = \Gamma_i \cap I_i^o$ and $\Gamma_i^{o,C}$
2. Determine the number of rows m of B_i:
   (a) Compute the number $s_\lambda^{(i)}$ of all multipliers related to Γ_i^o.
   (b) Compute $m = \sum_i^N s_\lambda^{(i)}$.
3. Send the index set $\Gamma_i^{o,C}$ to all neighboring subdomains.
4. For each received set $\Gamma_j^{o,C}$ from a neighboring subdomain Ω_j compute $\Gamma_{i,j}^o = \Gamma_i^o \cap \Gamma_j^{o,C}$. Send to subdomain Ω_j the information for which multipliers, associated with the dofs in $\Gamma_{i,j}^o$, a $+1$ or -1 has to be inserted into B_j.

Fig. 6: Construction of the local jump operators B_i.

Furthermore, a subdomain may not have enough vertices or edges to ensure the invertibility of certain subdomain matrices in FETI-DP methods. Here, we sometimes need to introduce additional primal constraints by subdivision of faces or edges to constrain the low energy modes of all components of a subdomain. Let us remark that our method can still lead to faces or edges that are not connected.

As mentioned in section 4, we have to take care of the Dirichlet boundary condition. These are also handled by the `AffineConstraints` class, as the hanging node constraints. Therefore, for the computation of the interface, we need to extract the information about the hanging nodes dofs from an instance where the Dirichlet boundary condition have not been set.

Construction of the FETI-DP jump operator

A crucial element of FETI-DP methods is the jump operator B which imposes the continuity of the solution. This operator has a row for each Lagrange multiplier and each row consists of exact two entries, a $+1$ and a -1.

The Lagrange multipliers are related to the locally active interface dofs. Hence, we partition them, and manage the computation of the local parts of B, by the locally owned part of the interface, see, Figure 6.

5 Numerical Results

We use Q1 finite elements. The deal.II library uses the p4est library to compute the domain decomposition, as in [10]. Our FETI-DP implementation is based on [11, 8]. Our coarse space uses vertices, edges, and, certain additional point constraints on faces. We perform 5 mesh refinement steps, using the Kelly error estimator. We use GMRES.

In Figure 7 (left) we show the eigenstrain distributions that are non-zero inside the loops and zero outside. This quick transition of the eigenstrain value is responsible for very high (for non-regularized problems: diverging) stresses that require a sufficiently fine mesh for obtaining an accurate solution.

In Tables 1 and 2, we report on the global problem size ("Global") the size of the coarse problem ("Coarse"), the number of Krylov iterations ("it."), the solver and assembly time ("solve" and "ass."). We also report timings to build the interface Γ, to build faces, edges, and vertices (denoted "f/e/v"), and to build the FETI-DP jump operator B.

First, we observe that the deal.II infrastructure can provide, within a fraction of a second (for the smaller problems) to a few seconds (for the larger problems), the necessary connectivity information to construct the FETI-DP preconditioner, i.e., the interface, the face, edges, and vertices, and the information to build the B-matrix.

We also observe that the number of iterations increases slightly when refining the mesh. Note that the problem is anisotropic (see section 2) which results in higher iteration counts compared to standard benchmark problems.

For the same refinement cycle, the problem sizes for 512 cores are larger by a factor between 7 and 8 compared with 64 cores. Since the number of cores is larger by a factor of 8, we can roughly compare the timings for 512 cores and 64 cores in the sense of weak scaling. In this sense, when comparing refinement step 5, we observe acceptable parallel scalability for the solver time (29.8s vs. 18.8s) and the total time (203s vs. 174s). This is also the case when summing the total time over all refinement steps, i.e., we have 305s (512 cores) and 237s (64 cores). Since this is not weak scalability in the strict sense, we refrain from providing parallel efficiency numbers. Note that the assembly does not scale perfectly since a certain load imbalance is introduced by the additional computations involved with the dislocations.

Performing the same computations using a larger number of cores, i.e., 216 and 1728 cores, we see that the solver time starts to be dominated by the coarse solver, since the coarse problem is quite large, i.e., $> 70\,000$ dof for the last two refinement cycles. This is also a result of our attempts to create a robust coarse space. As a result of the deteriorating solver scalability, the total time to solution, summed over all refinement steps, is 510s (1728 cores) to be compared with 277s (216 cores). This indicates that we need to reduce the coarse problem size by modifying our coarse space. Alternatively, we can move to a three- or multi-level method as in [10].

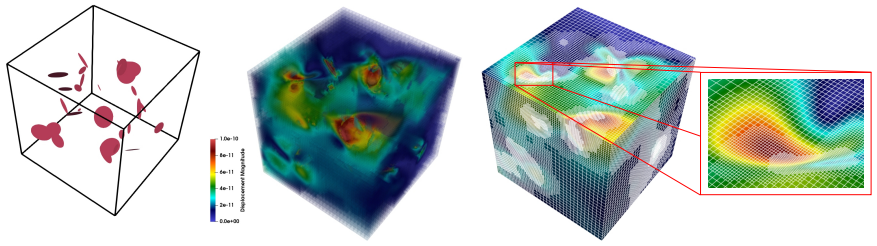

Fig. 7: Left: eigenstrain resulting from 28 dislocation loops (red color denotes a non-zero eigenstrain). **Middle:** Solution. **Right:** Solution and adaptive mesh for the fifth refinement cycle; problem size: 10.4 million dofs using 512 MPI ranks.

Table 1: Results for 5 refinement cycles on 64 cores and 512 cores.

		d.o.f.			Time in s					
#Cores	Refinem.	Global	Coarse	it.	solve	ass.	build Γ	f/e/v	build B	total time
64	1	14 739	405	23	0.16	9.41	< 0.01	0.01	< 0.01	10.0
	2	49 173	1 569	39	0.54	12.6	0.06	0.03	0.02	13.3
	3	153 420	1 680	46	1.23	20.3	0.24	0.03	0.01	22.1
	4	476 502	1 785	58	4.19	47.6	1.06	0.06	0.03	53.6
	5	1 475 034	1 896	55	18.8	150	3.80	0.19	0.06	174
	sum				24.9	240	5.17	0.32	0.13	273
512	1	107 811	4 557	22	1.39	5.07	0.02	0.01	0.02	6.94
	2	349 404	16 842	49	4.84	9.25	0.10	0.04	0.03	14.6
	3	1 087 689	19 275	51	6.57	15.3	0.23	0.05	0.06	22.8
	4	3 353 052	20 604	56	10.6	45.2	0.94	0.11	0.12	58.0
	5	10 358 751	22 254	53	29.8	165	4.59	0.22	0.23	203
	sum				53.2	240	5.88	0.86	0.46	305

Table 2: Results for 5 refinement cycles on 216 cores and 1728 cores.

		d.o.f.			Time in s					
#Cores	Refinem.	Global	Coarse	it.	solve	ass.	build Γ	f/e/v	build B	total time
216	1	46 875	1 725	23	0.38	7.53	0.01	0.01	0.01	8.12
	2	153 801	6 492	43	3.07	10.1	0.07	0.03	0.02	13.5
	3	479 475	7 701	44	3.63	16.9	0.27	0.05	0.03	21.3
	4	1 477 617	7 839	52	7.29	41.6	0.91	0.08	0.06	50.7
	5	4 570 413	8 139	54	23.9	152	4.90	0.17	0.11	183
	sum				46	223	6.16	0.33	0.22	277
1728	1	352 947	17 061	22	6.56	4.39	0.02	0.02	0.07	11.8
	2	1 133 949	61 266	50	32.1	8.05	0.10	0.05	0.11	41.6
	3	3 509 349	67 716	49	35.9	14.1	0.25	0.07	0.18	52.0
	4	10 820 382	71 811	63	47.1	61.4	0.79	0.11	0.34	112
	5	33 427 005	76 125	60	76.9	207	3.78	0.22	0.68	292
	sum				199	295	4.94	0.47	1.34	510

Acknowledgements The authors acknowledge computing time on the Compute Cluster of the Fakultät für Mathematik und Informatik of Technische Universität Freiberg (DFG project number 397252409), operated by the university computing center (URZ). The first and second author would like to thank Guido Kanschat and Daniel Arndt for the fruitful discussions on the deal.II data structures. The third author acknowledges financial support from the European Research Council

through the ERC Grant Agreement No. 759419 MuDiLingo (“A MultiscaleDislocation Language for Data-Driven Materials Science”)

References

1. Arndt, D., Bangerth, W., Blais, B., Clevenger, T.C., Fehling, M., Grayver, A.V., Heister, T., Heltai, L., Kronbichler, M., Maier, M., Munch, P., Pelteret, J.P., Rastak, R., Thomas, I., Turcksin, B., Wang, Z., Wells, D.: The `deal.II` library, version 9.2. J. Numer. Math. **28**(3), 131–146 (2020). DOI 10.1515/jnma-2020-0043
2. Bangerth, W., Burstedde, C., Heister, T., Kronbichler, M.: Algorithms and data structures for massively parallel generic adaptive finite element codes. ACM Transactions on Mathematical Software (TOMS) **38**(2), 14 (2011)
3. Cai, W., Arsenlis, A., Weinberger, C.R., Bulatov, V.V.: A non-singular continuum theory of dislocations. J. Mech. Phys. Solids **54**(3), 561–587 (2006). DOI 10.1016/j.jmps.2005.09.005
4. Dohrmann, C.R.: A preconditioner for substructuring based on constrained energy minimization. SIAM J. Sci. Comput. **25**(1), 246–258 (2003). DOI 10.1137/S1064827502412887
5. Eshelby, J.D., Peierls, R.E.: The determination of the elastic field of an ellipsoidal inclusion, and related problems. Pro. of the Royal Society of London. Series A. Math. and Phy. Sciences **241**(1226), 376–396 (1957). DOI 10.1098/rspa.1957.0133
6. Farhat, C., Lesoinne, M., LeTallec, P., Pierson, K., Rixen, D.: FETI-DP: A Dual–Primal Unified FETI Method, part I: A faster alternative to the two-level FETI method. Internat. J. Numer. Methods Engrg. **50**(7), 1523–1544 (2001). DOI 10.1002/nme.76
7. Jamond, O., Gatti, R., Roos, A., Devincre, B.: Consistent formulation for the discrete-continuous model: Improving complex dislocation dynamics simulations. International Journal of Plasticity **80**, 19–37 (2016). DOI 10.1016/j.ijplas.2015.12.011
8. Klawonn, A., Rheinbach, O.: A parallel implementation of dual-primal FETI methods for three-dimensional linear elasticity using a transformation of basis. SIAM J. Sci. Comput. **28**(5), 1886–1906 (2006). DOI 10.1137/050624364
9. Köhler, S., Rheinbach, O., Sandfeld, S., Steinberger, D.: FETI-DP Solvers and Deal. II for Problems in Dislocation Mechanics. PAMM **19**(1), e201900292 (2019). DOI 10.1002/pamm.201900292
10. Kůs, P., Šístek, J.: Coupling parallel adaptive mesh refinement with a nonoverlapping domain decomposition solver. Advances in Engineering Software **110**, 34–54 (2017). DOI 10.1016/j.advengsoft.2017.03.012
11. Rheinbach, O.: Parallel iterative substructuring in structural mechanics. Arch. Comput. Methods Eng. **16**(4), 425–463 (2009). DOI 10.1007/s11831-009-9035-4
12. Sandfeld, S., Monavari, M., Zaiser, M.: From systems of discrete dislocations to a continuous field description: stresses and averaging aspects. Modelling and Simulation in Materials Science and Engineering **21**(8), 085006 (2013). DOI 10.1088/0965-0393/21/8/085006
13. Steinberger, D., Gatti, R., Sandfeld, S.: A Universal Approach Towards Computational Characterization of Dislocation Microstructure. JOM **68**(8), 2065–2072 (2016). DOI 10.1007/s11837-016-1967-1