DisCoTec: Distributed higher-dimensional HPC simulations with the sparse grid combination technique

**Theresa Pollinger** [3,1,¶], **Marcel Hurler**[1], **Alexander Van Craen** [1], **Michael Obersteiner**[2], **and Dirk Pflüger** [1]

1 University of Stuttgart, Scientific Computing, Stuttgart, Germany **2** Technical University of Munich, Chair of Scientific Computing, Munich, Germany **3** RIKEN Center for Computational Science (R-CCS), Kobe, Japan ¶ Corresponding author

Summary

DisCoTec is a C++ framework for the sparse grid combination technique (Griebel et al., 1992), designed for massively parallel settings. It is implemented with shared-memory parallelism via OpenMP and distributed-memory parallelism via MPI, and is intended to be used in conjunction with existing simulation codes. For simulation codes that can handle nested structured grids, little to no adaptation work is needed for use with the DisCoTec framework. The combination technique with DisCoTec demonstrates its superiority in precision-per-memory for higher-dimensional time-dependent simulations, such as high-fidelity plasma turbulence simulations in four to six dimensions and even for simulations in two dimensions, improvements can be observed (Pollinger et al., 2023).

A central part of the combination technique at scale is the transformation of grid coefficients into a multi-scale basis. DisCoTec provides a selection of three different lifting wavelets for this purpose: hierachical hat basis, biorthogonal, and fullweighting basis. In addition, any code that can operate on nested structured grids can benefit from the model order reduction provided by the underlying sparse grid approach used by DisCoTec, without requiring any multi-scale operations. An additional feature of DisCoTec is the possibility of performing widely-distributed simulations of higher-dimensional problems, where multiple High-Performance Computing (HPC) systems collaborate to solve a joint simulation, as demonstrated in Pollinger et al. (2024). Thus, DisCoTec can leverage the compute power and main memory of multiple HPC systems, with comparatively low and manageable transfer costs due to the combination technique.

Statement of need

Higher-dimensional problems (by which we mean more than three space dimensions and one time dimension) quickly require infeasible amounts of computational resources such as memory and core-hours as the problem size increases—they are haunted by the so-called 'curse of dimensionality'. An example of this are high-fidelity plasma turbulence simulations in the field of confined fusion research. Currently employed approaches to this problem include dimensionally-reduced models, such as gyrokinetics (Brizard & Hahm, 2007) (which may not always be applicable), particle-in-cell methods (which suffer from inherent noise (Verboncoeur, 2005)), and restricting computations to a very limited resolution. A further—still developing but very promising—approach to the problem is low-rank methods (Einkemmer & Joseph, 2021). Multi-scale (hierarchical) methods, such as the sparse grid combination technique (CT) that DisCoTec employs, provide an alternative approach to addressing the curse of dimensionality by considering only those resolutions where the highest amount of information

is expected (Bungartz & Griebel, 2004). While some implementations of the CT are available, there is currently no other implementation for parallel simulations that require distributed computing. DisCoTec is a C++ framework for massively-parallel time-dependent problems with the CT, which fills this gap.

Methods: Sparse grid combination technique and implementation

The sparse grid combination technique (with time-stepping) is a multi-scale approach for solving higher-dimensional problems. Instead of solving the problem on one grid that is very finely resolved in all dimensions, the problem is solved on the so-called 'component grids' that are all rather coarsely resolved—each of them differently in the different dimensions. For instance, the following schematic shows a two-dimensional combination scheme, consisting of seven component grids.
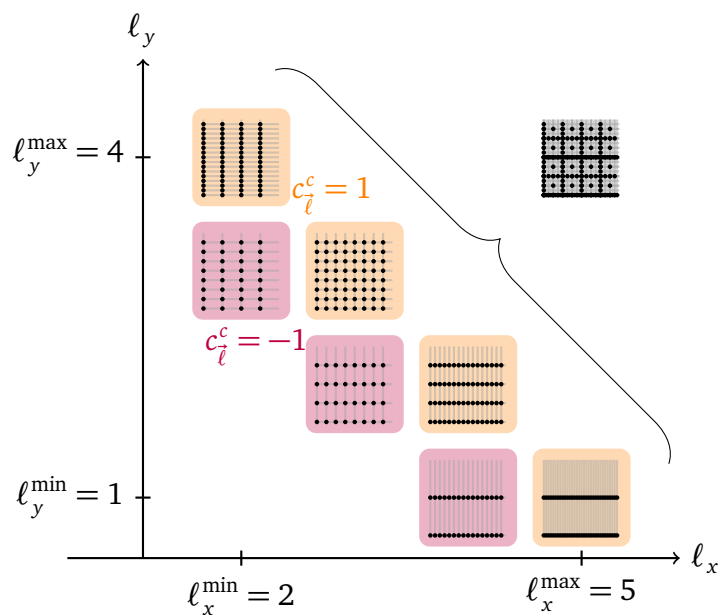

Figure 1: Combination scheme in two dimensions with $\vec{\ell}^{\text{min}} = (2, 1)$ and $\vec{\ell}^{\text{max}} = (5, 4)$, periodic boundary conditions. Figure first published in Pollinger (2024).

By updating each other's information throughout the simulation, the component grids still obtain an accurate solution of the overall problem (Griebel et al., 1992). This is enabled by an intermediate transformation into a multi-scale (hierarchical) basis, and application of the combination formula

$$f^{(\mathsf{s})} = \sum_{\vec{\ell} \in \mathcal{I}} c_{\vec{\ell}} f_{\vec{\ell}}$$

where $f^{(\mathsf{s})}$ is the sparse grid approximation, and $f_{\vec{\ell}}$ are the component grid functions. The set of all used levels $\vec{\ell}$ is often called a combination scheme $\mathcal{I}$. In Figure 1, the coefficients $c_{\vec{\ell}}$ are -1 for the coarser component grids (red background) and 1 for the finer component grids (orange background). In summary, each of the grids will run (one or more) time steps of the simulation, then exchange information with the other grids, and repeat this process until the simulation is finished.

DisCoTec provides the necessary infrastructure for the combination technique with a black-box approach, enabling massive parallelism—suitable for existing distributed solvers that use structured grids. An important feature is the usage of 'process groups', where multiple MPI

ranks will collaborate on a set of component grids, and the solver's existing parallelism can be re-used. The process groups are displayed as pg_i in Figure 2.
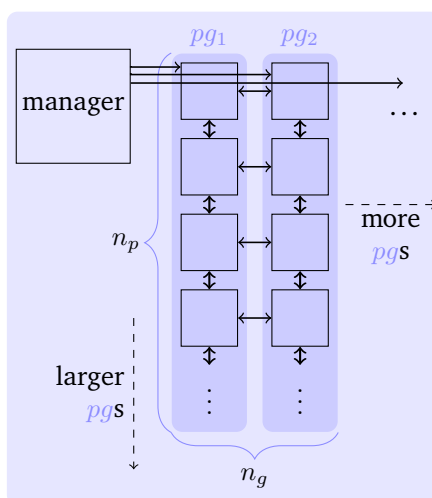

Figure 2: DisCoTec process groups: Each black square denotes one MPI rank. The ranks are grouped into the so-called 'process groups'. Distributed operations in DisCoTec require either communication in the process group, or perpendicular to it—there is no need for global communication or synchronization, which avoids a major scaling bottleneck. The manager rank is optional. Figure first published in Pollinger (2024).

In addition, the number of process groups can be increased to leverage the combination technique's embarrassing parallelism in the solver time steps. In Figure 2, this would be equivalent to adding more and more process groups to the right.

Using DisCoTec, kinetic simulations were demonstrated to scale up to hundreds of thousands of CPU cores (Pollinger, 2024). By putting a special focus on saving memory, most of the memory is available for use by the black-box solver, even at high core counts. In addition, OpenMP parallelism can be used to further increase parallelism while being more lightweight than MPI in terms of memory.

Through highly parallel I/O operations, DisCoTec can be used to perform simulations on multiple High Performance Computing systems simultaneously, if there exists a tool for sufficiently fast file transfer between the systems (Pollinger, 2024). The DisCoTec repository contains example scripts and documentation for utilizing UFTP as an example of a transfer tool, but the approach is not limited to UFTP.

DisCoTec provides a conveniently automated way of installation using a spack package (Gamblin et al., 2015), which can be used to install DisCoTec and its whole dependency tree in an automated manner optimized for HPC hardware.

State of the field

Besides DisCoTec there exist other frameworks that allow the usage of sparse grids and the combination technique. We will give a brief overview and outline the differences and application areas of the codes.

The C++ code SG++ (SG++ development team, 2018) provides a direct interface to sparse grids and applying them to a variety of different tasks such as interpolation, quadrature, optimization, PDEs, regression, and classification. With the help of wrappers, the framework can be used from various other programming languages such as Python and Matlab. The

code targets direct implementations within sparse grids and provides a basic implementation of the combination technique. Although offering parallelization for some of the tasks, the code mainly targets single-node computations.

The `Sparse Grids Matlab Kit` (Tamellini et al., 2024) by Piazzola and Tamellini was originally designed for teaching purposes and uncertainty quantification with the combination technique (Piazzola & Tamellini, 2024). It offers a user friendly MATLAB interface for the combination technique. In addition, dimensional adaptivity is available for nested and non-nested sequences of component grid collocation points. The code is designed for usage on a single node which limits the parallelism to shared memory.

The `sparseSpACE` (Obersteiner, 2019) project offers different variants of the combination technique including a spatially adaptive combination technique. It provides implementations for various applications such as numerical integration, interpolation, uncertainty quantification, sparse grid density estimation (for classification and clustering), regression, and PDE calculations. The code is completely written in Python and is mostly sequential. The main novelty of this project is the possibility to add spatial adaptivity to the combination technique.

This demonstrates that there exist multiple codes for sparse grids and the combination technique. However, `DisCoTec` is the only code that offers distributed parallelization with the combination technique and has demonstrated that it can scale up to full supercomputers and beyond. In addition, `DisCoTec` uses the most sophisticated approach to utilize the combination technique with time-dependent PDEs by employing recombinations, which increases the overall numerical accuracy (Pollinger et al., 2023).

Acknowledgements

References

Brizard, A. J., & Hahm, T. S. (2007). Foundations of nonlinear gyrokinetic theory. *Reviews of Modern Physics*, *79*(2), 421. https://doi.org/10.1103/revmodphys.79.421

Bungartz, H.-J., & Griebel, M. (2004). Sparse grids. *Acta Numerica*, *13*, 147–269.

Einkemmer, L., & Joseph, I. (2021). A mass, momentum, and energy conservative dynamical low-rank scheme for the Vlasov equation. *Journal of Computational Physics*, *443*, 110495. https://doi.org/10.1016/j.jcp.2021.110495

Gamblin, T., LeGendre, M., Collette, M. R., Lee, G. L., Moody, A., Supinski, B. R. de, & Futral, S. (2015). *The Spack package manager: Bringing order to HPC software chaos*. 1–12. https://doi.org/10.1145/2807591.2807623

Griebel, M., Schneider, M., & Zenger, C. (1992). A combination technique for the solution of sparse grid problems. In P. de Groen & R. Beauwens (Eds.), *Iterative Methods in Linear Algebra* (pp. 263–281). IMACS, Elsevier, North Holland. https://ins.uni-bonn.de/media/public/publication-media/griesiam.ps.gz

Obersteiner, M. (2019). sparseSpACE - the sparse grid spatially adaptive combination environment. In *GitHub repository*. GitHub. https://github.com/obersteiner/sparseSpACE

Obersteiner, M. (2021). *A spatially adaptive and massively parallel implementation of the fault-tolerant combination technique* [PhD thesis, Technische Universität München]. https://mediatum.ub.tum.de/doc/1613369/1613369.pdf

Piazzola, C., & Tamellini, L. (2024). Algorithm 1040: The sparse grids Matlab kit - a Matlab implementation of sparse grids for high-dimensional function approximation and uncertainty quantification. *ACM Transactions on Mathematical Software*, *50*(1). https://doi.org/10.1145/3630023

Pollinger, T. (2024). *Stable and mass-conserving high-dimensional simulations with the sparse grid combination technique for full HPC systems and beyond* [PhD thesis]. https://doi.org/10.18419/opus-14210

Pollinger, T., Craen, A. V., Offenhäuser, P., & Pflüger, D. (2024). *Realizing joint extreme-scale simulations on multiple supercomputers—two superfacility case studies*. 1568–1584. https://doi.org/10.1109/SC41406.2024.00104

Pollinger, T., Rentrop, J., Pflüger, D., & Kormann, K. (2023). A stable and mass-conserving sparse grid combination technique with biorthogonal hierarchical basis functions for kinetic simulations. *Journal of Computational Physics*, 112338. https://doi.org/10.1016/j.jcp.2023.112338

SG++ development team. (2018). SGpp. In *GitHub repository*. GitHub. https://github.com/SGpp/SGpp

Tamellini, L., Piazzola, C., Nobile, F., Sprungk, B., Porta, G., Guignard, D., & Tesei, F. (2024). *The sparse grids Matlab kit*. https://sites.google.com/view/sparse-grids-kit/home

Verboncoeur, J. P. (2005). Particle simulation of plasmas: Review and advances. *Plasma Physics and Controlled Fusion*, *47*, A231. https://doi.org/10.1088/0741-3335/47/5a/017