EXAHD: An Exa-scalable Two-Level Sparse Grid Approach for Higher-Dimensional Problems in Plasma Physics and Beyond

Dirk Pflüger[1], Hans-Joachim Bungartz[2], Michael Griebel[3], Frank Jenko[4], Tilman Dannert[5], Mario Heene[1], Christoph Kowitz[2], Alfredo Parra Hinojosa[2], and Peter Zaspel[3]

[1] Institute for Parallel and Distributed Systems, University of Stuttgart, Germany
[2] Chair of Scientific Computing, Technische Universität München, Germany
[3] Institute for Numerical Simulation, University of Bonn, Germany
[4] Max Planck Institute for Plasma Physics, Germany
[5] Computing Centre of the Max Planck Society and the MPI for Plasma Physics, Germany

Abstract. High-dimensional problems pose a challenge for tomorrow's supercomputing. Problems that require the joint discretization of more dimensions than space and time are among the most compute-hungry ones and thus standard candidates for exascale computing and even beyond. This project tackles such problems by a hierarchical extrapolation approach, the sparse grid combination technique. The method not only enables their treatment in the first place. The hierarchical approach also provides novel ways to deal with central problems in high-performance computing such as scalability and resilience: Global communication can be avoided and reduced to a small subset, and faults can be compensated for without the need for recomputations or checkpoint-restart. As an exemplary prototype for high-dimensional problems, turbulence simulations in plasma physics are studied.

1 Introduction

The emergence of future exascale systems requires the development of new algorithms and software to harness the computational power that will be available in the near future. Classical parallelizations that scale even up to petaflop systems will encounter limits on these "mega-node kilo-core giga-Hertz" architectures [15], and the rise of accelerator cards in HPC further increases the hardware complexity. On these future systems, three main challenges will be scalability, resilience, and load balancing, which are addressed in this project.

High-dimensional mesh-based problems require the joint discretization of more than the classical four dimensions, space and time. Straightforward approaches fully suffer the so-called curse of dimensionality: requiring M degrees of freedom in each dimension, M^d unknowns are required in d dimensions. The effort grows exponentially in the dimensionality, and the need for at least exascale computing becomes obvious even for moderate $d > 3$.

This project exemplarily considers turbulence simulations of hot fusion plasmas, where in the gyrokinetic formulation five dimensions plus time have to be dealt with. Fusion energy is one of the most attractive options to meet the growing global electricity demands in a sustainable and carbon-free way. To achieve this in a controlled way, 100 million degree hot plasma has to be kept away from the reactor walls by means of a strong magnetic field. On the way to the international ITER project, one of the most challenging scientific endeavors ever undertaken in an international joint effort, plasma turbulence simulations play a key role. In our project, time-dependent problems and eigenvalue problems are studied based on the gyrokinetic simulation code GENE.

For high-dimensional problems, a hierarchical approach comes to the rescue, the sparse grid combination technique [6]. It mitigates the curse of dimensionality on the one hand and reduces the number of unknowns in the discretization significantly. On the other hand, it allows one to deal with the exascale challenges mentioned above in a novel, compelling way. It decouples the overal problem into multiple problems of reduced size and breaks the demand for global communication, reducing the synchronization bottleneck significantly. A second level of parallelism is introduced, which offers new approaches to load balancing, and a hierarchical superposition can be exploited to deal with faults.

To achieve the goals of this project, new algorithmic and numerical approaches have to be developed. First, we give a short overview on the problem and the numerical method in Sect. 2. In Sect. 3, which is the core of this work, we describe the state of the art and current developments in our project, followed by an outlook on next steps in Sect. 4.

2 Plasma Physics and the Combination Technique

Besides a description by means of magnetohydrodynamics, plasmas can also be modelled kinetically by the six-dimensional Vlasov equation

$$\frac{\partial g}{\partial t} + \boldsymbol{v} \cdot \frac{\partial g}{\partial \boldsymbol{x}} + \frac{q}{m}(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}) \cdot \frac{\partial g}{\partial \boldsymbol{v}} = C(g). \tag{1}$$

Due to the restricted movement of the plasma particles of charge q and mass m around the magnetic field-lines (gyration), these equations can be reduced to the five-dimensional set of gyrokinetic equations with g representing the distribution in 5D phase-space consisting of three spatial coordinates (x, y and z) and two velocity coordinates $v_{\|}$ (velocity parallel to the magnetic field line) and μ (the magnetic moment). The collisions operator C governs the interaction of particles by collision, which is usually weak compared to the forces induced by the electric and magnetic fields $\boldsymbol{E}$ and $\boldsymbol{B}$ and will thus be neglected.

The gyrokinetic simulation code GENE discretizes the gyrokinetic equations by an Eulerian approach. A five-dimensional Cartesian grid is spanned throughout the domain, where the x and y coordinates are transformed to Fourier space. After discretization and other approximations, the equations implemented in GENE roughly have the structure $\frac{\partial \boldsymbol{g}}{\partial t} = \mathcal{L}(\boldsymbol{g}) + \mathcal{N}(\boldsymbol{g})$, with $\boldsymbol{g}$ representing the

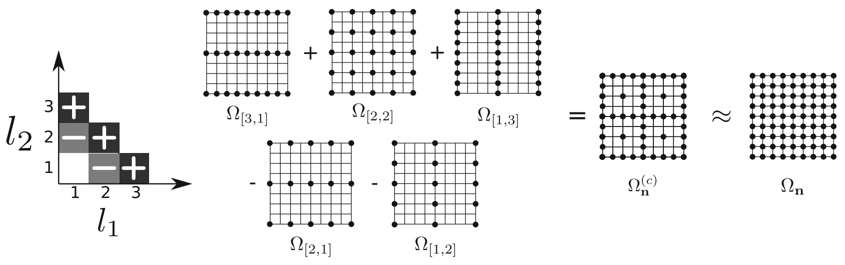


Fig. 1. The combination technique for a two-dimensional problem with $\boldsymbol{n} = [3, 3]^T$ and $\boldsymbol{l}_{\min} = [1, 1]^T$

distribution function discretized on the Cartesian grid. The operators $\mathcal{L}$ and $\mathcal{N}$ act on this vector and comprise all linear and non-linear terms, respectively. For an extensive description of the equations implemented in GENE, we refer to [3]. The linear operator $\mathcal{L}$ already describes the basic behavior of the plasma and allows studies of instabilities and estimates of turbulent transport. It will be used for the tests of the combination technique in this paper.

The combination technique [6] computes a sparse grid approximation of a function $f_{\boldsymbol{n}}$ defined on a regular Cartesian grid $\Omega_{\boldsymbol{n}}$. In general, an anisotropic grid $\Omega_{\boldsymbol{l}}$ can be defined by a level-vector $\boldsymbol{l}$ that determines the uniform mesh-width 2^{-l_k} in dimension $k = 1, \ldots, d$. The combination technique approximation $f_{\boldsymbol{n}}^{(c)} \approx f_{\boldsymbol{n}}$ can then be written as a sum of m full anisotropic Cartesian grids of smaller size, where each grid is weighted with its combination coefficient $c_{\boldsymbol{l}}$,

$$f_{\boldsymbol{n}}^{(c)}(\boldsymbol{x}) = \sum_{\boldsymbol{l} \in \mathcal{I}} c_{\boldsymbol{l}} f_{\boldsymbol{l}}(\boldsymbol{x}) , \tag{2}$$

with $\mathcal{I}$ being the set of level-vectors of the grids used for the combination, see Fig. 1 for an illustration. Here, we consider the space of piecewise d-linear functions and thus interpolate d-linear between the grid points. Different approaches to determine the appropriate combination coefficients $\boldsymbol{c}$ and index set $\mathcal{I}$ exist [12], with

$$f_{\boldsymbol{n}}^{(c)}(\boldsymbol{x}) = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{n},q}} f_{\boldsymbol{l}}(\boldsymbol{x}) \tag{3}$$

being the classical combination technique with the index set [6]

$$\mathcal{I}_{\boldsymbol{n},q} = \{\boldsymbol{l} \in \mathbb{N}^d : |\boldsymbol{l}|_1 = |\boldsymbol{l}_{\min}|_1 + c - q : \boldsymbol{n} \geq \boldsymbol{l} \geq \boldsymbol{l}_{\min}\} , \tag{4}$$

where $\boldsymbol{l}_{\min} = \boldsymbol{n} - c \cdot \mathbf{1}$, $c \in \mathbb{N}_0$ s.th. $l_{\min} \geq \mathbf{1}$, specifies a minimal resolution level in each direction. The hierarchical sparse grid approach thus decomposes a single problem (discretized on a full grid with a high resolution) into multiple smaller, anisotropic problems that can be computed independently and in parallel, and standard solvers working on anisotropic grids can be employed.

Bringing the combination technique and GENE together required only minor modifications of GENE, specifically, slightly shifting and stretching the original

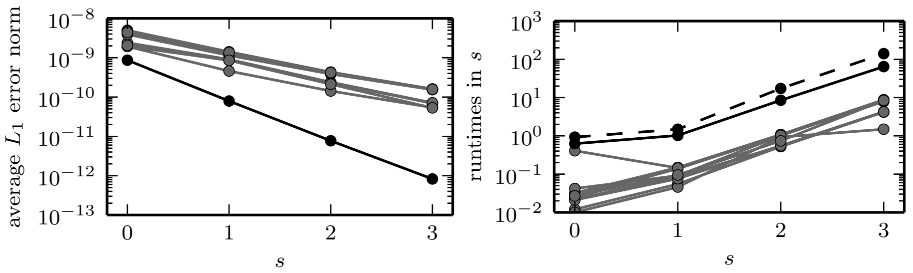


Fig. 2. Left: the error of the combined solution $f_{\boldsymbol{n}}^{(c)}$ compared to the reference solution $f_{\boldsymbol{n}}$ (*black*) and the GENE results $f_{\boldsymbol{l}}$ computed on $\Omega_{\boldsymbol{l}}$ with $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{n}}$ (*gray*). The error is the norm of the L_1 error normalized by the number of unknowns. **Right**: the computation time of the combined solution (*black*), of each partial solution (*gray*) and of the full grid solution (*dashed*). Obtaining the combined solution only requires half of the time compared to the full grid solution.

GENE grid in each dimension. We apply the combination technique separately for the real part and the imaginary part of GENE's complex-valued output.

GENE provides test cases of typical application scenarios, the simplest being the linear simulation of an unstable ion-temperature-gradient (ITG) mode [18]. To demonstrate the feasibility, we use for each $\Omega_{\boldsymbol{l}}$ the same physical parameters and time-steps and combine the result once in the end, and refer to [16] for other scenarios that we have studied. In order to test the combination technique for different resolutions, we varied $\boldsymbol{l}_{\min}$ and $\boldsymbol{n}$ according to

$$\boldsymbol{n} = \boldsymbol{l}_{\min,s} + [2,2,2]^T \quad \boldsymbol{l}_{\min,s} = [2,3,2]^T + s \cdot [1,1,1]^T \quad s \in \{0,1,2,3\}\,,$$

with $\boldsymbol{l}$ being the level vector for dimensions $(\mu, v_{\|}, z)$. The resolution of x and y was fixed to 1. In Fig. 2 one can see that the combined solution is actually close to the solution on the reference grid $\Omega_{\boldsymbol{n}}$, and that each of the partial solutions has a much higher error than the combined solution. Already for this rather small setup on a desktop system, the combination technique retrieves an approximation of the reference solution with a diminished runtime. Larger GENE runs are addressed in Sec. 3.1.

3 Exa-Challenges and -Solutions

In the following, we address the challenges that we face towards exascale simulations which will be required for full-scale simulations of the numerical ITER fusion experiment. As the project studies all aspects that are required, this reaches from load balancing, scalability, and resilience via the usage of hybrid parallelizations up to novel numerical schemes.

3.1 Load Balancing

Achieving full scalability with the combination technique on an exascale system requires effective load balancing. The anisotropy in the discretization of the partial solutions influences the convergence rate and stability of the underlying numerical solvers. This results in larger numbers of iterations and enforces smaller time-step sizes for very anisotropic discretizations compared to more isotropic ones. For our application code GENE, the anisotropy additionally influences the efficiency of the parallelization. We measured a difference in execution time of more than a factor of three for partial solutions computed with GENE with roughly the same number of unknowns.

The combination technique enables two levels of parallelization: on the coarse level, the individual partial solutions can be computed independently of each other in parallel. On the fine level, each partial solution can be solved in parallel using the parallelization concept of the application, see Sect. 3.4 for the latter one. In order to exploit the two-level parallelization, we use a manager-worker concept to distribute the partial solutions onto the available number of nodes of an HPC system. This concept has already successfully been used for the combination technique in [5]. A manager process distributes the partial solutions to a number of process groups using MPI.

In order to minimize the total runtime by optimally distributing the partial solutions onto the process groups, we have developed a load model [9] which predicts the execution time of a partial solution. The two parameters used for the model are the number of unknowns of the partial solution, $N := 2^{|\boldsymbol{l}|_1}$, and the anisotropy $\boldsymbol{s_l}$, with $s_{\boldsymbol{l},i} = \frac{l_i}{|\boldsymbol{l}|_1}$, of the corresponding grid $\Omega_{\boldsymbol{l}}$. It holds $|\boldsymbol{s_l}|_1 = 1$. Thus, a high value in one dimension will result in a low value in at least one of the other dimensions. For a perfectly isotropic grid it holds $s_{\boldsymbol{l},i} = \frac{1}{d}$. With this notation we can express the anisotropy of the grid completely decoupled from the number of grid points. Our load model then has the form

$$t(\boldsymbol{l}) = t(N, \boldsymbol{s_l}) = r(N)h(\boldsymbol{s_l}). \qquad (5)$$

The function $r(N)$ models the dependence of the execution time of a partial solution on the number of unknowns. The value provided by $r(N)$ is scaled by the function $h(\boldsymbol{s_l})$, which solely depends on the anisotropy of the discretization. The parameters of $r(N)$ and $h(\boldsymbol{s_l})$ are determined by fitting the functions to measurement data in the least squares sense.

Figure 3 shows, for different numbers of process groups, the predicted parallel efficiency E_p for the anisotropy model (AM) in comparison to a simple linear model (LM) that depends only on the number of unknowns. The predictions are based on measured execution times of other partial solutions. In this experiment, a process group that computes one partial solution at a time, corresponds to one node of Hermit (HLRS). We used the ITG test case described in Sec. 2 with $\boldsymbol{n} = [17, 17, 17, 17]^T$ and $\boldsymbol{l}_{\min} = [3, 3, 3, 3]^T$ for $(\mu, v_{\|}, z, x)$. The resolution of y was fixed to 1. Thus, the test case consisted of 425 partial solutions in essentially four dimensions. For *LM*, we have $t(\boldsymbol{l}) = 2^{|\boldsymbol{l}|_1}$ and only consider the number of unknowns, but not the anisotropy. Furthermore, Fig. 3 includes

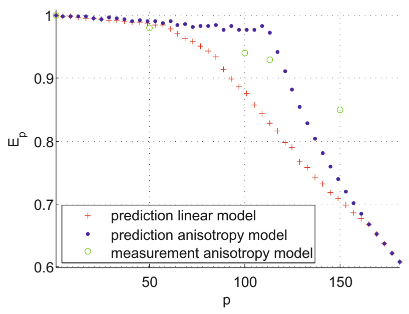


Fig. 3. Parallel efficiency E_p over the number p of process groups

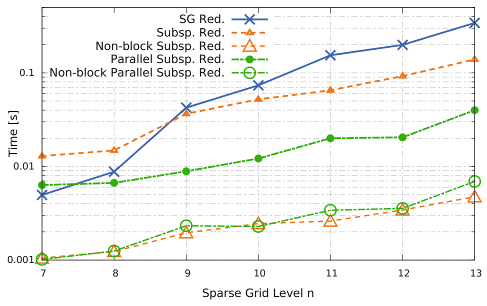


Fig. 4. Communication time for a 5-dimensional SG on 456 nodes of Hermit [13]

actual measurements using the AM in our manager-worker concept. The parallel efficiency $E_p = T_1/(pT_p)$ is predicted to decrease significantly above $p = 50$ using the LM, which is confirmed in practice. The anisotropic model, in contrast, predicts more than 97% until $p = 113$ solutions are computed in parallel. While the measurements fit well to the predictions for $p = 50$ and are a bit optimistic for $p = 100$ and $p = 113$, the model is slightly too pessimistic for a large degree of parallelism and works much better than it's own prediction. Eventually, E_p has to decrease, of course, as not more process groups can be spent than partial solutions exist.

3.2 Global Communication

The hierarchical approach allows one to decouple a single problem with global dependencies into independent partial problems. For initial value computations with the combination technique, it is furthermore necessary to combine the partial solutions every several time steps and to distribute the combination solution back to avoid divergence. This gather-scatter step is the remaining synchronization bottleneck. Therefore, we have developed global communication schemes for the combination technique, which minimize the communication time by exploiting the hierarchical structure of the combination solution [13].

The combination is assembled in the hierarchical sparse grid function space, not in the full grid nodal basis, see [14] for details. The idea of the communication scheme *Sparse Grid Reduce (SGR)* is to transform each partial solution to the sparse grid space and to sum them up according to the combination coefficients. This is the straightforward approach and will serve as the baseline. The summation can be expressed as a standard reduce operation like *MPI_Allreduce* on a set of vectors containing the sparse grids' coefficients. Note that each partial solution includes only a subset of the hierarchical subspaces of the sparse grid solution. Thus, the transformation has to interpolate the others in the hierarchical basis, which corresponds to a fill-in with zeros. This results in a high overhead of communicated data not containing any information.

A new *Subspace Reduce (SubR)* scheme avoids this overhead and communicates only the minimum necessary amount of data. The idea of this method is to reduce the hierarchical subspaces individually by using an efficient standard implementation like *MPI_Allreduce*. When reducing a particular hierarchical subspace, only the nodes actually containing this subspace contribute to the reduce operation. Thus, we do not need to communicate any data that contains no information, but the number of messages sent increases significantly compared to *SGR*, which only requires one reduce operation. If the sets of nodes contributing to the reduce operation of two particular subspaces are disjoint, the subspaces can be reduced in parallel. *Parallel Subspace Reduce (ParSubR)* further improves the run time by reducing the hierarchical subspaces in an order that enables a higher degree of parallelism than *SubR*.

We were able to significantly speed up *SubR* on Hermit by using the non-blocking *MPI_Iallreduce* of the MPI 3.0 standard (*Non-blocking Subspace Reduce (NB-SubR)*). This enables the MPI system to rearrange the substeps of the reduction operations on a fine granular level. This resulted in significantly lower run times than we were able to achieve by just rearranging the order of the library calls in *ParSubR*. Using the non-blocking operations for *ParSubR* resulted in a similar, though not systematically better, performance.

Figure 4 shows the run time of the communication step for dimension $d = 5$ and different discretization levels n. The experiments were done on 456 nodes of the supercomputer Hermit (HLRS). *SGR* is only faster than *SubR* for low n since the overhead is small. However, with increasing n the total communicated volume becomes the dominating factor and *SubR* is faster. Reordering the reduction operations with *ParSubR* significantly improved the performance of *SubR*. For $n = 13$, *ParSubR* was 8.5 times faster than *SGR*. An even larger speed up was achieved using non-blocking collective operations. For $n = 13$, *NB-SubR* was 72 times faster than *SGR*.

3.3 Fault Tolerance

Large scale simulations require large computation times. If we assume one hardware failure each week on current HPC systems, we will be down to failure rates in the range of minutes on future exascale systems. And this does not even take into account that smaller hardware integration will lead to higher failure rates. Thus it will be a necessity to deal with faults. In the context of the combination technique, this means that some of the component grids will not be computed successfully, and one cannot carry out the combination step properly. Recalculating the solutions on those grids in case of failures would require the rescheduling of tasks, which can increase the overall computation time and fool the load balancing schemes.

We therefore opt for an algorithm-based fault tolerant (ABFT) approach to overcome this problem. Several methods have been developed that attempt to recover the combined sparse grid solution in the case of processor failure [17,7,8] without checkpoint-restart. One of the most promising modifies the set of successfully calculated partial solutions and combines them with new coefficients,

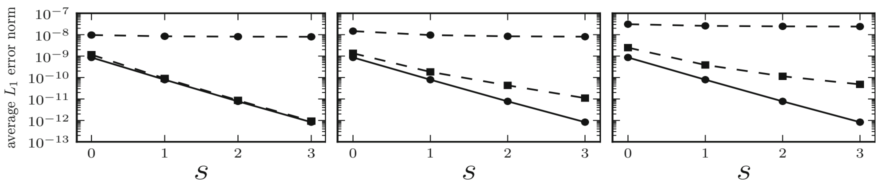


Fig. 5. Left: The error of the usual combination solution (*solid, circles*, no faults), the error with one grid missing on the highest level (*dashed, circles*), and the error after the recovery scheme has been applied (*dashed, squares*). **Middle:** the same as **Left**, but with two missing grids: one on the highest level, and one on the level below. **Right:** we added one more fault on the second level (3 faults in total).

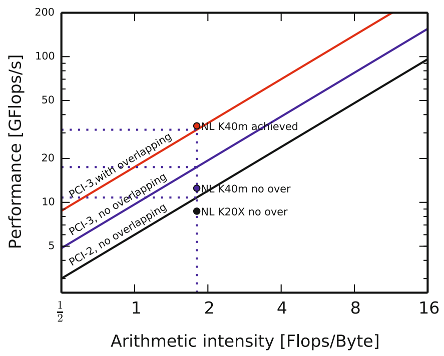
following existing ideas from adaptive sparse grids [10]. More sophisticated approaches involve inter- and extrapolation, and some error bounds for the different approaches are detailed in [8].

We carried out several tests on GENE with simulated faults. Our recovery strategy approximates the lost partial solutions by a linear combination of available ones. This simple approach already reveals promising features, as illustrated in Fig. 5, where we repeated the simulations from Sec. 2, now including faults. All combination schemes involve 10 grids, and we simulated one, two, and three faults on different levels. Note that in the last scenario 30% of all computations fail, but these can be compensated incredibly well and without the need for further (re)computations. We expect that more sophisticated approaches further reduce the recombination error.

3.4 GPU Computing

Many of the actual supercomputers are of heterogeneous type. Usually they are large Linux-based compute clusters where some or all nodes are equipped with an additional accelerator card, mainly of the GPU or Intel Xeon Phi type. To fully use the amazing performance of these accelerated cluster systems, a first attempt to port GENE to GPUs was carried out. We focused on the computation of the non-linear part $\mathcal{N}$ in (2) in a global (full-torus) simulation [2], since this part takes around half of the total runtime of the time loop. Therefore, it is a promising candidate for acceleration. The non-linear part consists of the following steps: Transposition to exchange the x and k_y directions, extension of the k_y direction for dealiasing, Fourier transform in the k_y direction, and multiplication of two extended, transposed, and transformed arrays to get the nonlinear term. The latter is then processed the same way backwards. For the Fourier transform, we used the `cuFFT` library which is part of CUDA, while all other operations had to be written as CUDA-C kernels, nearly doubling the number of code lines.

First performance comparisons of older Nehalem CPU cores with older Nvidia Fermi cards were promising, as they showed a speedup (always defined as the reduction of runtime of the optimized code on a whole CPU socket with GPU to the runtime on a whole CPU socket without GPU) of 4–5 for the nonlinearity.



	1 SNB	2 SNB
CPU only	4.4s	2.1s
Fermi M2090	7.4s	3.5s
Kepler K20X	4.1s	2.3s

Fig. 6. Left: part of the roofline model of a Kepler GPU with respect to floating point performance and transfer bandwidth for low arithmetic intensity. **Right**: performance of the nonlinearity with GPU acceleration.

Figure 6 (right) shows a more recent comparison of new architectures. On the CPU side, an 8-core SandyBridge socket and on the GPU side, a Kepler K20X card, did not hold these nice performance results. In the end, we found that the 8-core CPU alone performs as powerful as the CPU-GPU combination.

Investigating this result with the help of the "roofline model" [19], we could identify the slow PCIexpress 2.0 bus in combination with the relatively low arithmetic intensity (defined as the number of floating point operations per amount of data transferred via PCIexpress bus to the GPU), which is only 0.38 flops/byte for the nonlinearity, as the key to understanding the low performance. Since the transfer is slow, the computing power of the GPU cannot be fully exploited while waiting for the transferred data. As a remedy, one could increase the amount of computation per data transfer by porting the whole right-hand side computation to the GPU. From the roofline model, for the whole right-hand side we expect then again a factor 4–5 of performance gain, depending on the problem size and the quality of the kernel implementations.

A second possibility to speed up the CPU-GPU performance is to use a faster bus between host and device. This can be achieved in the simplest case by using the PCIexpress gen. 3 bus, which nominally doubles the bandwidth. A computationally similar (around 8% faster) K40m Nvidia GPU has been used, which has more memory and can use the faster bus in combination with an Ivybridge CPU. We measured 4.4s on the CPU and 3.1s with the Kepler K40m.

Figure 6 (left) shows a roofline plot including the results. The two bandwidth ceilings for pure PCI-2 and PCI-3 are shown as black and blue lines, together with the achieved performance for the nonlinearity (black and blue data points) if overlapping of computation and transfer is switched off. If two streams are used and the work is distributed over these streams, one can overlap part of the transfer by computation on the GPU. This can be taken as a kind of increased bandwidth as the idle time where only transfer occurs decreases. The red ceiling represents the improved bandwidth when the overlapping is switched on. Hence,

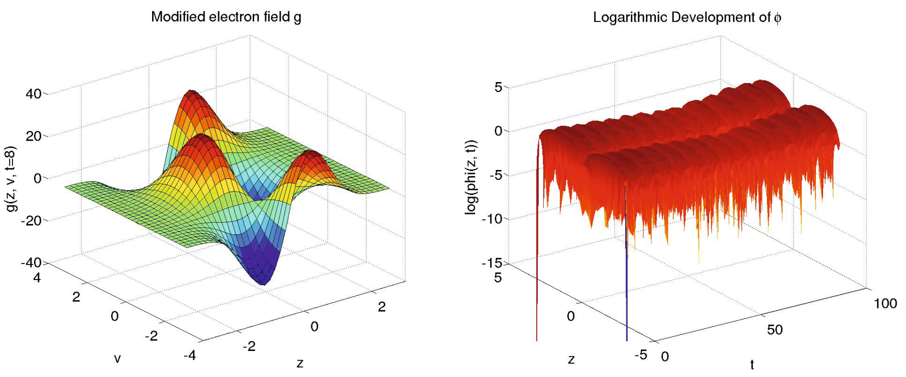


Fig. 7. Left: initial value runs of the (1+1)D simplified Vlasov-model with the modified electron field g at $t = 8.0$. **Right**: the electrostatic potential ϕ over time are small model problems for full 5D Vlasov simulations.

we see a significant improvement with a faster transfer, but still the gain of roughly 30% compared to a pure-CPU implementation is not worth the effort.

3.5 Numerics

In addition to the standard sparse grid combination technique, an iterated version of the classical and optimized combination technique [11] is under investigation for Vlasov initial value and eigenvalue problems. It will guarantee convergence of the standard combination technique to the corresponding sparse grid solution for complex PDEs. The iterated combination technique applies a residual correction method [4]. It is a second way to deal with errors at almost no additional cost.

To examine our method, we introduced a set of small-scale model problems that run independently of the large-scale simulation code GENE, including the Poisson problem and the Poisson eigenvalue problem. As a small-scale version of the full Vlasov equations, a driftkinetic version of the Vlasov-Maxwell equations in (1+1)D phase space is considered, see [1]. It can be used both as an initial value problem and an eigenvalue problem if the spatial operator is analysed for its spectral structure. An efficient GPU implementation is available. Simulation runs, see Fig. 7, were validated against results from the literature.

The classical combination technique was applied both to the Poisson problem and the small-scale Vlasov initial value problem leading to numerical reference solutions. Next, the iterated combination technique was analysed for the Poisson problem. The resulting numerical scheme converges to a fixpoint, cf. Fig. 8. We are currently assessing the quality of this solution and developing a new sparse grid residual evaluation scheme constructed by subspace problems.

In addition to the initial value problem, iterative correction methods for eigenvalue problems are under investigation. As a simplified model, the Poisson eigenvalue problem was first considered. An operator-based sparse grid combination technique is proposed, which avoids the evaluation of the Rayleigh quotient and

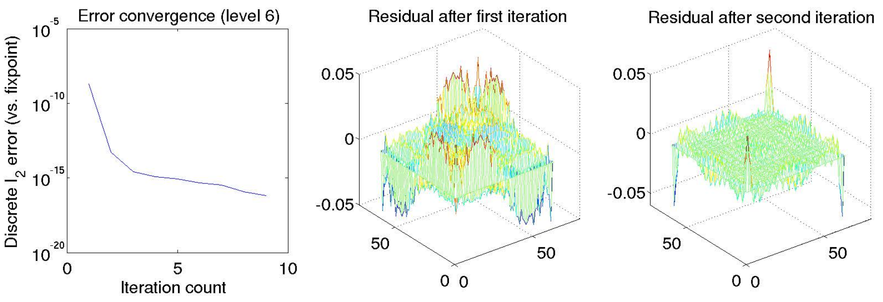


Fig. 8. **Left**: the iterated sparse grid combination technique converges to a fixpoint, **Middle, Right**: with some structure in the iterates of the residual

the eigenvalue identification problem available in previous approaches. This is done by applying the combination technique to the discrete operator instead of the generated eigenvalue/eigenvector pairs. First tests comparing numerical results of the Poisson eingenvalue problem with the numerically known eigenvalues of the Poisson operator are very promising. They suggest that the new scheme converges for a subset of the full-grid Poisson eigenvalue problem. Replacing interpolation in the operator-based sparse grid combination technique by a discrete l_2-projection allows to further improve convergence.

4 Conclusions and Future Work

To successfully solve high-dimensional problems on future exascale systems, novel algorithms, implementations, and numerical schemes have been developed. The hierarchical discretization scheme, which reduces the number of unknowns and will make full-scale high-resolution simulations possible on tomorrow's HPC systems, provides new methods to deal with the exa-challenges of scalability (breaking the need for global communication), resilience (without checkpoint-restart) and load balancing (due to a second, coarse-grain level of parallelism). We have shown the feasibility of our approach for turbulence simulations in plasma physics, presented new load models, communication schemes, first approaches to fault tolerance, and results on a hybrid implementation, as well as sketches of new iterative numerical schemes.

The next steps include a load model generated at runtime and refined as soon as new runtime data is available, also extendable to non-linear and eigenvalue runs. For our communication schemes, the transformation of the partial solutions into the hierarchical basis (required for the gather-scatter step) has to be done in a distributed way, at latest if the size of the overall solution exceeds the memory available on a single node. To deal with faults, we will examine whether it pays off to precompute partial solutions from additional, coarser discretization levels to speed up the recovery algorithms. Considering numerics, the iterated operator-based sparse grid eigenvalue problem will be considered and extended to full GENE runs. We will develop new numerical schemes in a simplified test-bed by extracting and analyzing the linear operator of GENE as a matrix.

Acknowledgements. This work was supported by the German Research Foundation (DFG) through the Priority Programme *Software for Exascale Computing (SPPEXA)*. The author Alfredo Parra Hinojosa additionally thanks CONACYT, Mexico.

References

1. Dannert, T., Jenko, F.: Vlasov simulation of kinetic shear alfvén waves. Computer Physics Communications 163(2), 67–78 (2004)
2. Dannert, T., Marek, A., Rampp, M.: Porting large HPC applications to GPU clusters: The codes GENE and VERTEX. Parallel Computing: Accelerating CSE. Advances in Parallel Computing 25, 305–314 (2014)
3. Görler, T.: Multiscale effects in plasma microturbulence. Ph.D. thesis, Universität Ulm (2009)
4. Griebel, M.: A domain decomposition method using sparse grids. In: Contemporary Mathematics, DDM6, vol. 157, pp. 255–261. Am. Math. Soc (1994)
5. Griebel, M., Huber, W., Rüde, U., Störtkuhl, T.: The combination technique for parallel sparse-grid-preconditioning or -solution of PDEs on workstation networks. In: Parallel Processing: CONPAR 92 VAPP V, LNCS, vol. 634 (1992)
6. Griebel, M., Schneider, M., Zenger, C.: A combination technique for the solution of sparse grid problems. In: Iterative Methods in Lin. Alg, pp. 263–281 (1992)
7. Harding, B., Hegland, M.: A robust combination technique. In: CTAC-2012. ANZIAM J., vol. 54, pp. 394–411 (August 2013)
8. Harding, B., Hegland, M.: Robust solutions to PDEs with multiple grids. In: Garcke, J., Pfluger, D. (eds.) Sparse Grids and Applications 2012. LNCISE, vol. 97, pp. 171–193. Springer, Heidelberg (2014)
9. Heene, M., Kowitz, C., Pflüger, D.: Load balancing for massively parallel computations with the sparse grid combination technique. In: Parallel Computing: Accelerating CSE. Advances in Parallel Computing, vol. 25, pp. 574–583 (2014)
10. Hegland, M.: Adaptive sparse grids. In: Proc. of 10th Computational Techniques and Applications Conference CTAC-2001., vol. 44, pp. C335–C353 (2003)
11. Hegland, M., Garcke, J., Challis, V.: The combination technique and some generalisations. Linear Algebra and its Applications 420(2–3), 249–275 (2007)
12. Hegland, M.: Adaptive sparse grids. ANZIAM Journal 44, C335–C353 (2003)
13. Hupp, P., Heene, M., Jacob, R., Pflüger, D.: Global communication schemes for the numerical solution of high-dimensional PDEs. Parallel Computing (submitted)
14. Hupp, P., Jacob, R., Heene, M., Pflüger, D., Hegland, M.: Global communication schemes for the sparse grid combination technique. Parallel Computing: Accelerating CSE. Advances in Parallel Computing 25, 564–573 (2014)
15. Keyes, D.E.: Exaflop/s: The why and the how. Comptes Rendus Mcanique 339(2–3), 70–77 (2011)
16. Kowitz, C., Pflüger, D., Jenko, F., Hegland, M.: The Combination Technique for the Initial Value Problem in Linear Gyrokinetics. In: Garcke, J., Griebel, M. (eds.) Sparse Grids and Applications. LNCSE, vol. 88, pp. 205–222. Springer, Heidelberg (2013)
17. Larson, J., Hegland, M., et al.: Fault-tolerant grid-based solvers: Combining concepts from sparse grids and mapreduce. Proc. Comp. Science 18, 130–139 (2013)
18. Wesson, J., Campbell, D.: Tokamaks. International Series of Monographs on Physics. OUP Oxford (2011)
19. Williams, S., Waterman, A., Patterson, D.: Roofline: an insightful visual performance model for multicore architectures. Comm. of the ACM 52(4), 65–76 (2009)