

# Advanced Stencil-Code Engineering

Edited by

Christian Lengauer<sup>1</sup>, Matthias Bolten<sup>2</sup>, Robert D. Falgout<sup>3</sup>, and  
Olaf Schenk<sup>4</sup>

- 1 Universität Passau, DE, christian.lengauer@uni-passau.de
- 2 Bergische Universität Wuppertal, DE, bolten@math.uni-wuppertal.de
- 3 Lawrence Livermore National Lab., US, falgout2@llnl.gov
- 4 University of Lugano, CH, olaf.schenk@usi.ch

---

## Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 15161 “Advanced Stencil-Code Engineering”. The seminar was hosted by the DFG project with the same name (ExaStencils for short) in the DFG priority programme “Software for Exascale Computing” (SPPEXA). It brought together experts from mathematics, computer science and applications to explore the challenges of very high performance and massive parallelism in solving partial differential equations. Its aim was to lay the basis for a new interdisciplinary research community on high-performance stencil codes.

**Seminar** April 12–17, 2015 – <http://www.dagstuhl.de/15161>

**1998 ACM Subject Classification** C.4 Performance of Systems, D.1.3 Concurrent Programming, D.2.6 Programming Environments, D.3.3 Language Constructs and Features, G.1.8 Partial Differential Equations, G.4 Mathematical Software

**Keywords and phrases** Code generation, domain-specific languages, exascale computing, high-performance computing, massive parallelism, multigrid, partial differential equations, program optimization, program parallelization, stencil codes

**Digital Object Identifier** 10.4230/DagRep.5.4.56

## 1 Executive Summary

*Christian Lengauer*

*Matthias Bolten*

*Robert D. Falgout*

*Olaf Schenk*

**License**  Creative Commons BY 3.0 Unported license  
© Christian Lengauer, Matthias Bolten, Robert D. Falgout, and Olaf Schenk

## Stencil Codes

Stencil codes are compute-intensive algorithms, in which data points arranged in a large grid are being recomputed repeatedly from the values of data points in a predefined neighborhood. This fixed neighborhood pattern is called a stencil. Stencil codes see wide-spread use in computing the discrete solutions of partial differential equations and systems composed of such equations. Connected to the implementation of stencil codes is the use of efficient solver technology, i.e., iterative solvers that rely on the application of a stencil and that provide good convergence properties like multigrid methods. Major application areas are the natural sciences and engineering. Although, in many of these applications, unstructured



Except where otherwise noted, content of this report is licensed  
under a Creative Commons BY 3.0 Unported license

Advanced Stencil-Code Engineering, *Dagstuhl Reports*, Vol. 5, Issue 4, pp. 56–75

Editors: Christian Lengauer, Matthias Bolten, Robert D. Falgout, and Olaf Schenk



DAGSTUHL  
REPORTS

Dagstuhl Reports  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

adaptive discretizations are employed for an efficient use of exascale supercomputers whose architectures possibly include accelerators or are of a heterogeneous nature, the use of structured discretizations and, thus, stencil codes has turned out to be helpful.

Stencil codes come in large varieties: there are many thousands! Deriving each of them individually, even if by code modification from one another, is not practical. The goal of the seminar is to raise the level of abstraction for application programmers significantly and to support this raise with an automated software technology that generates highly efficient massively parallel implementations which are tuned to the specific problem at hand and the execution platform used.

## Research Challenges

Stencil codes are algorithms with a pleasantly high regularity: the data structures are higher-dimensional grids and the computations follow a static, locally contained dependence pattern and are typically arranged in nested loops with linearly affine bounds. This invites massive parallelism and raises the hope for easily achieved high performance. However, serious challenges remain:

- Because of the large numbers and varieties of stencil code implementations, deriving each of them individually, even if by code modification from one another, is not practical. Not even the use of program libraries is practical; instead, a domain-specific metaprogramming approach is needed.
- Reaching petascale to exascale execution speed is a challenge in the frequently used so-called multigrid algorithms, which work on a hierarchy of increasingly larger grids. The coarse grids in the upper part of the hierarchy are too small for massive parallelism.
- Efficiency, i.e., a high ratio of speedup to the degree of parallelism, is impaired by the low mathematical density, i.e., the low ratio of computation steps to data transfers of stencil codes.
- An inappropriate use of the execution platform may act as a performance brake.

Stencil-code engineering has received increased attention in the last few years, which is evidenced by the appearance of a number of stencil-code programming languages and frameworks. To reach the highest possible execution speed and to conserve hardware resources and energy, the stencil code must be tuned cleverly to the specific application problem at hand and the execution platform used. One approach that could be followed has been demonstrated by the previous U.S. project SPIRAL, whose target was the domain of linear transforms: domain-specific optimization at several levels of abstraction – from the mathematical equations over an abstract, domain-specific program and, in further steps, to the actual target code on the execution platform used. At each level, one makes aggressive use of knowledge of the problem and platform and employs up-to-date, automated software technology suitable for that level.

## Questions and Issues Addressed

The charter of the seminar was to foster international cooperation in the development of a radically new, automatic, optimizing software technology for the effective and flexible exploitation of massively parallel architectures for dedicated, well delineated problem domains.

The central approaches in achieving this technology are:

- the aggressive use of domain knowledge for optimization at different levels of abstraction
- the exploitation of commonalities and variabilities in application codes via product-line technology and domain engineering

- the use of powerful models for program optimization, like the polyhedron model for loop parallelization and feature-orientation for software product lines

The application domain investigated in the seminar was stencil codes. It is envisaged that the approach can be ported to other well delineated domains – of course, with the substitution of suitable domain-specific content.

Among the issues discussed were:

- What are suitable abstraction, modularization, composition and generation mechanisms for stencil codes?
- What are the appropriate language features of a domain-specific language for stencil codes?
- What are the commonalities and variabilities of stencil codes?
- What are the computational performance barriers, especially, of multigrid methods using stencils and how can they be overcome?
- What are the performance barriers caused by data exchanges and how can they be overcome? How can communication be avoided in multilevel algorithms?
- What are the roles of nested loops and divide-and-conquer recursions in stencil codes?
- How can other solvers and preconditioners benefit from autotuned stencil codes?
- What role should techniques like autotuning and machine learning play in the optimization of stencil codes?
- What options of mapping stencil codes to a heterogeneous execution platform exist and how can an educated choice be made?
- Which techniques can be employed to make clever use of large-scale hybrid architectures, e.g., by the combination of multigrid with mathematical domain decomposition?

On the informatics side, one important role of the seminar was to inform the international stencils community about the techniques used in ExaStencils: software product lines, polyhedral loop optimization and architectural metaprogramming. Equally important was for ExaStencils members to learn about the experiences made with other techniques like divide-and-conquer, multicore optimization in parallel algorithms or autotuning. The application experts contributed to a realistic grounding of the research questions.

On the mathematics side, the seminar fostered the cooperation of experts in parallel solver technology with the groups from informatics to enable them to make use of the advanced techniques available. Further, different strategies for improving the scalability of iterative methods were discussed and the awareness of the opportunities and complexities of modern architectures in the numerical mathematics community was advanced.

## 2 Table of Contents

### Executive Summary

*Christian Lengauer, Matthias Bolten, Robert D. Falgout, and Olaf Schenk* . . . . . 56

### Overview of Talks

From stencils to elliptic PDE solvers <i>Ulrich Rüde</i> . . . . .	61
Maintaining performance in a general-purpose FEM code <i>Christian Engwer</i> . . . . .	61
Optimization opportunities of stencil codes via analytic performance modeling <i>Georg Hager</i> . . . . .	62
SPPEXA und ExaStencils <i>Christian Lengauer</i> . . . . .	62
Local Fourier analysis for multigrid on semi-structured meshes <i>Carmen Rodrigo Cardiel</i> . . . . .	62
Predicting the numerical performance of methods for evolutionary problems <i>Stephanie Friedhoff</i> . . . . .	63
An extension of hypre's structured and semi-structured matrix classes <i>Ulrike Meyer-Yang</i> . . . . .	64
The PyOP2 abstraction <i>Paul H. J. Kelly</i> . . . . .	64
The Pochoir stencil compiler <i>Bradley Kuszmaul</i> . . . . .	64
Formal synthesis of computational kernels <i>Franz Franchetti</i> . . . . .	65
ExaSlang and the ExaStencils code generator <i>Christian Schmitt, Stefan Kronawitter, Sebastian Kuckuk</i> . . . . .	65
Variability management in ExaStencils <i>Alexander Grebhahn</i> . . . . .	66
From general-purpose to stencil DSL code <i>Armando Solar-Lezama</i> . . . . .	66
STELLA: A domain-specific language for stencil computations <i>Carlos Osuna Escamilla</i> . . . . .	67
Designing GridTools <i>Mauro Bianco</i> . . . . .	67
Redesign of preconditioned Krylov methods around stencil compilers <i>Wim Vanroose</i> . . . . .	68
PolyMage: High-performance compilation for heterogeneous stencils <i>Uday Bondhugula</i> . . . . .	68
On the characterization of the data movement complexity of algorithms <i>P. Sadayappan</i> . . . . .	69

The mathematics of ExaStencils <i>Hannah Rittich</i> . . . . .	69
Stencils for hierarchical bases <i>Dirk Pflüger</i> . . . . .	69
Mapping stencils to FPGAs and synthesizable accelerators <i>Louis-Noel Pouchet</i> . . . . .	70
The stencil optimization framework MODESTO <i>Tobias Grosser</i> . . . . .	70
Autotuning divide-and-conquer stencil computations <i>Ekanathan Palamadai Natarajan</i> . . . . .	70
Things you can do with stencils <i>Gabriel Wittum</i> . . . . .	71
Optimization of higher-order stencils <i>P. Sadayappan</i> . . . . .	72
DSL for stencils in non-Newtonian fluids simulation? <i>Gundolf Haase</i> . . . . .	72
<b>Evening Discussion Sessions</b> . . . . .	72
<b>Conclusions</b> . . . . .	74
<b>Participants</b> . . . . .	75

### 3 Overview of Talks

40-min talk slots covered the programme until Thursday mid-afternoon. Many talks had multiple authors; in one, the presentation was shared by all authors. The latter part of the seminar was devoted to the planning of future collaborations. A list of talks follows in the order in which they were presented.

#### 3.1 From stencils to elliptic PDE solvers

*Ulrich Rüde (Friedrich-Alexander-Universität Erlangen-Nürnberg – Erlangen, DE)*

License © Creative Commons BY 3.0 Unported license  
© Ulrich Rüde

The talk addresses three aspects of stencil codes:

- What techniques can we use to speed up stencil codes? These are blocking and tiling techniques, but also memory layout transformations such as padding and multi-color-splits.
- What stencil codes should be considered? Here it is important to notice that the algorithms more often than not will need to accomplish a global exchange of data and that any attempt to avoid this will only result in inefficient algorithms. In other words: there is no way to avoid the complexities as they occur, e.g., in multigrid algorithms – the question is rather to find ways to implement such algorithmic structures as efficiently as possible.
- Where do we stand? Here the prototype HHG package shows that large systems with in excess of  $10^{12}$  (a trillion) unknowns can be solved in a matter of minutes using highly optimized multigrid algorithms on parallel systems running up to a million parallel threads.

#### 3.2 Maintaining performance in a general-purpose FEM code

*Christian Engwer (Westfälische Wilhelms-Universität – Münster, DE)*

License © Creative Commons BY 3.0 Unported license  
© Christian Engwer  
Joint work of P. Bastian, C. Engwer, J. Fahlke, S. Müthing


For solving partial differential equations the finite element method (FEM) is an attractive powerful tool. In many engineering applications, the FEM on unstructured meshes is used to account for the complicated geometry.

The text book stencil approaches gain high performance from their very simple predefined structure. FEM, on the other hand, gets the flexibility from its ability to deal with arbitrary unstructured meshes. To obtain the good performance of stencils and, at the same time, keep the flexibility of FEM, we adopt certain concepts from stencils. To achieve this, we introduce local structure – either by locally structured refinement or by higher-order methods.

A particular challenge arises from the fact that the PDE model is given in terms of user code and is executed in the inner most loop. We discuss how to restructure the interfaces to exploit the local structure and obtain a significant portion of peak performance while keeping the flexibility at the user level.

### 3.3 Optimization opportunities of stencils codes via analytic performance modeling

*Georg Hager (Friedrich-Alexander-Universität Erlangen-Nürnberg – Erlangen, DE)*

License  Creative Commons BY 3.0 Unported license  
© Georg Hager

Much effort has been put into optimized implementations of stencil algorithms. Such activities are usually not guided by performance models that provide estimates of expected speedup. Understanding the performance properties and bottlenecks by performance modeling enables a clear view on promising optimization opportunities. We use the recently developed Execution-Cache-Memory (ECM) model to quantify the performance bottlenecks of stencil algorithms on a contemporary Intel processor. Single-core performance and scalability predictions for typical “corner-case” stencil loop kernels are given. Guided by the ECM model, we accurately quantify the significance of “layer conditions”, which are required to estimate the data traffic through the memory hierarchy, and study the impact of typical optimization approaches such as spatial blocking, strength reduction, and temporal blocking for their expected benefits. We also compare the ECM model to the widely known Roofline model and pinpoint the limitations of both. In an outlook, we demonstrate a simple tool that can automatically construct the Roofline and ECM models for streaming kernels (including stencils).

### 3.4 SPPEXA und ExaStencils


*Christian Lengauer (Universität Passau – Passau, DE)*

License  Creative Commons BY 3.0 Unported license  
© Christian Lengauer

The DFG Priority Programme SPP 1648 “Software for Exascale Computing” (SPPEXA) is introduced briefly and one of its thirteen projects is sketched: “Advanced Stencil Code Engineering” (ExaStencils). The goals of the project are stated and justified, and the structure of the ExaStencils development framework is reviewed. Three further talks in the seminar provide details.

### 3.5 Local Fourier analysis for multigrid on semi-structured meshes

*Carmen Rodrigo Cardiel (Universidad de Zaragoza – Zaragoza, ES)*

License  Creative Commons BY 3.0 Unported license  
© Carmen Rodrigo Cardiel

Joint work of A. Arrarás, F. J. Gaspar, B. Gmeiner, T. Gradl, F. J. Lisbona, L. Porter, C. Rodrigo Cardiel, U. Råde, P. Salinas

To approximate solutions of problems defined on complex domains, it is very common to apply a regular refinement to an unstructured input grid which fits the geometry of the domain. In this way, a hierarchy of globally unstructured grids with regular structured patches is generated. This kind of mesh is suitable for the use with geometric multigrid methods and allows us to use stencil-based data structures which reduce the memory requirements drastically.

In this setting, we are interested in the design of efficient geometric multigrid methods on such semi-structured triangular grids. To design these algorithms, a local Fourier analysis for non-orthogonal grids is used. This tool is based on the Fourier transform and provides very accurate predictions of the asymptotic convergence of geometric multigrid methods. It is a useful technique for the choice of the suitable components of your algorithm. This analysis is applied on each structured patch of the grid in order to choose the most efficient components on each block of the semi-structured grid.

This strategy was initially applied to linear finite-element discretizations of scalar partial differential equations in two dimensions, and was later extended and generalized to such discretizations for systems of PDEs, three-dimensional tetrahedral grids, high-order finite element discretizations, finite-volume cell-centered schemes, and even to time-dependent non-linear problems by combining the approach with splitting schemes in time.

Note that each of these extensions requires the design of specific smoothers appropriate for the problems encountered with the different discretizations and, most of the time, it is also a special approach of the local Fourier analysis is necessary.

### 3.6 Predicting the numerical performance of methods for evolutionary problems

*Stephanie Friedhoff (University of Leuven – Leuven, BE)*

License © Creative Commons BY 3.0 Unported license  
© Stephanie Friedhoff

Joint work of S. Friedhoff, S. MacLachlan

With current trends in computer architectures leading towards systems with more, but not faster, processors, faster time to solution must come from greater parallelism. These trends particularly impact the numerical solution of the linear systems arising from the discretization of partial differential equations (PDEs) with evolutionary behavior, such as parabolic problems. The multigrid-reduction-in-time (MGRIT) algorithm is a truly multi-level approach to parallel-in-time integration, which directly uses an existing time propagator and, thus, can easily exploit substantially more computational resources than standard sequential time stepping. Multigrid waveform relaxation is another effective multigrid method on space-time grids for parabolic problems. However, a large gap exists between the theoretical analysis of these algorithms and their actual performance.

We present a generalization of the well-known local-mode (often local Fourier) analysis (LFA) approach. The proposed semi-algebraic mode analysis (SAMA) approach couples standard LFA with tractable numerical computation that accounts for the non-local character of operators in the class of evolutionary problems. We demonstrate that SAMA provides an advantage for parabolic problems, obtaining robust predictivity of performance independent of the length of the time domain – in sharp contrast to LFA, which only becomes predictive for extremely long time integration.



### 3.7 An extension of hypre’s structured and semi-structured matrix classes

*Ulrike Meier Yang (LLNL – Livermore, US)*

**License** © Creative Commons BY 3.0 Unported license  
© Ulrike Meyer-Yang

**Joint work of** R. Falgout, U. Meyer-Yang

The hypre software library provides high-performance preconditioners and solvers for the solution of large sparse linear systems on massively parallel computers. One of its attractive features is the provision of conceptual interfaces, which include a structured interface, a semi-structured interface, and a traditional linear-algebra-based interface. The (semi-)structured interfaces are an alternative to the standard matrix-based interface that describes rows, columns, and coefficients of a matrix. Here, instead, matrices are described primarily in terms of stencils and logically structured grids. These interfaces give application users a more natural means for describing their linear systems, and provide access to methods such as structured multigrid solvers, which can take advantage of the additional information beyond just the matrix. Since current architecture trends are favoring regular compute patterns to achieve high performance, the ability to express structure has become much more important.

We describe a new structured-grid matrix class that supports rectangular matrices and constant coefficients and a semi-structured-grid matrix class that builds on the new structured-grid matrix. We anticipate that an efficient implementation of these new classes will lead to better performance of matrix kernels and algorithms on current and future architectures than hypre’s current matrix classes.

### 3.8 The PyOP2 abstraction

*Paul H. J. Kelly (Imperial College – London, UK)*

**License** © Creative Commons BY 3.0 Unported license  
© Paul H. J. Kelly

**Joint work of** G. Bercea, C. Bertolli, C. Cantwell, M. Giles, G. Gorman, D. A. Ham, P. H. J. Kelly, C. Krieger, F. Luporini, G. R. Markall, M. Mills Strout, G. Mudalige, C. Olschanowsky, R. Ramanujam, F. Rathgeber, I. Reguly, G. Rokos, S. Sherwin

PyOP2 is a stencil-like abstraction for parallel loops over unstructured meshes. It is used as an intermediate representation in Firedrake, an automated system for the portable solution of partial differential equations using the finite element method. This talk explores some of the opportunities exposed by PyOP2, for unstructured and extruded meshes and for locality, parallelisation and vectorisation.

### 3.9 The Pochoir stencil compiler

*Bradley Kuszmaul (Massachusetts Institute of Technology – Boston, MA, US)*

**License** © Creative Commons BY 3.0 Unported license  
© Bradley Kuszmaul

**Joint work of** R. A. Chowdhury, B. Kuszmaul, C. E. Leiserson, C.-K. Luk, Y. Tang

A stencil computation repeatedly updates each point of a  $d$ -dimensional grid as a function of itself and its near neighbors. Parallel cache-efficient stencil algorithms based on “trapezoidal

decompositions” are known, but most programmers find them difficult to write. The Pochoir stencil compiler allows a programmer to write a simple specification of a stencil in a domain-specific stencil language embedded in C++ which the Pochoir compiler then translates into high-performing Cilk code that employs an efficient parallel cache-oblivious algorithm. Pochoir supports general  $d$ -dimensional stencils and handles both periodic and aperiodic boundary conditions in one unified algorithm. The Pochoir system provides a C++ template library that allows the user’s stencil specification to be executed directly in C++ without the Pochoir compiler (albeit more slowly), which simplifies user debugging and greatly simplified the implementation of the Pochoir compiler itself. A host of stencil benchmarks run on a modern multicore machine demonstrates that Pochoir outperforms standard parallel-loop implementations, typically running 2–10 times faster. The algorithm behind Pochoir improves on prior cache-efficient algorithms on multidimensional grids by making “hyperspace” cuts, which yield asymptotically more parallelism for the same cache efficiency.

### 3.10 Formal synthesis of computational kernels

*Franz Franchetti (Carnegie Mellon University – Pittsburgh, PA, US)*

**License** © Creative Commons BY 3.0 Unported license  
© Franz Franchetti

**Joint work of** research groups SPIRAL and HACMS

We address the question of how to map computational kernels automatically across a wide range of computing platforms to highly efficient code, and prove the correctness of the synthesized code. This addresses two fundamental problems that software developers are faced with: performance portability across the ever-changing landscape of parallel platforms, and verifiable correctness of sophisticated floating-point code. We have implemented this approach as part of the SPIRAL system where we have formalized a selection of computational kernels from the signal and image processing domain, software-defined radio, and robotic vehicle control.

### 3.11 ExaSlang and the ExaStencils code generator

*Christian Schmitt (Friedrich-Alexander-Universität Erlangen-Nürnberg – Erlangen, DE)*

*Stefan Kronawitter (Universität Passau – Passau, DE)*

*Sebastian Kuckuk (Friedrich-Alexander-Universität Erlangen-Nürnberg – Erlangen, DE)*

**License** © Creative Commons BY 3.0 Unported license  
© Christian Schmitt, Stefan Kronawitter, Sebastian Kuckuk

**Joint work of** F. Hannig, H. Köstler, S. Kronawitter, S. Kuckuk, C. Lengauer, U. Rüde, C. Schmitt, J. Teich

Many problems in computational science and engineering involve elliptic partial differential equations and require the numerical solution of large, sparse (non-)linear systems of equations. Multigrid is known to be one of the most efficient methods for this purpose. However, the concrete multigrid algorithm and its implementation depend highly on the underlying problem and hardware.

Project ExaStencils aims at a compiler and underlying code generation framework capable of generating automatically highly parallel and highly efficient geometric multigrid solvers from a very abstract description, while selecting the most performant program composition.

In our presentation, we focus on the aspects of code generation, node-level performance optimization and parallelization. More specifically, we provide an insight into the different components of our compiler framework and introduce some of the polyhedral as well as the low-level optimizations which are applied automatically. Furthermore, we introduce our approach to domain partitioning as well as details on employed communication strategies. Finally, we present several experimental results, ranging from node-level performance improvements over a case study of adding generator support for FPGAs to weak-scaling results on JUQUEEN.

### 3.12 Variability management in ExaStencils

*Alexander Grebhahn (Universität Passau – Passau, DE)*

**License** © Creative Commons BY 3.0 Unported license  
© Alexander Grebhahn

**Joint work of** S. Apel, A. Grebhahn, N. Siegmund

The automatic generation of multigrid solvers leads to the possibility of creating a high number of different variants that are optimal for a wide range of different hardware. However, identifying the optimal variant for a specific hardware is a challenging task due to the inherent variability of the solvers. To master this challenge, we created a machine-learning approach for the derivation of a performance-influence model. Such a model describes all relevant influences of configuration options and their interactions on the performance of all possible variants. To identify a performance-influence model, we use an iterative approach that relies on a number of measurements gathered, using several structured sampling heuristics. In a series of experiments, we demonstrated the feasibility of our approach in terms of accuracy of the derived models.

### 3.13 From general-purpose to stencil DSL code

*Armando Solar-Lezama (Massachusetts Institute of Technology – Boston, MA, US)*

**License** © Creative Commons BY 3.0 Unported license  
© Armando Solar-Lezama

The talk describes a new technique to allow high-performance stencil DSLs to be used to optimize existing legacy code. The key idea is to use synthesis technology to derive a high-level specification from a block of low-level code implementing a stencil. This high-level specification can then be mechanically translated into a target DSL.

In addition to deriving the code, the technique also derives the invariants necessary to prove the equivalence between the code and the extracted specification. The talk presented some initial results that suggest that the technique can extract a specification from some non-trivial stencils, including some that have undergone some hand optimization.

### 3.14 STELLA: A domain-specific language for stencil computations

*Carlos Osuna Escamilla (Eidgenössische Technische Hochschule – Zürich, CH)*

**License** © Creative Commons BY 3.0 Unported license  
© Carlos Osuna Escamilla

**Joint work of** M. Bianco, O. Fuhrer, T. Gysi, C. Osuna Escamilla, T. C. Schulthess

The dynamical core of many weather and climate simulation models are implemented as stencil methods solving partial differential equations (PDEs) on structured grids. STELLA has been developed as a domain-specific language, based on C++ template metaprogramming, for stencil codes on structured grids. The library abstracts the loop logic and parallelization of the stencils as well as other hardware-dependent optimizations like memory layout of data fields, loop and kernel fusion or software manages cache techniques. We show the use and performance of different parallelization algorithms within STELLA, like a parallel tridiagonal solver. These new parallelization modes increase the level of parallelism in GPUs for the algorithmic motifs used in COSMO, which improves the strong scaling behaviour and therefore time to solution on real use cases. A full rewrite of the COSMO dynamical core shows the usefulness of STELLA for production codes, when stencil computations often have to interoperate with other parts of the model using different programming models and even programming languages. Using STELLA, we achieved a speedup factor of 1.8x for CPUs and 5.8x for NVIDIA GPUs for the dynamical core of COSMO.

### 3.15 Designing GridTools

*Mauro Bianco (Eidgenössische Technische Hochschule – Zürich, CH)*

**License** © Creative Commons BY 3.0 Unported license  
© Mauro Bianco

The complexity and diversity of contemporary computer architectures make the effort of developing portable and efficient monolithic scientific applications a challenging endeavor. An application may use different algorithmic motifs, which have different requirements and optimal solution strategies.

Previous experience, such as STELLA, showed that investing in generalizing high-level application-specific libraries for portions of an application, provides performance portability. Thanks to this, the application can exploit new energy-efficient architectures, thus enabling innovative solutions of scientific problems, like increasing the resolution of numerical simulations for weather forecast.

GridTools is a set of C++ generic APIs to encapsulate the main algorithmic motifs in grid applications, such as weather and climate simulations. Specifically, GridTools provides a DSL for stencils computations, plus other facilities for halo-exchange communications, boundary conditions treatment, etc. This allows an application to be specified at high level and, at the same time, take advantage of the diverse architectural features, such as, multiple address spaces in modern computing nodes.

By means of its clean design and concepts, and being written in a very well established programming language, the GridTools API set is engineered to be expanded and improved over time, providing production a quality implementation of its components.

### 3.16 Redesign of preconditioned Krylov methods around stencil compilers

*Wim Vanroose (University of Antwerp – Antwerp, BE)*


License  Creative Commons BY 3.0 Unported license  
© Wim Vanroose

Joint work of S. Donfack, P. Ghysels, B. Reys, O. Schenk, W. Vanroose

The performance of preconditioned Krylov solvers is severely hampered by the limited memory bandwidth. Each of the building blocks of a multigrid preconditioned Krylov solver is an operation of low arithmetic intensity. We discuss how the algorithm can be organized using stencil compilers such the arithmetic intensity is raised, so we can benefit from SIMD operations.

### 3.17 PolyMage: High-performance compilation for heterogeneous stencils

*Uday Bondhugula (Indian Institute of Science – Bangalore, IN)*

License  Creative Commons BY 3.0 Unported license  
© Uday Bondhugula

Joint work of U. Bondhugula, R. T. Mullapudi, V. Vasista

This talk presents the design and implementation of PolyMage, a domain-specific language and compiler for image processing pipelines. An image processing pipeline can be viewed as a graph of interconnected stages which process images successively. Each stage typically performs one of point-wise, stencil, reduction or data-dependent operations on image pixels. Individual stages in a pipeline typically exhibit abundant data parallelism that can be exploited with relative ease. However, the stages also require high memory bandwidth preventing effective utilization of parallelism available on modern architectures. For applications that demand high performance, the traditional options are to use optimized libraries like OpenCV or to optimize manually. While using libraries precludes optimization across library routines, manual optimization accounting for both parallelism and locality is very tedious.

The focus of our system, PolyMage, is on automatically generating high-performance implementations of image processing pipelines expressed in a high-level declarative language. Our optimization approach primarily relies on the transformation and code generation capabilities of the polyhedral compiler framework. To the best of our knowledge, this is the first model-driven compiler for image processing pipelines that performs complex fusion, tiling, and storage optimization automatically. Experimental results on a modern multicore system show that the performance achieved by our automatic approach is up to 1.81x better than that achieved through manual tuning in Halide, a state-of-the-art language and compiler for image processing pipelines. For a camera raw image processing pipeline, our performance is comparable to that of a hand-tuned implementation.

### 3.18 On the characterization of the data movement complexity of algorithms

*P. Sadayappan (Ohio State University – Columbus, OH, US)*

**License** © Creative Commons BY 3.0 Unported license

© P. Sadayappan

**Joint work of** V. Elango, L.-N. Pouchet, J. Ramanujam, F. Rastello, P. Sadayappan

Data movement costs represent a significant factor in determining the execution time and energy expenditure of algorithms on current/emerging computers. Hence, characterizing the data movement cost is becoming increasingly important. This talk introduces the problem of finding lower bounds on the data movement complexity of algorithms and summarizes recent progress along multiple directions: a new approach to lower bounds using graph min-cut, automated analysis of affine codes for parametric asymptotic characterization of data movement lower bounds, and algorithm-architecture co-design using lower bounds on data movement.

### 3.19 The mathematics of ExaStencils

*Hannah Rittich (Bergische Universität Wuppertal – Wuppertal, DE)*

**License** © Creative Commons BY 3.0 Unported license

© Hannah Rittich

**Joint work of** M. Bolten, K. Kahl, H. Rittich

Local Fourier analysis (LFA) is a well known tool for analyzing and predicting the convergence behavior of multigrid methods. Originally, LFA has been introduced to analyze operators with constant coefficients. However, for some problems this assumption is too restrictive. We present a generalization of LFA that allows to analyze more general operators. The coefficients of these operators may vary in a block structured way. This enables us to analyze complex operators like operators with jumping coefficients and block smoothers.

### 3.20 Stencils for hierarchical bases

*Dirk Pflüger (Universität Stuttgart – Stuttgart, DE)*

**License** © Creative Commons BY 3.0 Unported license

© Dirk Pflüger

The discretization and solution of higher-dimensional PDEs suffers the curse of dimensionality. In these settings, hierarchical approaches such as sparse grids can help to push the limit of the dimensionality that can be dealt with. Sparse grids are based on hierarchical basis functions with local support and a truncation of the resulting expansion into higher-dimensional increment spaces. However, a FEM approach leads to matrices that are no longer sparse and to bilinear forms that result in “hierarchical stencils”. Thus, solution techniques and data structures that are optimized for sparse linear systems cannot be applied any more. But algorithms for the matrix-vector multiplication in optimal (linear) complexity do exist, even though an explicit assembly of the corresponding matrix would lead to more non-zero entries. In this talk, we present the challenges that arise when employing hierarchical bases and discuss the current state of the art with respect to the solution of PDEs. Novel ideas from other fields are more than welcome.

### 3.21 Mapping stencils to FPGAs and synthesizable accelerators

*Louis-Noel Pouchet (Ohio State University – Columbus, OH, US)*


License  Creative Commons BY 3.0 Unported license  
© Louis-Noel Pouchet

Stencils are recurring patterns in numerous application domains, ranging from image processing to PDE solving. The medical imaging domain leverages numerous algorithms using stencils, and achieving portable high performance for these codes requires an integrated approach, from application modeling to low-level compiler optimization and hardware design.

In this talk, I present an overview of the research at the Center for Domain-Specific Computing, which aims at accelerating medical imaging applications by combining domain-specific modeling of applications, domain- and pattern-specific optimizing compilers, and a customizable heterogeneous computing platform. Focus will be made on the high-level modeling of the application using macro-dataflow concepts, and domain-specific optimization for stencils on CPUs and FPGAs.

### 3.22 The stencil optimization framework MODESTO

*Tobias Grosser (Eidgenössische Technische Hochschule – Zürich, CH)*

License  Creative Commons BY 3.0 Unported license  
© Tobias Grosser

Joint work of Tobias Grosser, Tobias Gysi, Torsten Hoefler

Code transformations, such as loop tiling and loop fusion, are of key importance for the efficient implementation of stencil computations. However, their direct application to a large code base is costly and severely impacts program maintainability. While recently introduced domain-specific languages facilitate the application of such transformations, they typically still require manual tuning or auto-tuning techniques to select the transformations that yield optimal performance. We introduce MODESTO, a model-driven stencil optimization framework, that for a stencil program suggests program transformations optimized for a given target architecture. Initially, we review and categorize data locality transformations for stencil programs and introduce a stencil algebra that allows the expression and enumeration of different stencil program implementation variants. Combining this algebra with a compile-time performance model, we show how to automatically tune stencil programs. We use our framework to model the STELLA library and optimize kernels used by the COSMO atmospheric model on multi-core and hybrid CPU-GPU architectures. Compared to naive and expert-tuned variants, the automatically tuned kernels attain a 2.0–3.1x and a 1.0–1.8x speedup, respectively.

### 3.23 Autotuning divide-and-conquer stencil computations

*Ekanathan Palamadai Natarajan (Massachusetts Institute of Technology – Boston, MA, US)*

License  Creative Commons BY 3.0 Unported license  
© Ekanathan Palamadai Natarajan

Joint work of C. E. Leiserson, E. Palamadai Natarajan

Ztune is an application-specific autotuner for optimizing serial divide-and-conquer stencil computations modeled on the trapezoidal-decomposition algorithm due to Frigo and Strumpen. Each recursive step of the algorithm divides a space-time hypertrapezoidal region, called a

“zoid”, into subzoids, or if the size of the zoid falls beneath a given threshold, executes a base case that loops across the space-time dimensions to compute the stencil at each point in the zoid. Ztune defines a search domain that generalizes this algorithm, where at each zoid in the recursion tree, a parameter is created that chooses whether to divide or perform the base case and, if divide, chooses the space-time dimension to be divided. Ztune searches this domain of possible choices to find the fastest plan for executing the stencil computation. Although Ztune, in principle, performs an exhaustive search of the search domain, it uses three properties – space-time equivalence, divide subsumption, and favored dimension – to prune the search domain. These three properties reduce the autotuning time by orders of magnitude without significantly sacrificing runtime.

We implemented Ztune to autotune the Trap algorithm used in the open-source Pochoir stencil compiler (disabling parallelism). We then compared the performance of the Ztuned code with that of Pochoir’s default code on nine application benchmarks across two machines with different hardware configurations. On these benchmarks, the Ztuned code ran 5%–8% faster (geometric mean) than Pochoir’s hand-optimized code.

We also compared Ztune with state-of-the-art heuristic autotuning. The sheer number of choice parameters in Ztune’s search domain, however, renders naive heuristic autotuning infeasible. Consequently, we used the open-source OpenTuner framework to implement a heuristic autotuner called Otune that optimizes only the size of the base case along each dimension. Whereas the time for Ztune to autotune each of the benchmarks could be measured in minutes, to achieve comparable results, the autotuning time for Otune was typically measured in hours or days. Surprisingly, for some benchmarks, Ztune actually autotuned faster than the time it takes to perform the stencil computation once.

### 3.24 Things you can do with stencils

*Gabriel Wittum (Goethe-Universität – Frankfurt, DE)*

**License** © Creative Commons BY 3.0 Unported license  
© Gabriel Wittum

Numerical simulation with supercomputers has become one of the major topics in computational science. To promote modelling and simulation of complex problems, new strategies are needed allowing for the solution of large, complex model systems. Crucial issues for such strategies are reliability, efficiency, robustness, scalability, usability, and versatility.

After introducing some basic notions of stencil notation and calculus, we discuss what is necessary for computing with a fixed or almost fixed stencil. To demonstrate this, we discuss the computation of seiches of Lake Constance using a domain adapted but structured grid, which still yields an operator with almost fixed pattern.

We discuss advantages and disadvantages and show that a numerical approach combining adaptivity, parallelism and multigrid methods allows for extreme parallel scalability while still maintaining flexibility to adapt numerical methods to the problem is more general and allows to gain acceleration by factors of up to  $10^6$  using sensible adaptive numerics. These strategies are combined in the novel simulation system UG 4 (“Unstructured Grids”).



### 3.25 Optimization of higher-order stencils

*P. Sadayappan (Ohio State University – Columbus, OH, US)*

License  Creative Commons BY 3.0 Unported license  
© P. Sadayappan

Joint work of M. Kong, L.-N. Pouchet, J. Ramanujam, F. Rastello, P. Sadayappan, K. Stock

It is well known that the associativity of operations like addition and multiplication offer opportunities for reordering of operations to enable better parallelization. We discuss a complementary use of associativity of operations: to improve data locality. We consider the optimization of high-order (or long-range) stencil computations. High-order stencil computations exhibit a much higher operational intensity (ratio of arithmetic operations to data moved from/to main-memory) than simple nearest-neighbor stencil operations. Although high-order stencil computations are not memory-bandwidth-bound due to high operational intensity, they nevertheless do not achieve much higher performance than low-order stencils. This is because of severe register pressure. We discuss an associative reordering strategy that interleaves computations targeting multiple neighboring grid sites and thereby significantly reduces register pressure. Experimental results demonstrate significant performance improvement through use of the associative reordering.

### 3.26 DSL for stencils in non-Newtonian fluids simulation?

*Gundolf Haase (Karl-Franzens-Universität – Graz, AT)*

License  Creative Commons BY 3.0 Unported license  
© Gundolf Haase

Joint work of D. Vasco

The talk introduces the formulation of non-Newtonian fluids as a coupled system of non-linear PDEs. The included Navier-Stokes equations contain an additional non-Newtonian term originating from the non-linear shear-stress tensor. The non-linear system of equations in each time step is solved by the SIMPLE algorithm and the discretization is the 7-point stencil taking into account the staggered grid for the pressure and the temperature.

Several issues for DSLs in this context are discussed including the automatic generation of a geometric multigrid for the coupled equations of the linearized system of equations. The final goal would be a full approximation scheme (FAS) automatically generated for multicore CPUs/GPUs/Xeon Phi and MPI.

## 4 Evening Discussion Sessions

### Stencil Codes: Walls, Challenges, Goals

The initial goal of the discussion was to pinpoint the problems which stand in the way of the scalability of stencil codes and the productivity of their engineering. Ulrich Rüde pointed out that when solving linear systems where the system matrix is described by a stencil using direct methods is not feasible as the inverse of such a matrix is usually almost dense. Using separators does not solve this problem either: because the Schur complement is usually dense, an iterative solution results in too many iterations, thus multigrid approaches are needed. While asymptotically multigrid is the fastest solver in many cases, in practice they

are sometimes not the best choice as the constants involved depend on the problem at hand and not only the algorithm determines complexity.

In many cases Krylov subspace methods are used, but Saday Sadayappan mentioned that Krylov subspace methods are inefficient with respect to data movement. Another issue is the missing algorithmic scalability, i.e., the dependence of the number of iterations on the problem size. This could be overcome by multigrid preconditioning but essentially this puts the hierarchy necessary to tackle the problem in the preconditioner.

Paul Kelly and Gabriel Wittum further explained that the performance of the method heavily depends on the problem, e.g., in compressible or incompressible flows or when dealing with parabolic or hyperbolic PDEs. Christian Engwer noted that the building blocks are the same, although as countered by Saday Sadayappan the applications are different.

Harald Köstler added that people from both the compiler community and applied mathematics are needed to solve a problem. Further, at the interface of both worlds, experts are needed. Saday Sadayappan replied that the computer science people only need to know about the loops. It was argued that the algorithmic choice heavily influences the time to solution but this could be handled by optimizing all different algorithmic options.

According to Franz Franchetti one should start with a short compact way to describe the algorithm and optimize from there. Harald Köstler added that actually a top-down approach starting from the model is the right approach. According to Christian Engwer the optimal way would be if mathematicians provide a bilinear form plus information about the mesh and that computer scientists optimize starting from this. Therefore a rich language is needed that stated by Armando Solar-Lezama should be high-level enough. Sven Apel added that a domain scoping is necessary and that abstraction hides complexity. In Franz Franchetti's opinion several layers of DSLs are needed. In the following discussion it was noted that designing a DSL targeting HPC is difficult as some concepts are not scalable or optimizable. Further, it was added that currently hand-written and generated codes are compared. Understanding the domain makes creating optimized implementations easier, resulting in a huge increase of productivity. The necessary domain-specific optimization needs semantic knowledge or should at least benefit from it stated Paul Kelly. Franz Franchetti closed the discussion by pointing out that one has to be able to express what is known.

## Suitable Representations of Stencil Codes

One issue at the seminar was multi-layered domain-specific optimization. Three approaches were presented: FireDrake, SPIRAL and ExaStencils. Each one offers at the most abstract, a language of mathematical expressions in the considered domain and at the more concrete levels executable representations that include successively aspects of the computation and architecture.

Starting point of the discussion was a slide (slide 20) of Franz Franchetti's talk showing the SPIRAL languages at different levels of abstraction for the three domains that SPIRAL has handled in the past (linear transforms, linear algebra, ...). In his talk, Franchetti had left open whether such representations could be developed for multigrid solving.

The second-most abstract and first executable level in SPIRAL has a representation in point-free combinator style. This means that the equations are in the space of the functions themselves and not in the range of the functions. Syntactically it is distinguished by the absence of function arguments in the expressions. It was explored whether and how this could be achieved for multigrid. The issue of matrix-freedom was also discussed. Matrix-freedom

gets rid of the need to store intermediate matrices in a computation. The conclusion was that multigrid solving seems to lend itself well to the SPIRAL style of software engineering. This will be explored in further collaborations.

A discussion as to the sensibility of restricted domains ensued. The criticism voiced was that a domain-specific approach has little relevance because much – maybe, most – application software will not be covered by it. It was countered by two arguments: (1) it is unlikely that a general-purpose software technology can be as powerful as a domain-specific one and (2) if the domain has a wide enough market, there is no reason to penalize its customers with a software technology that is less effective than it could be, just because others will not be able to benefit from it.

### Future Collaborations

One major goal of the seminar was to encourage participants of the different research communities to collaborate on stencil research in the future. Paper titles for a potential postproceedings volume were negotiated and collected in a discussion session on Thursday evening. Until its appearance, details remain confidential.

## 5 Conclusions

With 46 participants, the seminar was fully booked. A good spread of contemporary projects on stencil codes and high-performance DSLs was represented:

- from SPPEXA: ExaStencils, EXA-DUNE, TERRA-NEO, and ESSEX
- from elsewhere: Pochoir, PATUS, SPIRAL, STELLA, PolyMage, PyOP2, Polly

The main challenge was to reach an understanding between the members of the mutual needs of the diverse research communities – math, CS, applications. The culmination of the seminar’s success would consist of the appearance of a postproceedings and of sustained future collaborations.

## Participants

- Sven Apel  
Universität Passau, DE
- Mauro Bianco  
CSCS – Lugano, CH
- Matthias Bolten  
Bergische Univ. Wuppertal, DE
- Uday Bondhugula  
Indian Institute of Science –  
Bangalore, IN
- Rezaul Chowdhury  
Stony Brook University, US
- Simplicio Donfack  
University of Lugano, CH
- Christian Engwer  
Universität Münster, DE
- Robert D. Falgout  
LLNL – Livermore, US
- Franz Franchetti  
Carnegie Mellon University –  
Pittsburgh, US
- Michael Freitag  
Universität Passau, DE
- Stephanie Friedhoff  
KU Leuven, BE
- Francisco Jose Gaspar  
University of Zaragoza, ES
- Björn Gmeiner  
Univ. Erlangen-Nürnberg, DE
- Alexander Grebhahn  
Universität Passau, DE
- Armin Größlinger  
Universität Passau, DE
- Tobias Grosser  
ETH Zürich, CH
- Gundolf Haase  
Universität Graz, AT
- Georg Hager  
Univ. Erlangen-Nürnberg, DE
- Frank Hannig  
Univ. Erlangen-Nürnberg, DE
- Juraj Kardos  
Brno Univ. of Technology, CZ
- Paul H. J. Kelly  
Imperial College London, GB
- Hans-Peter Kersken  
DLR – Köln, DE
- Harald Köstler  
Univ. Erlangen-Nürnberg, DE
- Stefan Kronawitter  
Universität Passau, DE
- Sebastian Kuckuk  
Univ. Erlangen-Nürnberg, DE
- Bradley C. Kuszmaul  
MIT – Cambridge, US
- Christian Lengauer  
Universität Passau, DE
- Dmitry Mikushin  
University of Lugano, CH
- Marcus Mohr  
LMU München, DE
- Carlos Osuna Escamilla  
ETH Zürich, CH
- Ekanathan Palamadai  
Natarajan  
MIT – Cambridge, US
- Dirk Pflüger  
Universität Stuttgart, DE
- Louis-Noël Pouchet  
Ohio State University –  
Columbus, US
- Hannah Rittich  
Bergische Univ. Wuppertal, DE
- Carmen Rodrigo Cardiel  
University of Zaragoza, ES
- Ulrich Rüdiger  
Univ. Erlangen-Nürnberg, DE
- P. Saday Sadayappan  
Ohio State University –  
Columbus, US
- Olaf Schenk  
University of Lugano, CH
- Christian Schmitt  
Univ. Erlangen-Nürnberg, DE
- Armando Solar-Lezama  
MIT – Cambridge, US
- Jürgen Teich  
Univ. Erlangen-Nürnberg, DE
- Wim Vanroose  
University of Antwerp, BE
- Christian Waluga  
TU München, DE
- Gabriel Wittum  
Universität Frankfurt, DE
- Barbara Wohlmuth  
TU München, DE
- Ulrike Meier Yang  
LLNL – Livermore, US

