Report from Dagstuhl Seminar 20111

Tensor Computations: Applications and Optimization

Edited by

Paolo Bientinesi[1], David Ham[2], Furong Huang[3], Paul H. J. Kelly[4], Christian Lengauer[5], and Saday Sadayappan[6]

1 **Umeå University, Sweden,** pauldj@cs.umu.se
2 **Imperial College London, GB,** david.ham@imperial.ac.uk
3 **University of Maryland, USA,** furongh@cs.umd.edu
4 **Imperial College London, GB,** p.kelly@imperial.ac.uk
5 **University of Passau, DE,** christian.lengauer@uni-passau.de
6 **University of Utah and Pacific Northwest National Laboratory, USA,** saday@cs.utah.edu

Abstract

Tensors are higher-dimensional analogs of matrices, and represent a key data abstraction for many applications in computational science and data science. In contrast to the wide availability on diverse hardware platforms of high-performance numerical libraries for matrix computations, only limited software infrastructure exists today for high-performance tensor computations.

Recent research developments have resulted in the formulation of many machine learning algorithms in terms of tensor computations. Tensor computations have also emerged as fundamental building blocks for many algorithms in data science and computational science. Therefore, several concurrent efforts have targeted the development of libraries, frameworks, and domain-specific compilers to support the rising demand for high-performance tensor computations. However, there is currently very little coordination among the various groups of developers. Further, the groups developing high-performance libraries/frameworks for tensor computations are still rather disconnected from the research community that develops applications using tensors as a key data abstraction.

The main goal of this Dagstuhl Seminar has been to bring together the following two communities: first researchers from disciplines developing applications centered around tensor computations, and second researchers developing software infrastructure for efficient tensor computation primitives. Invitees from the former group included experts in machine learning and data analytics, and computational scientists developing tensor-based applications. Invitees from the latter group spanned experts in compiler optimization and experts in numerical methods.

A very fruitful exchange of ideas across these four research communities took place, with discussions on the variety of needs and use-cases for tensor computations and the challenges/opportunities in the development of high-performance software to satisfy those needs.

Seminar March 8–13, 2020 – http://www.dagstuhl.de/20111
2012 ACM Subject Classification Information systems → Data structures, Software and its engineering → Compilers
Keywords and phrases compilers, computational science, linear algebra, machine learning, numerical methods
Digital Object Identifier 10.4230/DagRep.10.3.58
Edited in cooperation with Vorderwuelbecke, Sophia

Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license
Tensor Computations: Applications and Optimization, *Dagstuhl Reports*, Vol. 10, Issue 3, pp. 58–70
Editors: Paolo Bientinesi, David Ham, Furong Huang, Paul H. J. Kelly, Christian Lengauer and Saday Sadayappan

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

1 Executive Summary

Paolo Bientinesi David Ham
Furong Huang
Paul H. J. Kelly
Christian Lengauer
Saday Sadayappan

License Creative Commons BY 3.0 Unported license
© Paolo Bientinesi, David Ham, Furong Huang, Paul H. J. Kelly, Christian Lengauer and Saday Sadayappan

This seminar was planned for 40 participants, but due to travel restrictions resulting from Covid-19, only 15 were able to attend – though several key talks were delivered via teleconferencing. As a result, the Seminar was very focused, and very productive. Some aspects were lost, perhaps in particular representation in-person of the full breadth of applications communities.

It was very evident from the presentations and lively discussions at the Seminar that the field of "Tensor Computations" is vibrant, multi-faceted, interdisciplinary, and fundamental to progress in a diverse range of important areas which are driving researchers in different fields to search for common foundations and common tools.

One of the communities with an interest in tensor computations can be described as "classical" computational science, focusing, for example, on partial differential equations in fluid dynamics, and electronic structure computations in chemistry and materials science. Tensor contractions have been identified as a powerful way of representing the computational structure in the architecture of compilers for domain-specific languages serving these communities. Exploiting this algebraic intermediate representation in the compiler has enabled important performance optimizations far beyond the scope of conventional compilers based on loop nests and polyhedral techniques.

Another major community is primarily concerned with tensor decomposition – finding low-rank approximations of tensors. This is fundamental to data analytics and machine learning applications. Tensor factorization also provides a powerful framework for deep learning and representation learning, and provides a promising strategy for weight compression in convolutional neural networks.

Tensor contractions, in the form of tensor networks, have enormous importance as a tool for understanding and computation in particle and quantum physics. Indeed mapping the connections between these topics, as exposed through the structure of the tensor network representation, offers an exciting frontier with the potential to underpin these different disciplines with common language and shared software.

The Seminar developed a focus, to some extent as a result of the participants able to attend, on tensor contractions, recognising that this provides a foundation for implementation of numerical methods for tensor decompositions. Revisiting this is a key topic to be addressed in following up this Seminar in the future.

A major focus for progress was identified in characterization of safety and correctness properties – ensuring that tensor contraction expressions are well-formed and meaningful. A related topic that was identified as critical concerns how structure is captured, represented and used. This is not only conceptually valuable, but provides a pathway to exploiting block, band and symmetry structure in generating efficient code.

An open question remains in how to capture, track and exploit the properties of tensors with unstructured (i.e. data-dependent) sparsity.

A key outcome from the Seminar was to recognize the massive replication of efforts in terms of software development. Many tools and libraries are being re-developed within different communities, while failing to share techniques and experience in high-performance implementation. The aim of this Seminar was to address this lack of cohesive, coherent community effort to develop computational building blocks. The results of this effort are being realized in the form of a "white paper", offering a manifesto for how to bridge the discipline divides and realize the potential for tensor computations in the future.

2 Table of Contents

3 Overview of Talks

3.1 Building blocks: from matrix to tensor operations

Paolo Bientinesi and Lars Karlsson (Umeå University, SE)

License Creative Commons BY 3.0 Unported license
© Paolo Bientinesi and Lars Karlsson

The entire domain of matrix computations is tightly connected to the concept of building blocks, i.e., computational kernels that support one well defined mathematical operation. For almost 50 years, the linear algebra community has been identifying, layering, implementing, and optimizing building blocks. Significant benefits include portable performance, self-documenting quality of code, and robust implementations. Furthermore, standardization and wide adoption of interfaces paved the road towards automated approaches to code generation and performance tuning, and enabled abstraction (via high-level languages). Nowadays there exists a sophisticated and comprehensive hierarchy of libraries for dense and sparse linear algebra (e.g., BLAS, LAPACK, PETSc, etc.); these libraries provide invaluable support for a vast ecosystem of applications.

We are convinced that the tensor community could benefit from similar ideas. The need for a better computational infrastructure was publicly recognized already in 2009 (if not earlier) at a workshop organized by Charles Van Loan [1]. Despite many years of development, in 2020 the software landscape for tensor computations is still heavily fragmented and dominated by sophisticated MATLAB toolboxes and niche C++ libraries. Libraries similar to the BLAS and LAPACK are not even on the radar. We believe that it is (still) time for a major community effort to agree on the functionality and possibly the interface of a few low-level tensor operations, to then focus on high performance and parallel scalability.

References

1 E Acar, O Alter et al. *Workshop Report: Future Directions in Tensor-Based Computation and Modeling (2009).* https://www.cs.cornell.edu/cv/TenWork/FinalReport.pdf 10.13140/2.1.4040.4807

3.2 Theory and practice of tensor computations

Peter Braam (University of Oxford, GB)

License Creative Commons BY 3.0 Unported license
© Peter Braam

Tensor computation packages usually silently include and omit easily identified components such as DSLs or notation, reduction of arithmetic intensity, data and index organization and code generation and optimizations for hardware platforms. In many new domains, the existing features offered by such components don't suffice as illustrated by the failure to find a package for the significant computations required by the SKA telescope. We will review the components and ask if there is a structure for these packages that is more broadly useful across domains like there is for compilers, operating systems etc?

3.3 A universal representation for safe tensor computations

Charisee Chiw (Galois – Portland, US)

License Creative Commons BY 3.0 Unported license
© Charisee Chiw
Joint work of Charisee Chiw, Eric Davis

My previous work involved representing, compiling, generating, and testing code for tensor computations. I'm wondering if some of those tasks are shared among the participants at the seminar, and I expect they have been. My goal is to find out, if there is a way to leverage all the work being done in this field, into a unifying framework.

I worked on Diderot [1], a domain-specific language designed for scientific visualization and image analysis. New Visualization techniques are implemented to improve what we can understand from the data. Without our DSL, the user would need to write a lot of bespoke python code that can be error-prone and tedious. To enable new types of algorithms the scientists would compute mathematically complicated computations. The DSL we created ended up needing to be retargeted. Instead of it being a DSL designed to enable visualization, it needed to be a DSL designed for math, specifically tensor math.

Our design goals included tensor operations on and between tensors and tensor fields including differentiation. The compiler shouldn't just stop working with a certain combination of operators, and shapes, which is what some tools currently do. Our new goal was to support a flexible, rich tensor calculus based language. To enable that, we designed EIN, an intermediate representation based on Einstein Index Notation. We adapted regular compiler techniques and leveraged domain-specific optimizations to move past compilation issues. For us, correctness meant creating this automatic test generation tool, DATm. DATm compared the output of tensor computations to the output when using a symbol tool in Python.

Working at Galois, I've wondered if Diderot's standard of correctness holds up. There are various strategies in this avenue; coq is considered the gold standard, function verification can help prove properties about data, automatic test generation (metamorphic, differential testing), policy enforcement, and user defined DSLs.

Could we create a generic tensor representation framework? Once we have a universal framework, can we significantly lower the lift to add new backends, add optimization passes, and do test case generation? Can we add a separate parallel track to check for correctness of our tensor computations? Connecting our ideas will both improve the expressivity and performance we have in various user-facing DSLS and tools. By losing our domain knowledge, are we losing performance-enhancing knowledge? If so, can we tag the data and/or computations to have a certain property so it hints to the compiler what optimization and execution paths to take?

References

1 Kindlmann, Gordon & Chiw, Charisee & Seltzer, Nicholas & Samuels, Lamont & Reppy, John. (2015). *Diderot: A Domain-Specific Language for Portable Parallel Scientific Visualization and Image Analysis*. IEEE Transactions on Visualization and Computer Graphics. 22. 1-1. doi:10.1109/TVCG.2015.2467449.

3.4 Low tensor rank approximations and computations, a short introduction

Jeremy Cohen (CNRS – IRISA – Rennes, FR)

License Creative Commons BY 3.0 Unported license
© Jeremy Cohen

Tensor decompositions can be seen as a field with specific, complicated notions such as tensor rank, and even more complicated notations. In this talk, I try to explain in simple terms, using matrix factorizations as a starting point, where these notions and notations come from. Moreover, I show how in particular the MTTKRP appears when trying to solve the tensor low-rank approximation problem.

3.5 Efficient implementation of tensor contractions for electronic structure computations

Anna Engels-Putzka (DLR – Köln, DE)

License Creative Commons BY 3.0 Unported license
© Anna Engels-Putzka
Joint work of Anna Engels-Putzka, Michael Hanrath
Main reference Michael Hanrath, Anna Engels-Putzka: "An efficient matrix-matrix multiplication based antisymmetric tensor contraction engine for general order coupled cluster", J Chem Phys. 2010;133(6):064108
URL https://doi.org/10.1063/1.3467878

The equations of wave function-based methods in quantum chemistry, like Configuration interaction (CI) or Coupled Cluster (CC), consist of products of one or more excitation operator coefficients (amplitudes) with matrix elements of the Hamilton operator (one- and two-electron integrals), which can be interpreted as tensor contractions.

Since the terms have to be evaluated multiple times during the iterative solution of the equations, an efficient implementation of these contractions is crucial for such methods, in particular for high excitation levels.

The talk discusses two aspects of an implementation aimed at arbitrary excitation levels, on the one hand the determination of an optimized contraction order and choice of intermediates (together referred to as factorization), on the other hand the contractions themselves.

Special emphasis is put on the internal structure of the tensors, which leads to a conflict between storage efficiency and easy accessibility of the entries. On the one hand, there is the antisymmetry with respect to certain index permutations due to the indistinguishability of the electrons, which leads to many entries being redundant or zero. On the other hand, if spatial symmetries of the considered molecule or certain spin properties of the wave function are exploited, only those tensor entries are non-zero where the indices fulfil certain restrictions. For contractions, the tensor entries are rearranged so that the actual multiplication can be carried out as a sequence of highly efficient matrix-matrix multiplications.

By using optimized addressing structures, the overhead of this step can be reduced such that – for sufficiently large problems – the matrix multiplication dominates the total computational costs.

References

1 A. Engels-Putzka and M. Hanrath. *A fully simultaneously optimizing genetic approach to the highly excited coupled-cluster factorization problem*. The Journal of Chemical Physics 134, 124106, 2011

2 M. Hanrath and A. Engels-Putzka. *An efficient matrix-matrix multiplication based antisymmetric tensor contraction engine for general order coupled cluster*, The Journal of Chemical Physics 133, 064108, 2010

3.6 Understanding generalization in deep learning through tensor methods

Furong Huang (University of Maryland – College Park, US)

License Creative Commons BY 3.0 Unported license
© Furong Huang
Joint work of Jingling Li, Yanchao Sun, Jiahao Su, Taiji Suzuki, Furong Huang
Main reference Jingling Li, Yanchao Sun, Jiahao Su, Taiji Suzuki, Furong Huang: "Understanding Generalization in Deep Learning via Tensor Methods", CoRR, Vol. abs/2001.05070, 2020.
URL https://arxiv.org/abs/2001.05070

Deep neural networks generalize well on unseen data though the number of parameters often far exceeds the number of training examples. We advance the understanding of the relations between the network's architecture and its generalizability from the compression perspective. Using tensor analysis, we propose a series of intuitive, data-dependent and easily-measurable properties that tightly characterize the compressibility and generalizability of neural networks; thus, in practice, our generalization bound outperforms the previous compression-based ones, especially for neural networks using tensors as their weight kernels (eg CNNs). Moreover, these intuitive measurements provide further insights into designing neural network architectures with properties favorable for better/guaranteed generalizability.

3.7 Tensors as morphisms in a linear fusion category: from mathematics to high performance numerics using Julia

Jutho Haegeman (Ghent University, BE)

License Creative Commons BY 3.0 Unported license
© Jutho Haegeman
URL https://github.com/Jutho/TensorKit.jl

I will discuss a Julia package for representing and manipulating tensors, where tensors can have an internal structure that is dictated by the rules of (unitary) tensor fusion categories. In particular, this encompasses tensors with a block sparsity pattern (corresponding to abelian symmetries), but also goes beyond this to the setting of non-abelian group symmetries as well as more general fusion categories with non-trivial braiding and twists, associated with the physics of fermions and anyons. Such tensors arise in the context of tensor network simulations in quantum physics. The implementation has elements from category theory built in at the lowest level, but tries to hide these from the user whenever possible, and at the same time aims at high efficiency and flexibility, e.g. to exploit multithreading and GPUs.

3.8 The tensor problems in finite element assembly

David Ham (Imperial College London, GB), Paul H. J. Kelly (Imperial College London, GB), Lawrence Mitchell (Durham University, GB), Tianjiao Sun, and Sophia Vorderwuelbecke (Imperial College London, GB)

License Creative Commons BY 3.0 Unported license
© David Ham, Paul H. J. Kelly, Lawrence Mitchell, Tianjiao Sun, and Sophia Vorderwuelbecke
Main reference Miklós Homolya, Lawrence Mitchell, Fabio Luporini, David A. Ham: "TSFC: A Structure-Preserving Form Compiler", SIAM J. Sci. Comput., Vol. 40(3), 2018.
URL http://dx.doi.org/10.1137/17M1130642

Finite element assembly can be represented as a sequence of (relatively) small tensor operations. I will discuss how the Firedrake system exploits tensor computation abstractions to achieve performance optimizations such as sum factorization and vectorization. I will also indicate the limitations of our current approach, in particular with respect to affine index extents.

3.9 Parallel non-negative CP decomposition of dense tensors

Koby Hayashi (Georgia Institute of Technology – Atlanta, US)

License Creative Commons BY 3.0 Unported license
© Koby Hayashi

The CP tensor decomposition is a low-rank approximation of a tensor. We present a distributed-memory parallel algorithm and implementation of an alternating optimization method for computing a CP decomposition of dense tensor data that can enforce nonnegativity of the computed low-rank factors. The principal task is to parallelize the matricized-tensor times Khatri-Rao product (MTTKRP) bottleneck subcomputation. The algorithm is computation efficient, using dimension trees to avoid redundant computation across MTTKRPs within the alternating method. Our approach is also communication efficient, using a data distribution and parallel algorithm across a multidimensional processor grid that can be tuned to minimize communication. We benchmark our software on synthetic as well as hyperspectral image and neuroscience dynamic functional connectivity data, demonstrating that our algorithm scales well to 100s of nodes (up to 4096 cores) and is faster and more general than the currently available parallel software.

3.10 Compiling for data not code

Paul H. J. Kelly (Imperial College London, GB)

License Creative Commons BY 3.0 Unported license
© Paul H. J. Kelly
Joint work of Paul H. J. Kelly, Thomas Debrunner, Sajad Saeedi, BD Wozniak, FD Witherden, FP Russell, PE Vincent, Mehedi Paribartan
Main reference Thomas Debrunner, Sajad Saeedi, Paul H. J. Kelly: "AUKE: Automatic Kernel Code Generation for an Analogue SIMD Focal-Plane Sensor-Processor Array", TACO, Vol. 15(4), pp. 59:1–59:26, 2019.
URL https://doi.org/10.1137/17M1130642

Compiling is like skiing: we struggle up the mountain, trying to analyse the program the programmer wrote. We build, at the mountaintop, a representation of everything we know. From this, we now can go downhill – synthesising optimized code, exploiting the space

of possible code generation alternatives allowed by the dependence information we have gathered and guided by other information, like sizes and loop bounds and data volumes. But in this talk I reflect on our experience in building various compilers whose input is not code, but data. I mentioned our AUKE tool for generation code for convolutions on the SCAMP-5 analog SIMD image sensor. I mentioned our GiMMiK code generator for small-matrix multiplications, as found in the PyFR tool for computational fluid dynamics. I talked about the libxsmm JIT, developed at Intel, for small matrix and convolution operations. However, this work starts with the raw numerical values – we are again struggling up the analysis mountain. We should aim instead to take a helicopter ride to the top – and start from a compact, abstract description of the structure of the data.

3.11 Representing structure in computational meshes

Lawrence Mitchell (Durham University, GB)

License Creative Commons BY 3.0 Unported license
© Lawrence Mitchell
Joint work of Lawrence Mitchell, Christian Engwer, Marcel Koch

Many mesh-based simulations occur on meshes with some regular structure. Examples include, but are not limited to: composite macro-elements and high continuity spline discretizations; regular refinement in geometric multigrid; structured mesh extrusion (common in ocean, ice sheet, and atmospheric modelling); and octree-like AMR.

An unanswered question is how to handle these and other kinds of structure in meshes in a composable way. For example, what if I want a macro-element on an extruded mesh must I program that case by hand?

I presented the way in which we (in the Firedrake project) represent unstructured meshes and provided some nascent musing on symbolic interfaces for code generators that might exploit some structure. A particular focus was maintaining the ability to perform topological queries of the "symbolic" meshes.

3.12 Safety and correctness: essential for tensor languages?

Norman Rink (TU Dresden, DE)

License Creative Commons BY 3.0 Unported license
© Norman Rink
Joint work of Norman A. Rink, Jeronimo Castrillon
Main reference Norman A. Rink, Jerónimo Castrillón: "TeIL: a type-safe imperative tensor intermediate language", in Proc. of the 6th ACM SIGPLAN International Workshop on Libraries, Languages and Compilers for Array Programming, ARRAY@PLDI 2019, Phoenix, AZ, USA, June 22, 2019, pp. 57–68, ACM, 2019.
URL http://dx.doi.org/10.1145/3315454.3329959

Recent years have seen an inflationary rise of tensor frameworks, each with its own tensor language (formerly "array language"). The programming language community has a long-standing tradition of analyzing languages, including array languages, for properties such as memory safety. In addition, the last two decades have produced significant work on provably correct compilers and runtime systems – in the sense that these tools adhere strictly to the definition of the language they implement. Little of this existing work on safety and correctness has found its way into the recent tensor frameworks or, more generally, into tools

that are used in developing tensor-based applications. With this talk, I would like to start a discussion about whether this should change. In other words, how much pain is caused by the lack of safety and correctness results for tensor or array languages and tools?

3.13 Representation learning and tensor factorization

Volker Tresp (Siemens AG – München, DE)

License Creative Commons BY 3.0 Unported license
© Volker Tresp

Representation learning is hugely successful in natural language processing and knowledge graph modelling. The basic concept is that entities or words are mapped to latent vectors of a given dimension and a shallow or a deep neural network then maps those representations to one or several probabilistic statements. We discuss relationships between representation learning and tensor factorization. We show how tensor embedding models can be used in selection tools for industrial products, in scene graph analysis in vision and in cognitive models for perception and memory.

References

1 Nickel, M., Tresp, V., & Kriegel, H. P. (2011, June). *A three-way model for collective learning on multi-relational data*. In Icml (Vol. 11, pp. 809-816).
2 Nickel, M., Murphy, K., Tresp, V., & Gabrilovich, E. (2015). *A review of relational machine learning for knowledge graphs*. Proceedings of the IEEE, 104(1), 11-33.
3 Yang, Y., Krompass, D., & Tresp, V. (2017, August). *Tensor-train recurrent neural networks for video classification*. In Proceedings of the 34th International Conference on Machine Learning-Volume 70 (pp. 3891-3900). JMLR. org.
4 Baier, S., Ma, Y., & Tresp, V. (2017, October). *Improving visual relationship detection using semantic modeling of scene descriptions*. In International Semantic Web Conference (pp. 53-68). Springer, Cham.
5 Tresp, V., Esteban, C., Yang, Y., Baier, S., & Krompaß, D. (2015). *Learning with memory embeddings*. arXiv preprint arXiv:1511.07972.

3.14 A BLAS for stencil computation

Richard M. Veras (Louisiana State Univ. – Baton Rouge, US)

License Creative Commons BY 3.0 Unported license
© Richard M. Veras

Stencil computations are an important class of problems that arise in a variety of fields where performance across a diverse set of architectures is critical. This includes areas ranging from physics, engineering, bioinformatics and machine learning. Thus any improvement in the efficiency of stencil computations yields increased computational productivity in those fields. Translating the regularity of a stencil into performance across SIMD vectors, multiple cores and deep memory hierarchies involves the coordination of many moving parts and requires knowledge of the target problem that may not be accessible at the compiler level. In this work we develop a systematic approach for generating high performance code for stencil

kernels from the parameters of the target architecture. We then propose a BLAS-like library for stencil computations that casts high-order and high-dimensional stencil operations in terms of these generated kernels. The result is performance portable stencil computations that reach near machine peak performance.

3.15 Slate: a system for linear algebra operaton tensor expressions of finite element problems

Sophia Vorderwuelbecke (Imperial College London, GB)

License Creative Commons BY 3.0 Unported license
© Sophia Vorderwuelbecke
Joint work of Thomas H. Gibson, Lawrence Mitchell, David A. Ham, Colin J. Cotter
Main reference T. H. Gibson, L. Mitchell, D. A. Ham, C. J. Cotter: "Slate: extending Firedrake's domain-specific abstraction to hybridized solvers for geoscience and beyond", Geoscientific Model Development, Vol. 13(2), pp. 735–761, 2020.
URL https://doi.org/10.5194/gmd-13-735-2020

Hybridization is a technique for mixed or discontinuous finite element problems, which loosens inter element coupling. It comes with the advantage that algebraic manipulations to produce condensed systems, involving expensive operations, can happen on a local level on smaller systems. Slate is a domain-specific language in the Firedrake framework encapsulating local tensor algebra on finite element expressions, for example for condensation. I explained key components, current drawbacks and potential optimizations of the language and its compiler.

References

1 Rathgeber, F., Ham, D. A., Mitchell, L., Lange, M., Luporini, F., McRae, A. T., Bercea, G.T., Markall, G.R., & Kelly, P. H. (2016). *Firedrake: automating the finite element method by composing abstractions*. ACM Transactions on Mathematical Software (TOMS), 43(3), 1-27.

2 Gibson, T. H., Mitchell, L., Ham, D. A., & Cotter, C. J. (2018). *A domain-specific language for the hybridization and static condensation of finite element methods*. arXiv preprint arXiv:1802.00303.

Participants

- Peter Braam
University of Oxford, GB
- Charisee Chiw
Galois – Portland, US
- Jeremy Cohen
CNRS – IRISA – Rennes, FR
- Anna Engels-Putzka
DLR – Köln, DE
- Jutho Haegeman
Ghent University, BE
- David Ham
Imperial College London, GB
- Koby Hayashi
Georgia Institute of Technology – Atlanta, US
- Paul H. J. Kelly
Imperial College London, GB
- Christian Lengauer
Köln, DE
- Lawrence Mitchell
Durham University, GB
- Norman Rink
TU Dresden, DE
- Volker Tresp
Siemens AG – München, DE
- Richard M. Veras
Louisiana State Univ. – Baton Rouge, US
- Frank Verstraete
Ghent University, BE
- Sophia Vorderwuelbecke
Imperial College London, GB