
Welcome

NSF CDI Workshop

Combining Symbolic, Numeric and Algebraic Computation:

Can Scientific Software keep up with Moore's Law?

A Transformational Grand Challenge

Workshop Organizers:

Erich Kaltofen, NC State

Lenore Mullin, NSF CISE CCF

Al Thaler, NSF MPS DMS



Agenda Day 1 AM

Final Schedule

8:30 - 9:00: Registration, coffee, juice, mini muffins (Room 1235), and Poster Set-Up (Atrium)

9:00 - 9:05: *Opening of Workshop* by Co-Chairs **Erich Kaltofen** (NCSU), **Lenore Mullin** (NSF CISE CCF), **Alvin Thaler** (NSF MPS DMS)

9:05 - 9:20: *Welcome* from **NSF Deputy Assistant Directors Debbie Crawford** (CISE) and **Jack Lightbody** (MPS)

9:20 - 9:35: *Welcome* from **NSF Division Directors Mike Foster** (CISE CCF) and **Executive Officer Deborah Lockhart** (MPS DMS)

9:35-10:00: *What is CDI?* **Tom Russell** Co-Chair **CDI Committee** (MPS DMS)



Agenda Day 1 AM

10:00-11:00: Panel I: *What are the Grand Challenges for Symbolic, Numeric and Algebraic Scientific Computing?*

Erich Kaltofen (Moderator): James Demmel, Randolph Franklin, Jeremy Johnson, Marianna Safronova, Jan Verschelde.

11:00-12:00: Panel II: *Applicable Computer Science Foundations*

Lenore Mullin (Moderator): Gene Copperman, Matthew Knepley, Jeremy Siek, Cleve Moler, Arun Rodrigues, Stephen Watt

12:00-2:00: Lunch at local restaurants



Agenda Day 1 PM

2:00-3:00: Panel III: *Directions Towards CDI & Beyond: Where Do We Go From Here?* Workshops, BoFs, Teaming

Alvin Thaler (Moderator); Gilbert Baumslag, Alyson Reeves; other members to be determined

Physics (Session Chair: Barry Schneider)

3:00-3:25: Invited Talk: Anthony Kennedy *Symbolic Supercomputing*

3:25-3:45: Break

Symbolic Computation (Session Chair: Chris Brown)

3:45-4:10: **Invited Talk:** Mike Dewar Tentative Title *Experiences with the UK High End Computing Terascale Resource (HECToR)*

4:10-4:35: **Invited Talk:** B. David Saunders *Why Solve Linear Systems Exactly?*

4:35-5:00: **Invited Talk:** Stephen Watt *The Role of Categorical Languages*

5:00-8:00: *Closing remarks* by NSF CISE Assistant Director Jeannette Wing

Followed by **Reception and *Poster Session* in the NSF Atrium.**

Posters by Chris Brown, Randolph Franklin, Robert Miner, David Padua, Marianna Safronova, Lihong Zhi; others TBA



Agenda Day 2 AM

8:30-9:00: Coffee, juice, biscotti

Computer Science (Session Chair: Robert Miner)

9:00-9:25: **Invited Talk:** David Padua *Software in the Era of Parallelism*

9:25-9:50: **Invited Talk:** Matthew Knepley (Argonne National Laboratories)
Refactoring Finite Element Computation

9:50-10:15: **Invited Talk:** Fernando Perez (Univ. Colorado) *Modern Algorithms in Mathematical Research, Parallelism and Languages: the Intersection of Theoretical and Practical Issues*

10:15-10:35: Break

Mathematics (Session Chair: Vicki Powers)

10:35-11:00 **Invited Talk:** Dan Grayson (Univ. Illinois Urbana-Champaign)
Cyber-Aided Mathematics

11:00-12:00 **Panel IV: Perspectives by Organizers and Committee Members:**
Erich Kaltofen, Lenore Mullin, Alvin Thaler (Co-Chairs), Lee Jameson (NSF MPS DMS), Jeremy Johnson (Drexel Univ.), Eun Park (NSF CISE CCF), Emil Volcheck (ACM), Stephen Watt (Univ. Western Ontario, Canada)



Panel II
Applicable Computer Science
Foundations

**Gene Cooperman, Mathew Knepley,
Jeremy Siek, Cleve Moler, Arun
Rodrigues, Stephen Watt**



Gene Cooperman

Northeastern University

Gene Cooperman has been a professor at the College of Computer and Information Science at Northeastern University since 1986. He works extensively in computational group theory, and also in high performance computing. He and his PhD students in the High Performance Computing Laboratory currently have a special interest in overcoming architectural bottlenecks for very large computations in computational group theory and related fields. Most recently, his group showed that 26 moves suffice to solve Rubik's cube. The two primary techniques were:

- (1) the use of seven terabytes of distributed disk space; and
- (2) a fast coset-generator multiplication algorithm reaching 10 million multiplications per second.



Jeremy Siek

University of Colorado

Jeremy Siek is an Assistant Professor at the University of Colorado at Boulder and a participant in the C++ Standards Committee. Jeremy's research interests are in generic programming, programming language design, type systems, and optimizing compilers for domain specific languages. Jeremy helped design the "concept" extension that will better support generic programming in the next version of the C++ Standard. Before that Jeremy developed several generic libraries including the Boost Graph Library and the Matrix Template Library.



Matt Knepley

Argonne National Laboratories

Matt Knepley received his doctorate in Computer Science from Purdue University in 2000 under the supervision of Ahmed Sameh. He then worked for Akamai Technologies before taking his current position in the MCS division of Argonne National Laboratory. He is also a visiting Research Professor at Rush University Medical Center.



Cleve Moler

MathWorks

Cleve Moler is chairman and chief scientist at The MathWorks. Moler was a professor of math and computer science for almost 20 years at the University of Michigan, Stanford University and the University of New Mexico. He spent five years with two computer hardware manufacturers, the Intel Hypercube organization and Ardent Computer, before joining The MathWorks full-time in 1989. In addition to being the author of the first version of MATLAB, Moler is one of the authors of the LINPACK and EISPACK scientific subroutine libraries. He is co-author of three textbooks on numerical methods.



Stephen Watt

University of Western Ontario

Stephen Watt is an internationally recognized expert in the area of software systems for computer algebra. He received his PhD from the University of Waterloo in 1986, where he was one of the original authors of Maple. Later, at IBM Research, he was one of the principal authors of the Axiom computer algebra system and the architect of the Aldor programming language for mathematical computing. Both Axiom and Aldor were distributed by the Numerical Algorithms Group and are now open source. Dr Watt's work has broadened the scope of symbolic mathematical computing: He was one of the principal authors of MathML, the W3C Standard for mathematics on the web that is now used for the mathematics in US Patents. He also helped open the area of symbolic-numeric algorithms for polynomials, a current hot topic in the field. He has been elected Chair of ACM SIGSAM (the special interest group on symbolic and algebraic computing), elected first Chair of the ISSAC Steering Committee, and was the founding director of ORCCA (the Ontario Research Centre for Computer Algebra).



Arun Rodrigues

Sandia National Laboratories

Arun Rodrigues started life as an MPI library developer before taking a sharp turn towards nonsilicon computing. His current interests are in multithreading, processors in memory, network acceleration, and architectural simulation. He received his PhD from Notre Dame in 2006 and is currently at Sandia National Labs in the Scalable Computer Architecture group. He is the chief maintainer of the Structural Simulation Toolkit.



Panel II

Applicable Computer Science Foundations

Assume:

- you need to run huge ab-initio simulations
- you need to easily vary parameters and get results as rapidly as possible, ideally, at machine speeds.
- you could dream up AND realize **your ideal numeric AND symbolic programming language**. It would run optimally on one or many processors given the problem size . It could scale and port with ease. Also assume you always know where the data is.
- What would be a sufficient algebra to support many important domain sciences in support of Computational Science -> Computational Thinking? **Correctness is a requirement! No specific languages are to be mentioned except in the abstract.**



Panel II

Applicable Computer Science Foundations

Assuming now that you have your ideal language:

What would the ideal machine have to be like?

What would the network have to be like?

What would the processor-memory look like?



Panel II

Applicable Computer Science Foundations

There are **numerous ways to implement symbolic** computation.

What are the ideal ways and what programming paradigms must be utilized to implement them? Generic programming and Expression Templates in C++, Resolution Theorem Proving, Grammars in Compilers and Interpreters? **What communities explore research in these areas? Are they here?** Consequently, if we are using pattern matching of some form what keeps us from proving that the mathematical models we use to design (and build) programming languages are correct; that scientific software is correct?



Panel II

Applicable Computer Science Foundations

What can we do to implement scalable, portable, and optimal code?

Ideally we wish to not only port and scale applications to peta-scale computers with complicated processor memory hierarchies, **but we wish to have the ideal of porting and scaling to any future architecture while running at hardware speeds.** That is can we get scientific numeric, symbolic, and algebraic software to keep up with Moore's Law? **Do we need to rethink compilers and interpreters? Do we need to rethink Operating Systems?**



Panel II

Applicable Computer Science Foundations

What domains utilize the same mathematical models and is there a way to capitalize on the common mathematics they use? That is, are there algorithms, data structures, algebras that could capitalize on processor memory hierarchies because of locality of access patterns? **What data structures have deterministic access patterns? What algebras use these data structures? What scientific domains use these algebras?**



Panel II

Applicable Computer Science Foundations

What applications are limited by time (CPU and Memory speeds), **and space** (cache, RAM, shared, distributed memory, network bandwidths and speeds, poor memory management, temporaries, lack of cache locality), etc.? **Are there Grand Challenge Applications that need numeric-symbolic computing and optimizations?**



Panel II

Applicable Computer Science Foundations

How do we guarantee **reproducible correct computational experiments**? How do we enable scientists with the ability to not only reproduce their experiments but also have a memory of when they were previously done so as not to run them again? Thus we would need enormous space AND computational resources.

