

# ERNIE CHAN

---

United States citizen

## Education

The University of Texas at Austin Austin, TX  
*Doctor of Philosophy* in Computer Science, advised by Robert van de Geijn August 2005 – August 2010

The University of Texas at Austin Austin, TX  
*Master of Science* in Computer Science August 2005 – August 2009  
– Parallel Masters study to Ph.D. program

The University of Texas at Austin Austin, TX  
*Bachelor of Science* in Computer Science August 2000 – May 2004  
– Graduated with Honors and Special Department Honors in Computer Science

## Professional Experience

NVIDIA Santa Clara, CA  
*Compute Architect*, managed by Michael Lightstone February 2011 – Present  
– Aid in the design of graphics processing units for high-performance computing applications

The University of Texas at Austin Austin, TX  
*Postdoctoral Fellow*, supervised by Robert van de Geijn August 2010 – January 2011  
– Investigated the scheduling techniques required for the scalable utilization of exotic computer architectures such as Intel Single-chip Cloud Computer using RCCE and graphics processing units using CUDA, both of which require the explicit management of memory

National Instruments Austin, TX  
*Software Engineer Intern*, supervised by Jim Nagle June 2007 – August 2007 & June 2008 – August 2008  
– Created a scripting tool for statically constructing directed acyclic graphs of dense matrix computations using the graphical programming language G whereby utilizing the compiler and runtime system of LabVIEW to exploit parallelism  
– Interfaced and adapted the open source library `libflame` to replace the dense linear algebra operations in LabVIEW provided by the commercial library Intel MKL

Argonne National Laboratory Argonne, IL  
*Research Aide Appointment*, supervised by William Gropp and Rajeev Thakur June 2005 – August 2005  
– Developed and implemented collective communication algorithms for IBM Blue Gene/L in C using MPI leveraging the fact that a node can simultaneously send and receive to each of its six neighbors within the three-dimensional torus point-to-point communication network

Texas Advanced Computing Center Austin, TX  
*Undergraduate Research Assistant*, supervised by Avi Purkayastha February 2004 – August 2004  
– Studied the mapping of a virtual multidimensional topology to a set of nodes in order to compose hybrid collective communication algorithms where different algorithms are selected within each dimension according to the size of the message being sent

## Teaching Experience

University of Virginia  
*Graduate Teaching Assistant*

Charlottesville, VA  
August 2004 – May 2005

– Information Assurance, Intro to Computer Science, and Discrete Mathematics II

## Dissertation

Ernie Chan. *Application of Dependence Analysis and Runtime Data Flow Graph Scheduling to Matrix Computations*. Ph.D. dissertation, Department of Computer Science, The University of Texas at Austin, August 2010.

We present a methodology for exploiting shared-memory parallelism within matrix computations by expressing linear algebra algorithms as directed acyclic graphs. Our solution involves a separation of concerns that completely hides the exploitation of parallelism from the code that implements the linear algebra algorithms. This approach to the problem is fundamentally different since we also address the issue of programmability instead of strictly focusing on parallelization. Using the separation of concerns, we present a framework for analyzing and developing scheduling algorithms and heuristics for this problem domain. As such, we develop a theory and practice of scheduling concepts for matrix computations in this dissertation.

## Journals

Francisco D. Igual, Ernie Chan, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí, Robert A. van de Geijn, and Field G. Van Zee. The FLAME approach: From dense linear algebra algorithms to high-performance multi-accelerator implementations. *Journal of Parallel and Distributed Computing*, November 2011.

Parallel accelerators are playing an increasingly important role in scientific computing. However, it is perceived that their weakness nowadays is their reduced “programmability” in comparison with traditional general-purpose CPUs. For the domain of dense linear algebra, we demonstrate that this is not necessarily the case. We show how the libflame library carefully layers routines and abstracts details related to storage and computation, so that extending it to take advantage of multiple accelerators is achievable without introducing platform specific complexity into the library code base. We focus on the experience of the library developer as he develops a library routine for a new operation, reduction of a generalized Hermitian positive definite eigenvalue problem to a standard Hermitian form, and configures the library to target a multi-GPU platform. It becomes obvious that the library developer does not need to know about the parallelization or the details of the multi-accelerator platform. Excellent performance on a system with four NVIDIA Tesla C2050 GPUs is reported. This makes libflame the first library to be released that incorporates multi-GPU functionality for dense matrix computations, setting a new standard for performance.

Bryan Marker, Ernie Chan, Jack Poulson, Robert van de Geijn, Rob F. Van der Wijngaart, Timothy G. Mattson, and Theodore E. Kubaska. Programming many-core architectures - a case study: dense matrix computations on the Intel single-chip cloud computer processor. *Concurrency and Computation: Practice and Experience*, August 2011.

A message passing, distributed-memory parallel computer on a chip is one possible design for future, many-core architectures. We discuss initial experiences with the Intel Single-chip Cloud Computer research processor, which is a prototype architecture that incorporates 48 cores on a single die that can communicate via a small, shared, on-die buffer. The experiment is to port a state-of-the-art, distributed-memory, dense matrix library, Elemental, to this architecture and gain insight from the experience. We show that programmability addressed by this library, especially the proper abstraction for collective communication, greatly aids the porting effort. This enables us to support a wide range of functionality with limited changes to the library code.

Field G. Van Zee, Ernie Chan, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí, and Robert A. van de Geijn. Introducing: The `libflame` library for dense matrix computations. *IEEE Computing in Science & Engineering*, 11(6):56-62, November 2009.

As part of the FLAME project, we have been diligently developing new methodologies for analyzing, designing, and implementing linear algebra libraries. While we did not know it when we started, these techniques appear to solve many of the programmability problems that now face us with the advent of multicore and many-core architectures. These efforts have culminated in a new library, `libflame`, which strives to replace similar libraries that date back to the late 20th century. With this paper, we introduce the scientific computing community to this library.

Gregorio Quintana-Ortí, Enrique S. Quintana-Ortí, Robert A. van de Geijn, Field G. Van Zee, and Ernie Chan. Programming matrix algorithms-by-blocks for thread-level parallelism. *ACM Transactions on Mathematical Software*, 36(3):14:1-14:26, July 2009.

With the emergence of thread-level parallelism as the primary means for continued performance improvement, the programmability issue has reemerged as an obstacle to the use of architectural advances. We argue that evolving legacy libraries for dense and banded linear algebra is not a viable solution due to constraints imposed by early design decisions. We propose a philosophy of abstraction and separation of concerns that provides a promising solution in this problem domain. The first abstraction, FLASH, allows algorithms to express computation with matrices consisting of contiguous blocks, facilitating algorithms-by-blocks. Operand descriptions are registered for a particular operation *a priori* by the library implementor. A runtime system, SuperMatrix, uses this information to identify data dependencies between suboperations, allowing them to be scheduled to threads out-of-order and executed in parallel. But not all classical algorithms in linear algebra lend themselves to conversion to algorithms-by-blocks. We show how our recently proposed LU factorization with incremental pivoting and a closely related algorithm-by-blocks for the QR factorization, both originally designed for out-of-core computation, overcome this difficulty. Anecdotal evidence regarding the development of routines with a core functionality demonstrates how the methodology supports high productivity while experimental results suggest that high performance is abundantly achievable.

Ernie Chan, Marcel Heimlich, Avi Purkayastha, and Robert van de Geijn. Collective communication: theory, practice, and experience. *Concurrency and Computation: Practice and Experience*, 19(13):1749-1783, July 5, 2007.

We discuss the design and high-performance implementation of collective communications operations on distributed-memory computer architectures. Using a combination of known techniques (many of which were first proposed in the 1980s and early 1990s) along with careful exploitation of communication modes supported by MPI, we have developed implementations that have improved performance in most situations compared to those currently supported by public domain implementations of MPI such as MPICH. Performance results from a large Intel Xeon/Pentium 4 (R) processor cluster are included.

## Conferences

Ernie Chan, Andrew Chapman, and Robert van de Geijn. Managing the complexity of lookahead for LU factorization with pivoting. In *SPAA'10: Proceedings of the Twenty-Second Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 200-208, Santorini, Greece, June 13-15, 2010.

We describe parallel implementations of LU factorization with pivoting for multicore architectures. Implementations that differ in two different dimensions are discussed: (1) using classical partial pivoting versus recently proposed incremental pivoting and (2) extracting parallelism only within the Basic Linear Algebra Subprograms versus building and scheduling a directed acyclic graph of tasks. Performance comparisons are given on two different systems.

Ernie Chan, Jim Nagle, Robert van de Geijn, and Field G. Van Zee. Transforming linear algebra libraries: From abstraction to parallelism. In *HIPS'10: Proceedings of Fifteenth International Workshop on High-Level Parallel Programming Models and Supportive Environments*, Atlanta, GA, USA, April 19, 2010.

We have built a body of evidence which shows that, given a mathematical specification of a dense linear algebra operation to be implemented, it is possible to mechanically derive families of algorithms and subsequently to mechanically translate these algorithms into high-performing code. In this paper, we add to this evidence by showing that the algorithms can be statically analyzed and translated into directed acyclic graphs (DAGs) of coarse-grained operations that are to be performed. DAGs naturally express parallelism, which we illustrate by representing the DAGs with the G graphical programming language used by LabVIEW. The LabVIEW compiler and runtime execution system then exploit parallelism from the resulting code. Respectable speedup on a sixteen core architecture is reported.

Gregorio Quintana-Ortí, Enrique S. Quintana-Ortí, Ernie Chan, Robert van de Geijn, and Field G. Van Zee. Design of scalable dense linear algebra libraries for multithreaded architectures: the LU factorization. In *MTAAP'08: Proceedings of the 2008 Workshop on Multithreaded Architectures and Applications*, Miami, FL, USA, April 18, 2008.

The scalable parallel implementation, targeting SMP and/or multicore architectures, of dense linear algebra libraries is analyzed. Using the LU factorization as a case study, it is shown that an algorithm-by-blocks exposes a higher degree of parallelism than traditional implementations based on multithreaded BLAS. The implementation of this algorithm using the SuperMatrix runtime system is discussed and the scalability of the solution is demonstrated on two different platforms with 16 processors.

Ernie Chan, Field G. Van Zee, Paolo Bientinesi, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí, and Robert van de Geijn. SuperMatrix: A multithreaded runtime scheduling system for algorithms-by-blocks. In *PPoPP'08: Proceedings of the Thirteenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 123-132, Salt Lake City, UT, USA, February 20-23, 2008.

This paper describes SuperMatrix, a runtime system that parallelizes matrix operations for SMP and/or multi-core architectures. We use this system to demonstrate how code described at a high level of abstraction can achieve high performance on such architectures while completely hiding the parallelism from the library programmer. The key insight entails viewing matrices hierarchically, consisting of blocks that serve as units of data where operations over those blocks are treated as units of computation. The implementation transparently enqueues the required operations, internally tracking dependencies, and then executes the operations utilizing out-of-order execution techniques inspired by superscalar microarchitectures. This separation of concerns allows library developers to implement algorithms without concerning themselves with the parallelization aspect of the problem. Different heuristics for scheduling operations can be implemented in the runtime system independent of the code that enqueues the operations. Results gathered on a 16 CPU ccNUMA Itanium2 server demonstrate excellent performance.

Gregorio Quintana-Ortí, Enrique S. Quintana-Ortí, Ernie Chan, Robert A. van de Geijn, and Field G. Van Zee. Scheduling of QR factorization algorithms on SMP and multi-core architectures. In *PDP'08: Proceedings of the Sixteenth EuroMicro International Conference on Parallel, Distributed and network-based Processing*, pages 301-310, Toulouse, France, February 13-15, 2008.

This paper examines the scalable parallel implementation of QR factorization of a general matrix, targeting SMP and multi-core architectures. Two implementations of algorithms-by-blocks are presented. Each implementation views a block of a matrix as the fundamental unit of data, and likewise, operations over these blocks as the primary unit of computation. The first is a conventional blocked algorithm similar to those included in libFLAME and LAPACK but expressed in a way that allows operations in the so-called critical path of execution to be computed as soon as their dependencies are satisfied. The second algorithm

captures a higher degree of parallelism with an approach based on Givens rotations while preserving the performance benefits of algorithms based on blocked Householder transformations. We show that the implementation effort is greatly simplified by expressing the algorithms in code with the FLAME/FLASH API, which allows matrices stored by blocks to be viewed and managed as matrices of matrix blocks. The SuperMatrix run-time system utilizes FLASH to assemble and represent matrices but also provides out-of-order scheduling of operations that is transparent to the programmer. Scalability of the solution is demonstrated on a ccNUMA platform with 16 processors.

Ernie Chan, Field G. Van Zee, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí, and Robert van de Geijn. Satisfying your dependencies with SuperMatrix. In *Cluster'07: Proceedings of the 2007 IEEE International Conference on Cluster Computing*, pages 91-99, Austin, TX, USA, September 17-20, 2007.

SuperMatrix out-of-order scheduling leverages high-level abstractions and straightforward data dependency analysis to provide a general-purpose mechanism for obtaining parallelism from a wide range of linear algebra operations. Viewing submatrices as the fundamental unit of data allows us to decompose operations into component tasks that operate upon these submatrices. Data dependencies between tasks are determined by observing the submatrix blocks read from and written to by each task. We employ the same dynamic out-of-order execution techniques traditionally exploited by modern superscalar micro-architectures to execute tasks in parallel according to data dependencies within linear algebra operations. This paper provides a general explanation of the SuperMatrix implementation followed by empirical evidence of its broad applicability through performance results of several standard linear algebra operations on a wide range of computer architectures.

Ernie Chan, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí, and Robert van de Geijn. SuperMatrix out-of-order scheduling of matrix operations for SMP and multi-core architectures. In *SPAA'07: Proceedings of the Nineteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 116-125, San Diego, CA, USA, June 9-11, 2007.

We discuss the high-performance parallel implementation and execution of dense linear algebra matrix operations on SMP architectures, with an eye towards multi-core processors with many cores. We argue that traditional implementations, as those incorporated in LAPACK, cannot be easily modified to render high performance as well as scalability on these architectures. The solution we propose is to arrange the data structures and algorithms so that matrix blocks become the fundamental units of data, and operations on these blocks become the fundamental units of computation, resulting in algorithms-by-blocks as opposed to the more traditional blocked algorithms. We show that this facilitates the adoption of techniques akin to dynamic scheduling and out-of-order execution usual in superscalar processors, which we name *SuperMatrix Out-of-Order scheduling*. Performance results on a 16 CPU Itanium2-based server are used to highlight opportunities and issues related to this new approach.

Ernie Chan, William Gropp, Rajeev Thakur, and Robert van de Geijn. Collective communication on architectures that support simultaneous communication over multiple links. In *PPoPP'06: Proceedings of the Eleventh ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 2-11, New York, NY, USA, March 29-31, 2006.

Traditional collective communication algorithms are designed with the assumption that a node can communicate with only one other node at a time. On new parallel architectures such as the IBM Blue Gene/L, a node can communicate with multiple nodes simultaneously. We have redesigned and reimplemented many of the MPI collective communication algorithms to take advantage of this ability to send simultaneously, including broadcast, reduce(-to-one), scatter, gather, allgather, reduce-scatter, and allreduce. We show that these new algorithms have lower expected costs than the previously known lower bounds based on old models of parallel computation. Results are included comparing their performance to the default implementations in IBM's MPI.

Ernie W. Chan, Marcel F. Heimlich, Avi Purkayastha, and Robert A. van de Geijn. On optimizing collective communication. In *Cluster'04: Proceedings of the 2004 IEEE International Conference on Cluster Computing*, pages 145-155, San Diego, CA, USA, September 20-23, 2004.

In this paper we discuss issues related to the high-performance implementation of collective communications operations on distributed-memory computer architectures. Using a combination of known techniques (many of which were first proposed in the 1980s and early 1990s) along with careful exploitation of communication modes supported by MPI, we have developed implementations that have improved performance in most situations compared to those currently supported by public domain implementations of MPI such as MPICH. Performance results from a large Intel Pentium 4 (R) processor cluster are included.

## Technical Reports

Bryan Marker, Ernie Chan, Jack Poulson, Robert van de Geijn, Rob F. Van Der Wijngaart, Timothy G. Mattson, and Theodore E. Kubaska. "Programming Many-Core Architectures - A Case Study: Dense Matrix Computations on the Intel SCC Processor." The University of Texas at Austin, Department of Computer Science. Technical Report TR-11-03. January 24, 2011. 18 pages.

Ernie Chan and Francisco D. Igual. "Runtime Data Flow Graph Scheduling of Matrix Computations with Multiple Hardware Accelerators." The University of Texas at Austin, Department of Computer Science. Technical Report TR-10-36. October 11, 2010. 12 pages.

Ernie Chan. "RCCE\_comm: A Collective Communication Library for the Intel Single-chip Cloud Computer." Intel Corporation, Many-core Applications Research Community. September 10, 2010. 6 pages.

Ernie Chan, Andrew Chapman, and Robert van de Geijn. "Managing the Complexity of Lookahead for LU Factorization with Pivoting." The University of Texas at Austin, Department of Computer Science. Technical Report TR-09-30. October 15, 2009. 15 pages.

Ernie Chan. "The Foundations of Thread-level Parallelism in the SuperMatrix Runtime System." The University of Texas at Austin, Department of Computer Science. Technical Report TR-09-27. September 2, 2009. 15 pages.

Ernie Chan. "Runtime Data Flow Scheduling of Matrix Computations." The University of Texas at Austin, Department of Computer Science. Technical Report TR-09-22. August 10, 2009. 13 pages.

Ernie Chan, Jim Nagle, Robert van de Geijn, and Field G. Van Zee. "Transforming Linear Algebra Libraries: From Abstraction to Parallelism." The University of Texas at Austin, Department of Computer Science. Technical Report TR-09-17. May 27, 2009. 12 pages.

Maribel Castillo, Ernie Chan, Francisco D. Igual, Rafael Mayo, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí, Robert van de Geijn, and Field G. Van Zee. "Making Programming Synonymous with Programming for Linear Algebra Libraries." The University of Texas at Austin, Department of Computer Science. Technical Report TR-08-20. April 17, 2008. 13 pages.

Gregorio Quintana-Ortí, Enrique S. Quintana-Ortí, Robert van de Geijn, Field G. Van Zee, and Ernie Chan. "Programming Algorithms-by-Blocks for Matrix Computations on Multithreaded Architectures." The University of Texas at Austin, Department of Computer Science. Technical Report TR-08-04. January 15, 2008. 20 pages.

Gregorio Quintana-Ortí, Enrique S. Quintana-Ortí, Ernie Chan, Robert A. van de Geijn, and Field G. Van Zee. "Design and Scheduling of an Algorithm-by-Blocks for LU Factorization on Multithreaded

Architectures.” The University of Texas at Austin, Department of Computer Science. Technical Report TR-07-50. September 19, 2007. 16 pages.

Ernie Chan, Field G. Van Zee, Paolo Bientinesi, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí, and Robert van de Geijn. “SuperMatrix: A Multithreaded Runtime Scheduling System for Algorithms-by-Blocks.” The University of Texas at Austin, Department of Computer Science. Technical Report TR-07-41. August 22, 2007. 13 pages.

Gregorio Quintana-Ortí, Enrique S. Quintana-Ortí, Ernie Chan, Robert A. van de Geijn, and Field G. Van Zee. “Scheduling of QR Factorization Algorithms on SMP and Multi-Core Architectures.” The University of Texas at Austin, Department of Computer Science. Technical Report TR-07-37. July 31, 2007. 15 pages.  
Ernie Chan, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí, and Robert van de Geijn. “SuperMatrix Out-of-Order Scheduling of Matrix Operations for SMP and Multi-Core Architectures.” The University of Texas at Austin, Department of Computer Science. Technical Report TR-06-67. December 18, 2006. 12 pages.

Ernie Chan, Marcel Heimlich, Avi Purkayastha, and Robert van de Geijn. “Collective Communication: Theory, Practice, and Experience.” The University of Texas at Austin, Department of Computer Science. Technical Report TR-06-44. September 26, 2006. 32 pages.

## **Presentations**

*Programming Dense Matrix Computations Using Distributed and Off-Chip Shared-Memory on Many-Core Architectures.* First Intel Many-core Applications Research Community Symposium, November 2010.

*Programming Dense Matrix Computations Using Distributed and Off-Chip Shared-Memory on Many-Core Architectures.* University of Vienna, November 2010. Host: Jesper Larsson Träff.

*Application of Dependence Analysis and Runtime Data Flow Graph Scheduling to Matrix Computations.* Argonne National Laboratory, August 2010. Host: Michael Wilde.

*Runtime Data Flow Scheduling of Matrix Computations.* RWTH Aachen University, June 2010. Host: Paolo Bientinesi.

*Runtime Data Flow Scheduling of Matrix Computations.* SIAM Conference on Parallel Processing for Scientific Computing, February 2010.

*Runtime Data Flow Scheduling of Matrix Computations.* Microsoft Corporation, August 2009. Host: Laurent Visconti.

*Dense Linear Algebra on Multicore Architectures: What Kind of Parallelism?* SIAM Conference on Parallel Processing for Scientific Computing, March 2008.

*SuperMatrix Out-of-Order Scheduling of Matrix Operations for SMP and Multi-Core Architectures.* IBM T. J. Watson Research Center, March 2007. Host: John Gunnels.

*Collective Communication: Theory, Implementation, and Experience.* Argonne National Laboratory, March 2005. Host: Rajeev Thakur.

## Awards

Award at MARC Symposium, November 2010. First contributor to Intel SCC software repository.

Certificate of Achievement at SCC Symposium, February 2010. First alpha team to write and execute RCCE application code on Intel Single-chip Cloud Computer.

Student Travel Grant at *SLAM PP'10* conference, February 2010.

UTCS Student Travel Grant for *SLAM PP'08* conference, March 2008.

Student Travel Grant at *PPoPP'08* conference, February 2008.

Student Travel Grant at *PPoPP'06* conference, March 2006.

Student Travel Grant at *Cluster'04* conference, September 2004.

## Service

Conference reviewer for *InPar'12*, January 2012.

Conference reviewer for *SC'11*, May 2011.

Journal reviewer for *ACM Transactions on Mathematical Software*, March 2010.

Undergraduate Studies Faculty Committee for Department of Computer Science at The University of Texas at Austin, September 2007 – August 2009. Review and revamp undergraduate curriculum.

Journal reviewer for *Scientific Programming Special Issue on High Performance Computing on Cell B.E. Processors*, June 2008.

Conference reviewer for *SPAA'07*, January 2007.

Journal reviewer for *IEEE Transactions on Parallel and Distributed Systems*, August 2006.

## Software

**NVIDIA LINPACK:** A complete reimplementation of the LINPACK benchmark for obtaining TOP500 scores on clusters, written in C using MPI and ported to CUDA for GPU acceleration

**SuperMatrix:** Runtime system that parallelizes dense matrix computations for shared-memory computer architectures, written in C using OpenMP and POSIX threads API and built into `libflame`  
<http://z.cs.utexas.edu/wiki/flame.wiki/libflame/>

**RCCE\_comm:** A collective communication library for the Intel Single-chip Cloud Computer, written in C using the RCCE API  
[http://marcbug.scc-dc.com/svn/repository/trunk/rcce\\_applications/UT/RCCE\\_comm/](http://marcbug.scc-dc.com/svn/repository/trunk/rcce_applications/UT/RCCE_comm/)

**InterCol:** A plug compatible collective communication library for MPI, written in C and is portable to any large, commodity cluster  
<http://www.tacc.utexas.edu/resources/software/>

## **Languages**

Fluent in English

Conversational in Chinese (Cantonese)

## **References**

Prof. William Gropp, Department of Computer Science, University of Illinois at Urbana-Champaign.  
Phone: (217) 244-6720, Fax: (217) 265-6738, [wgropp@uiuc.edu](mailto:wgropp@uiuc.edu)

Prof. Enrique S. Quintana-Ortí, Departamento de Ingeniería y Ciencia de Computadores, Universidad Jaime I, Spain. Phone: (+34) 964-728257, Fax: (+34) 964-728486, [quintana@icc.uji.es](mailto:quintana@icc.uji.es)

Prof. Robert van de Geijn, Department of Computer Science, The University of Texas at Austin.  
Phone: (512) 471-9720, Fax: (512) 471-8885, [rvdg@cs.utexas.edu](mailto:rvdg@cs.utexas.edu)