Measurement and Standards for Computational Science and Engineering[1]

Computation has become a critical tool for the practice of science and engineering. Modern research and development leading to new products such as semiconductors, drugs, disk drives, automobiles, and aircraft engines could not be done without substantial investment in large-scale computation. Similarly, complex processes and procedures, from scheduling to exploration for oil to pricing derivatives on Wall Street, have been made much more reliable due to advances in algorithms and their implementation in computer software. As the language of science, mathematics lies at the heart of all work in these fields. Thus, the efficiency and reliability of mathematical methods, algorithms, and software are crucial to the advance of modern science and engineering.

ITL promotes advancement in computational science and engineering by providing fundamental mathematical reference data useful for developing new algorithms and software, as well as for assessing their effectiveness. ITL also plays a key role in a variety of efforts to develop standards for mathematical software which improve portability, interoperability, and performance of products. Several of these efforts will be described in this report. They range from the development of reference data for mathematical functions to the assessment of the state-of-the-art in micromagnetic modeling. The work described here is centered in the ITL Mathematical and Computational Sciences Division (MCSD), http://math.nist.gov/mcsd/.

FUNDAMENTAL MATHEMATICAL REFERENCE DATA

The as part of its goal to develop infrastructure necessary for the practice of science and engineering, the NIST Measurement and Standards Laboratories develop and provide fundamental reference data. Reference data is of great importance in mathematics.

Digital Library of Mathematical Functions

Standard mathematical functions play a critical role in mathematical modeling, a central activity in modern science and engineering. To make effective use of functions, the community must agree upon their exact definition and notation. Working with functions requires knowledge of their many properties and complex relationships. Computing with functions requires understanding of approximations or access to efficient computer software.

A *function* can be thought of as a prescription for generating values of dependent variables from given values of independent variables, or, more simply, for producing outputs from inputs. Simple functions, familiar to everyone, are powers and roots of independent variables. Important classes of functions are polynomials, rational functions, elementary functions, and higher functions. *Polynomials* and *rational functions* are used, for example, in numerical analysis to approximate more complicated functions and to compute numerical solutions of problems in science and engineering. *Elementary functions* include the familiar trigonometric functions. These lie at the heart of signal processing, for example, a field that has wide application in telephony, radio, television, guidance and control of aircraft and spacecraft, medical imaging, oil exploration technology, and computers. *Higher functions* serve many of the same purposes as elementary functions but in much more complicated settings. Examples are Bessel and Legendre functions, which arise naturally as solutions of basic boundary-value problems in physics, and the incomplete gamma function, which is a foundation of statistical theory and methodology.

To be effective in solving real-world problems, practitioners need ready access to a reliable source of information about mathematical functions. Since 1964, the primary source for such information has been the NBS *Handbook of Mathematical Functions* (M. Abramowitz and I. Stegun, eds., 1964). More than 148,000 copies of this publication have been sold by the U.S. Government Printing Office; commercial publishers have sold many more. A study of citations over the period 1974-1995 illustrates that, remarkably, the rate of increase in citations to the *Handbook* is greater than the rate of increase in citations for scientific literature as a whole!

---

[1] This article appeared in the March 1999 issue of the *ITL Bulletin*, issued by the Information Technology Laboratory of the National Institute of Standards and Technology. Its authors are Ronald Boisvert, James Blue, Michael Donahue, Daniel Lozier, William Mitchell, Donald Porter, and Roldan Pozo.

In spite of its persistent usefulness, the *Handbook* is now out-of-date in many respects. Since its publication, numerous advances in mathematical knowledge have been made:

- New functions have entered the realm of practical importance, e.g. q-series.
- New fields of application have emerged, e.g. in nonlinear dynamics.
- Analytical developments have occurred, e.g. in asymptotics.
- New properties, e.g. integral representations and addition formulas, have been discovered.
- Numerical developments, e.g. interval analysis and Padé approximations, have occurred.
- Computer algebra and symbolic processing have come into wide use.
- An enormous increase in computing power has made obsolete standard numerical processes of the 1950's, such as table-making and interpolation, while increasing the value of others.
- Comprehensive software packages have been constructed for working and computing with functions.

At the same time, dissemination of information is being revolutionized by the rapid development of the Internet and World Wide Web. A modern successor to the *Handbook* should provide new capabilities unavailable in print media, such as:

- Generic representation of mathematical artifacts such as formulas, graphs, tables and diagrams.
- Advanced search, with the ability to locate formulas based on mathematical subexpressions.
- Downloading of mathematical artifacts into document processors.
- Importing of formulas directly into symbolic computing systems.
- Continuous updating to incorporate corrections, additions and extensions.
- Maintenance of communication channels between users and developers, with a public record of usage.
- Support for external *application modules* to provide tutorials, application notes or research monographs in fields that use mathematical functions.
- Recommendations of algorithms and software for computing functions, with links to sources.
- Generation of numerical tables and graphs for user-specified ranges of input variables

As a result of the continuing need and these new opportunities, NIST has begun the process of completely rewriting the *Handbook* with presentation as an on-line resource. The result will be the NIST *Digital Library of Mathematical Functions*. Together with NIST staff from other divisions and subject specialists from outside NIST, MCSD mathematicians are working to (a) gather all pertinent mathematical information, (b) construct a state-of-the-art reference database with all necessary tools for long-term maintenance, (c) present validated reference information on the Web, and (d) develop application modules in quantum mechanics, electromagnetism, and an adaptive learning system for mathematical functions.

A prototype of the Digital Library can be inspected at http://math.nist.gov/DigitalMathLib/. Completion of the system is expected in 2002.

REFERENCE ARTIFACTS FOR SOFTWARE TESTING

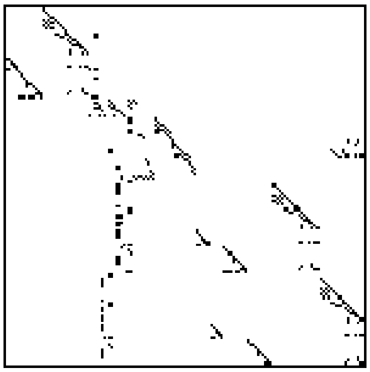
A frequently applied method for the testing of numerical software is to exercise it on a battery of representative problems. Often such problems are generated randomly, insuring that a large number of test cases can be applied. Unfortunately, this is rarely sufficient for serious numerical software testing. Errors or, more likely, numerical difficulties typically occur for highly structured problems or for those near to the boundaries of applicability of the underlying algorithm. These parts of the domain are rarely sampled in random problem generation, and hence testing must also be done on problem sets that illustrate particular behaviors. These are often quite difficult to produce, and, thus, researchers often exchange sample problem sets. Such data sets serve a variety of additional purposes:

1. Defining the state-of-the-art.
2. Characterizing industrial-grade applications.
3. Catalyzing research by posing challenges.
4. Providing a baseline of performance for software developers.
5. Providing data for users who want to gain confidence in software.

Unfortunately, these collections are often lost when the underlying technology is picked up by the commercial sector, leaving software developers and users without an important tool to use in judging the capability of their products.

The Matrix Market

The decomposition, solution and eigenanalysis of systems of linear equations remain important problems in scientific computation for which new algorithms and software packages are continually being developed. Of particular importance are linear systems of equations that are represented by *sparse* matrices. A sparse matrix is a matrix in which most elements are zero. The adjacent figure is a sparsity plot for such a matrix; the dots show where nonzeros are located. Problems of this type arise in modeling based on partial differential equations, such as in fluid flow and structural analysis. The behavior of algorithms and software for such problems is highly dependent on the sparsity structure and the numerical properties derived from the underlying problem. As a result, in order to make reliable, reproducible and quantitative assessments of the value of new algorithmic developments it is useful to have a common collection of representative problems through which methods can be compared. Researchers in this area have exchanged problem sets of this type informally for some time. One of the difficulties with such collections is that their size and diversity makes them unwieldy to manage and use effectively. As a result, such collections have not been used as much as they should, and matrices useful for testing are not easy to find.



Recent developments in network communications infrastructure, such as the World Wide Web (WWW), are providing new possibilities for improving access to and usability of test corpora of this type. The NIST Matrix Market (http://math.nist.gov/MatrixMarket/) is one such collection.

The Matrix Market is a repository of matrices for use in the comparative analysis of algorithms and software for numerical linear algebra. More than 500 matrices of size 9 by 9 to 90,449 by 90,449 from a wide variety of applications are made available in the Matrix Market. Matrices are gathered together into *sets*. Matrices in a set are related by application area or contributed from a single source. Sets are grouped further into *collections*, such as the well-known Harwell-Boeing collection. Individual matrices may be stored explicitly as dense or sparse matrices, or may be available implicitly via a code that generates them. Matrix generators are either run at NIST remotely via Web-based form, or run locally as a Java applet in a Web browser. In other cases, Fortran code may be downloaded for inclusion in a local testing application. Available matrices are of a wide variety of types, e.g. real, complex, symmetric, nonsymmetric, Hermitian. Some are representations of nonzero patterns only. Others include supplementary data such as right-hand sides, solution vectors, or initial vectors for iterative solvers. We store matrices and associated material one-per-file, in both the well-known Harwell-Boeing format, as well as in a new Matrix Market format. Software for reading and writing such matrices in Fortran, C, and Matlab are provided.

For each matrix we provide a summary page in HyperText Markup Language (HTML) outlining the properties of the matrix and displaying a graphical representation of its properties. Graphics include sparsity plots such as shown above, and three-dimensional representations in Virtual Reality Markup Language (VRML) which can be manipulated graphically in a Web browser. Spectral portraits, which illustrate the sensitivity of matrix eigenvalues, are also available in many cases. Similarly, we have developed a Web page for each set that gives its background (e.g., source and application area), references, and a thumbnail sketch of each matrix's nonzero pattern. We maintain a separate database that contains all of the information on these pages in a highly structured form. This allows us to manipulate the data in various ways; for example, all of the Web pages for matrices and sets are automatically generated from this database. The database also supports both structured and free-text retrieval. The Matrix Market search tool, for example, allows users to locate matrices with particular special properties, e.g. all real symmetric positive definite matrices with more than 10,000 rows and less than 0.1 percent density.

The Matrix Market has supported linear algebra researchers and software developers since 1997. Its Web site is visited by more than 100 users daily.

Measuring the Performance of Micromagnetic Modeling Software

The magnetic disk drive industry currently has sales of over $40 billion per year and is growing rapidly. Each year's new drives are larger, faster, and cheaper per bit than the previous year's, and each bit occupies less area on the disk. At the current rate, the recording density increases by 60 percent per year. To maintain this progress, mathematical modeling at the micromagnetic level is necessary.

Micromagnetics modeling is done at submicroscopic levels, but at a scale large enough so that the material can be treated as a continuum, rather than looking at individual atoms. Since the nonlinear equations of micromagnetics cannot be solved analytically for any realistic system, computer models are required. Computer programs for micromagnetic modeling are complex, and their accuracy should be evaluated before using them for designing commercial magnetic devices. Standard problems are needed to assess the accuracy and speed of the computer programs.

Researchers from the Information Technology Laboratory and the Materials Science Engineering Laboratory, in cooperation with µMAG (the Micromagnetic Modeling Activity Group, whose members come from NIST, industry, and universities) have developed several standard problems for micromagnetics. In 1996 and 1997, the results of eight different computer programs for the first standard problem were submitted anonymously. The standard problems and submitted solutions can be found at http://www.ctcms.nist.gov/~rdm/mumag.org.html. These results graphically demonstrate the need for standard problems for assessing micromagnetic computer programs.

While the results of the first standard problem highlighted the challenges in micromagnetic modeling, submitted results to the second and third standard problems have shown much greater agreement, and are finding use as benchmarks for newly developed code.

SOFTWARE STANDARDS

The ITL Mathematical and Computational Sciences Division is also involved in a number of activities related to standardization in the numerical software area. The goal of these efforts is to promote portability, interoperability and high performance. By having agreed-upon interfaces to common low- and mid-level kernels, users can write software that can be more easily ported to a wide variety of environments. Hardware and software vendors can then concentrate on providing highly optimized versions of kernel operations for their particular environment. The result is a higher level of *performance portability*, i.e. software that not only runs in a variety of operating environments, but also runs at near optimal levels of performance.

In most cases, this work involves ad-hoc community-based efforts rather than formal standardization bodies. In most cases, standards are developed in open forums with participation of industrial, government and academic stakeholders. Reference implementations are developed along with proposed standards; these are made freely available on the Web.

The Sparse BLAS

Numerical simulation and modeling applications are big consumers of computer cycles. A single run of such an application may take anywhere from a few hours to few months, and computational scientists are always looking for ways to speed up these simulations. When analyzed, it is often found that much of the is spent on elementary linear algebra operations such as multiplying two matrices together. This occurs because simulations involving optimization or the solution of partial differential equations often lead to the solution of large linear systems. In short, if linear algebra can be speeded up, the overall simulation speeds up as well.

That is exactly what happened twenty years ago, when a small Fortran library of kernel routines was introduced to the scientific computing community. Dubbed the Basic Linear Algebra Subprograms (BLAS), these routines included vector operations such as dot products and vector scale/add. The BLAS had two effects on numerical programs. First, they made such code more modular by replacing the explicit loops for these computations with well-structured subroutines. Second, they allowed hardware vendors to tailor their own BLAS kernels to exploit best the underlying architecture, greatly improving performance in a portable way. The result was numerical codes that looked cleaner and had improved performance by as much as ten-fold.

It was not long before vendor libraries were providing a uniform interface for matrix/vector operations via the BLAS. During the late 1980s, extensions to the BLAS were proposed that included larger grained matrix/matrix operations, allowing even greater potential for optimization. By exploiting block-based algorithms, more of the numeric data could be kept closer to the floating-point hardware, resulting in increased performance for cache-based computer architectures. Today nearly every high performance computer manufacturer provides a BLAS numeric library for their machines.

During the past few years, a new standardization effort has been underway to extend this optimization strategy to other linear algebra computational kernels as well as to other computer languages. This new consortium, called the BLAS Technical Forum consists of various hardware vendors as well academic and research institutions specializing in high performance numerical computing.

One of the most important new extensions deals with sparse matrices. While the original BLAS focused on dense matrices, the matrices encountered in most large-scale industrial applications are large and sparse. Sparse structures are much more difficult to optimize, because, unlike their dense counterparts, their layout in memory is based on their nonzero structure and this is different for each application. To further complicate matters, there are several storage schemes commonly used for representing sparse matrices, each with subtle performance tradeoffs that vary between application domains and computer architectures.

Nevertheless, computational experiments carried out at NIST show that code optimization strategies exist that can lead to significant performance improvements. Some of these techniques include loop unrolling, pre-fetching, and minimizing indirect addressing. For large sparse matrices, it is often beneficial to try to reorganize the elements of the matrix into small blocks. These blocks (rather than single entries) become the basic unit of computation and are much better able to achieve high performance on cache-based architectures by exploiting locality of the original problem.

The Sparse BLAS proposal under consideration by the BLAS Technical Forum supports two basic operations: (1) multiplying a sparse matrix with a dense vector or matrix, and (2) solving a sparse triangular system with single or multiple right-hand sides. These two operations are fundamental, for example, in solving sparse linear systems using iterative techniques, and can also be used as building blocks for direct methods. The Sparse BLAS proposal also supports initialization from nine different storage formats. However, because it is based on a generic interface, vendors and independent library developers have completely flexibility in transforming and computing with optimized data structures best suited for their computer architecture and/or application domain.

Using this framework, it is not only possible to improve the performance of many sparse matrix codes, but also to provide a consistent interface for sparse matrix computation. Thus, the Sparse BLAS achieve the important goals that motivated the original work on standardizing linear algebra kernels: o make numerical codes faster, more portable, and easier to understand. The current proposed standard, along with a reference implementation, can be viewed at http://math.nist.gov/spblas/.

Java Numerics

While Java has already made an enormous impact on the computing industry, it has seen very little use in the computational sciences. Nevertheless, there has begun to be interest in the use of Java for compute-intensive applications. The reasons for this include

- the portability of Java code based on the Java Virtual Machine (JVM),
- access to a significant collection of class libraries to aid in developing graphical user interfaces,
- facilities for network-based computing built into the language, and
- its elegant object-oriented design.

Because of these features, Java could well become an excellent environment for the development of large-scale numerical applications. However, Java suffers from a variety of performance and other design problems that may severely hamper its usefulness in this area.

In order to develop a clearer assessment of the suitability of Java for large-scale ("Grande") applications, and to promote necessary developments, NIST helped form the Java Grande Forum (JGF) in March 1998. The JGF is an open forum of industrial, government, and academic partners (http://www.javagrande.org/) which aims to clearly articulate both the capabilities and current problems with Java for application areas requiring the highest levels of performance. It develops requirements and suggestions for specific changes to Java, and works with the research community to help set standards in such areas as scientific application programmer interfaces (APIs) and frameworks for computing services marshaled by Java. The JGF contains two major working groups: the Numerics Working Group and the Applications and Concurrency Working Group. Six meetings of these groups took place in the JGF's first year of operation.

The Numerics Working Group, which is chaired by MCSD, issued its first report on Java numerics, in October 1998. In its initial assessment, the working group focussed on five critical areas where improvements to the Java language are needed: (a) floating-point arithmetic, (b) complex arithmetic, (c) multidimensional arrays, (d) lightweight classes, and (e) operator overloading. Complex arithmetic and multidimensional arrays are fundamental requirements for scientific computing which are currently missing from the language. Lightweight classes and operator overloading provide key facilities which would lead to highly efficient and usable classes for complex arithmetic and multidimensional arrays. In addition, they would admit efficient implementation of alternative mathematical systems such as interval arithmetic. The working group's proposal for changes to Java's floating point semantics balances the need for efficiency with the need for predictability, which is that hallmark of Java's design. Java's current strict model for floating-point semantics imposes requirements that cannot be achieved on many current microprocessors without a 3-fold to 10-fold penalty in performance. The working group's report can be found at the Java Numerics web site at NIST: http://math.nist.gov/javanumerics/.

The in Numerics Working Group is also working to catalyze efforts to develop standard APIs for numerical computing in Java. Strawman APIs for complex arithmetic, array operations, linear algebra, and special functions, along with complete reference implementations have already been proposed and are being evaluated. These may be downloaded from the Java Numerics web page. Additional APIs in the areas of interval arithmetic, multiprecision arithmetic, and Fourier transforms are also planned.

Finally, NIST has developed a benchmark of representative numerical kernels in order to assess the current state-of-the-art of Java compilers and run-time environments. Scimark, available at http://math.nist.gov/scimark/, is a Java applet that executes five kernels: FFT, LU factorization, Gauss-Seidel relaxation, Monte-Carlo integration, and sparse matrix multiplication. Performance results reported to NIST from a wide variety of platforms can be viewed at the SciMark Web site.

Portable Interactive Graphics for Fortran Programmers

A large portion of software for scientific computing is written in Fortran. Typically, such applications generate a very large volume of data, which is nearly impossible to make sense of without sophisticated three-dimensional graphical visualizations. For portability, it is desirable to use a standardized, widely available graphics library to display the required graphics. One example is OpenGL, an interface for interactive two- and three-dimensional graphics that is independent of operating system, window system, and hardware operations (see http://www.opengl.org/). OpenGL is an open, industry supported standard with implementations on most hardware platforms, providing low-level support from which sophisticated graphical displays may be produced. For example, objects drawn with OpenGL are built from polygons positioned in three-dimensional space. The polygons are given material properties and color to attain the desired appearance, lighting sources and the orientation of the view.

OpenGL was originally developed for use in a C-based programming environment, so it does not immediately satisfy the needs of Fortran programmers. Although OpenGL has had Fortran 77 bindings for some time, these bindings rely upon extensions to the Fortran 77 standard. Some of these extensions are commonly used by Fortran compilers, but others are not widely supported, which makes OpenGL difficult or impossible to use from some Fortran processors. Also, some of the OpenGL functionality cannot be achieved by any Fortran processor under the Fortran 77 bindings.

By using the new features of Fortran 90 it is possible to define bindings for OpenGL that do not depend on any extensions to the standard and provide access to the full functionality of OpenGL. They can also increase the robustness and portability in user application code, and increase the similarity between the Fortran and C interfaces. Such an interface was developed several years ago by MCSD for internal use at NIST.

Thinking that this might be more widely useful, MCSD staff proposed these bindings to the OpenGL Architecture Review Board (ARB) in December 1996. The ARB is the governing board of OpenGL, and consists of members from a variety of companies with OpenGL products (3DLabs, Compaq, Evans & Sutherland, Hewlett-Packard, IBM, Intel, Intergraph, Microsoft, NVIDIA and Silicon Graphics). The proposal was favorably received, and in February 1998 the ARB formally adopted the proposal as the official Fortran 90 bindings for OpenGL. The bindings are publicly available at http://www.opengl.org/Documentation/Fortran.html.

Adoption of the OpenGL Fortran 90 bindings represents a significant development for scientific visualization in the Fortran community. With the new bindings, a Fortran programmer can write standard-conforming graphics applications that are portable over most computing platforms. Such a capability is very important in the computational science community.

In conjunction with this work, MCSD has developed f90gl, a public domain reference implementation of the Fortran 90 bindings, which is publicly available at http://math.nist.gov/f90gl/. Since its initial release two years ago, f90gl has been downloaded approximately 2,500 times and is being deployed in a wide variety of applications. A number of vendors of Fortran-based products have already expressed interest in this work and it is expected that OpenGL products based on f90gl will emerge soon.

An OOMMF for Micromagnetic Modeling

Mathematical modeling is becoming increasingly important in the micromagnetics industry. Unfortunately, computer programs for micromagnetic modeling are both expensive and time-consuming to develop. Most existing programs are proprietary, and few commercial programs are available for licensing. After seeing the results of eight proposed solutions from seven different computer programs for the first standard micromagnetics problem (*see Measuring the Performance of Micomagnetic Modeling Software* in this report), ITL researchers decided that a publicly available reference code was needed. The OOMMF (Object Oriented MicroMagnetic Framework) project was started in 1997 to produce a micromagnetic modeling code.

The goal of the OOMMF project is to develop a portable, extensible public domain micromagnetic program and associated tools. This code will form a completely functional micromagnetics package, but will also have a well documented, flexible programmer's interface so that people developing new code can swap their own code in and out as desired.

In order to allow a programmer not familiar with the code as a whole to add modifications and new functionality, an object oriented approach is critical. The C++ language is a good compromise with respect to availability, functionality, and portability. In order to allow the code to run on a wide variety of systems, the interface and graphics code is in Tcl/Tk. This enables the code to operate across a wide range of Unix platforms, Windows NT, and Windows 95/98.

A two-dimensional version of the code and an associated program for magnetization display are now available at http://math.nist.gov/oommf/. The full three-dimensional version of the code will be available later this year.

URLs

Resource	URL
Digital Library of Mathematical Functions	http://math.nist.gov/DigitalMathLib/
Fortran 90 Interface to OpenGL	http://math.nist.gov/f90gl/
Java Numerics	http://math.nist.gov/javanumerics/
Matrix Market	http://math.nist.gov/MatrixMarket/
Micromagnetic Modeling	http://math.nist.gov/oommf/
Sparse BLAS	http://math.nist.gov/spblas/

Further Reading

Digital Library of Mathematical Functions:

- D. W. Lozier, Toward a Revised NBS Handbook of Mathematical Functions, NISTIR 6072, National Institute of Standards and Technology, Gaithersburg, MD, September 1997. Also available as http://math.nist.gov/DigitalMathLib/publications/nistir6072/
- D. W. Lozier, B. R. Miller and B. V. Saunders, Design of a Digital Mathematical Library for Science, Technology and Education, NISTIR 6297, National Institute of Standards and Technology, Gaithersburg, MD, February 1999. Also available as http://math.nist.gov/DigitalMathLib/publications/nistir6297/

Fortran 90 Interface to OpenGL:

- W. Mitchell, A Fortran 90 Interface to OpenGL, NISTIR 6134, National Institute of Standards and Technology, Gaithersburg, MD, February 1998. Also available as http://www.opengl.org/Documentation/Fortran90/nistir6134.pdf

Java Numerics:

- R. F. Boisvert, J. J. Dongarra, R. Pozo, K. A. Remington, and G. W. Stewart, Developing Numerical Libraries in Java, *Concurrency: Practice and Experience*, Vol. 10, Nos. 11-13, 1998, pp. 1117-1129.

Matrix Market:

- R. Boisvert, R. Pozo, K. Remington, R. Barrett, and J. J. Dongarra, Matrix Market: a web resource for test matrix collections, in *The Quality of Numerical Software: Assessment and Enhancement*, (R. Boisvert, ed.), Chapman & Hall, London, 1997, pp. 125-137. Also available as http://math.nist.gov/MatrixMarket/reports/MMpaper.ps.gz

Sparse BLAS:

- Iain Duff, Michele Marrone, Giuseppe Radicati and Carlo Vittoli, Level 3 Basic Linear Algebra Subprograms for Sparse Matrices: a User-Level Interface, *ACM Transactions on Mathematical Software*, Vol. 23, No. 3, 1998, pp. 379-401. Also available as http://www.acm.org/pubs/citations/journals/toms/1997-23-3/p379-duff/

Acknowledgements

The following staff of the ITL Mathematical and Computational Sciences contributed to the development of this report: James Blue, Ronald Boisvert, Michael Donahue, Daniel Lozier, William Mitchell, Donald Porter, and Roldan Pozo.