

# Genblas: Basic Linear Algebra Subroutines in C++ for All Fields.

AHMET DURAN & DAVID SAUNDERS

University of Delaware

duran@mail.eecis.udel.edu, saunders@mail.eecis.udel.edu

The BLAS, Basic Linear Algebra Subprograms, has existed in various forms for over two decades. It has long been heavily used in numerical linear algebra and now is finding application in symbolic linear algebraic computation.

Recently, the BLAS Technical Forum [1] standardized the bindings (interface) for Fortran 77, Fortran 95, and C calls to BLAS routines. C++ bindings are still in development. We here propose and illustrate a system of C++ binding which we call *Genblas*. Genblas exploits the template mechanism to be generic with respect to the element type. The element type may be any type which supports the arithmetic functions in the familiar C++ operator forms. Thus float, double, complex, are all valid element types. But also types representing finite fields may be used such as `zz_p` and `ZZ_p` of the NTL system [5]. The type `int` may be used except in `trsv` (the triangular solver) when the matrix is not unit triangular. In the BLAST standard, the more object oriented Fortran 95 bindings simplify and extend the older Fortran 77 bindings. Our Genblas C++ bindings are related to the C bindings in a parallel way.

Genblas is also generic with respect to matrix representation issues. Both dense and sparse matrix BLAS bindings are supported. For sparse matrices representations including the standard *blas\_sparse\_matrix*, SuperMatrix, and the blackbox sparse matrix type in LinBox [3] may be used. Genblas operates by wrapping C calls, so can be used with any C implementation of the BLAS meeting the standard. However, implementations of the sparse matrix bindings conforming to the new BLAST Forum standard do not yet exist. For the sparse bindings we wrap calls as used in the SuperLU system [2]. Timings indicate that there is negligible overhead for calling blas routines through the Genblas wrappers.

Finally, Genblas can be used to improve performance of matrix computations over finite fields. Suppose you wish to solve a linear system over  $GF_p$ , for a prime  $p$ . Normal modular arithmetic is done with a remainder mod  $p$  computation as part of each basic field operation. However one can perform larger block matrix-vector and matrix-matrix products using non-normalized arithmetic and then mod out only after the block operation. This substantially speeds up the computation due to two factors. First, many fewer of the expensive quotient-remainder are performed. Second, the basic block vector and matrix operations may use Genblas to exploit highly tuned BLAS codes. This has been demonstrated by Dumas, Gautier, and Pernet [4] in their FFLAS codes.

We expect to call Genblas from Linbox algorithms much as blas is used underneath Lapack. Linbox is a library for exact linear algebra emphasizing blackbox techniques. Linbox is structured so that the field in which a computation is done is an object, not a type. Thus functions typically have a field in the argument list. We plan to implement such functions so as to call generic Genblas or the FFLAS algorithms directly, or to call Genblas with template specializations for the FFLAS calls.

## References

- [1] Basic Linear Algebra Subprograms Technical (BLAST) Forum, January 25, 2001. <http://www.netlib.org/blas/blast-forum/> .

- [2] J.W. Demmel, J.R. Gilbert, X.S. Li: SuperLU User's Guide, September 21, 1999.
- [3] Jean-Guillaume Dumas, Thierry Gautier, Mark Giesbrecht, Pascal Giorgi, Bradford Hovinen, Erich Kaltofen, B. David Saunders, Will J. Turner, and Gilles Villard. LinBox: A generic library for exact linear algebra. In *Proceedings of the 2002 International Congress of Mathematical Software, Beijing, China*. World Scientific Pub, August 2002.
- [4] Dumas, Gautier, Pernet, Finite Field Linear Algebra Subroutines, Preprint, <http://www-lmc.imag.fr/lmc-mosaic/Jean-Guillaume.Dumas/FFLAS/#Papers> .
- [5] V. Shoup, NTL: A Library for doing Number Theory, <http://www.shoup.net/ntl/> .