

# Cholesky Decomposition and Linear Programming on a GPU \* †

[Extended Abstract]

Jin Hyuk Jung  
 Department of Computer Science  
 University of Maryland, College Park  
 jjung@cs.umd.edu

Dianne P. O’Leary  
 Department of Computer Science  
 University of Maryland, College Park  
 oleary@cs.umd.edu

## ABSTRACT

The rapid evolution of Graphics Processing Units (GPUs) in performance, architecture, and programmability provides computational potential beyond their primary purpose, graphics processing. In this work we present an efficient algorithm for solving symmetric and positive definite linear systems using triangular update on a GPU. Using the decomposition algorithm and other basic building blocks for linear algebra on the GPU, we demonstrate a GPU-powered linear program solver based on a Primal-Dual Interior-Point Method.

**Contributions:** We present a new algorithm to decompose symmetric and positive definite dense matrices through a set of kernel calls with minimum copying operations to maximize performance. Using our algorithm and other BLAS kernels, we demonstrate how to build a GPU-powered primal-dual interior-point method with minimal feedback to the CPU. We use:

- Triangular domain updating to exploit the symmetric structure.
- Texture coordinate mapping and index swizzling for efficient texture fetching.

## Keywords

GPGPU, Cholesky, Matrix Decomposition, Linear Programming, Interior Point Method

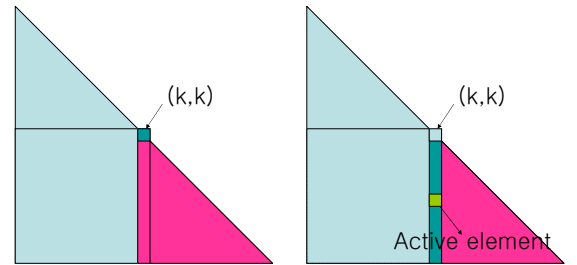
## 1. Introduction

Strong competition in the gaming industry is driving graphics processing hardware development. ATI’s Radeon and NVIDIA’s GeForce series, the dominant products in the market, offer highly programmable parallel engines for processing graphics problems. Due to competition and the rapidly growing game market, GPUs are now cheap parallel SIMD machines available to any users.

## 2. Cholesky Decomposition Using Triangular Update on a GPU

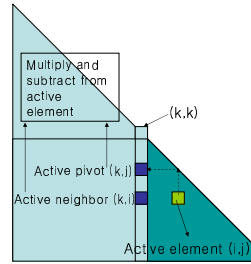
\*A full version of this paper can be obtained at <http://www.cs.umd.edu/~jjung/pub/cholgpu.pdf>

†This work was supported in part by the US Department of Energy under Grant DEFG0204ER25655.

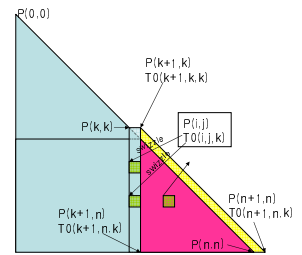


(a) **Square root:** Compute square root of the diagonal element at the  $k^{th}$  column and row

(b) **Normalization:** Divide each element in the  $k^{th}$  column below the diagonal by the diagonal



(c) **Submatrix update** Compute the product of index triad is needed when the two elements, active a 3D texture coordinate is applied to each vertex.



(d) **Only one interpolated index triad is needed when the two elements, active a 3D texture coordinate is applied to each vertex.**

Figure 1:  $k^{th}$  iteration of Cholesky decomposition

A system of linear equations,  $\mathbf{Ax} = \mathbf{b}$ , where  $\mathbf{A}$  is a large, dense  $n \times n$  matrix, and  $\mathbf{x}$  and  $\mathbf{b}$  are column vectors of size  $n$ , can be efficiently solved using a decomposition technique, LU for instance. If the matrix is symmetric and positive definite, Cholesky decomposition is the most efficient in solving the system [3]. The decomposition algorithm involves 3 routines (square rooting, normalizing, and updating submatrix) at each iteration. Each routine can be well matched to a GPU kernel as described in Figure 1a, 1b, and 1c. The most time consuming part among the 3 is the submatrix update routine. The update must take place only in the triangular region to be most efficient, but specialized languages

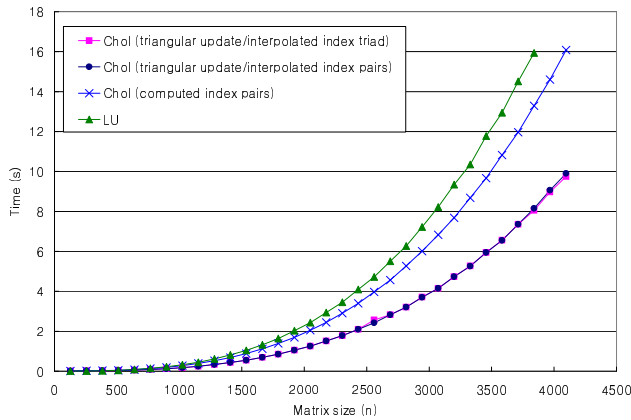


Figure 2: OpenGL API implementation. Execution time of Cholesky and LU decomposition without pivoting measured on Windows XP.

such as BrookGPU [1] don't support designating a triangular domain as an output target. A column-wise multipass update can be used to mimic the triangular update, but this approach cannot fully utilize the parallel architecture because each pass has several processors unused. However, singlepass update over a triangular domain can be implemented with the OpenGL API as drawing a triangle is truly a native feature of the GPU. An oversized triangle, the vertices of which are at  $(k + 1, k)$ ,  $(k + 1, n)$ , and  $(n + 1, n)$ , is drawn to cover the lower triangular matrix at the  $k^{th}$  iteration as presented in Figure 1d.

Two sets of texture coordinates can be assigned to the vertices to generate indexes for the active pivot and the active neighbor. In fact, the index pairs passed to a fragment processor are the same as in the LU algorithm [2]. However, the index pair for the pivot must be treated differently, since our Cholesky decomposition doesn't update the upper triangular part. The index pair needs to be swapped, which can be handled by a swizzle operator at no cost. The swizzle operator reorders the coordinates of a multidimensional variable. This technique reduces the number of instructions from 6 to 4. In addition, the two index pairs share an invariant index  $k$ . Thus, it can be put at a  $z$  coordinate which is invariant to the  $x$  and  $y$  coordinates. The pairs can be packed into, interpolated from, and restored (using swizzle) from a single 3D texture coordinate to reduce the rasterizer's work.

### 3. Interior-Point Method for Linear Programming on a GPU

Linear programming is the problem of optimizing a linear objective function subject to satisfying a set of linear constraints, either equalities or inequalities. The Primal-dual interior-point method is one of the most widely used algorithms to solve the linear programming problem. Solving the linear programming problem is equivalent to finding a solution to the KKT (Karush-Kuhn-Tucker) conditions, for which a modified Newton's method is used. Finding the search direction at each iteration of the method usually involves solving either a perturbed Newton system or normal equations in which a symmetric positive definite matrix is

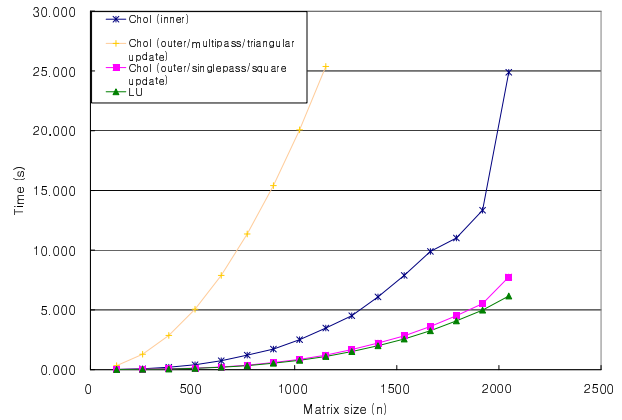


Figure 3: BrookGPU implementation measured on Windows XP with an OpenGL backend. Average execution time of Cholesky algorithms via inner product and outer product (singlepass square update), and LU decomposition.

found. The system of normal equations is preferred to the Newton system, because it is much more compact and it can be solved via Cholesky decomposition, which is very efficient and numerically stable.

## 4. Conclusion

In the interior point method, matrix formation and decomposition are most demanding of computational resources. Thus, support for the triangular output domain is essential in order to exploit the symmetric structure. As seen in Figure 2 and 3, our experiments show that Cholesky decomposition achieves its full performance gain over LU only when the triangular update technique is deployed. Developers of streaming languages targeting GPUs should seriously consider incorporating this feature in their model.

## 5. REFERENCES

- [1] I. Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, and P. Hanrahan. Brook for GPUs: stream computing on graphics hardware. *ACM Transactions on Graphics*, 23(3):777–786, 2004.
- [2] N. Galoppo, N. K. Govindaraju, M. Henson, and D. Manocha. LU-GPU: Efficient Algorithms for Solving Dense Linear Systems on Graphics Hardware. In *SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, page 3, Washington, DC, USA, 2005. IEEE Computer Society.
- [3] G. H. Golub and C. F. V. Loan. *Matrix Computations (3rd ed.)*, chapter 4. Special Linear Systems, pages 133–205. Johns Hopkins University Press, Baltimore, MD, USA, 3rd edition, 1996.