LDLT FACTORIZATION IN CUDA

for systems solving

Sunday 22nd March, 2020

Maxime Darrin   Pierre Guetschel

M2A - Sorbonne Université

Our project

The project

We build our project in two parts:

1. The factorization algorithm
2. The solver (using a factorized form)

Hardware

Our experiments have been conducted on a GTX 1060 for laptop.

Data storage

Storage of n matrix of size $d * d$

Matrices L and D :

$D^1_{1,1}$	$D^1_{2,2}$	$\dots$	$D^1_{d,d}$	$L^1_{1,1}$	$L^1_{2,1}$	$\dots$	$L^1_{d,1}$	$\dots$	$L^1_{d,d}$	$\dots$
$\dots D^2_{1,1}$	$\dots$	$L^n_{d,d}$								

Matrix A :

$A^1_{1,1}$	$A^1_{2,2}$	$\dots$	$A^1_{d,d}$	$\emptyset$	$A^1_{2,1}$	$\dots$	$A^1_{d,1}$	$\dots$	$\emptyset$	$\dots$
$\dots A^2_{1,1}$	$\dots$	$\emptyset$								

with $M^k_{i,j}$ being the element (i, j) of the k^{th} matrix M.

We choosed to store the diagonal elements of L to simplify our code.

This confguration allows us to compute the factorization in place.

The factorization

	Max Col	Max k (row)	row + shared memory
Execution time	0.322 ms	0.322 ms	0.0000

Figure 1: Comparison on small matrices. (100 matrices of size 32x32)

	Max Col	Max k (row)	row + shared memory
Execution time	1125.9 ms	1108.9 ms	0.007936 ms

Figure 2: Comparison on large matrices. (100 matrices of size 512x512)

Error propagation

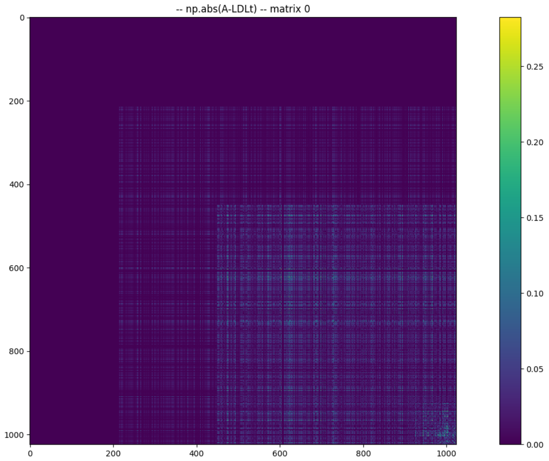

Figure 3: Error propagation on a big matrice

The solver

d =	16	128	512
Execution time	0.084 ms	0.960 ms	12.50 ms

Figure 4: Comparison with 128 threads and 100 matrices (on per block)

Behavior

We have a gain of time which is linear in the number of threads.

The full pipeline

	Max Col	Max k (row)	row + shared memory
Execution time	1108.7ms	1163.1 ms	0.0091 ms
Solving time	13.9 ms	13.9 ms	13.9 ms

Figure 5: Comparison on large matrices. (100 matrices of size 512x512)

The end

Figure 6: A pangolin, probably the source of our current sorrows.