Batched Least Squares of Tall Skinny Matrices on GPUs

Tim Schmielau and Jacques du Toit

Linear Least Squares via SVD

We consider linear least squares problems

$$\min_{\beta \in \mathbb{R}^n} \left\| y - X\beta \right\|_2$$

for $y \in \mathbb{R}^m$ and $X \in \mathbb{R}^{m \times n}$ where X is "tall and skinny", i.e. $m \gg n$. We focus on problems where

$$500 \leq m \leq 100,000 \quad \text{and} \quad 3 \leq n \leq 40$$

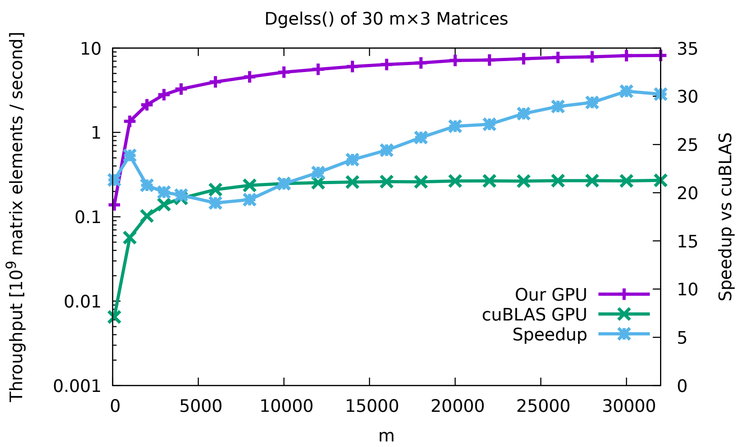
although we can also handle matrices outside this range. The classical algorithm takes a QR factorization of X followed by an SVD of the n by n matrix R, and we do the same. The problem is that QR scales badly for tall skinny X since it seeks parallelism across columns.

Batching Independent Problems Together

Better performance is possible if independent problems are batched together. This is often possible, e.g. XVA applications in finance. NAG has made a highly optimized GPU code allowing different sized matrices in the batch.

Performance vs. NVIDIA Libraries on P100

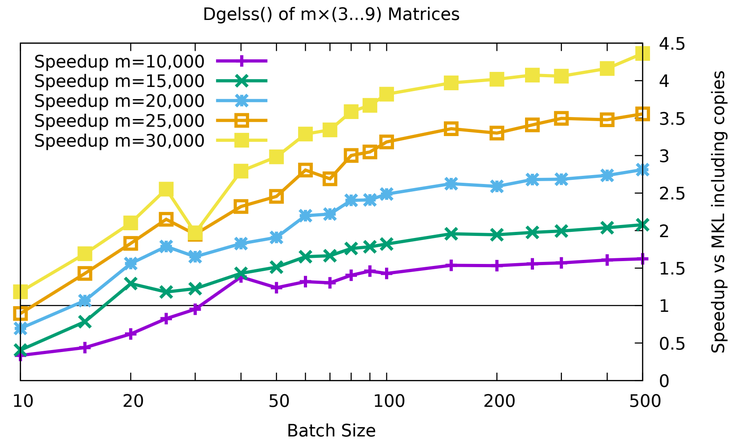
A batched GPU solver can be made using `cuBLAS` and `cuSolver`, but performance is not good and all matrices in the batch must be of the same size.



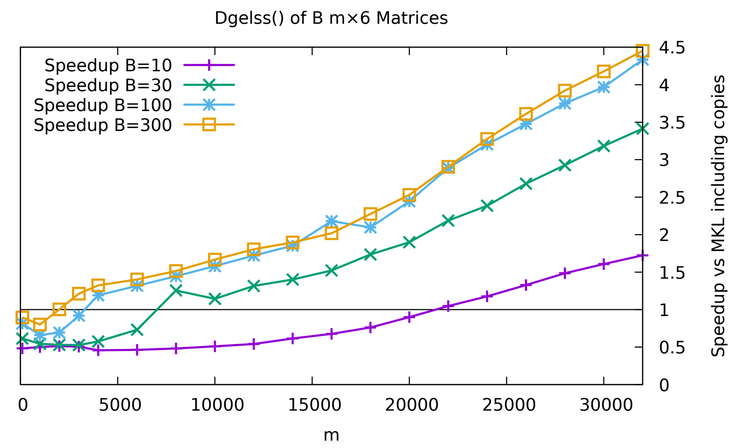
This graph shows our code vs. the NVIDIA libraries for 30 double precision m by 3 matrices. Our code is much faster. Results for different size batches and matrices and for single precision are similar.

Use in CPU-Only Applications

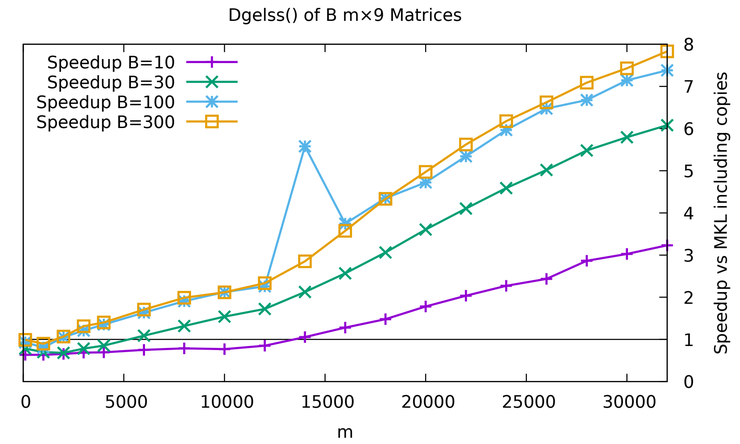
On CPU we loop in parallel over all the matrices in the batch, and then process each matrix serially by calling `sgelss` or `dgelss` in MKL. This gives good performance on our test system Intel Core i7-7700K @ 4.3GHz with 8 threads. It can solve relatively small problems in the time it would take to transfer the problem to the GPU through the PCIe3 bus. Although our code on P100 is over 20x faster than the CPU, when data transfers are included we see speedups only for larger problems.



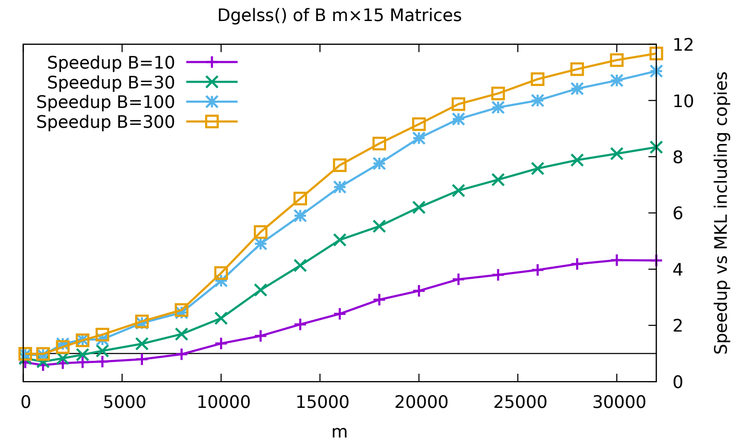
This plot shows speedup vs. MKL including transfers for batches where each matrix has m rows and a random Uniform$(3, 9)$ number of columns.



The plot above shows speedup vs. MKL including all transfers for batches of m by 6 matrices. Larger problem sizes give larger speedups.



These plots show results for batches of m by 9 and m by 15 matrices. The speedups increase strongly the more columns each matrix has.



Single precision results are similar.

Matrices Arising from Basis Functions

Often X represents basis functions (e.g. low order polynomials) evaluated at m realizations of one or two underlying factors. In this case, significant savings are possible by only transferring the factors to the GPU and evaluating the basis functions there to construct the matrices. Speedups of 10x to 20x should be possible depending on the problem.

Code Availability

The code is available to trial. Please contact **support@nag.co.uk** to arrange access.