D9

Tensor Core Acceleration of Math Intensive Kernels in QUDA

M. A. Clark (mclark@nvidia.com), Jiqun Tu (jtu@nvidia.com), Mathias Wagner (mathiasw@nvidia.com) and Evan Weinberg (eweinberg@nvidia.com)

Tensor Core on NVIDIA GPUs

The matrix multiply-accumulate (MMA) operations are drastically sped up on NVIDIA GPUs with the tensor core units, which are available starting from the Volta architecture. Starting from the Ampere architecture, a variety of data types, including FP64, FP16, BF16 (Brain Floating Point Format, or bfloat16), TF32 (TensorFloat-32), INT8, INT4, and Binary, are supported. On an A100 GPU, the theoretical peak half precision MMA (HMMA) performance reaches a stunning 312 TFLOPS. We have been working on using the tensor core units to accelerates the math intensive workflow in lattice QCD calculations under the framework of QUDA, a library for performing calculations in lattice QCD on GPUs. A light-weighted abstraction of the CUDA PTX MMA instruction is added in order to efficiently stage data through the different layers of GPU memory.

Multi-BLAS Operations

The multi-BLAS operations, which calculates block inner products of the shape,

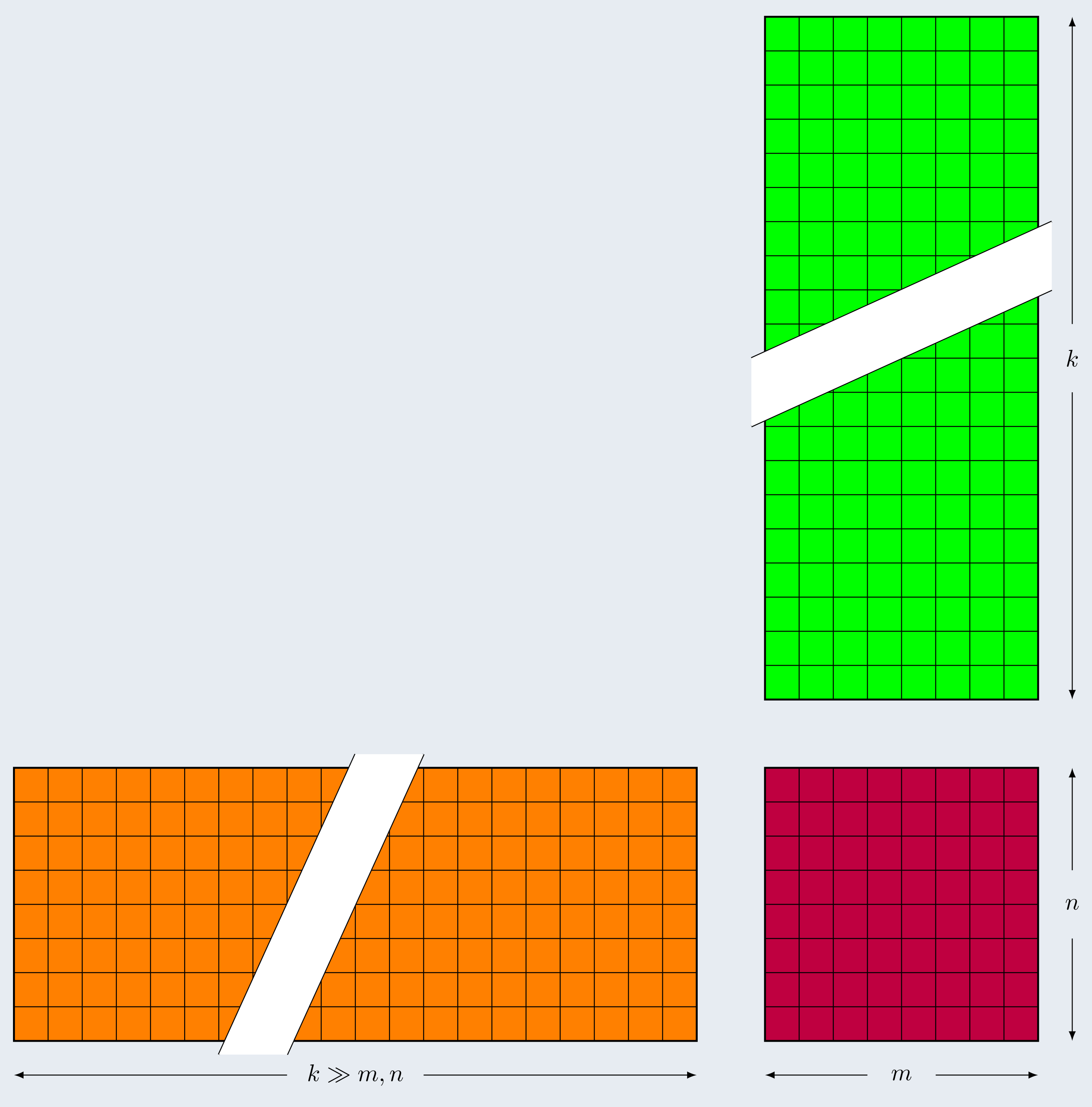
$$c_{i,j} = \langle a_i, b_j \rangle,\ i \leq m,\ j \leq n,$$

are equivalent to general matrix multiplications (GEMMs) of shape (m,n,k), i.e the multiplication of an m-by-k matrix A with an k-by-n matrix B to get an m-by-n matrix C, where $k \gg m,n$ is the degrees of freedom of the underlying vectors. They become compute bound as m and n go larger, e.g. in the block-orthorgonalization part of the multi-grid setup steps. The tensor core units, especially the double precision MMA (DMMA) instructions, could be used to speed up these math intensive operations. It is also possible to use three BF16 MMA to reach the accuracy of a 16-bit MMA for multi-BLAS performed with 16-bit half precision: this is done by representing the 15 ($= 16-1$ due to the sign bit) bits with two BF16 numbers (each has 7 bits for mantissa) whose total mantissa bits are sufficient to cover the needed 15 bits, i.e. $(1+7)\times 2 > 15$. The arithmetic is done in the natural high-low way:

$$(a_{high}+a_{low})\times(b_{high}+b_{low}) = a_{high}\times b_{high} + a_{high}\times b_{low} + a_{low}\times b_{high} + \cancel{a_{low}\times b_{low}},$$

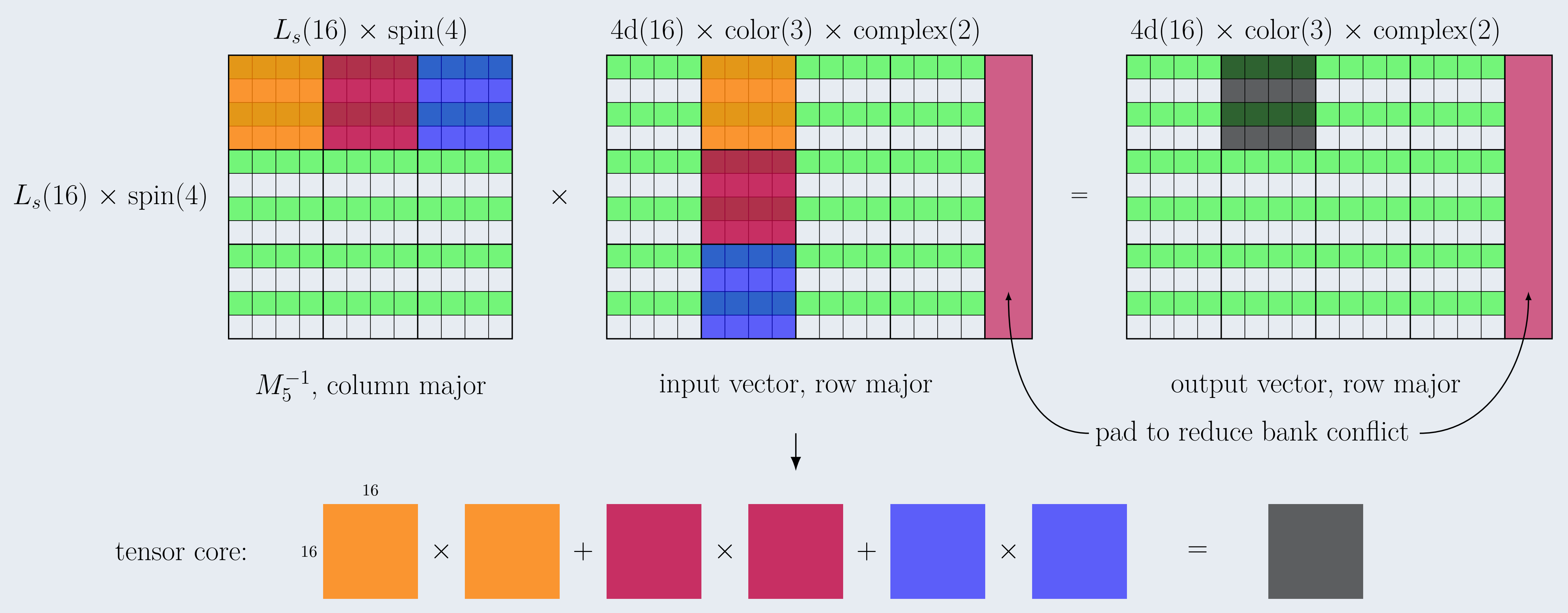
where the last contribution is dropped.

This is still work in progress.



Local Preconditioning for Domain Wall Fermion

In [arXiv:2104.05615] the multi-splitting algorithm is used as a preconditioner for the domain wall Dirac operator in order to reduce the inter-node communication cost, at the expense of performing more on-node floating point and memory operations. The local preconditioner, whose performance is critical to achieving an overall speed up with the algorithm, is implemented with, among other strategies, the half precision MMA (HMMA) instruction to perform the application of the M_5^{-1} operator, which is in nature a dense matrix multiplication.



Coarse Link Construction in Multi-grid Setup

Tensor cores are used to accelerate the dense matrix multiplications in coarse link construction in the multi-grid setup steps [arXiv:1612.07873], where dense complex matrix multiplications are performed between fine grid gauge links U (or the generalized coarse links Y in the case of coarsening the coarse links) and near-null vectors V:

$$Y = V^{\dagger}UV.$$

In practice the link is constructed by first computing the UV product and then $V^{\dagger}UV$. The resulting coarse grid operator takes the form

$$\hat{D} = X\delta_{\hat{x},\hat{y}} + \sum_{\mu} Y^{\mu}\delta_{\hat{x},\hat{y}-\hat{\mu}} + Y^{-\mu}\delta_{\hat{x},\hat{y}+\hat{\mu}},$$

where X is the diagonal part of $V^{\dagger}UV$ (the contribution when U does not cross the sub-block boundaries), $\hat{x}$ and $\hat{y}$ are the coarse grid coordinates. To take advantage of the Schur decomposition the even-odd preconditioned coarse links $\hat{Y}$ are to be calculated,

$$\hat{Y} = X^{-1}Y.$$

We need to take a custom approach towards what would otherwise be a batched-GEMM operation to take advantage of QUDA's unique 16-bit fixed-precision format, exploit maximum parallelism, and maintain determinism:

1. Coarse links are aggregates of fine-grid contractions within sub-blocks. To maximize parallelism while maintaining determinism, portions of $V^{\dagger}UV$ are computed locally via tensor core acceleration, then converted to a 32-bit integer format before atomic accumulation.
2. The 16-bit fixed-point representation (half precision) requires a scale set by the maximum absolute value of the coarse links. To optimize the dynamic range of the 16-bit format, a two-pass algorithm is needed: in the first pass the matrix multiplication is computed but only the maximum absolute value among the entries of the output matrix is recorded. The output matrix is computed and stored in the second pass after the input matrices are scaled according to the maximum value obtained from the first pass.

We see significant speed up from using tensor cores in all stages of coarse link construction. The following numbers are measured on a workflow coarsening from a $8\times8\times8\times16$ grid with $N_c = 24$ to a $4\times4\times4\times8$ grid with $N_c = 64$. 16-bit half precision are used so the TFLOPS and speed up numbers cover both passes.

Stage	V100 TFLOPS	V100 speed up	A100 TFLOPS	A100 speed up
UV	22.7	13.0	48.4	12.1
$V^{\dagger}UV$	6.5	1.5	11.4	2.0
$X^{-1}Y$	42.0	9.2	70.6	11.6