

Triangular Matrix Inversion

(a survey of sequential approaches)

Nicholas Knight

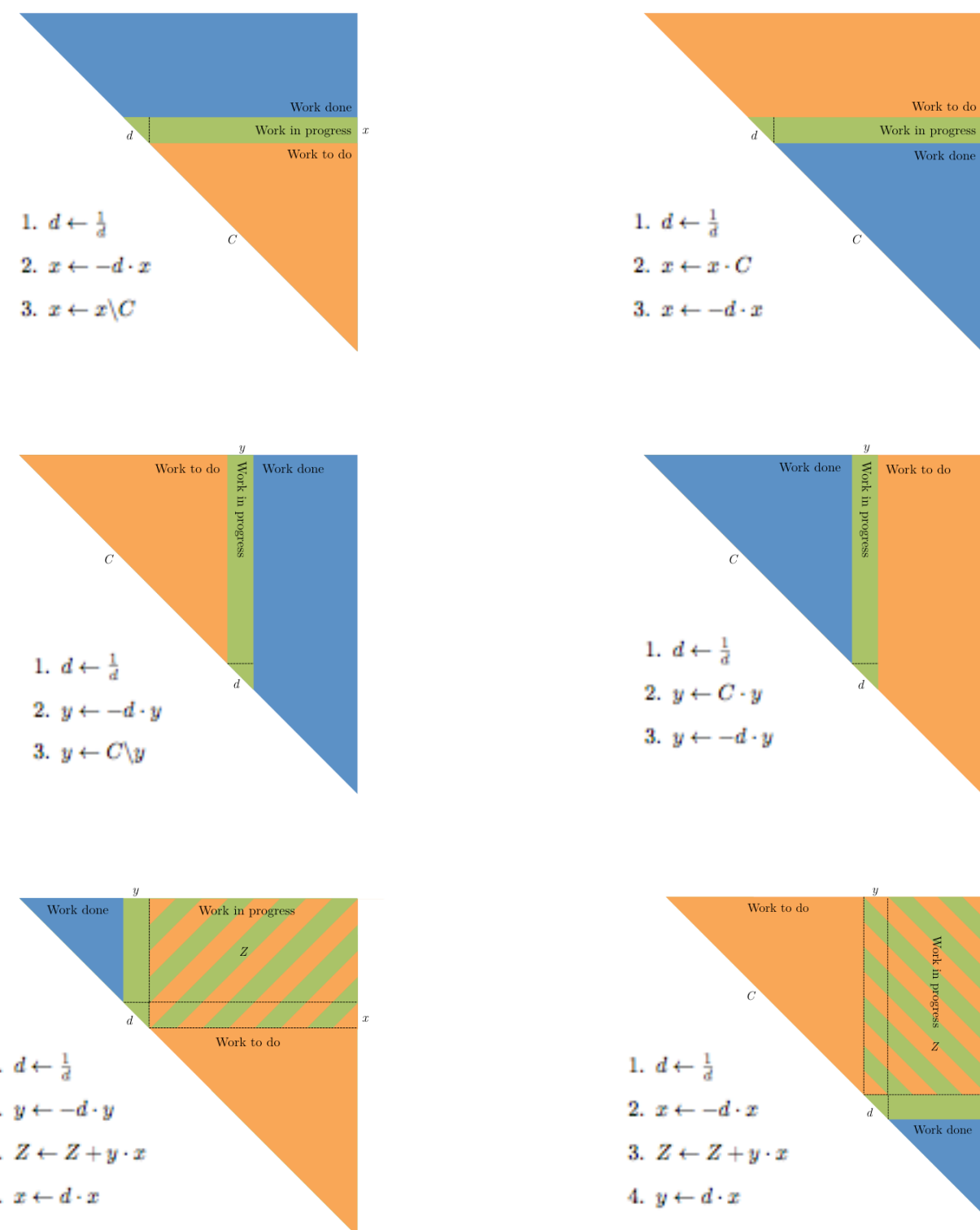
Research supported by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (Award #DIG07-10227)

Background

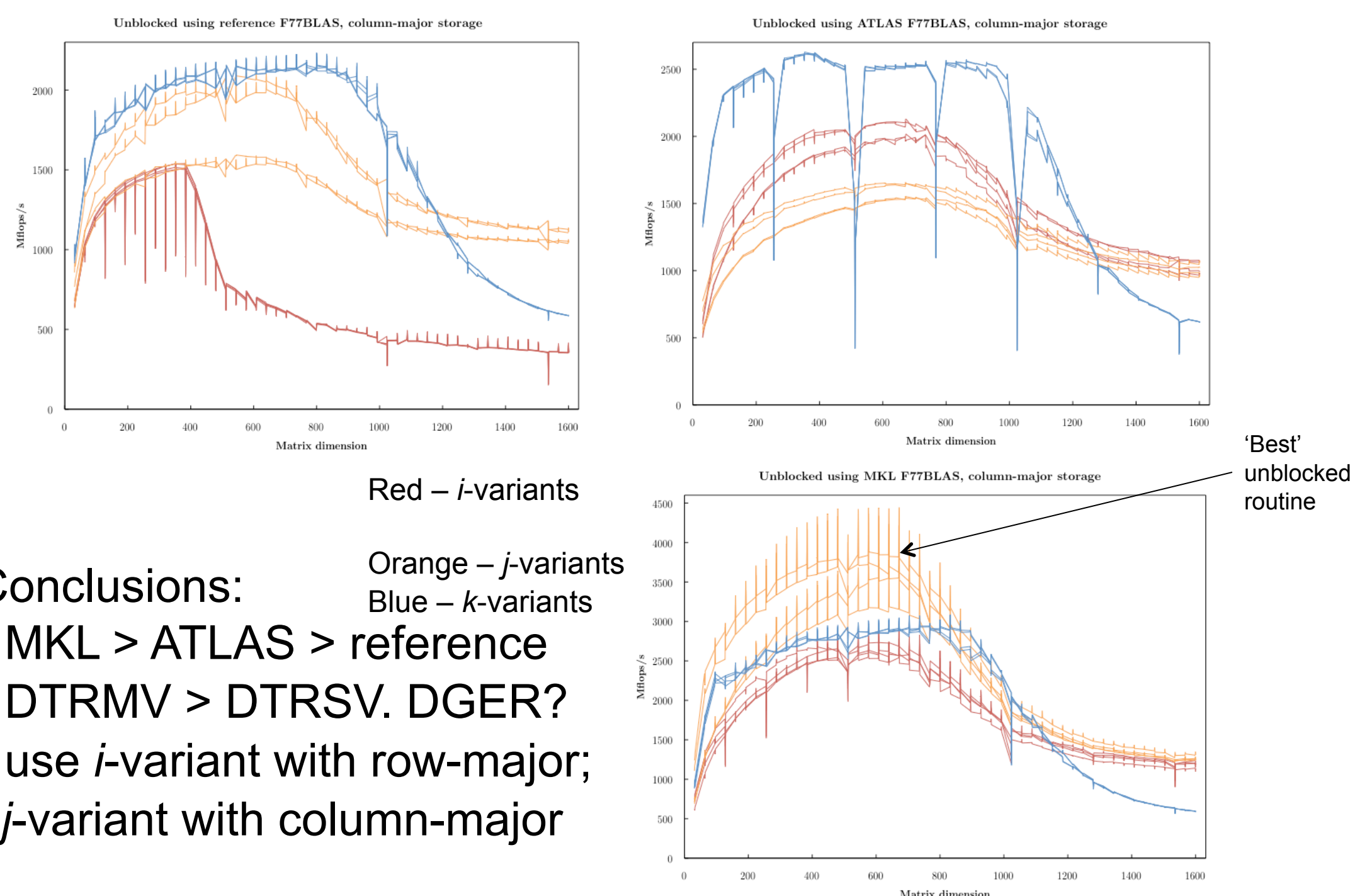
Need a matrix inverse (explicitly) calculated?

- Take the LU factorization, invert the L and/or U matrices using triangular matrix inversion (TMI).
- TMI costs $n^3/3$ flops (using $O(n^3)$ matrix multiply, etc.)
- Standard implementation: LAPACK's xTRTRI.

Unblocked algorithms



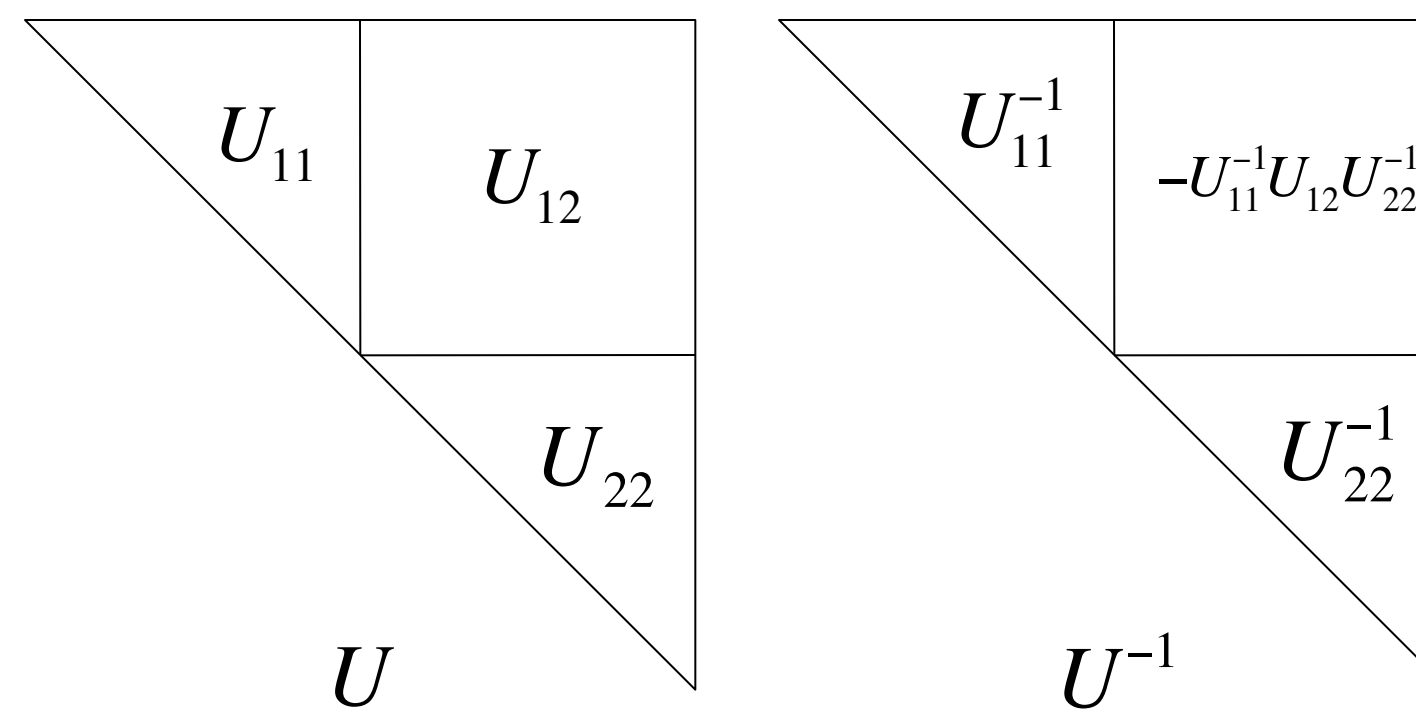
Results



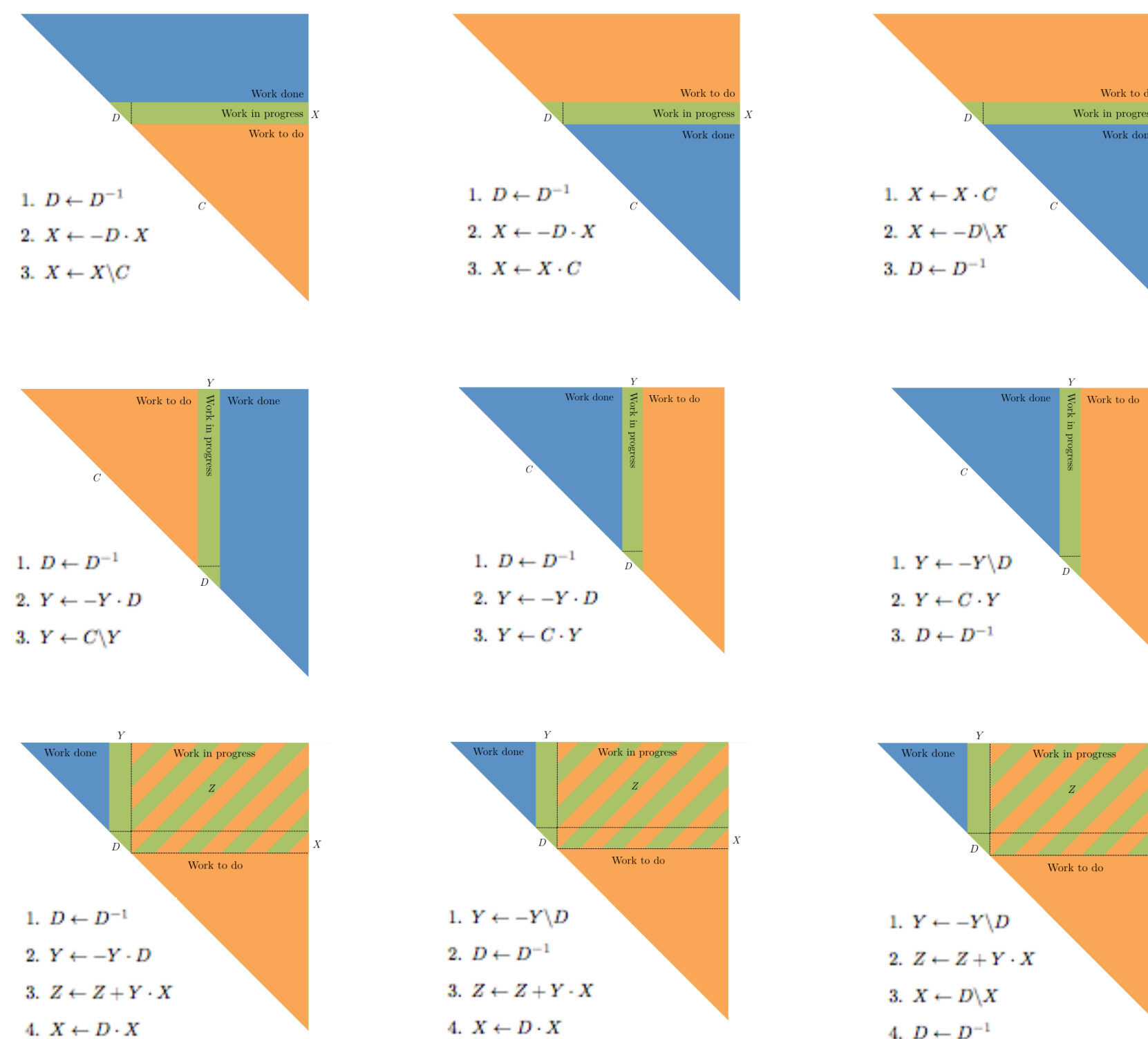
- Conclusions:
- MKL > ATLAS > reference
 - DTRMV > DTRSV. DGER?
 - use i -variant with row-major; j -variant with column-major

Algorithm Design I

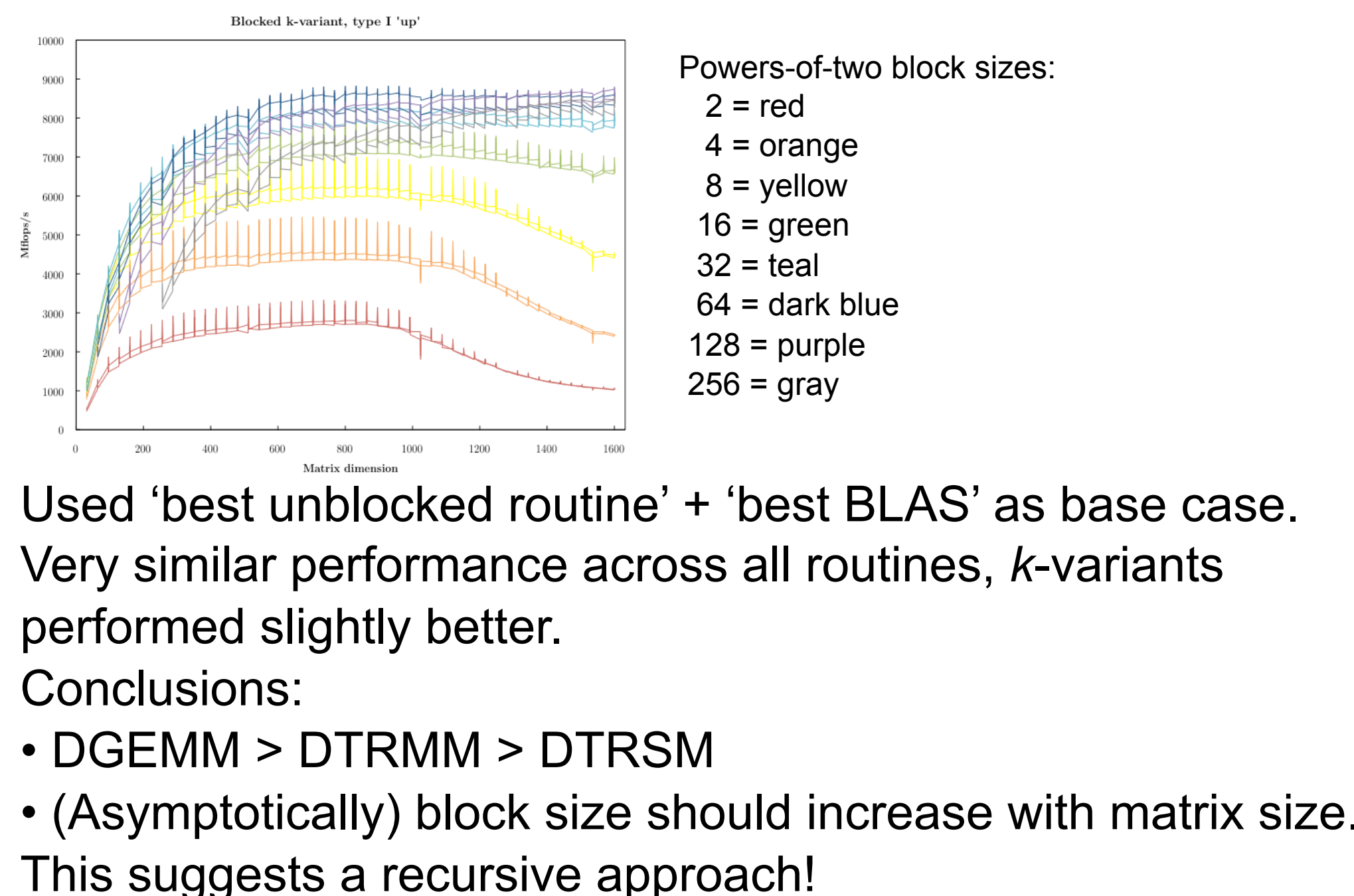
Lots of possible algorithms, since the inverse of a triangular matrix has a special structure:



Blocked algorithms



Results

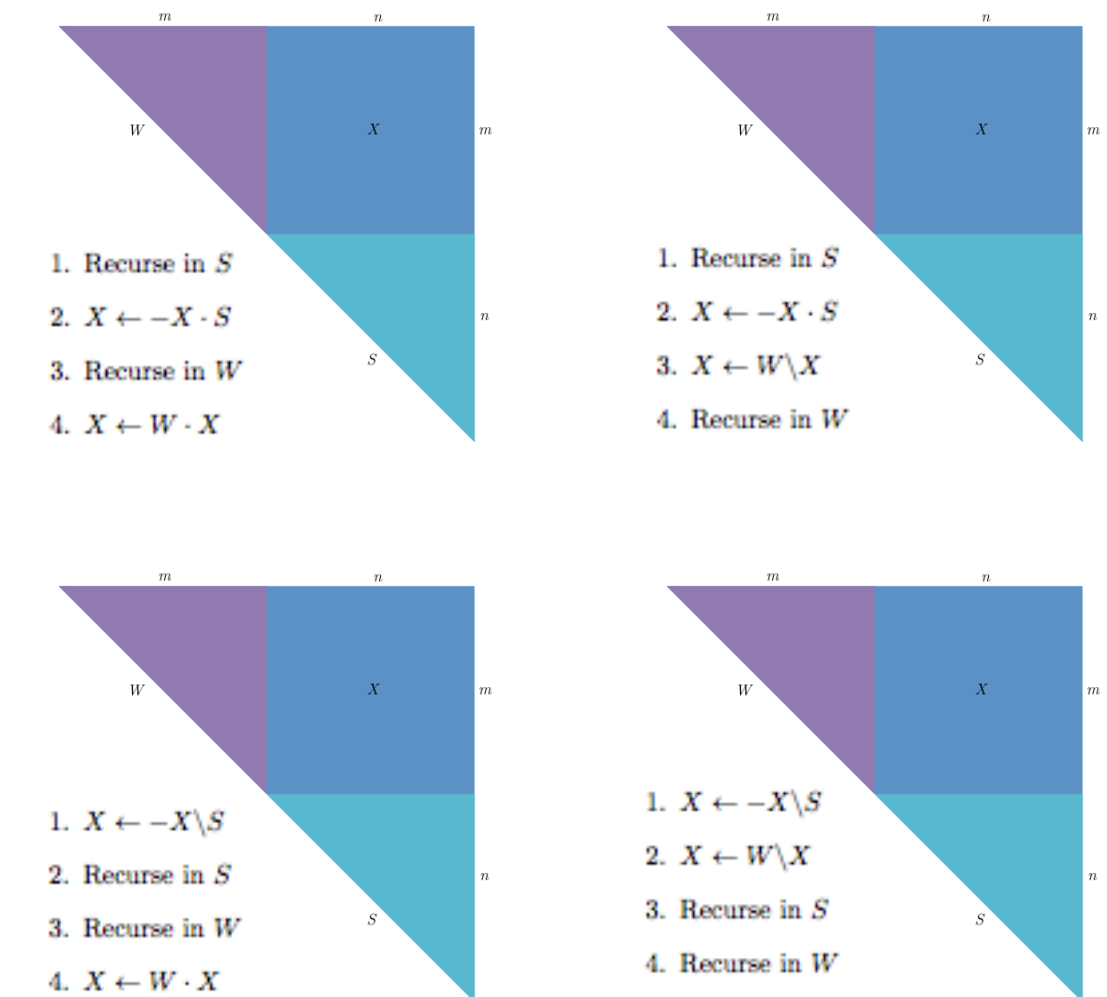


Algorithm Design II

My design considerations:

- Unblocked vs. blocked vs. recursive
- i -, j -, and k -variants
- xTRMV vs. xTRSV; xTRMM vs. xTRSM
- BLAS library choice (reference, ATLAS, MKL)
- Column-major vs. row-major storage
- Only consider double-precision real ('D')

Recursive algorithms

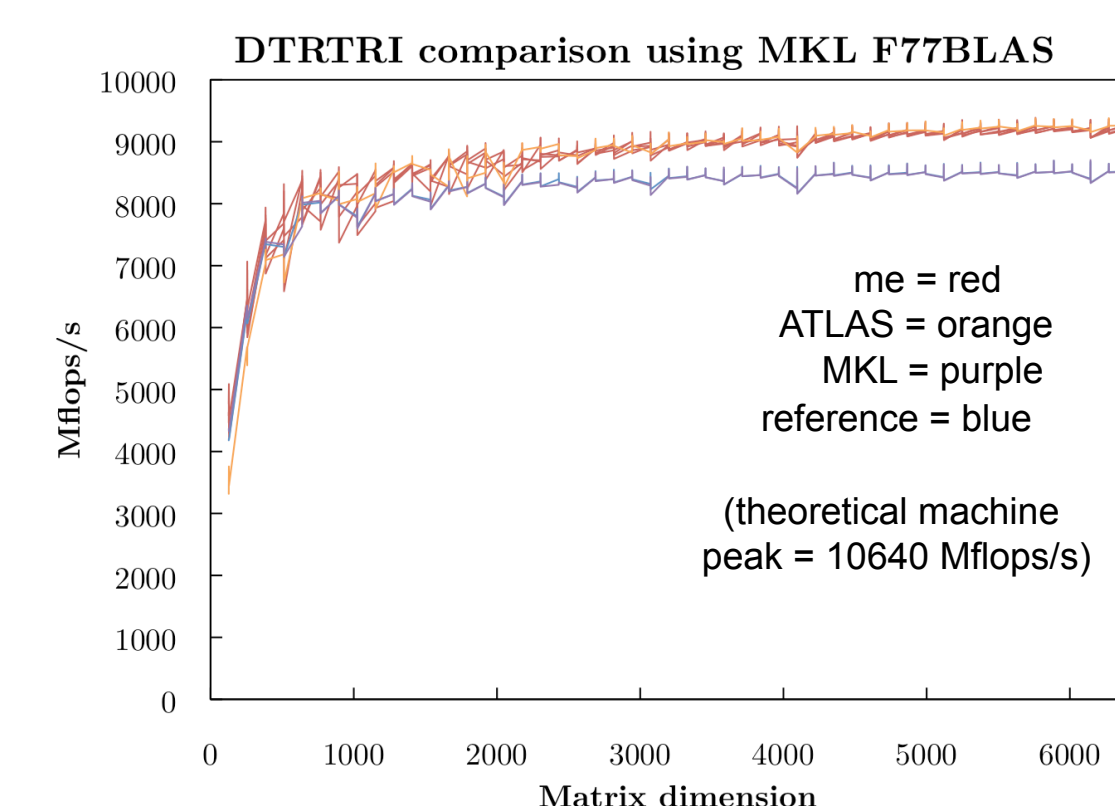
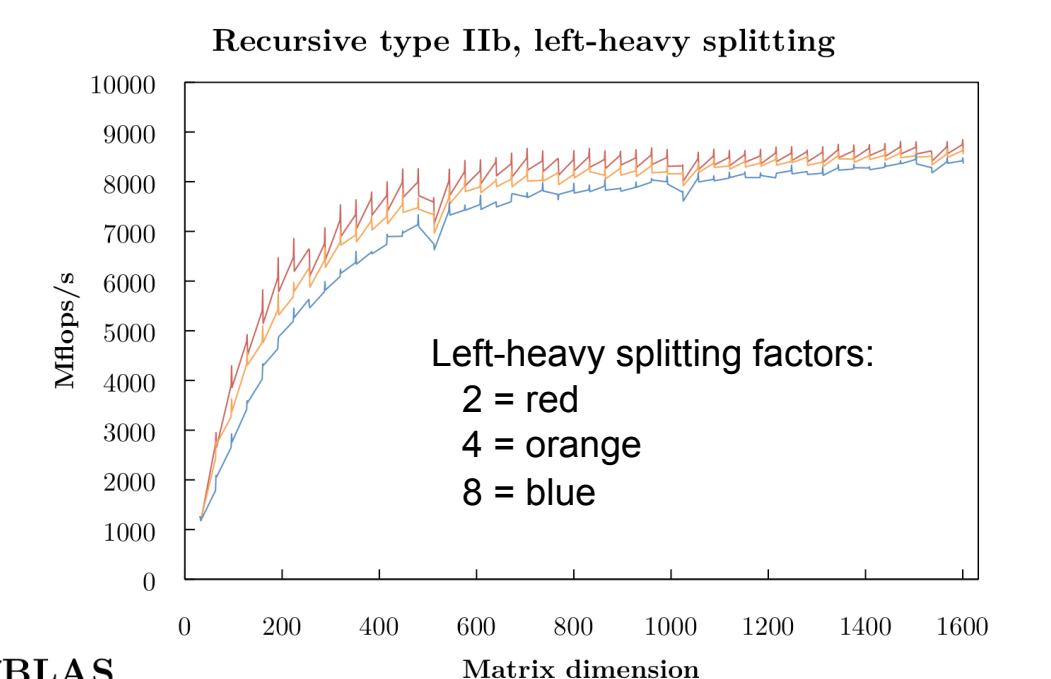


Results

Conclusions:

- All base-case sizes asymptotically the same – $b=32$ seems best over these matrices.
- No significant difference between DTRSM-heavy and DTRMM-heavy routines for larger matrices.
- Unbalanced subproblem 'splitting' not helpful:

- Recursive formulation is best! (Appears that MKL is using a blocked code):



- My codes are best for small ($N < 512$) matrices, and a very close second-place for larger matrices.

