Triangular Matrix Inversion
(a survey of sequential approaches)
Nicholas Knight

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.

Algorithm Design I

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

\[
\begin{bmatrix}
U_{11} & U_{12} \\
U_{21} & U_{22}
\end{bmatrix}
\rightarrow
\begin{bmatrix}
U_{11} & -U_{12}U_{22}^{-1} \\
0 & U_{22}
\end{bmatrix}
\]

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’)

Unblocked algorithms

Blocked algorithms

Results

Conclusions:
• MKL > ATLAS > reference
• DTRMV > DTRSV. DGER?

Conclusions:
• DGEMM > DTRMM > DTRSM
• (Asymptotically) block size should increase with matrix size.

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!

Results

Recursive algorithms

 Used ‘best unblocked routine’ * ‘best BLAS’ as base case. Very similar performance across all routines, $k$-variants performed slightly better.

Conclusions:
• DGEMM > DTRMM > DTRSM
• (Asymptotically) block size should increase with matrix size. This suggests a recursive approach!

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