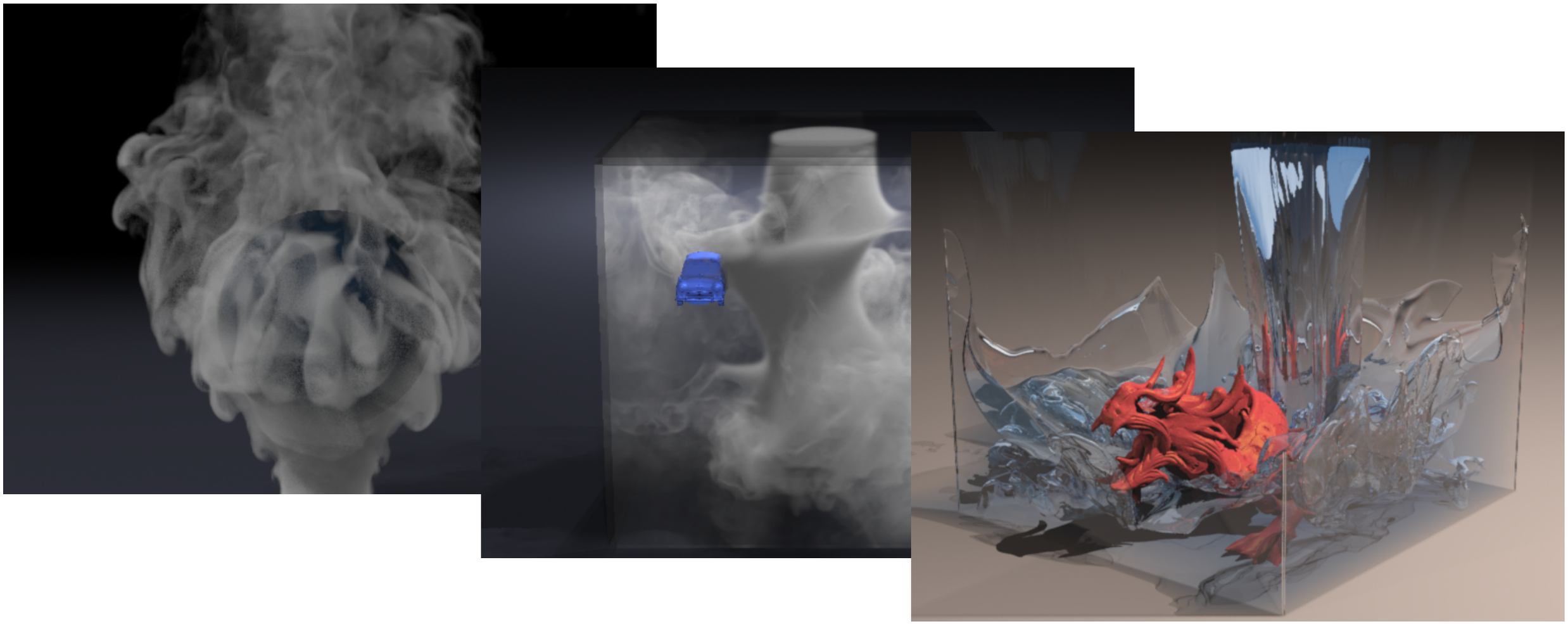

Dense Matrix Computations
Additional BLAS Level 3 Operations
(MKL interface, matrix formats and conventions)

Introduction

Last few lectures we focused on GEMM (General Matrix-Matrix Multiply) operations, and surveyed possible optimizations they can use

Key takeaways:
- Utilization of memory (using cache lines fully and efficiently, creating "local" copies of frequently reusable memory when needed) was essential for some of the most impactful optimizations
- Best use of parallel computation potential is possible when we can make a kernel be compute-bound (as opposed to memory-bound) and data transformations (blocking) can help with that

Optimization best practices:
- Always have checks/tests that validate the correctness of your implementation against a trusted "baseline" alternative (in the case of MKL, the alternative was highly optimized, but generally you will be seeking to do that optimization)
- For the most invasive of optimizations (e.g. those that require careful profiling or looking at assembly code) try to factor out the code you are focusing on into as small units as possible. This makes analysis easier
- Only sparingly go down to the level of intrinsics, and only if you have to.

Today's lecture

We survey additional dense linear algebra routines in the repertoire of MKL, mostly involving matrix-matrix operations
(they offer the best chances for parallel accelerations)

We will not descend into low-level implementation details; however:
- Many of these routines rely on tricks similar to GEMM (so we can discuss them at high-level and understand the spirit of the optimizations used)
- We can often appreciate what is the potential of these kernels for parallel performance based on similar ones we analyzed in detail

Categories of routines in our survey:
- Dense BLAS Level 3 routines beyond GEMM
(still involving matrix-matrix operations)
- LAPACK routines on space-saving matrix formats
- A few other special-purpose routines

BLAS Level 3 Routines

BLAS Level 3 routines perform matrix-matrix operations. The following table lists the BLAS Level 3 routine groups and the data types associated with them.

BLAS Level 3 Routine Groups and Their Data Types		
Routine Group	**Data Types**	**Description**
`cblas_?gemm`	s, d, c, z	Computes a matrix-matrix product with general matrices.
`cblas_?hemm`	c, z	Computes a matrix-matrix product where one input matrix is Hermitian.
`cblas_?herk`	c, z	Performs a Hermitian rank-k update.
`cblas_?her2k`	c, z	Performs a Hermitian rank-2k update.
`cblas_?symm`	s, d, c, z	Computes a matrix-matrix product where one input matrix is symmetric.
`cblas_?syrk`	s, d, c, z	Performs a symmetric rank-k update.
`cblas_?syr2k`	s, d, c, z	Performs a symmetric rank-2k update.
`cblas_?trmm`	s, d, c, z	Computes a matrix-matrix product where one input matrix is triangular.
`cblas_?trsm`	s, d, c, z	Solves a triangular matrix equation.

BLAS Level 3 Routines

BLAS Level 3 routines perform matrix-matrix operations. The following table lists the BLAS Level 3 routine groups and the data types associated with them.

BLAS Level 3 Routine Groups and Their Data Types		
Routine Group	**Data Types**	**Description**
cblas_?gemm	s, d, c, z	Computes a matrix-matrix product with general matrices.
cblas_?hemm	c, z	Computes a matrix-matrix product where one input matrix is Hermitian.
cblas_?herk	c, z	Performs a Hermitian rank-k update.
cblas_?her2k	c, z	Performs a Hermitian rank-2k update.
cblas_?symm	s, d, c, z	Computes a matrix-matrix product where one input matrix is symmetric.
cblas_?syrk	s, d, c, z	Performs a symmetric rank-k update.
cblas_?syr2k	s, d, c, z	Performs a symmetric rank-2k update.
cblas_?trmm	s, d, c, z	Computes a matrix-matrix product where one input matrix is triangular.
cblas_?trsm	s, d, c, z	Solves a triangular matrix equation.

Refresher on forward substitution

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_i \\ \vdots \\ b_N \end{pmatrix}$$

Refresher on forward substitution

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_i \\ \vdots \\ b_N \end{pmatrix}$$

```
for i = 1...N
    x_i ← b_i
    for j = 1...(i-1)
        x_i ← x_i − l_ij x_j
    endfor
    x_i ← x_i / l_ii
endfor
```

Refresher on forward substitution

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \boxed{x_i} \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ \boxed{b_i} \\ \vdots \\ b_N \end{pmatrix}$$

```
for i = 1...N
    x_i ← b_i
    for j = 1...(i-1)
        x_i ← x_i − l_ij x_j
    endfor
    x_i ← x_i / l_ii
endfor
```

Refresher on forward substitution

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_i \\ \vdots \\ b_N \end{pmatrix}$$

```
for i = 1...N
    x_i ← b_i
    for j = 1...(i-1)
        x_i ← x_i − l_ij x_j
    endfor
    x_i ← x_i / l_ii
endfor
```

Refresher on forward substitution

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_i \\ \vdots \\ b_N \end{pmatrix}$$

```
for i = 1...N
    x_i ← b_i
    for j = 1...(i-1)
        x_i ← x_i − l_ij x_j
    endfor
    x_i ← x_i / l_ii
endfor
```

Refresher on forward substitution

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_i \\ \vdots \\ b_N \end{pmatrix}$$

```
for i = 1...N
    x_i ← b_i
    for j = 1...(i-1)
        x_i ← x_i − l_ij x_j
    endfor
    x_i ← x_i / l_ii
endfor
```

Forward substitution on matrices

$$LX = B$$

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} x_{11} & \dots & x_{1M} \\ x_{21} & \dots & x_{2M} \\ x_{31} & \dots & x_{3M} \\ \vdots & & \vdots \\ x_{i1} & \dots & x_{iM} \\ \vdots & & \vdots \\ x_{N1} & \dots & x_{NM} \end{pmatrix} = \begin{pmatrix} b_{11} & \dots & b_{1M} \\ b_{21} & \dots & b_{2M} \\ b_{31} & \dots & b_{3M} \\ \vdots & & \vdots \\ b_{i1} & \dots & b_{iM} \\ \vdots & & \vdots \\ b_{N1} & \dots & b_{NM} \end{pmatrix}$$

Forward substitution on matrices

$$LX = B$$

cblas_?trsm

Solves a triangular matrix equation.

Syntax

```
void cblas_strsm (const CBLAS_LAYOUT Layout, const CBLAS_SIDE side, const CBLAS_UPLO
uplo, const CBLAS_TRANSPOSE transa, const CBLAS_DIAG diag, const MKL_INT m, const
MKL_INT n, const float alpha, const float *a, const MKL_INT lda, float *b, const MKL_INT
ldb);

void cblas_dtrsm (const CBLAS_LAYOUT Layout, const CBLAS_SIDE side, const CBLAS_UPLO
uplo, const CBLAS_TRANSPOSE transa, const CBLAS_DIAG diag, const MKL_INT m, const
MKL_INT n, const double alpha, const double *a, const MKL_INT lda, double *b, const
MKL_INT ldb);
```

Forward substitution on matrices

$$LX = B$$

Description

The `?trsm` routines solve one of the following matrix equations:

```
op(A)*X = alpha*B,
```

or

```
X*op(A) = alpha*B,
```

where:

alpha is a scalar,

X and *B* are *m*-by-*n* matrices,

A is a unit, or non-unit, upper or lower triangular matrix, and

`op(A)` is one of `op(A) = A`, or `op(A) = A'`, or `op(A) = conjg(A')`.

The matrix *B* is overwritten by the solution matrix *X*.

Forward substitution on matrices

$$LX = B$$

Input Parameters

Layout	Specifies whether two-dimensional array storage is row-major (`CblasRowMajor`) or column-major (`CblasColMajor`).
side	Specifies whether `op(A)` appears on the left or right of `X` in the equation: if *side* = `CblasLeft`, then `op(A)*X = alpha*B`; if *side* = `CblasRight`, then `X*op(A) = alpha*B`.
uplo	Specifies whether the matrix `A` is upper or lower triangular. *uplo* = `CblasUpper` if *uplo* = `CblasLower`, then the matrix is low triangular.
transa	Specifies the form of `op(A)` used in the matrix multiplication: if *transa*=`CblasNoTrans`, then `op(A) = A`; if *transa*=`CblasTrans`; if *transa*=`CblasConjTrans`, then `op(A) = conjg(A')`.

Forward substitution on matrices

$$LX = B$$

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} x_{11} & \dots & x_{1M} \\ x_{21} & \dots & x_{2M} \\ x_{31} & \dots & x_{3M} \\ \vdots & & \vdots \\ x_{i1} & \dots & x_{iM} \\ \vdots & & \vdots \\ x_{N1} & \dots & x_{NM} \end{pmatrix} = \begin{pmatrix} b_{11} & \dots & b_{1M} \\ b_{21} & \dots & b_{2M} \\ b_{31} & \dots & b_{3M} \\ \vdots & & \vdots \\ b_{i1} & \dots & b_{iM} \\ \vdots & & \vdots \\ b_{N1} & \dots & b_{NM} \end{pmatrix}$$

Forward substitution on matrices

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} x_{11} & \dots & x_{1M} \\ x_{21} & \dots & x_{2M} \\ x_{31} & \dots & x_{3M} \\ \vdots & & \vdots \\ x_{i1} & \dots & x_{iM} \\ \vdots & & \vdots \\ x_{N1} & \dots & x_{NM} \end{pmatrix} = \begin{pmatrix} b_{11} & \dots & b_{1M} \\ b_{21} & \dots & b_{2M} \\ b_{31} & \dots & b_{3M} \\ \vdots & & \vdots \\ b_{i1} & \dots & b_{iM} \\ \vdots & & \vdots \\ b_{N1} & \dots & b_{NM} \end{pmatrix}$$

Forward substitution on matrices

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} x_{11} & \dots & x_{1M} \\ x_{21} & \dots & x_{2M} \\ x_{31} & \dots & x_{3M} \\ \vdots & & \vdots \\ x_{i1} & \dots & x_{iM} \\ \vdots & & \vdots \\ x_{N1} & \dots & x_{NM} \end{pmatrix} = \begin{pmatrix} b_{11} & \dots & b_{1M} \\ b_{21} & \dots & b_{2M} \\ b_{31} & \dots & b_{3M} \\ \vdots & & \vdots \\ b_{i1} & \dots & b_{iM} \\ \vdots & & \vdots \\ b_{N1} & \dots & b_{NM} \end{pmatrix}$$

Forward substitution on matrices

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} x_{11} & \dots & x_{1M} \\ x_{21} & \dots & x_{2M} \\ x_{31} & \dots & x_{3M} \\ \vdots & & \vdots \\ x_{i1} & \dots & x_{iM} \\ \vdots & & \vdots \\ x_{N1} & \dots & x_{NM} \end{pmatrix} = \begin{pmatrix} b_{11} & \dots & b_{1M} \\ b_{21} & \dots & b_{2M} \\ b_{31} & \dots & b_{3M} \\ \vdots & & \vdots \\ b_{i1} & \dots & b_{iM} \\ \vdots & & \vdots \\ b_{N1} & \dots & b_{NM} \end{pmatrix}$$

Forward substitution on matrices

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} x_{11} & \dots & x_{1M} \\ x_{21} & \dots & x_{2M} \\ x_{31} & \dots & x_{3M} \\ \vdots & & \vdots \\ x_{i1} & \dots & x_{iM} \\ \vdots & & \vdots \\ x_{N1} & \dots & x_{NM} \end{pmatrix} = \begin{pmatrix} b_{11} & \dots & b_{1M} \\ b_{21} & \dots & b_{2M} \\ b_{31} & \dots & b_{3M} \\ \vdots & & \vdots \\ b_{i1} & \dots & b_{iM} \\ \vdots & & \vdots \\ b_{N1} & \dots & b_{NM} \end{pmatrix}$$

Forward substitution on matrices

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} x_{11} & \dots & x_{1M} \\ x_{21} & \dots & x_{2M} \\ x_{31} & \dots & x_{3M} \\ \vdots & & \vdots \\ x_{i1} & \dots & x_{iM} \\ \vdots & & \vdots \\ x_{N1} & \dots & x_{NM} \end{pmatrix} = \begin{pmatrix} b_{11} & \dots & b_{1M} \\ b_{21} & \dots & b_{2M} \\ b_{31} & \dots & b_{3M} \\ \vdots & & \vdots \\ b_{i1} & \dots & b_{iM} \\ \vdots & & \vdots \\ b_{N1} & \dots & b_{NM} \end{pmatrix}$$

Could use multithreading to handle multiple columns, but:
- Doesn't help with the fact that the algorithm is memory-bound
(the left hand side has to be read separately by every thread)
- Doesn't help with the fact that the forward substitution algorithm
is still seemingly serial
(iterations of the outer for-loop are dependent on one another)

Forward substitution on matrices

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} \dots & \mathbf{x}_1^T & \dots \\ \dots & \mathbf{x}_2^T & \dots \\ \dots & \mathbf{x}_3^T & \dots \\ & \vdots & \\ \dots & \mathbf{x}_i^T & \dots \\ & \vdots & \\ \dots & \mathbf{x}_N^T & \dots \end{pmatrix} = \begin{pmatrix} \dots & \mathbf{b}_1^T & \dots \\ \dots & \mathbf{b}_2^T & \dots \\ \dots & \mathbf{b}_3^T & \dots \\ & \vdots & \\ \dots & \mathbf{b}_i^T & \dots \\ & \vdots & \\ \dots & \mathbf{b}_N^T & \dots \end{pmatrix}$$

*Idea: Start thinking of matrices **X** & **B** as consisting of "block" elements (each of which is an M-length row)*

Forward substitution on matrices

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} \dots & \mathbf{x}_1^T & \dots \\ \dots & \mathbf{x}_2^T & \dots \\ \dots & \mathbf{x}_3^T & \dots \\ & \vdots & \\ \dots & \mathbf{x}_i^T & \dots \\ & \vdots & \\ \dots & \mathbf{x}_N^T & \dots \end{pmatrix} = \begin{pmatrix} \dots & \mathbf{b}_1^T & \dots \\ \dots & \mathbf{b}_2^T & \dots \\ \dots & \mathbf{b}_3^T & \dots \\ & \vdots & \\ \dots & \mathbf{b}_i^T & \dots \\ & \vdots & \\ \dots & \mathbf{b}_N^T & \dots \end{pmatrix}$$

```
for i = 1...N
   x_i^T ← b_i^T
   for j = 1...(i-1)
      x_i ← x_i − l_ij x_j
   endfor
   x_i ← (l_ii^-1) x_i
endfor
```

Forward substitution on matrices

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} \dots & \mathbf{x}_1^T & \dots \\ \dots & \mathbf{x}_2^T & \dots \\ \dots & \mathbf{x}_3^T & \dots \\ & \vdots & \\ \dots & \mathbf{x}_i^T & \dots \\ & \vdots & \\ \dots & \mathbf{x}_N^T & \dots \end{pmatrix} = \begin{pmatrix} \dots & \mathbf{b}_1^T & \dots \\ \dots & \mathbf{b}_2^T & \dots \\ \dots & \mathbf{b}_3^T & \dots \\ & \vdots & \\ \dots & \mathbf{b}_i^T & \dots \\ & \vdots & \\ \dots & \mathbf{b}_N^T & \dots \end{pmatrix}$$

```
for i = 1...N
    x_i^T ← b_i^T
    for j = 1...(i-1)
        x_i ← x_i − l_ij x_j
    endfor
    x_i ← (l_ii^-1) x_i
endfor
```

Forward substitution on matrices

$$\begin{pmatrix} \mathbf{L}_{11} & & & & & & \\ \mathbf{L}_{21} & \mathbf{L}_{22} & & & & & \\ \mathbf{L}_{31} & \mathbf{L}_{32} & \mathbf{L}_{33} & & & & \\ \vdots & & & \ddots & & & \\ \mathbf{L}_{i1} & \mathbf{L}_{i2} & \mathbf{L}_{i3} & \dots & \mathbf{L}_{ii} & & \\ \vdots & & & & & \ddots & \\ \mathbf{L}_{B1} & \mathbf{L}_{B2} & \mathbf{L}_{B3} & \mathbf{L}_{B4} & \mathbf{L}_{B5} & \dots & \mathbf{L}_{BB} \end{pmatrix} \begin{pmatrix} \dots & \mathbf{X}_1 & \dots \\ \dots & \mathbf{X}_2 & \dots \\ \dots & \mathbf{X}_3 & \dots \\ & \vdots & \\ \dots & \mathbf{X}_i & \dots \\ & \vdots & \\ \dots & \mathbf{X}_B & \dots \end{pmatrix} = \begin{pmatrix} \dots & \mathbf{B}_1 & \dots \\ \dots & \mathbf{B}_2 & \dots \\ \dots & \mathbf{B}_3 & \dots \\ & \vdots & \\ \dots & \mathbf{B}_i & \dots \\ & \vdots & \\ \dots & \mathbf{B}_B & \dots \end{pmatrix}$$

```
for i = 1...N
   X_i ← B_i
   for j = 1...(i-1)
      X_i ← X_i − L_ij X_j
   endfor
   X_i ← (L_ii^-1) X_i
endfor
```

Compare with scalar version ...

$$\begin{pmatrix} l_{11} & & & & & & \\ l_{21} & l_{22} & & & & & \\ l_{31} & l_{32} & l_{33} & & & & \\ \vdots & & & \ddots & & & \\ l_{i1} & l_{i2} & l_{i3} & \dots & l_{ii} & & \\ \vdots & & & & & \ddots & \\ l_{N1} & l_{N2} & l_{N3} & l_{N4} & l_{N5} & \dots & l_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_i \\ \vdots \\ b_N \end{pmatrix}$$

```
for i = 1...N
    x_i ← b_i
    for j = 1...(i-1)
        x_i ← x_i - l_ij x_j
    endfor
    x_i ← x_i / l_ii
endfor
```

Forward substitution on matrices

$$\begin{pmatrix} \mathbf{L}_{11} & & & & & & \\ \mathbf{L}_{21} & \mathbf{L}_{22} & & & & & \\ \mathbf{L}_{31} & \mathbf{L}_{32} & \mathbf{L}_{33} & & & & \\ \vdots & & & \ddots & & & \\ \mathbf{L}_{i1} & \mathbf{L}_{i2} & \mathbf{L}_{i3} & \dots & \mathbf{L}_{ii} & & \\ \vdots & & & & & \ddots & \\ \mathbf{L}_{B1} & \mathbf{L}_{B2} & \mathbf{L}_{B3} & \mathbf{L}_{B4} & \mathbf{L}_{B5} & \dots & \mathbf{L}_{BB} \end{pmatrix} \begin{pmatrix} \dots & \mathbf{X}_1 & \dots \\ \dots & \mathbf{X}_2 & \dots \\ \dots & \mathbf{X}_3 & \dots \\ & \vdots & \\ \dots & \mathbf{X}_i & \dots \\ & \vdots & \\ \dots & \mathbf{X}_B & \dots \end{pmatrix} = \begin{pmatrix} \dots & \mathbf{B}_1 & \dots \\ \dots & \mathbf{B}_2 & \dots \\ \dots & \mathbf{B}_3 & \dots \\ & \vdots & \\ \dots & \mathbf{B}_i & \dots \\ & \vdots & \\ \dots & \mathbf{B}_B & \dots \end{pmatrix}$$

```
for i = 1...N
    X_i ← B_i
    for j = 1...(i-1)
        X_i ← X_i − L_ij X_j
    endfor
    X_i ← (L_ii^-1) X_i
endfor
```

This can be done with an in-place back substitution on $\mathbf{L}_{ii}\mathbf{X}_i=\mathbf{B}_i$

Forward substitution on matrices

$$\begin{pmatrix} \mathbf{L}_{11} & & & & & & \\ \mathbf{L}_{21} & \mathbf{L}_{22} & & & & & \\ \mathbf{L}_{31} & \mathbf{L}_{32} & \mathbf{L}_{33} & & & & \\ \vdots & & & \ddots & & & \\ \mathbf{L}_{i1} & \mathbf{L}_{i2} & \mathbf{L}_{i3} & \dots & \mathbf{L}_{ii} & & \\ \vdots & & & & & \ddots & \\ \mathbf{L}_{B1} & \mathbf{L}_{B2} & \mathbf{L}_{B3} & \mathbf{L}_{B4} & \mathbf{L}_{B5} & \dots & \mathbf{L}_{BB} \end{pmatrix} \begin{pmatrix} \dots & \mathbf{X}_1 & \dots \\ \dots & \mathbf{X}_2 & \dots \\ \dots & \mathbf{X}_3 & \dots \\ & \vdots & \\ \dots & \mathbf{X}_i & \dots \\ & \vdots & \\ \dots & \mathbf{X}_B & \dots \end{pmatrix} = \begin{pmatrix} \dots & \mathbf{B}_1 & \dots \\ \dots & \mathbf{B}_2 & \dots \\ \dots & \mathbf{B}_3 & \dots \\ & \vdots & \\ \dots & \mathbf{B}_i & \dots \\ & \vdots & \\ \dots & \mathbf{B}_B & \dots \end{pmatrix}$$

```
for i = 1...N
    X_i ← B_i
    for j = 1...(i-1)
        X_i ← X_i − L_ij X_j
    endfor
    X_i ← (L_ii^{-1}) X_i
endfor
```

Each product $\mathbf{L}_{ii}\mathbf{X}_i$ (for i=1,2,…) can be computed in parallel (with multithreading)

Forward substitution on matrices

$$\begin{pmatrix} \mathbf{L}_{11} & & & & & & \\ \mathbf{L}_{21} & \mathbf{L}_{22} & & & & & \\ \mathbf{L}_{31} & \mathbf{L}_{32} & \mathbf{L}_{33} & & & & \\ \vdots & & & \ddots & & & \\ \mathbf{L}_{i1} & \mathbf{L}_{i2} & \mathbf{L}_{i3} & \dots & \mathbf{L}_{ii} & & \\ \vdots & & & & & \ddots & \\ \mathbf{L}_{B1} & \mathbf{L}_{B2} & \mathbf{L}_{B3} & \mathbf{L}_{B4} & \mathbf{L}_{B5} & \dots & \mathbf{L}_{BB} \end{pmatrix} \begin{pmatrix} \dots & \mathbf{X}_1 & \dots \\ \dots & \mathbf{X}_2 & \dots \\ \dots & \mathbf{X}_3 & \dots \\ & \vdots & \\ \dots & \mathbf{X}_i & \dots \\ & \vdots & \\ \dots & \mathbf{X}_B & \dots \end{pmatrix} = \begin{pmatrix} \dots & \mathbf{B}_1 & \dots \\ \dots & \mathbf{B}_2 & \dots \\ \dots & \mathbf{B}_3 & \dots \\ & \vdots & \\ \dots & \mathbf{B}_i & \dots \\ & \vdots & \\ \dots & \mathbf{B}_B & \dots \end{pmatrix}$$

Each individual product $\mathbf{L}_{ii}\mathbf{X}_i$ can be vectorized such as the inner (block) mat-mat multiply in GEMM

```
for i = 1...N
   X_i ← B_i
   for j = 1...(i-1)
      X_i ← X_i − L_ij X_j
   endfor
   X_i ← (L_ii^{-1}) X_i
endfor
```

BLAS Level 3 Routines

BLAS Level 3 routines perform matrix-matrix operations. The follo
groups and the data types associated with them.

Matrix-Matrix Products with symmetric or triangular inputs (saves bandwidth by not reading entire matrix)

BLAS Level 3 Routine Groups and Th		
Routine Group	**Data Types**	
cblas_?gemm	s, d, c, z	Computes a matrix-matrix product with general matrices.
cblas_?hemm	c, z	Computes a matrix-matrix product where one input matrix is Hermitian.
cblas_?herk	c, z	Performs a Hermitian rank-k update.
cblas_?her2k	c, z	Performs a Hermitian rank-2k update.
cblas_?symm	s, d, c, z	Computes a matrix-matrix product where one input matrix is symmetric.
cblas_?syrk	s, d, c, z	Performs a symmetric rank-k update.
cblas_?syr2k	s, d, c, z	Performs a symmetric rank-2k update.
cblas_?trmm	s, d, c, z	Computes a matrix-matrix product where one input matrix is triangular.
cblas_?trsm	s, d, c, z	Solves a triangular matrix equation.

BLAS Level 3 Routines

BLAS Level 3 routines perform matrix-matrix operatio
groups and the data types associated with them.

Rank-k updates of symmetric matrices
(rank-2k is a less used variant)

$$S \leftarrow \beta S + \alpha A A^T$$

BLAS Level 3 Routine Gr		
Routine Group	**Data Types**	
cblas_?gemm	s, d, c, z	Computes a matrix-matrix product with general matrices.
cblas_?hemm	c, z	Computes a matrix-matrix product where one input matrix is Hermitian.
cblas_?herk	c, z	Performs a Hermitian rank-k update.
cblas_?her2k	c, z	Performs a Hermitian rank-2k update.
cblas_?symm	s, d, c, z	Computes a matrix-matrix product where one input matrix is symmetric.
cblas_?syrk	s, d, c, z	Performs a symmetric rank-k update.
cblas_?syr2k	s, d, c, z	Performs a symmetric rank-2k update.
cblas_?trmm	s, d, c, z	Computes a matrix-matrix product where one input matrix is triangular.
cblas_?trsm	s, d, c, z	Solves a triangular matrix equation.