CoE 163

Computing Architectures and Algorithms

Optimizing Gaussian Elimination

Recall from previous lesson

- Manufacturers/vendors implement BLAS that are optimized for their machines
- To write optimal code, best to make use of BLAS if possible
  - BLAS level 3 is the most efficient
  - Better to reorder algorithms to use BLAS 3 versus BLAS level 1 or 2

Objective of this lesson

- Show how an algo can be reordered to make use of BLAS 3
- Reordering uses blocking, so that we operate more on submatrices instead of vectors/scalars

Reordering Gaussian Elimination to use BLAS3

Primary reference: *Applied Numerical Linear Algebra* by James W. Demmel

Let’s look at an algorithm for LU Factorization

- Use Gaussian Elimination which can be defined as:
  - “Take each row and subtract multiplies of it from later rows to zero out the entries below the diagonal”

Let's look at an algorithm for LU Factorization

- Use an "in-place" algorithm, where L and U are overwritten on A:
- Example, for the given below

$$A = \begin{bmatrix} 2 & 2 & 3 \\ 5 & 9 & 10 \\ 4 & 1 & 2 \end{bmatrix}$$

$$A = LU$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 5/2 & 1 & 0 \\ 2 & -3/4 & 1 \end{bmatrix} \quad U = \begin{bmatrix} 2 & 2 & 3 \\ 0 & 4 & 5/2 \\ 0 & 0 & -17/8 \end{bmatrix}$$

Let's look at an algorithm for LU Factorization

- Use an "in-place" algorithm, where L and U are overwritten on A:
- Example, algorithm rewrites A with L and U as shown below:

$$A = \begin{bmatrix} 2 & 2 & 3 \\ 5/2 & 4 & 5/2 \\ 2 & -3/4 & -17/8 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 5/2 & 1 & 0 \\ 2 & -3/4 & 1 \end{bmatrix} \quad U = \begin{bmatrix} 2 & 2 & 3 \\ 0 & 4 & 5/2 \\ 0 & 0 & -17/8 \end{bmatrix}$$

Let's look at an algorithm for LU Factorization

```
for i = 1 to n-1
        /* apply permutations so a_ii ≠ 0 */
        for j = i+1 to n
                a_ji = a_ji/a_ii
        end for
        for j = i+1 to n
                for k = i+1 to n
                        a_jk = a_jk - a_ji * a_ik
                end for
        end for
end for
```

Let's look at an algorithm for LU Factorization

```
for i = 1 to n-1
        /* apply permutations so a_ii ≠ 0 */
        for j = i+1 to n
                a_ji = a_ji/a_ii
        end for
        for j = i+1 to n
                for k = i+1 to n
                        a_jk = a_jk - a_ji * a_ik
                end for
        end for
end for
```

"Take each row and subtract multiples of it from later rows to zero out the entries below the diagonal"

Let’s use Matlab notation:

```
for i=1 to n-1
       /*apply permutations*/
       A(i+1:n, i)=A(i+1:n, i)/A(i, i)
       A(i+1:n, i+1:n)=A(i+1:n, i+1:n) – A(i+1:n, i)*A(i, i+1:n)
end for
```

Let’s use Matlab notation:

```
for i=1 to n-1
        /*apply permutations*/
        A(i+1:n, i)=A(i+1:n, i)/A(i, i)
        A(i+1:n, i+1:n)=A(i+1:n, i+1:n) – A(i+1:n, i)*A(i, i+1:n)
end for
```

Level 1 BLAS
(multiply vector
by a scalar)

Let’s use Matlab notation:

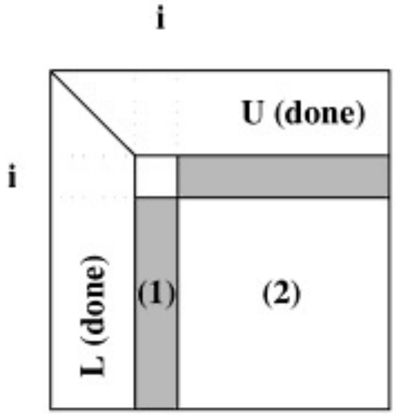
```
for i=1 to n-1
       /*apply permutations*/
       A(i+1:n, i)=A(i+1:n, i)/A(i, i)
       A(i+1:n, i+1:n)=A(i+1:n, i+1:n) – A(i+1:n, i)*A(i, i+1:n)
end for
```

Level 2 BLAS
(rank-1 update of the submatrix
$A(i+1{:}n, i+1{:}n)$)

We shall reorder the algo to use Level 3 BLAS

- Modify algorithm slightly to be used within our Level 3 version
- Matrix is now m-by-n

```
for i=1 to min(m-1,n)
          A(i+1:n, i)=A(i+1:n, i)/A(i, i)
          if i < n
              A(i+1:n,i+1:n)=A(i+1:n,i+1:n) –
              A(i+1:n,i)*A(i,i+1:n)
end for
```



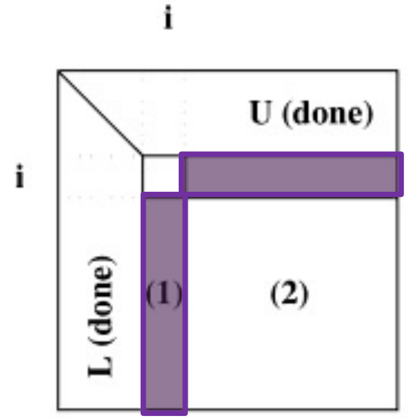
Step i of Level 2 BLAS Implementation of LU

From *Applied Numerical Algebra,* by Demmel

We shall reorder the algo to use Level 3 BLAS

- Modify algorithm slightly to be used within our Level 3 version
- Matrix is now m-by-n

```
for i=1 to min(m-1,n)
        A(i+1:n, i)=A(i+1:n, i)/A(i, i)
        if i < n
           A(i+1:n,i+1:n)=A(i+1:n,i+1:n) –
           A(i+1:n,i)*A(i,i+1:n)
end for
```



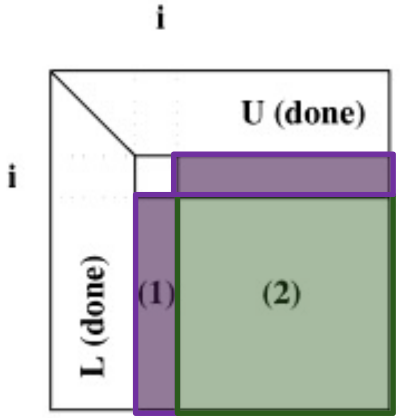
Step i of Level 2 BLAS Implementation of LU

From *Applied Numerical Algebra,* by Demmel

We shall reorder the algo to use Level 3 BLAS

- Modify algorithm slightly to be used within our Level 3 version
- Matrix is now m-by-n

```
for i=1 to min(m-1,n)
        A(i+1:n, i)=A(i+1:n, i)/A(i, i)
        if i < n
            A(i+1:n,i+1:n)=A(i+1:n,i+1:n) –
            A(i+1:n,i)*A(i,i+1:n)
end for
```



Step i of Level 2 BLAS Implementation of LU

From *Applied Numerical Algebra,* by Demmel

Our BLAS 3 version will use blocking

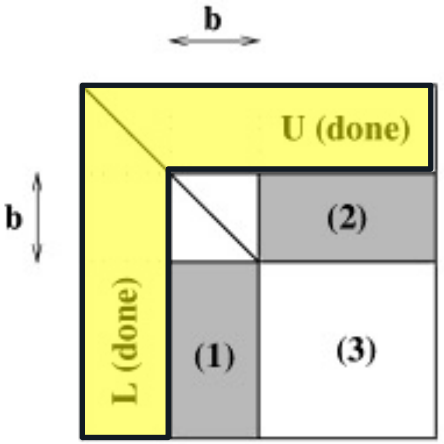
- We will “delay” the update of submatrix 2 by b steps, where b is our block size
- Apply b rank-1 updates all at once in one matrix-matrix multiplication

First, let's see what happens mathematically

- Suppose we are done computing first $i-1$ columns of L and rows of U

$$
\begin{array}{cc}
 & \begin{array}{ccc} i-1 & b & n-b-i+1 \end{array} \\
A = & \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \begin{array}{l} i-1 \\ b \\ n-b-i+1 \end{array}
\end{array}
$$

$$
A = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{11} & I & 0 \\ L_{11} & 0 & I \end{bmatrix} \cdot \begin{bmatrix} U_{11} & U_{21} & U_{31} \\ 0 & \tilde{A}_{22} & \tilde{A}_{23} \\ 0 & \tilde{A}_{32} & \tilde{A}_{33} \end{bmatrix}
$$



Step i of Level 3 BLAS Implementation of LU

From *Applied Numerical Algebra,* by Demmel

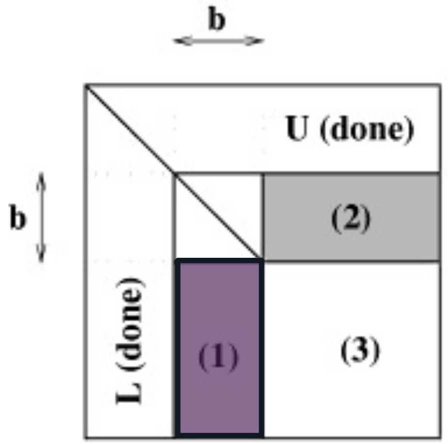
First, let's see what happens mathematically

- Apply our BLAS2 algorithm to submatrix $\begin{bmatrix} \tilde{A}_{22} \\ \tilde{A}_{32} \end{bmatrix}$ to get:

$$\begin{bmatrix} \tilde{A}_{22} \\ \tilde{A}_{32} \end{bmatrix} = \begin{bmatrix} L_{22} \\ L_{32} \end{bmatrix} \cdot U_{22} = \begin{bmatrix} L_{22}U_{22} \\ L_{32}U_{22} \end{bmatrix}$$

We can then write:

$$\begin{bmatrix} \tilde{A}_{22} & \tilde{A}_{23} \\ \tilde{A}_{32} & \tilde{A}_{33} \end{bmatrix} = \begin{bmatrix} L_{22}U_{22} & \tilde{A}_{23} \\ L_{32}U_{22} & \tilde{A}_{33} \end{bmatrix}$$



Step i of Level 3 BLAS Implementation of LU

From *Applied Numerical Algebra,* by Demmel

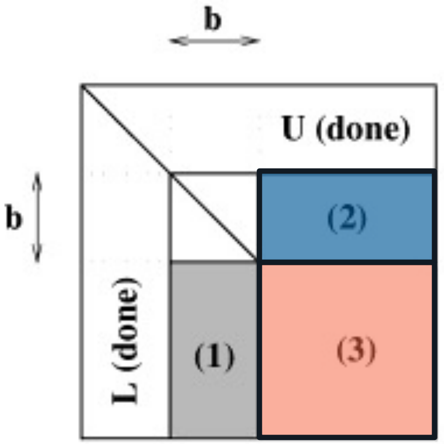
First, let's see what happens mathematically

$$\begin{bmatrix} \tilde{A}_{22} & \tilde{A}_{23} \\ \tilde{A}_{32} & \tilde{A}_{33} \end{bmatrix} = \begin{bmatrix} L_{22}U_{22} & \tilde{A}_{23} \\ L_{32}U_{22} & \tilde{A}_{33} \end{bmatrix}$$

$$= \begin{bmatrix} L_{22} & 0 \\ L_{32} & I \end{bmatrix} \cdot \begin{bmatrix} U_{22} & L_{22}^{-1}\tilde{A}_{23} \\ 0 & \tilde{A}_{33} - L_{32} \cdot \left(L_{22}^{-1}\tilde{A}_{23}\right) \end{bmatrix}$$

$$= \begin{bmatrix} L_{22} & 0 \\ L_{32} & I \end{bmatrix} \cdot \begin{bmatrix} U_{22} & U_{23} \\ 0 & \tilde{A}_{33} - L_{32} \cdot U_{23} \end{bmatrix}$$

$$= \begin{bmatrix} L_{22} & 0 \\ L_{32} & I \end{bmatrix} \cdot \begin{bmatrix} U_{22} & U_{23} \\ 0 & \widetilde{\widetilde{A}}_{33} \end{bmatrix}$$



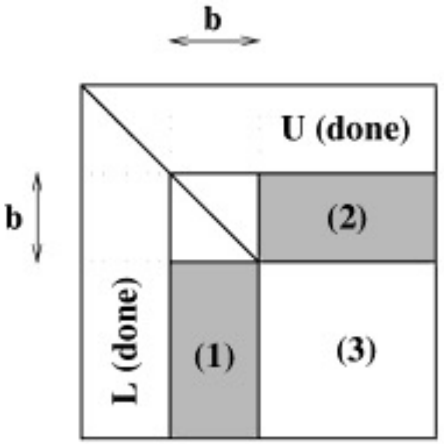
Step i of Level 3 BLAS Implementation of LU

From *Applied Numerical Algebra,* by Demmel

We get updated factorization with b more columns of L and U completed

$$= \begin{bmatrix} L_{22} & 0 \\ L_{32} & I \end{bmatrix} \cdot \begin{bmatrix} U_{22} & U_{23} \\ 0 & \widetilde{\widetilde{A}}_{33} \end{bmatrix}$$

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{23} & I \end{bmatrix} \cdot \begin{bmatrix} U_{11} & U_{21} & U_{31} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & \widetilde{\widetilde{A}}_{33} \end{bmatrix}$$



Step i of Level 3 BLAS Implementation of LU

From *Applied Numerical Algebra*, by Demmel

BLAS L3 Algorithm for LU Factorization

1) Use BLAS L2 Algorithm to factorize $\begin{bmatrix} \tilde{A}_{22} \\ \tilde{A}_{32} \end{bmatrix} = \begin{bmatrix} L_{22} \\ L_{32} \end{bmatrix} \cdot U_{22}$
2) Form $U_{23} = L_{22}^{-1} \tilde{A}_{23}$ (this is a BLAS L3 operation)
3) Form $\tilde{\tilde{A}}_{33} = \tilde{A}_{33} - L_{32} \cdot \left(L_{22}^{-1} \tilde{A}_{23} \right)$ (MMM operation, BLAS L3)

BLAS L3 Algorithm for LU Factorization

for $i = 1$ to $n - 1$ step b

Factorize $A(i{:}n, i{:}i+b) = \begin{bmatrix} L_{22} \\ L_{32} \end{bmatrix} U_{22}$

/* use BLAS L2 algo*/

$A(i{:}i+b-1, i+b{:}n) = L_{22}^{-1} \cdot A(i{:}i+b-1, i+b{:}n)$

/* Form U_{23} */

$A(i+b{:}n, i+b{:}n) = A(i+b{:}n, i+b{:}n) - A(i+b{:}n, i{:}i+b-1) \cdot A(i{:}i+b-1, i+b{:}n)$

/* Form $\widetilde{\widetilde{A}}_{33}$ */

end for

Additional remarks

- Need to choose block size to maximize speed
  - Large blocks to multiply larger matrices
  - Number of floating point operations by Level 2 and Level 1 BLAS in step 1 is about $\frac{n^2 b}{2}$ for small b
    - Grows as b grows, we don't want to pick b too large
- Commonly used values are $b = 32$ or $b = 64$
- Detailed implementations:
  - BLAS 2 Algo: `sgetf2` on LAPACK
  - BLAS 3 Algo: `sgetrf` on LAPACK
  - Search here: http://www.netlib.org/lapack/explore-html/modules.html