Team 6: LU Factorization

Optimizations targeting towards multicore processors

Ryan McKenna, Matthew Paul, James Kerrigan
Niko Gerassimakis, Neil Duffy

May 20, 2015

Linear Algebra

- The quintessential problem in linear algebra is solving a linear system of equations

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- We want to find values of x_1, x_2, and x_3 such that
  1. $a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$
  2. $a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$
  3. $a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$

Impact

- Linear algebra comes up in a lot of professions:
  - Physics
  - Partial differential equations
  - Graph theory
  - Statistics / Curve Fitting
  - Sports Ranking

Solving Linear Systems

- If A is an $n \times n$ matrix, solving a system of the form $Ax = b$ takes $O(n^3)$ time.
- If A is a triangular matrix, then solving the system takes $O(n^2)$ time

LU Factorization Background

- LU Factorization works by decomposing a square matrix A into a lower triangular matrix, L, and an upper triangular matrix, U:

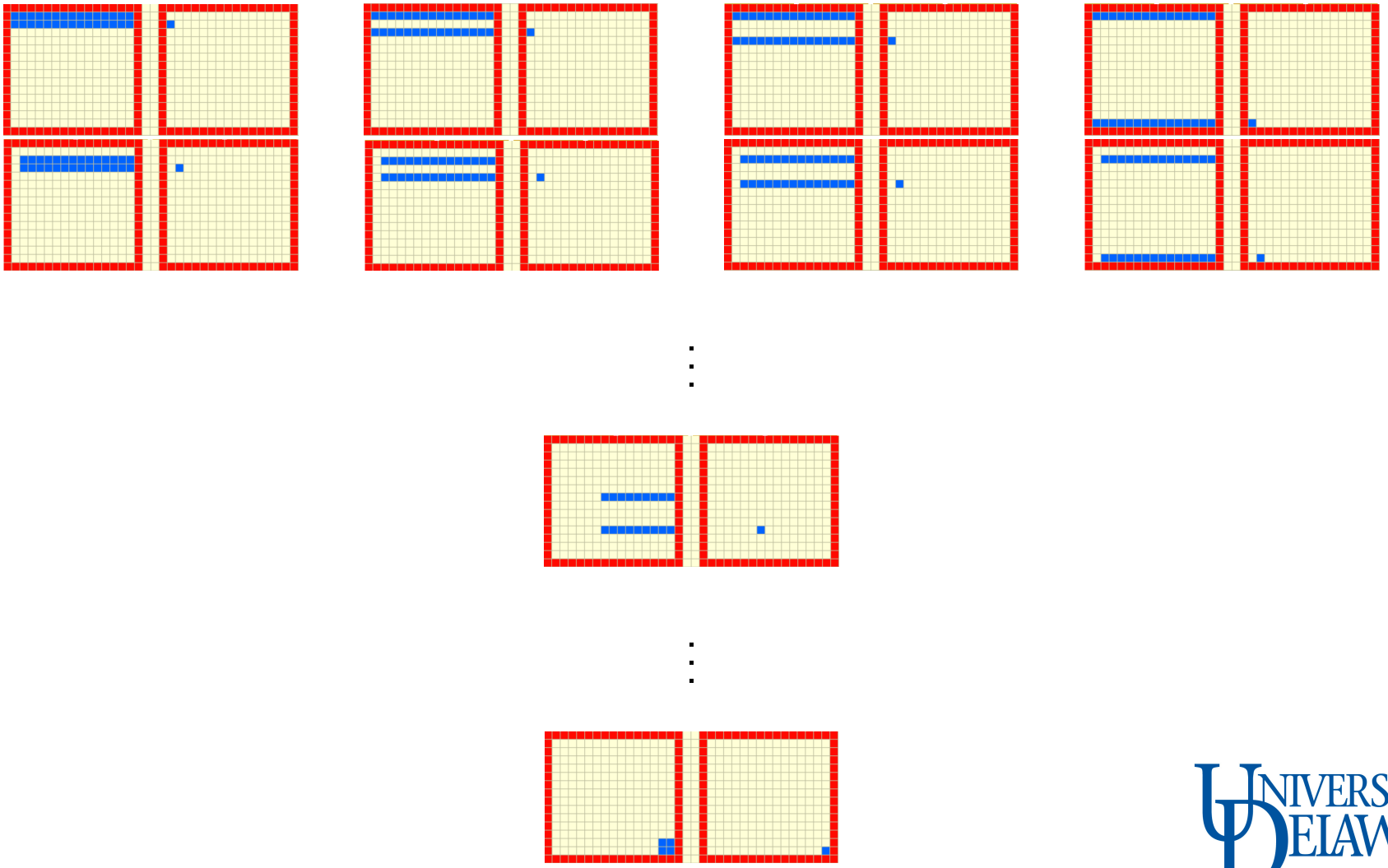
$$A = LU$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

- With L and U, we can solve $Ax = LUx = b$ in $O(n^2)$.

Algorithm Description / Access Pattern

LU factorization is an $O(n^3)$ algorithm:



Implementation

```
void lu(double **A, double **L, double **U, int n) {
    zero (L, n);
    copy (U, A, n);
    init (L, n);
    for(int j=0; j < n; j++) {
        for(int i=j+1; i < n; i++) {
            double m = U[i][j] / U[j][j];
            L[i][j] = m;
            for(int k=j; k < n; k++)
                U[i][k] -= m*U[j][k];
        }
    }
}
```

Approach

1. Generate random matrices up to 6400 x 6400.
2. Run and time 4 trials of the factorization algorithm for each matrix size.
3. Repeat for every optimization configuration.

Optimizations

- -O1, -O2, -O3
- loop unrolling
- vectorization
- native (architecture specific) optimizations
- openMP