ECS130 Scientific Computation **Handout A** **January 11, 2017**

1. Review of vector and matrix algebra

$$A \pm B, \quad A \cdot B, \quad A^T (= A'), \quad A \cdot x, \quad x^T \cdot y, \quad x \cdot y^T$$

   where x and y are vectors, and A and B are matrices.

2. BLAS = Basic Linear Algebra Subprograms, a de facto application programming interface standard to perform basic linear algebra operations such as vector and matrix multiplication.

   - Level 1: vector operations, such as $y \leftarrow \alpha x + y$, ...
   - Level 2: matrix-vector operations, such $y \leftarrow \alpha Ax + \beta y$, ...
   - Level 3: matrix-matrix operations, such $C \leftarrow \alpha AB + \beta C$, ...

   Highly optimized implementations of the BLAS have been developed by hardware vendors such as by MKL of Intel, ESSL of IBM, cuBLAS of NVIDIA.

3. Linear system of equations

$$Ax = b$$

   where A is a given square matrix of order n, b is a given column vector of n components, and x is an unknown column vector of n components.

   The following statements are equivalent:

   - for any vector b, the linear system has a solution x.
   - If a solution exists, it is unique.
   - If $Ax = 0$, then $x = 0$.
   - The columns (rows) of A are linearly independent.
   - There is a matrix A^{-1} such that $A^{-1} \cdot A = A \cdot A^{-1} = I$.
     (A is called nonsingular or invertible, A^{-1} is called the inverse of A)
   - $\det(A) \neq 0$.

4. Lower triangular linear system of equations:

$$Lx = b$$

   where $L = (\ell_{ij})$ is an $n \times n$ lower triangular, i.e., $\ell_{ij} = 0$ if $i < j$, and nonsingular (invertible), i.e., $\ell_{ii} \neq 0$ for $i = 1, \ldots, n$.

5. Forward substitution algorithm – *row-oriented*

   - Algorithm:
     for $i = 1, 2, \ldots, n$,
     $x_i = (b_i - l_{i,1}x_1 - l_{i,2}x_2 - \cdots - l_{i,i-1}x_{i-1}) / l_{i,i}$
   - Flops: n^2
   - M-scripts in componentwise form:

```
for i = 1:n
   x(i) = b(i);
   for j = 1:i-1
      x(i) = x(i) - L(i,j)*x(j);
   end
   x(i) = x(i)/L(i,i);
end
```

- M-scripts in vectorized form:

```
x(1) = b(1)/L(1,1);
for i = 2:n
   x(i) = (b(i) - L(i,1:i-1)*x(1:i-1))/L(i,i);
end
```

6. Forward substitution algorithm – *column-oriented*

- Algorithm: By the partition

$$\begin{array}{cc} & \begin{array}{cc} 1 & n-1 \end{array} \\ \begin{array}{c} 1 \\ n-1 \end{array} & \left[\begin{array}{cc} \ell_{11} & \\ L_{21} & L_{22} \end{array}\right] \end{array} \left[\begin{array}{c} x_1 \\ x_{(2:n)} \end{array}\right] = \left[\begin{array}{c} b_1 \\ b_{(2:n)} \end{array}\right]$$

we have

$$\begin{aligned} \ell_{11}x_1 &= b_1 \\ L_{21}x_1 + L_{22}x_{(2:n)} &= b_{(2:n)} \end{aligned}$$

Therefore, $x_1 = b_1/\ell_{11}$, and then after updating $\widehat{b}_{(2:n)} = b_{(2:n)} - L_{21}x_1$, we solve solve the same-kind of lower triangular system with dimension $n-1$:

$$L_{22}x_{(2:n)} = \widehat{b}_{(2:n)}.$$

The procedure is repeated until finding all entries of x.

- M-scripts in vectorized form:

```
x = zeros(n,1);
for j = 1:n-1
   x(j) = b(j)/L(j,j);
   b(j+1:n) = b(j+1:n) - L(j+1:n,j)*x(j);
end
x(n) = b(n)/L(n,n);
```

7. Row-oriented vs. column-oriented forward substitution

Language considerations:

- C stores double subscripted arrays by rows,
- Fortran stores by columns.

Memory considerations

- virtual memory
- page-hit/miss

- locality of reference.

8. Triangular systems with multiple right-hand sides:

$$LX = B,$$

where B is $n \times m$.

Algorithm 1: solve $Lx_k = b_k$ for $k = 1 : m$

Algorithm 2: vectorized on m

```
X = zeros(n,m);
for j = 1:n-1
   X(j,:) = B(j,:)/L(j,j);
   B(j+1:n,:) = B(j+1:n,:) - L(j+1:n,:)*X(j,:);
end
X(n,:) = B(n,:)/L(n,n);
```

9. **Exercise:** Upper triangular linear system of equations

$$Ux = b$$

can be treated similarly using back substitution, where U is an $n \times n$ upper triangular (i.e., $u_{ij} = 0$ for $i > j$) and nonsingular (i.e., $u_{ii} = 0$ for $i = 1, 2, \ldots, n$)