Most numerical programs do similar operations
90% time is at 10% of the code
If these 10% of the code is optimized, programs will be fast
Frequently used subroutines should be available
For numerical computations, common operations can be easily identified
Example:
daxpy(n, alpha, p, inc, w, inc);
daxpy(n, malpha, q, inc, r, inc);

rtr = ddot(n, r, inc, r, inc);
 rnorm = sqrt(rtr);
tnorm = sqrt(ddot(n, t, inc, t, inc));

- ddot: inner product
- daxpy: $ax + y$, $x, y$ are vectors and $a$ is a scalar
- The first BLAS paper:

ACM Trans. Math. Soft.: a major journal on numerical software

- Become de facto standard for the elementary vector operations
  (http://www.netlib.org/blas/)

Faster code than what you write
BLAS: Basic Linear Algebra Subroutines IV

- Netlib (http://www.netlib.org) is a site which contains freely available software, documents, and databases of interest to the numerical, scientific computing, and other communities (starting even before 1980).

- Interfaces from e-mail, ftp, gopher, x-based tool, to WWW

- Level 1 BLAS includes:
  - dot product
constant times a vector plus a vector rotation (don’t need to know what this is now)
copy vector $x$ to vector $y$
swap vector $x$ and vector $y$
length of a vector $(\sqrt{x_1^2 + \cdots + x_n^2})$
sum of absolute values $(|x_1| + \cdots + |x_n|)$
constant times a vector
index of element having maximum absolute value

**Programming convention**

$$dw = ddot(n, dx, incx, dy, incy)$$
BLAS: Basic Linear Algebra Subroutines

VI

\[ w = \sum_{i=1}^{n} dx_{1+(i-1)\text{incx}} dy_{1+(i-1)\text{incy}} \]

Example:

\[ dw = ddot(n, dx, 2, dy, 2) \]

\[ w = dx_1 dy_1 + dx_3 dy_3 + \cdots \]

- sdot is single precision, ddot is for double precision
- \( y = ax + y \)

call daxpy(n, da, dx, incx, dy, incy)
For researchers involving the design of BLAS, a difficulty is to decide which subroutines to be included.

Traditionally these subroutines are written in Fortran.

But we may call them from a different language.

C calls Fortran:

```c
rtr = ddot_(&n, r, &inc, r, &inc);
```

ddot_: calling Fortran subroutines (machine dependent)
BLAS: Basic Linear Algebra Subroutines

- &n: call by reference for Fortran subroutines
- Arrays: both call by reference
  - C: start with 0, Fortran: with 1
  - This should not cause problems here
- There is CBLAS interface
  - C calls C:
    \[
    rtr = \text{ddot}(n, r, \text{inc}, r, \text{inc});
    \]
  - So no need to handle the issues between C and Fortran
Now BLAS libraries are available on most computers. For example, on a Linux computer:

```bash
$ ls /usr/lib| grep blas
libblas
libblas.a
libblas.so
libblas.so.3
libblas.so.3gf
libcblas.so.3
libcblas.so.3gf
libf77blas.so.3
libf77blas.so.3gf
```
.a: static library, .so: dynamic library, .3 versus .3gf: g77 versus gfortran
The original BLAS contains only $O(n)$ operations. That is, vector operations.

Matrix-vector product takes more time.

Level 2 BLAS involves $O(n^2)$ operations, where $n$ is the size of matrices.

Level 2 BLAS II

- Matrix-vector product

\[ Ax : (Ax)_i = \sum_{j=1}^{n} A_{ij}x_j, \ i = 1, \ldots, m \]

This can be done by \( m \) inner products. However, it is inefficient if we use level 1 BLAS to implement this.

- Scope of level 2 BLAS:

- Matrix-vector product

\[ y = \alpha Ax + \beta y, \ y = \alpha A^T x + \beta y, \ y = \alpha \bar{A}^T x + \beta y \]
\( \alpha, \beta \) are scalars, \( x, y \) are vectors, \( A \) is a matrix, and \( \bar{A}^T \) is the conjugate transpose (or Hermitian transpose) of \( A \). For real matrices,

\[
\bar{A}^T = A^T
\]

- Lower- or upper-triangular matrix-vector products

\[
x = Tx, \ x = T^T x, \ x = \bar{T}^T x
\]

\( x \) is a vector and \( T \) is a lower or upper triangular matrix
Rank-one and rank-two updates

\[ A = \alpha xy^T + A, \quad H = \alpha x\bar{y}^T + \bar{\alpha}y\bar{x}^T + H \]

\( H \) is a Hermitian matrix \((H = \bar{H}^T, \text{ symmetric for real numbers})\)
rank: \# of independent rows (columns) of a matrix
Note that for a matrix, column rank = row rank
Level 2 BLAS V

- $xy^T$ is a rank one matrix. For example,

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 6 & 8 \end{bmatrix}$$

This calculation needs $O(n^2)$ operations

- $xy^T + yx^T$ is a rank-two matrix

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 6 & 8 \end{bmatrix}, \begin{bmatrix} 3 \\ 4 \end{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 4 & 8 \end{bmatrix}$$

- Solution of triangular equations
\[ x = T^{-1}y \]

\[
\begin{bmatrix}
T_{11} & T_{21} & \cdots & T_{n1} \\
T_{21} & T_{22} & \cdots & T_{n2} \\
\vdots & \vdots & \ddots & \vdots \\
T_{n1} & T_{n2} & \cdots & T_{nn}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n
\end{bmatrix}
= \begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_n
\end{bmatrix}
\]

The solution

\[
x_1 = \frac{y_1}{T_{11}}
\]

\[
x_2 = \frac{y_2 - T_{21}x_1}{T_{22}}
\]

\[
x_3 = \frac{y_3 - T_{31}x_1 - T_{32}x_2}{T_{33}}
\]
Number of multiplications/divisions:

\[ 1 + 2 + \cdots + n = \frac{n(n + 1)}{2} \]
Level 3 BLAS I

- Tasks that involve $O(n^3)$ operations
- Reference:
- Matrix-matrix products

$$C = \alpha AB + \beta C, \quad C = \alpha A^T B + \beta C,$$
$$C = \alpha AB^T + \beta C, \quad C = \alpha A^T B^T + \beta C$$
Level 3 BLAS II

- Rank-k and rank-2k updates
- Multiplying a matrix by a triangular matrix
  \[ B = \alpha TB \]
- Solving triangular systems of equations with multiple right-hand side:
  \[ B = \alpha T^{-1}B \]
- Naming conversions follow from those of level 2
- BLAS does not include subroutines for solving general linear systems or eigenvalues
  They are in the package LAPACK described later