BLAS-2 in LAPACK

How can BLAS 2.5 help?
Archetypal LAPACK Algorithm

A matrix is partitioned into $k$ blocks of columns, each block small enough to fit in cache memory.

For $j = 1 : k$,
    Perform BLAS 1 and 2 on block $j$;
For $m = j + 1 : k$
    perform BLAS 3;
End;
End;

The "ideal" cost is data transfers from main memory. Each step of the inner loop corresponds to accessing a block of data of $O(n^2/k)$ data. Since there are $O(k^2)$ steps overall, there are $O(n^2k)$ data transferred from main memory.
Algorithms Archetypically Implemented

- LU decomposition
- Cholesky decomposition
- Householder QR decomposition
- reduction to similar tridiagonal form (symmetric eigenvalue)
- reduction to similar Hessenberg form (unsymmetric eigenvalue)
- reduction to bidiagonal form (as part of SVD).
Algorithms Implemented with $O(n^2k)$ Data Transfers

- LU decomposition
- Cholesky decomposition
- Householder QR decomposition
Algorithms Implemented with \( O(n^3) \) Data Transfers

The archetype is used here but fails to reduce data transfer below \( O(n^3) \)

- reduction to similar tridiagonal form (symmetric eigenvalue)

- reduction to similar Hessenberg form (unsymmetric eigenvalue)

- reduction to bidiagonal form (as part of SVD).
What Goes Wrong?

A matrix is partitioned into $k$ blocks of columns, each block small enough to fit in cache memory.

For $j = 1 : k$,

Perform BLAS 1 and 2 on block $j$;
FOR EACH COLUMN,
A BLAS-2 GEMV
TRANSFERS THE WHOLE MATRIX.

For $m = j + 1 : k$
perform BLAS 3;
End;
End;

The predominant data transfers are for the BLAS-2 operations performed for each column.
For each of \( n \) columns \( O(n^2) \) data is transferred for a total of \( O(n^3) \) data transfers. In the "ideal" case for which computations are free and the only cost is data transfer, the computation is equivalent to unblocked LU decomposition.
SVD Bidiagonalization

For $j = 1 : k$,

Perform BLAS 1 and 2 on block $j$;

FOR EACH COLUMN,

Two Different calls to BLAS-2 GEMV TRANSFER THE WHOLE MATRIX.

For $m = j + 1 : k$

perform BLAS 3;

End;

End;

In the "ideal" case, takes twice as long as un-blocked LU decomposition.
BLAS 2.5 Improves

Bidiagonalization

_GEMVT performs

\[ y \leftarrow Ax; \quad z^t \leftarrow w^t A \]

Both matrix vector multiplies are performed in one access to the matrix.

- \( A \) is transferred from main memory only once per column.

- "Ideal" case as fast as unblocked LU.

- Vote?
BLAS 2.5 allows improved performance in parallel tridiagonalization

_SYMVMT performs

\[ y \leftarrow Ax, z^t \leftarrow w^t A \]

Both matrix vector multiplies are performed in one access to the matrix.

- \( A \) is transferred from main memory only once.

- Nearly doubles performance on half of the flops in reduction to tridiagonal form

- Will be incorporated in ScaLAPACK. Vote? programs.
Modified Gram-Schmidt

Given orthonormal vectors $v_1, v_2, \ldots v_m$ and vector $w$, produce an orthonormal set $v_1, v_2, \ldots v_{m+1}$ such that

$$\text{span}(v_1, \ldots v_{m+1}) = \text{span}(v_1, \ldots v_m, w)$$

$v_{m+1} = w$;
For $j = 1 : m$,
$$\alpha = v_j^T v_{m+1}; \text{BLAS - 1\_dot}$$
$$v_{m+1} = v_{m+1} - \alpha v_j; \text{BLAS - 1\_axpy}$$
End;
$$v_{m+1} = v_{m+1}/\|v_{m+1}\|_2$$
BLAS 1.5 Modified Gram-Schmidt

Frequently modified Gram-Schmidt runs with very large matrices so the vectors are long enough that BLAS 1 may not be as bad as it might seem. Provided that two vectors can be kept in fast memory, combining _dot and _axpy in one operation may be helpful.

\[ w \leftarrow w - (w^T v) v, \alpha \leftarrow w^T v \]

Actually, one vector in fast memory may be enough to give a good speedup.

Vote?
BLAS 3.5 allows a different archetype

When BLAS-2 operations involve the whole matrix, the predominant data transfer occurs in the BLAS-2 block. In a BLAS 3.5 based algorithm, the only data transfer is that associated with the BLAS-2 block. The BLAS 3.5 based algorithm is as easy to implement as the BLAS-2 block of the LAPACK archetype. The overall algorithm is more straightforward.
BLAS 3.5 Bidiagonalization

Call1 to _gemvr
For \( j = 2 : n - 2 \),
    Call2 to _gemvr
End;
Call 3 to _gemvr

In LAPACK _gesvd.f bidiagonalization there are 10 cases, each involving a comparison between number of columns \( m \) and rows \( n \). These do not appear to be necessary in the _gemvr implementation.
The BLAS 3.5 _gemvr operator

A single call to _gemvr performs the following. Rank NR1 update, when \( NR1 > 0 \),

\[
A \leftarrow A + W_1^T \ast Z_1
\]

Yin has KR columns. when \( KR > 0 \),

\[
Y_{out} \leftarrow A^T \ast Y_{in} + Y_{out}
\]

. Rank NR2 update, when \( NR2 > 0 \),

\[
A \leftarrow A + W_2^T \ast Z_2
\]

Xin has KC columns, when \( KC > 0 \),

\[
X_{out} \leftarrow A \ast X_{in} + X_{out}
\]

The default value for any of NR1, KR, NR2, KC is zero.
An automated program for optimizing _gemvr

Competing projects from UT and Berkeley optimize _gemv. Applying the same techniques to automation of _gemvr can allow automated optimization of

- Reduction to similar Hessenberg form by Householder transformations.

- Reduction to bidiagonal form by Householder transformations.

- Reduction to similar Hessenberg form by elementary similarity transformations.
BLAS 3.5 Allow Easier Development of New Algorithms

Allowing easy implementation of new algorithms and alternate implementations of LAPACK algorithms helps the field of numerical linear algebra as a field of continuing active research. As opposed to a study of already given classical algorithms. Living language vs. Latin.

Easy efficient coding allows competitive versions of code to develop. Economic theory and experience is that competition makes for more efficient and understandable entities (in this case computer programs).
Example

Orthogonal reduction to similar block Hessenberg form Movement of data to reduce to upper 2 by 2 block Hessenberg form is only half that of reducing to upper Hessenberg form. The cost of reducing the block Hessenberg to Hessenberg is negligible in comparison. Coding can be by the LAPACK archetype or by _gemvr. _gemvr coding seems easier.

Vote on _gemvr?