LAPACK Working Note 82
Call Conversion Interface (CCI) for LAPACK/ESSL[*][†]

Jack Dongarra and Michael Kolatis
Department of Computer Science
University of Tennessee
Knoxville, Tennessee 37996-1301

October 3, 1994

1 Overview

This document reviews the initial version of the Call Conversion Interface (CCI) from LAPACK to the Engineering and Scientific Subroutine Library (ESSL). The CCI substitutes a call to an ESSL subroutine in place of an LAPACK routine whenever ESSL subroutines provide either functional or near-functional equivalence. In either case, the ESSL subroutine will be used only if its calling sequence can be made to fit that of LAPACK in structure. Finally, the CCI consists of several parts: a list of possible subroutine matchings, interfacing requirements, the successes and failures of those matchings, timings (LAPACK vs. the CCI), and availability.

1.1 Minimum Software Requirements

This enablement requires the following software:

- AIX 3.2.5
- XLF 3.1
- ESSL 2.2.1.1
- LAPACK 2.0

2 Subroutine Matchings

[*]ESSL is a trademark of IBM.

[†]This work supported in part by the Office of Scientific Computing, U.S. Department of Energy, under Contract DE-AC05-84OR21400 and in part by IBM.

Subroutine Matchings					
Linear Algebra			Eigensystem		
types	LAPACK	ESSL	*types*	LAPACK	ESSL
S,D,C,Z	_GETRF	_GEF	S,D,C,Z	_GEEV	_GEEV
S,D,C,Z	_GETRS	_GESM	S,D,C,Z	_GEEVX	_GEEV
S,D	_GETRI	_GEICD	S,D	_SPEV	_SPEV
S,D,C,Z	_POTRF	_POF	S,D	_SPEVX	_SPSV
S,D,C,Z	_POTRS	_POSM	C,Z	_HPEV	_HPEV
S,D	_POTRI	_POICD	C,Z	_HPEVX	_HPSV
S,D	_PPTRF	_PPF	S,D	_GEGV	_GEGV
S,D	_PPTRI	_PPICD	S,D	_SYGV	_SYGV
S,D	_PBTRF	_PBCHF	Others		
S,D	_GESVD	_GESVF	S,D	_TRTRI	_TRI
S,D	_GELSS	_GESVS	S,D	_TPTRI	_TPI
S,D	_GELS	_GELLS	S,D,C,Z	_LASWP	_LASWP

Table 1: This is a list of all the possible subroutine pairings (LAPACK $\Leftarrow$ ESSL) for the initial version of the CCI.

Table 1 holds the list of LAPACK/ESSL subroutine pairings. The subroutine types (i.e. *types*) in Table 1 represent $\{\mathbf{S}\}$ single precision, $\{\mathbf{D}\}$ double precision, $\{\mathbf{C}\}$ complex and $\{\mathbf{Z}\}$ double complex.

3 Interfacing Requirements

The interface subroutines use the following structure. (See also Section 6.1).

1. The full header including the following is used verbatim from the replaced subroutine:
   - Subroutine statement and arguments.
   - LAPACK interface code, including modifications indicating that the particular subroutine is for the ESSL enablement.
   - Argument declarations, verbatim from LAPACK routine.
   - Purpose statement, verbatim from LAPACK.
   - ESSL special modifications clause, describing any special modifications necessary for the ESSL enablement.
   - Argument list, verbatim from LAPACK. The range of arguments allowed should be functionally equivalent to the range for LAPACK.
2. Local arguments (by and large, the same as for LAPACK-with some omissions).
3. **EINFO**, **ERRSAV**, **ERRSET** and **ERRSTR** are declared and externalized, as necessary.

4. The usual LAPACK argument checking is used.

5. All software and documentation conforms to LAPACK “standards”.

6. Calls to **EINFO**, **ERRSAV**, **ERRSET** and **ERRSTR** do not allow computational errors to cause loss of program control. All calls to **EINFO** are of the three argument form, with integers IERR1 and IERR2 acting as the error information arguments. All calls to **ERRSAV** and **ERRSTR** utilize a CHARACTER*8 variable in the form SV*xxxx* where *xxxx* is the four-digit ESSL error number. Finally, these calls are never made when they are not needed.

7. Call to the ESSL computational subroutine. Some calls include an alternate return for error handling.

8. Any postcomputation needed. Then, control skips around error handling code to common **ERRSTR** call to restore ESSL error handling.

9. Computation error handling. ESSL error codes are mapped onto LAPACK error codes, if possible. If not, the necessary LAPACK routines are called to determine the proper LAPACK error codes.

10. After ESSL **ERRSTR**, control is returned to calling program.

4 Working Description/Status

This section describes the level of ESSL enablement that was given to each LAPACK routine. If a routine is fully enabled, then it has passed the entire LAPACK test suite. If a routine could not be enabled, the limiting factors are described. If workspace limitations are present, the LAPACK workspace size requirement (usually, LWORK) is given with the ESSL workspace size requirement (always NAUX).

4.1 Linear Algebra: 26/32 routines

4.1.1 _GETRF $\Leftarrow$ _GEF: 4/4 routines

_GETRF calls _GEF when the input matrix is square (M = N). In all other cases, _GETRF is the LAPACK routine _GETRF.

4.1.2 _GETRS $\Leftarrow$ _GESM: 4/4 routines

This pair is fully enabled.

4.1.3 _GETRI $\Leftarrow$ _GEICD: 2/2 routines

This pair is fully enabled.

4.1.4 _POTRF $\Leftarrow$ _POF: 4/4 routines

This pair is fully enabled.

Architecture	Cache size	Line size	Cutover Value = Half Band Width
POWER1	32K	64B	72
POWER1	64K	64B	100
POWER1	64K	128B	100
POWER2	128K	128B	140
POWER2	256K	256B	190

Table 2: IBM-supplied tuning information for the crossover values in ESSL routine DPBCHF: the crossover value reflects when the subroutine uses either a narrow band or wide band algorithm for the factorization.

4.1.5 _POTRS $\Leftarrow$ _POSM: 4/4 routines

This pair is fully enabled.

4.1.6 _POTRI $\Leftarrow$ _POICD: 2/2 routines

This pair is fully enabled.

4.1.7 _PPTRF $\Leftarrow$ _PPF: 2/2 routines

_PPTRF calls _PPF when the input matrix A is stored in lower-packed format only. _PPF does not handle upper-packed format; in this case, _PPTRF is the LAPACK routine _PPTRF.

4.1.8 _PPTRI $\Leftarrow$ _PPICD: 2/2 routines

_PPTRI calls _PPICD when the input matrix A is stored in lower-packed format only. _PPICD does not handle upper-packed format; in this case, _PPTRI is the LAPACK routine _PPTRI.

4.1.9 _PBTRF $\Leftarrow$ _PBCHF: 2/2 routines

_PBTRF calls _PBCHF when the input matrix A is stored in lower-band-packed format only. _PBCHF does not handle upper-band-packed format; in this case, _PBTRF is the LAPACK routine _PBTRF.

For performance reasons, divides are done in a way that reduces the effective exponent range for which DPBCHF works properly, only when processing narrow band widths (see Table 2); therefore, elements less than $|2^{146}|$ are required.

4.1.10 _GESVD $\Leftarrow$ _GESVF: 0/2 routines

_GESVD modifies input matrix A based upon user request whereas _GESVF destroys A. Thus, in simply finding singular values, _GESVD returns A in bidiagonal form, when all the singular values fail to converge, with possible info from orthogonal matrices Q and P; _GESVF does not. However, this transformation of A is not documented for _GESVD; rather, it is a biproduct of a routine that is called by _GESVD, called _BDSQR. Also, if the SVD algorithm fails to converge, _GESVD returns the number of super- or subdiagonals that failed to converge in INFO; but, _GESVF returns the position of a singular value that did not converge. Since _GESVF destroys any information about A, it is not possible to duplicate the _GESVD INFO information. This is a functionality mismatch, and no enablement is possible. Further, the destruction of A and B by _GESVF makes it impossible to compute any of the orthogonal matrices U, U^T, V, and V^T due to limitations on workspace. (_GESVD workspace varies in size but is guaranteed to work any algorithm with LWORK $\geq$ MAX(3*MIN(M,N)+MAX(M,N),5*MIN(M,N)-4) & NAUX can range from needing at least N+MAX(M,N) to 2N+MAX(M,N,NB) where NB is equivalent to LAPACK's NRHS - the number of righthandsides.)

4.1.11 _GELSS $\Leftarrow$ _GESVS: 0/2 routines

In Section 4.1.10, _GESVD and _GESVF do compute correct factorizations; however, the results are specific to their respective solve routines, _GELSS and _GESVS. Consequently, without an enablement for _GESVD $\Leftarrow$ _GESVF, the wrong results would be produced. Thus, it appears possible to use the combination of _GESVF and _GESVS to enable _GELSS; but, all of the problems that occur in using _GESVF still apply. This is a functionality mismatch, and no enablement is possible. (_GELSS workspace varies in size but is guaranteed to work any algorithm with LWORK $\geq$ 3*N+MAX(2*N,NRHS,M) if M $\geq$ N else LWORK should be $\geq$ 3*M+MAX(2*M,NRHS,N) & _GESVS has no extra workspace requirement.)

4.1.12 _GELS $\Leftarrow$ _GELLS: 0/2 routines

_GELS and _GELLS both use QR to solve the minimal norm least squares solution of AX $\cong$ B. However, _GELS returns A with either its QR or LQ factorization; whereas, _GELLS destroys A. This is a functionality mismatch, and no enablement is possible. Also, for _GELLS, if LDB $\geq$ MAX(M,N), then matrix B (input) and matrix X (output) can be the same; otherwise, the results are unpredictable. Since _GELS always has matrix B and X the same, it would be necessary to have extra workspace (for _GELLS' X) to make sure _GELSS worked in all cases. Also, _GELS puts the residual in matrix B on output; but, _GELLS has a separate vector RN that takes on this value–thus, the need for more extra workspace. Further, _GELS already has an extra workspace requirement. (LWORK $\geq$ MIN(M,N) + MAX(1,M,N,NRHS) & NAUX $\geq$ 3N + MAX(N,NB) where NB = NRHS.)

4.2 Eigensystem: 0/20 routines

4.2.1 _GEEV $\Leftarrow$ _GEEV: 0/4 routines

There is a **name conflict** here. No enablement is possible. LAPACK stores S&D eigenvalues in real arrays WR and WI whereas ESSL uses complex array W. LAPACK stores S&D right eigenvectors in real matrix VR whereas ESSL uses complex array Z. These type mismatch problems do not exist in the C&Z routines. In eigenvector calculation, ESSL utilizes LOGICAL array SELECT to determine which eigenvectors are calculated, and ESSL only finds the right eigenvectors; conversely, LAPACK has no equivalent of SELECT and will find all the eigenvectors based on user request. Of course, it would be possible to calculate the left eigenvectors by using the ESSL routine on A^T, but A is destroyed by the ESSL routine & there is certainly not enough storage for a copy of A. So, it appears that as long as LAPACK's JOBVL = 'N' (left eigenvectors are not computed), then this routine will be functionally the same & LAPACK's WORK will be able to provide enough workspace for ESSL's AUX in S&D routines. But, in the C&Z routines, only the eigenvalues can be computed. (LAPACK's complex routines use RWORK with size 2N & ESSL's complex routines need NAUX $\geq$ 3N for eigenvector computation.)

4.2.2 _GEEVX $\Leftarrow$ _GEEV: 0/4 routines

There is a **name conflict** here (see Section 4.2.1). No enablement is possible. ESSL routine automatically balances input matrix A w/o capability of returning LAPACK's ILO, IHI, and SCALE array which contain all sorts of information used to balance A. Further, only a partial enablement would be possible given LAPACK's balancing options given by input variable BALANC. All the problems in _GEEV $\Leftarrow$ _GEEV enablement still apply.

4.2.3 _SPEV $\Leftarrow$ _SPEV: 0/2 routines

There is a **name conflict** here. No enablement is possible. A suggestion would be for ESSL to continue the use of the name _SLEV which was used in earlier versions of ESSL. ESSL does not offer information that will allow LAPACK's INFO to return with the number of off-diagonal elements of an intermediate tridiagonal form that failed to converge to zero. Also, LAPACK returns information about input matrix A's reduction to tridiagonal form in packed array AP; ESSL does not offer this information.

4.2.4 _SPEVX $\Leftarrow$ _SPSV: 0/2 routines

There is a **name conflict** here. LAPACK's linear algebra _SPSV conflicts with ESSL's eigensystem _SPSV. W/o a name change, it would be necessary to remove all of the dependencies on _SPSV in the LAPACK library. This seems to be a compromise of this enablement. ESSL does not offer information that will allow LAPACK's INFO and IFAIL to return with information about the number and indices of eigenvectors that failed to converge. Also, LAPACK returns information about input matrix A's reduction to tridiagonal form in packed array AP; ESSL does not offer this information. Finally, there is only enough workspace offered by LAPACK for ESSL to compute eigenvalues. (WORK's size is 8N & AUX needs $\geq$ 9N to compute eigenvectors.)

4.2.5 _HPEV $\Leftarrow$ _HPEV: 0/2 routines

There is a **name conflict** here. No enablement is possible. A suggestion would be for ESSL to continue the use of the name _HLEV which was used in earlier versions of ESSL. ESSL does not offer information that will allow LAPACK's INFO to return with the number of off-diagonal elements of an intermediate tridiagonal form that failed to converge to zero. Also, LAPACK returns information about input matrix A's reduction to tridiagonal form in packed array AP; ESSL does not offer this information.

4.2.6 _HPEVX $\Leftarrow$ _HPSV: 0/2 routines

There is a **name conflict** here. LAPACK's linear algebra _HPSV conflicts with ESSL's eigensystem _HPSV. W/o a name change, it would be necessary to remove all of the dependencies on _HPSV in the LAPACK library. This seems to be a compromise of this enablement. ESSL does not offer information that will allow LAPACK's INFO and IFAIL to return with information about the number and indices of eigenvectors that failed to converge. Also, LAPACK returns information about input matrix A's reduction to tridiagonal form in packed array AP; ESSL does not offer this information. Finally, there is only enough workspace offered by LAPACK for ESSL to compute eigenvalues. (RWORK's size is 6N & AUX needs $\geq$ 11N to compute eigenvectors.)

4.2.7 _GEGV $\Leftarrow$ _GEGV: 0/2 routines

There is a **name conflict** here. No enablement is possible. LAPACK stores alpha (the numerators of the eigenvalues) in real arrays ALPHAR and ALPHAI whereas ESSL uses complex array ALPHA. Eigenvectors are returned in LAPACK in real array VR; in ESSL, they are in complex array Z. At this time, LAPACK's INFO is extremely dependent on other LAPACK routines, and no match seems possible with ESSL.

4.2.8 _SYGV $\Leftarrow$ _SYGV: 0/2 routines

There is a **name conflict** here. No enablement is possible. Only a partial enablement would be allowed because ESSL only allows for A in lower storage mode only (LAPACK allows upper and lower). LAPACK's B returns the triangular factor U or L; ESSL's B is destroyed. Also, LAPACK's INFO returns the number of off-diagonal elements that failed to converge; ESSL does not offer this information. In LAPACK there is an option for the eigenvectors to be returned in A, but ESSL returns eigenvectors in Z. It is possible for Z to be copied to A, but there is no guarantee that enough workspace is provided. (LWORK $\geq$ MAX(1,3*N-1) & NAUX $\geq$ 2N and Z = N). However, there is enough workspace for an eigenvalue computation only.

4.3 Others: 4/8 routines

4.3.1 _TRTRI $\Leftarrow$ _TRI: 2/2 routines

This pair is fully enabled.

4.3.2 _TPTRI $\Leftarrow$ _TPI: 2/2 routines

This pair is fully enabled.

4.3.3 _LASWP $\Leftarrow$ _LASWP: 0/4 routines

There is a **name conflict** here. _LASWP does not exist for this version of ESSL, and no enablement is possible.

5 Timing Comparisons

Timing was performed on both an IBM RISC System/6000 model 550 and a model 590. The timer used was READRTC–an IBM-supplied microsecond timing routine. Compile switches were set as follows:

- On the 550, *-u -O3 -qMAXMEM=8192* were set.
- On the 590, *-u -O3 -qMAXMEM=8192 -qarch=pwrx* were set.

The standard LAPACK timing routines were used: these routines provide a comprehensive timing suite and are suitable for a performance comparison of the fortran LAPACK routines against the CCI routines. Most of the results for the {**D**} double precision routines are presented in the Tables.

Mflops for LAPACK vs. the CCI on an RS/6000-550												
NB	DGETRF		DPOTRF		DPBTRF[a]		DGETRI		DPOTRI		DTRTRI	
1	19.7	64.4	26.1	72.4	20.6	67.2	31.2	70.0	27.9	71.0	26.5	70.5
16	46.4	64.4	55.9	72.4	48.5	67.2	56.9	70.0	62.3	71.0	55.7	70.5
32	49.1	64.4	62.2	72.4	52.6	67.2	63.2	70.0	65.7	71.0	62.2	70.5
48	47.7	64.4	62.2	72.4	51.9	67.2	65.1	70.0	66.3	71.0	64.1	70.5
64	46.4	64.4	64.1	72.4	53.6	67.2	66.1	70.0	66.2	71.0	64.5	70.5

Table 3: There are two columns under each subroutine: the left is fortran LAPACK w/ ESSL BLAS, and the right is the CCI w/ ESSL BLAS. NB is the number of blocks for the fortran LAPACK routines only; ESSL times are independent of NB. For each NB, LDA = 513 (LDA is the leading dimension of the matrix) and, if necessary, UPLO = 'L' (operations performed with the lower triangle of the input matrix). Finally, all matrices are square w/ order N = 500.

[a]LDA = 602, N = 1000, K = 200 (the half band width)

Mflops for LAPACK vs. the CCI on an RS/6000-590												
NB	DGETRF		DPOTRF		DPBTRF[a]		DGETRI		DPOTRI		DTRTRI	
1	69.9	219.9	122.9	245.8	129.7	233.4	136.4	237.7	151.7	240.8	109.7	245.1
16	169.8	219.9	181.7	245.8	194.5	233.4	200.5	237.7	228.7	240.8	189.4	245.1
32	180.9	219.9	209.0	245.8	184.2	233.4	221.9	237.7	249.1	240.8	208.3	245.1
48	180.9	219.9	209.0	245.8	184.3	233.4	228.0	237.7	219.6	240.8	208.3	245.1
64	187.0	219.9	209.0	245.8	175.0	233.4	228.0	237.7	219.6	240.8	219.3	245.1

Table 4: There are two columns under each subroutine: the left is fortran LAPACK w/ ESSL BLAS, and the right is the CCI w/ ESSL BLAS. NB is the number of blocks for the fortran LAPACK routines only; ESSL times are independent of NB. For each NB, LDA = 513 (LDA is the leading dimension of the matrix) and, if necessary, UPLO = 'L' (operations performed with the lower triangle of the input matrix). Finally, all matrices are square w/ order N = 500.

[a]LDA = 602, N = 1000, K = 200 (the half band width)

Mflops for LAPACK vs. the CCI on both the 550 and 590								
	DGETRS				DPOTRS			
NRHS	550		590		550		590	
1	27.0	28.0	95.6	105.8	32.6	32.7	105.3	109.9
2	27.6	28.0	99.9	101.4	32.8	32.2	110.2	105.7
16	59.7	59.2	199.8	199.8	59.4	60.0	181.1	196.0
100	68.1	70.4	227.0	237.9	72.5	72.5	245.9	248.7

Table 5: There are two columns under each architecture: the left is fortran LAPACK w/ ESSL BLAS, and the right is the CCI w/ ESSL BLAS. NRHS is the number of right hand sides. For each NRHS, LDA = 513 (LDA is the leading dimension of the matrix) and, if necessary, UPLO = 'L' (operations performed with the lower triangle of the input matrix). Finally, all matrices are square w/ order N = 500.

Mflops for LAPACK vs. the CCI on both the 550 and 590												
	DPPTRF				DPPTRI				DTPTRI			
N	550		590		550		590		550		590	
50	18.8	48.2	57.7	170.5	18.5	46.5	48.6	155.4	15.2	44.8	45.1	145.6
100	24.4	64.5	93.6	232.6	31.3	53.0	91.4	213.6	27.6	57.7	81.3	209.6
200	23.7	59.2	115.1	235.1	35.7	61.5	136.5	224.8	33.2	63.3	125.5	224.8
300	23.5	64.0	113.1	226.1	37.8	66.2	133.7	233.9	35.8	65.4	133.3	227.3
400	23.6	67.1	100.4	227.1	38.9	66.0	118.7	230.5	37.1	65.2	118.5	237.0
500	23.3	67.5	99.5	232.2	39.7	66.1	117.5	231.8	37.7	66.0	115.7	231.5

Table 6: There are two columns under each architecture: the left is fortran LAPACK w/ ESSL BLAS, and the right is the CCI w/ ESSL BLAS. LDA = 513 (LDA is the leading dimension of the matrix) and, if necessary, UPLO = 'L' (operations performed with the lower triangle of the input matrix). N is the order of the square input matrix.

6 Availability

The CCI will be distributed similarly to LAPACK itself. Along with the CCI routines, three text files and one makefile will be distributed to guide the incorporation of the CCI. An example CCI routine plus the other four files follow.

6.1 Example

```
      SUBROUTINE DGETRF( M, N, A, LDA, IPIV, INFO )
*
*  -- LAPACK routine (version 2.0) --
*     Univ. of Tennessee, Univ. of California Berkeley, NAG Ltd.,
*     Courant Institute, Argonne National Lab, and Rice University
*     March 31, 1993
*
*  -- ESSL CCI enablement (version 1.0) --
*     Univ. of Tennessee, IBM Kingston and Yorktown,
*     August 1, 1994
*
*     .. Scalar Arguments ..
      INTEGER            INFO, LDA, M, N
*     ..
*     .. Array Arguments ..
      INTEGER            IPIV( * )
      DOUBLE PRECISION   A( LDA, * )
*     ..
*
*  Purpose
*  =======
*
*  DGETRF computes an LU factorization of a general M-by-N matrix A
*  using partial pivoting with row interchanges.
*
*  The factorization has the form
*     A = P * L * U
*  where P is a permutation matrix, L is lower triangular with unit
*  diagonal elements (lower trapezoidal if m > n), and U is upper
*  triangular (upper trapezoidal if m < n).
*
*  This is the right-looking Level 3 BLAS version of the algorithm.
*
*  ESSL Enablement Comments
*  ==== ========== ========
*
```

```
*  This is a stub routine that calls the ESSL subroutine DGEF when
*  the input matrix is square (M = N).  If M.ne.N, DGETRF is the
*  LAPACK routine DGETRF (the right-looking Level 3 BLAS version of
*  the algorithm).  In all cases, the results returned will be identical
*  in structure to those of the normal LAPACK routine DGETRF.
*
*  Arguments
*  =========
*
*  M       (input) INTEGER
*          The number of rows of the matrix A.  M >= 0.
*
*  N       (input) INTEGER
*          The number of columns of the matrix A.  N >= 0.
*
*  A       (input/output) DOUBLE PRECISION array, dimension (LDA,N)
*          On entry, the M-by-N matrix to be factored.
*          On exit, the factors L and U from the factorization
*          A = P*L*U; the unit diagonal elements of L are not stored.
*
*  LDA     (input) INTEGER
*          The leading dimension of the array A.  LDA >= max(1,M).
*
*  IPIV    (output) INTEGER array, dimension (min(M,N))
*          The pivot indices; for 1 <= i <= min(M,N), row i of the
*          matrix was interchanged with row IPIV(i).
*
*  INFO    (output) INTEGER
*          = 0:  successful exit
*          < 0:  if INFO = -i, the i-th argument had an illegal value
*          > 0:  if INFO = i, U(i,i) is exactly zero. The factorization
*                has been completed, but the factor U is exactly
*                singular, and division by zero will occur if it is used
*                to solve a system of equations.
*
*  =====================================================================
*
*     .. Parameters ..
      DOUBLE PRECISION   ONE, ZERO
      PARAMETER          ( ONE = 1.0D+0, ZERO = 0.0D+0 )
*     ..
*     .. Local Scalars ..
      CHARACTER*8        SV2103
      INTEGER            I, IERR1, IERR2, IINFO, J, JB, NB
```

```
*     ..
*     .. External Subroutines ..
      EXTERNAL           DGEF, DGEMM, DGETF2, DLASWP, DTRSM, EINFO,
     $                   ERRSAV, ERRSET, ERRSTR, XERBLA
*     ..
*     .. External Functions ..
      INTEGER            ILAENV
      EXTERNAL           ILAENV
*     ..
*     .. Intrinsic Functions ..
      INTRINSIC          MAX, MIN
*     ..
*     .. Executable Statements ..
*
*     Test the input parameters.
*
      INFO = 0
      IF( M.LT.0 ) THEN
         INFO = -1
      ELSE IF( N.LT.0 ) THEN
         INFO = -2
      ELSE IF( LDA.LT.MAX( 1, M ) ) THEN
         INFO = -4
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'DGETRF', -INFO )
         RETURN
      END IF
*
*     Quick return if possible
*
      IF( M.EQ.0 .OR. N.EQ.0 )
     $   RETURN
*
      IF( M.EQ.N ) THEN
*
*        Execute ESSL routine DGEF
*
*        ESSL error-handling initialization
*
         CALL EINFO( 0, IERR1, IERR2 )
         CALL ERRSAV( 2103, SV2103 )
         CALL ERRSET( 2103, 256, -1, 0, 0, 2103 )
*
```

```
*        Call to appropriate ESSL routine
*
         CALL DGEF( A, LDA, N, IPIV, *10 )
         GO TO 30
*
*        ESSL Run-time error:  use error information to
*        determine INFO and continue processing
*
   10    CONTINUE
         CALL EINFO( 2103, IERR1, IERR2 )
*
*        IERR1 gets the column number of the LAST zero diagonal element;
*        BUT, INFO needs to return the column number of the FIRST
*        zero diagonal element.  So, if an error is reported, we
*        re-examine all the elements on the diagonal up to IERR1
*        to find the correct value for INFO.
*
         DO 20 J = 1, IERR1
            IF( A( J, J ).EQ.ZERO ) THEN
               INFO = J
               GO TO 30
            END IF
   20    CONTINUE
*
*        Restore setting of parameters for error 2103
*
   30    CONTINUE
         CALL ERRSTR( 2103, SV2103 )
*
      ELSE
*
*        Execute LAPACK code for DGETRF
*
*        Determine the block size for this environment.
*
         NB = ILAENV( 1, 'DGETRF', ' ', M, N, -1, -1 )
         IF( NB.LE.1 .OR. NB.GE.MIN( M, N ) ) THEN
*
*           Use unblocked code.
*
            CALL DGETF2( M, N, A, LDA, IPIV, INFO )
         ELSE
*
*           Use blocked code.
```

```
*
         DO 50 J = 1, MIN( M, N ), NB
            JB = MIN( MIN( M, N )-J+1, NB )
*
*           Factor diagonal and subdiagonal blocks and test for exact
*           singularity.
*
            CALL DGETF2( M-J+1, JB, A( J, J ), LDA, IPIV( J ),
     $                   IINFO )
*
*           Adjust INFO and the pivot indices.
*
            IF( INFO.EQ.0 .AND. IINFO.GT.0 )
     $         INFO = IINFO + J - 1
            DO 40 I = J, MIN( M, J+JB-1 )
               IPIV( I ) = J - 1 + IPIV( I )
   40       CONTINUE
*
*           Apply interchanges to columns 1:J-1.
*
            CALL DLASWP( J-1, A, LDA, J, J+JB-1, IPIV, 1 )
*
            IF( J+JB.LE.N ) THEN
*
*              Apply interchanges to columns J+JB:N.
*
               CALL DLASWP( N-J-JB+1, A( 1, J+JB ), LDA, J, J+JB-1,
     $                      IPIV, 1 )
*
*              Compute block row of U.
*
               CALL DTRSM( 'Left', 'Lower', 'No transpose', 'Unit',
     $                     JB, N-J-JB+1, ONE, A( J, J ), LDA,
     $                     A( J, J+JB ), LDA )
               IF( J+JB.LE.M ) THEN
*
*                 Update trailing submatrix.
*
                  CALL DGEMM( 'No transpose', 'No transpose',
     $                        M-J-JB+1, N-J-JB+1, JB, -ONE,
     $                        A( J+JB, J ), LDA, A( J, J+JB ), LDA,
     $                        ONE, A( J+JB, J+JB ), LDA )
               END IF
            END IF
   50       CONTINUE
         END IF
      END IF
      RETURN
*
*     End of DGETRF
*
      END
```

6.2 CCI_README

```
===============
CCI README FILE
===============

VERSION 1.0 : August 1, 1994

DATE:  August 1, 1994

CCI is a Call Conversion Interface that allows LAPACK users to
incorporate the optimized performance of the Engineering and
Scientific Subroutine Library (ESSL) when using an IBM RS/6000
architecture.

CCI is available via netlib and xnetlib as a tar file.  A CCI
performance report is available as an LAPACK Working Note.
To receive a list of available reports, send email to
netlib@ornl.gov with a message of the form:
send index from lapack/lawns.

To utilize this package, the following software is required:

-- AIX 3.2.5
-- XLF 3.1
-- ESSL 2.2.1.1
-- LAPACK 2.0

The package includes several LAPACK/SRC/*.f files that will
access routines in ESSL to significantly speed up execution time
while maintaining LAPACK testing standards.

A CCI_README file containing the information in this letter and a
CCI_QUICK_INSTALL file containing a quick reference guide to the
installation process are located in the LAPACK directory.
Further, a new make.inc file is provided within the LAPACK
directory to make the CCI incorporation even easier.
A CCI_NOTES file contains all documentation on the CCI routines
available in this version.

Remember that LAPACK with the CCI will always need to be used
with the ESSL library when compiling your programs; further, in
order to prevent linking problems, LAPACK must be linked BEFORE
ESSL.  For example, on an IBM RS\6000, a file would be compiled
```

as follows:

```
xlf filename.f lapack.a -lessl
```

Please send comments, corrections, and suggestions to:

```
          Dr. Jack Dongarra
          107 Ayres Hall
          Department of Computer Science
          University of Tennessee
          Knoxville, TN   37996-1301

          Office: (615) 974-8295
          Fax   : (615) 974-8296

          Email : dongarra@cs.utk.edu
```

6.3 CCI_QUICK_INSTALL

```
============================================
Quick Reference Guide to Incorporate the CCI
============================================

1. Uudecode, uncompress and tar the file on
   top of the existing LAPACK directory.

   uudecode cci.uu
   uncompress cci.tar.Z
   tar xvf cci.tar

2. Remove the old LAPACK library so that it
   may be replaced by a new LAPACK library
   containing the CCI.

   cd LAPACK
   rm lapack.a

3. Edit the LAPACK/make.inc file to create the
   appropriate link to ESSL for the IBM
   architecture that you are using.

   For example, on an IBM RS/6000-550, choose

   BLASLIB      = -lessl

   as the library name for this architecture.

4. Make the new LAPACK library containing the
   CCI.

   make lapacklib
```

6.4 CCI_NOTES

```
==============
CCI NOTES FILE
==============

VERSION 1.0 : August 1, 1994

DATE:  August 1, 1994

This Notes file contains any available documentation that will
allow the best use of each subroutine available in the CCI.
Information is listed alphabetically by subroutine name.
Exact documentation can be obtained by examining the
ESSL Enablement Comments present in each CCI subroutine.

CGETRF:  supply square matrix (M = N)
CGETRS:  use as is
CPOTRF:  use as is
CPOTRS:  use as is
DGETRF:  supply square matrix (M = N)
DGETRI:  use as is
DGETRS:  use as is
DPBTRF:  supply lower band format (UPLO = 'L'), and
         scale input matrix so that exponents < 10**146
DPOTRF:  use as is
DPOTRI:  use as is
DPOTRS:  use as is
DPPTRF:  supply lower packed format (UPLO = 'L')
DPPTRI:  supply lower packed format (UPLO = 'L')
DTPTRI:  use as is
DTRTRI:  use as is
SGETRF:  supply square matrix (M = N)
SGETRI:  use as is
SGETRS:  use as is
SPBTRF:  supply lower band format (UPLO = 'L')
SPOTRF:  use as is
SPOTRI:  use as is
SPOTRS:  use as is
SPPTRF:  supply lower packed format (UPLO = 'L')
SPPTRI:  supply lower packed format (UPLO = 'L')
STPTRI:  use as is
STRTRI:  use as is
ZGETRF:  supply square matrix (M = N)
```

```
ZGETRS:  use as is
ZPOTRF:  use as is
ZPOTRS:  use as is
```

6.5 make.inc

```
####################################################################
#  LAPACK make include file.                                       #
#  LAPACK, Version 2.0                                             #
#  June 30, 1994                                                   #
#  Modified to incorporate  ESSL CCI (version 1.0),                #
#  August 1, 1994.                                                 #
####################################################################
#
#  The machine (platform) identifier to append to the library names
#
PLAT = _rs6k
#
#  Modify the FORTRAN and OPTS definitions to refer to the
#  compiler and desired compiler options for your machine.  NOOPT
#  refers to the compiler options desired when NO OPTIMIZATION is
#  selected.  Define LOADER and LOADOPTS to refer to the loader and
#  desired load options for your machine.
#
FORTRAN  = f77
OPTS     = -O3 -qMAXMEM=8192 -u
NOOPT    = -u
LOADER   = f77
LOADOPTS =
#
#  The archiver and the flag(s) to use when building archive (library)
#  If you system has no ranlib, set RANLIB = echo.
#
ARCH     = ar
ARCHFLAGS= cr
RANLIB   = ranlib
#
#  The location of the libraries to which you will link.  (The
#  machine-specific, optimized BLAS library is contained within IBM's
#  ESSL.  Thus, the BLASLIB identifier should be used for both the
#  BLAS library and the ESSL library identifier.  Select the
#  appropriate library name for the IBM architecture you are using
#  by removing the # in front of the BLASLIB identifiers below.)
#
#BLASLIB      = -lessl
#BLASLIB      = -lesslp2
LAPACKLIB    = lapack$(PLAT).a
TMGLIB       = tmglib$(PLAT).a
```

```
EIGSRCLIB    = eigsrc$(PLAT).a
```