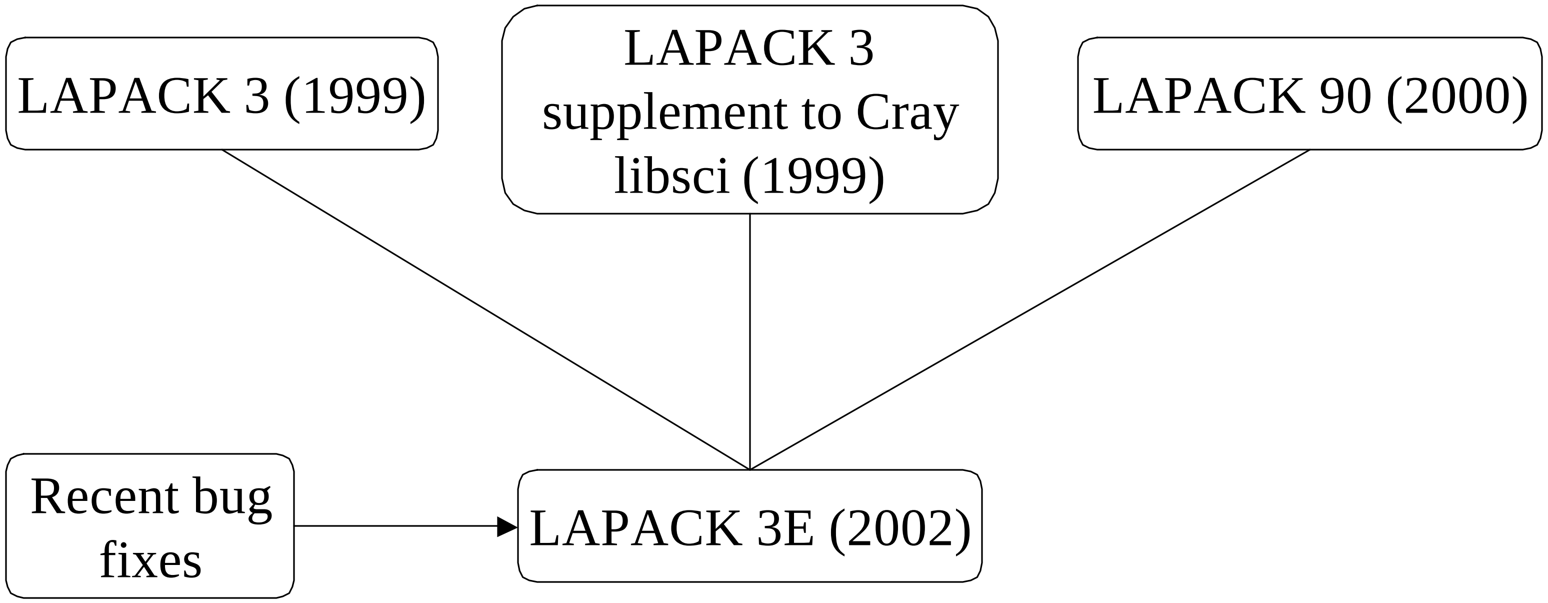
LAPACK 3E – A Fortran 90-enhanced version of LAPACK

Edward Anderson
Lockheed Martin Technology Services
Anderson.Edward@epa.gov

August 8, 2002

What is LAPACK 3E?

Project to integrate my past Cray enhancements with LAPACK 3E using LAPACK 90-style generic interfaces



Application

Molecular modeling using methods of computational biophysical chemistry

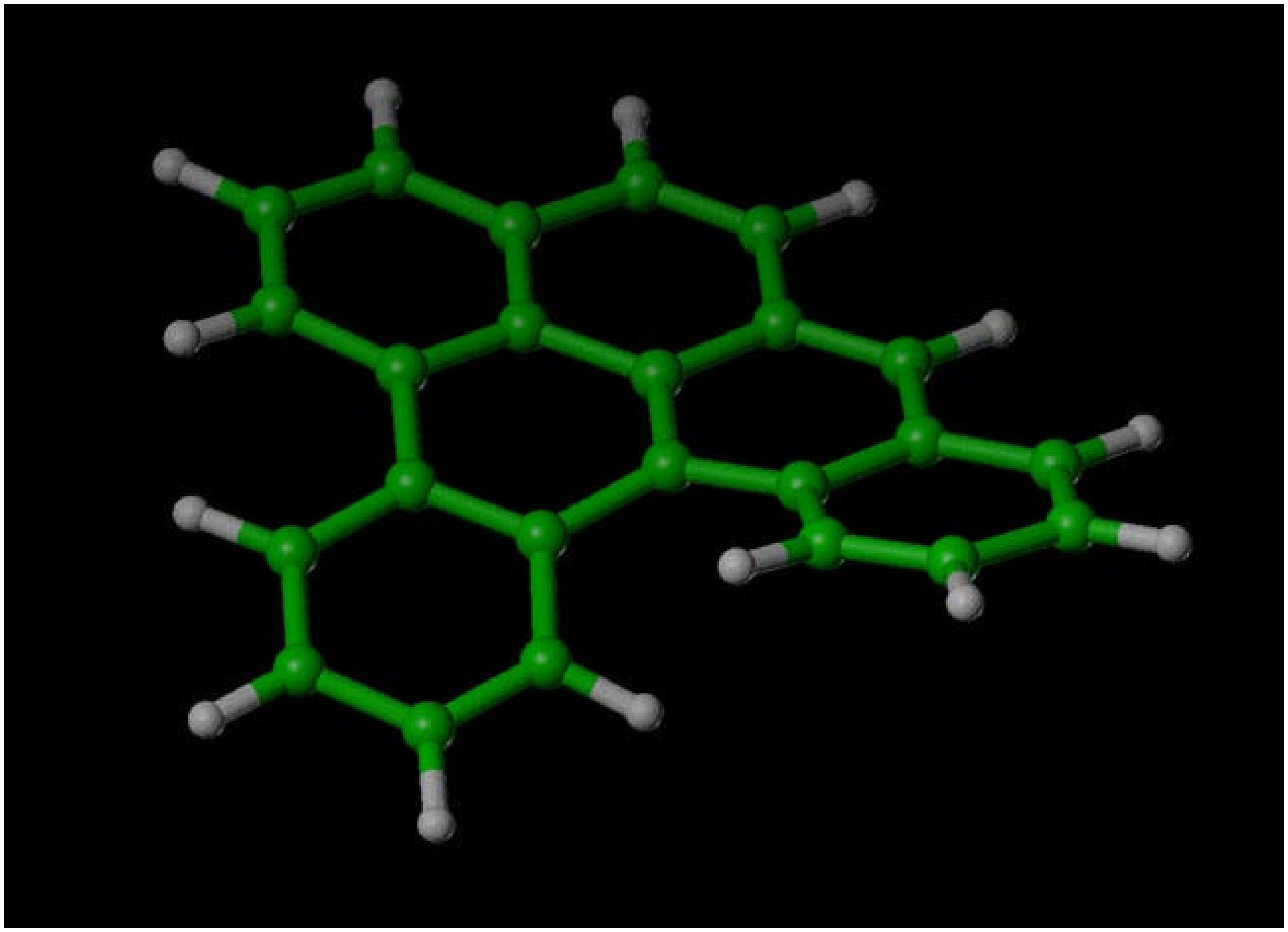
Studying Endocrine Disruptor Chemicals (EDCs) and Polycyclic Aromatic Hydrocarbons (PAHs) formed from incomplete combustion of organic materials

Modeling

- Capacity of metabolites of these environmental chemicals to bind to the ligand binding domain of the estrogen receptor and to DNA
- Effect this binding has on DNA structure

Goal: Assess the risk of cancers of the endocrine system (breast, prostate, thyroid) induced by environmental chemicals

Benzo(a,l)pyrene



Technical approach

- Use semi-empirical Hartree-Fock methods to obtain initial molecular geometries (Gaussian, AMSOL, MOPAC)
- Use ab-initio H-F methods to obtain more complete molecular structures, properties and interaction energies (Gaussian, GAMESS)
- For post H-F calculations, use density functional methods in Gaussian or "Divide and conquer", a quantum mechanical method for determining the energetic of the binding of environmental molecules to biopolymers

The D&C program uses a recursive bisection method for particle-particle interactions.

It calls LA_SYGVD from LAPACK 90!

LAPACK 95 at NESC

- Cray libsci was based on LAPACK 2.
- I installed LAPACK 3 supplement to libsci and LAPACK 95 on our CRAY T3E.
- Now we want the same libraries on our IBM SP.

Two main issues:

- Libsci supplement used Cray naming conventions (S = 64-bit real, C = 64-bit complex), wanted IEEE conventions on the IBM
- Needed thread-safe version of LAPACK 3 to use shared-memory parallelism

SAVE statements in LAPACK

Two contexts:

1) Reverse communication in xLACON and xLASQ3

   Add arguments to calling list and rename

2) Computed constants, e.g.,

```
LOGICAL FIRST
DATA FIRST / .TRUE. /
SAVE FIRST, …
IF( FIRST ) THEN
   …
   FIRST = .FALSE.
END IF
```

$\left.\right\}$ Compute constants first time only to reduce overhead

LAPACK 3E design

- Eliminate SAVE statements for thread safety
- Use PARAMETERS for replicated constants
- Parameterize KIND to allow common source for single and double precision
- Use generic interfaces defined in modules for all subroutine calls
- Use preprocessor for renaming at compile time
- Include bug fixes and improvements
- Replicate LAPACK 90 naming conventions

→ Modules and generic interfaces require Fortran 90! ←

Replicated constants

The LAPACK auxiliary routine SLAMCH is called to compute floating point model parameters that are intrinsics in Fortran 90.

EPS = SLAMCH(‘Epsilon’) ≅ EPSILON(1.0)
SAFMIN = SLAMCH(‘Safe minimum’) ≅ TINY(1.0)
SAFMAX = SLAMCH(‘Overflow’) ≅ HUGE(1.0)

SMLNUM is variously computed as

SAFMIN	SAFMIN*(N / EPS)
SAFMIN*REAL(MAX(1, N))	SQRT(SAFMIN / EPS)
SAFMIN / EPS	SQRT(SAFMIN) / EPS

In LAPACK 3E: make EPS, SAFMIN, etc. all PARAMETERs

Common source for different KINDs

Use KIND-specific declarations:

REAL(WP) instead of “REAL” or “DOUBLE PRECISION”

WP is defined in a module:

```
MODULE LA_CONSTANTS
     INTEGER, PARAMETER :: WP=8
     . . .
END MODULE LA_CONSTANTS
```

`WP=4` in module `LA_CONSTANTS32`

This module is used in every subroutine:

```
SUBROUTINE SGETRF( … )
    USE LA_CONSTANTS
    . . .
```

`PARAMETER`s in `LA_CONSTANTS`

WP	EPS, ULP
ZERO, CZERO	SAFMIN
HALF, CHALF	SAFMAX (= 1 / SAFMIN)
ONE, CONE	SMLNUM (= SAFMIN / ULP)
TWO	BIGNUM
THREE	RTMIN (= sqrt(SMLNUM))
FOUR	RTMAX
EIGHT	SPREFIX (‘S’ or ‘D’)
TEN	CPREFIX (‘C’ or ‘Z’)

`#ifdef _CRAY` and `#ifdef _CRAYMPP` are used to set numerical constants and subroutine prefixes correctly for Cray architectures

Generic interfaces

Following LAPACK95, create generic interfaces (in a module) for all BLAS and LAPACK routines:

```
MODULE LA_XFOO

INTERFACE LA_FOO

SUBROUTINE SFOO( X )
   USE LA_CONSTANTS32, ONLY: WP
   REAL(WP), INTENT(INOUT) :: X(*)
END SUBROUTINE SFOO

SUBROUTINE DFOO( X )
   USE LA_CONSTANTS, ONLY: WP
   REAL(WP), INTENT(INOUT) :: X(*)
END SUBROUTINE DFOO
```

```
SUBROUTINE CFOO( X )
   USE LA_CONSTANTS32, ONLY: WP
   COMPLEX(WP), INTENT(INOUT) :: X(*)
END SUBROUTINE CFOO

SUBROUTINE ZFOO( X )
   USE LA_CONSTANTS, ONLY: WP
   COMPLEX(WP), INTENT(INOUT) :: X(*)
END SUBROUTINE ZFOO

END INTERFACE ! LA_FOO

END MODULE LA_XFOO
```

Use of generic interfaces

Old style:

```
PROGRAM MAIN
REAL X(100)
EXTERNAL SFOO
CALL SFOO(X)
```

New style:

```
PROGRAM MAIN
USE LA_CONSTANTS
USE LA_XFOO
REAL(WP) :: X(100)
CALL LA_FOO(X)
```

What about:

```
   CALL LA_FOO(X(10)) !?
```

Mismatched interfaces

The calling site must match one of the interface specs for every argument exactly in type, kind, and rank.

If it doesn’t match, you can

a) Match the interface to the call
b) Match the call to the interface

LAPACK 3E modules define both the natural interace and a “point” interface for BLAS and LAPACK generic interfaces.

- Natural interface: just like the subroutine definition
- Point interface: all arrays are indexed (such as `A(I,J)` or `X(1)`)
- If the calling site doesn’t match the natural interface, index all the arrays to use the point interface
- Point interface is default – natural interface is a wrapper to it

Point and natural interfaces

Point interfaces allow argument matching by position and type without rank for use with indexed arrays.

```
MODULE LA_XCOPY

INTERFACE LA_COPY

!  Point interface for xCOPY1

SUBROUTINE SCOPY1( N, X, Y )
   USE LA_CONSTANTS32, ONLY: WP
   INTEGER, INTENT(IN) :: N
   REAL(WP), INTENT(IN) :: X
   REAL(WP), INTENT(OUT) :: Y
END SUBROUTINE SCOPY1

MODULE PROCEDURE SCOPY1_X1Y1

END INTERFACE ! LA_COPY
PRIVATE SCOPY1_X1Y1
```

```
CONTAINS

!  Natural interface for xCOPY1

SUBROUTINE SCOPY1_X1Y1( N, X, Y )
   USE LA_CONSTANTS32, ONLY: WP
   INTEGER, INTENT(IN) :: N
   REAL(WP), INTENT(IN) :: X(*)
   REAL(WP), INTENT(OUT) :: Y(*)
   CALL SCOPY1( N, X(1), Y(1) )
END SUBROUTINE SCOPY1_X1Y1

END MODULE LA_XCOPY
```

Interface modules in LAPACK 3E

- LA_BLAS1
- LA_BLAS2
- LA_BLAS3
- LA_AUXILIARY (commonly used auxiliaries)
- LA_LAPACK (many copied from LAPACK95)
- LA_XYYZZZ (infrequently used auxiliaries)

With USE, I always specify the interfaces needed:

```
USE LA_AUXILIARY, ONLY: ILAENV, XERBLA, LA_LARFG
```

Renaming in the preprocessor

Every LAPACK 3E routine has as its first line

```
   #include “lapacknames.inc”
```

The structure of this file is:

```
   #if LA_REALSIZE == 4 || LA_REALSIZE == 32
       #ifdef _CRAY
             #define SAXPY HAXPY
             . . .
       #end if
       #define LA_CONSTANTS LA_CONSTANTS32
   #else
       #ifndef _CRAY
             #define SAXPY DAXPY
             . . .
       #endif
   #endif
```

} S → H and C → G in 32 bits for Cray

} S → D and C → Z in 64 bits for IEEE

Invoking the preprocessor

On Cray, files with .F or .F90 extension invoke the preprocesor:

```
f90 –F –DLA_REALSIZE=4 –o hgetrf.o –c sgetrf.F
f90 –c sgetrf.F
```

On IBM, files with .F extension invoke the preprocessor:

```
xlf –WF,-DLA_REALSIZE=4 –c sgetrf.F
xlf –o dgetrf.o –c sgetrf.F
```

Summary of common source changes

```
#include "lapacknames.inc"
      SUBROUTINE SGETRF( M, N, A, LDA, IPIV, INFO )
      USE LA_CONSTANTS
      USE LA_AUXILIARY, ONLY: ILAENV, XERBLA, LA_LASWP
      USE LA_BLAS3, ONLY: LA_GEMM, LA_TRSM
      USE LA_XGETF2
*
*  -- LAPACK routine (version 3.0) --
*     Univ. of Tennessee, Univ. of California Berkeley, NAG Ltd.,
*     Courant Institute, Argonne National Lab, and Rice University
*     March 31, 1993
*     04-09-02:  LAPACK 3E version (eca)
*
*     .. Scalar Arguments ..
      INTEGER            INFO, LDA, M, N
*     ..
*     .. Array Arguments ..
      INTEGER            IPIV( * )
      REAL(WP)           A( LDA, * )
*     ..
```

} Translated from EXTERNAL stmts

```
*     .. Local Scalars ..
      INTEGER            I, IINFO, J, JB, NB
*     ..
*     .. Intrinsic Functions ..
      INTRINSIC          MAX, MIN
*     ..
*     .. Executable Statements ..
*
      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( SPREFIX // 'GETRF', -INFO )
         RETURN
      END IF
*
*     Quick return if possible
*
      IF( M.EQ.0 .OR. N.EQ.0 )
     $   RETURN
```

} No PARAMETERS and EXTERNAL statements

```
      NB = ILAENV( 1, SPREFIX // 'GETRF', ' ', M, N, -1, -1 )
      IF( NB.LE.1 .OR. NB.GE.MIN( M, N ) ) THEN
*
*        Use unblocked code.
*
         CALL LA_GETF2( M, N, A, LDA, IPIV, INFO )
      ELSE
*
*        Use blocked code.
*
         DO 20 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 LA_GETF2( 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 10 I = J, MIN( M, J+JB-1 )
               IPIV( I ) = J - 1 + IPIV( I )
   10       CONTINUE
*
*           Apply interchanges to columns 1:J-1.
*
            CALL LA_LASWP( J-1, A(1,1), LDA, J, J+JB-1, IPIV(1), 1 )
```

} Natural interface

Use point interfaces
inside loop for efficiency

```
            IF( J+JB.LE.N ) THEN
*
*              Apply interchanges to columns J+JB:N.
*
               CALL LA_LASWP( N-J-JB+1, A( 1, J+JB ), LDA, J, J+JB-1,
     $                        IPIV( 1 ), 1 )
*
*              Compute block row of U.
*
               CALL LA_TRSM( '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 LA_GEMM( '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
   20    CONTINUE
      END IF
      RETURN
      END
```

Mixed interface – convert to point interface at call

LAWN 126 improvements

LAWN 126 = “Performance improvements to LAPACK for the Cray Scientific Library”, with M. Fahey (1997)

- Parallel linear system solves with NRHS > 1
- Vastly better SLASSQ
- Cleaner SLARTG, SLARFG
- Faster SGEBAL
- Faster SSTEIN (using MGS)
- Add UPLO argument to CPTSV / CTPSVX (only incompatibility with LAPACK 3)
- Call Level 3 LAPACK routines, not Level 2 directly

What’s not in LAPACK 3E

- Fortran 90-style argument lists beyond LAPACK 95
- Allocatable work arrays
- Internal subroutines
- New error handler (still using XERBLA)
- Checks for NaN and INF arguments
- Extended precision arithmetic

Current status

- All 655 LAPACK routines converted
- Every routine has a generic INTERFACE
- Compiled successfully on IBM SP and CRAY T3E
- Standard tests pass on IBM SP
- No test routines have been converted
- Will be made available on netlib
- Target availability is September 30, 2002