Software Product Description

PRODUCT NAME: HP Fortran for Tru64 UNIX Alpha Systems Version 5.6 SPD 37.54.17

DESCRIPTION

HP Fortran for Tru64 UNIX Alpha Systems Version 5.6 contains both the HP Fortran 95/90 Version 5.6 software and the HP Fortran 77 Version 5.6 software as well as the Compaq Extended Math Library (CXML) Version 5.20. In this Software Product Description (SPD), HP Fortran refers to HP Fortran 95/90 unless a specific reference to the 95/90 or 77 product is needed to distinguish between the two software products.

HP Fortran is an implementation of the Fortran programming language that supports the FORTRAN 66, FORTRAN 77, Fortran 90, and Fortran 95 standards. HP Fortran 95/90 and HP Fortran 77 fully support the following standards:

- ANSI X3.9-1966 (FORTRAN 66)
- ANSI X3.9-1978 (FORTRAN 77)
- ISO 1539-1980(E) (FORTRAN 77)
- MIL-STD-1753
- FIPS-69-1 (HP Fortran meets the requirements of this standard by conforming to the ANSI Standard and by including a flagger. The flagger optionally produces diagnostic messages for compile-time elements that do not conform to the Full-Level ANSI Fortran Standard.)

HP Fortran 95/90 supports all of the standards that HP Fortran 77 supports plus the following new standards:

- ANSI X3.198-1992 (Fortran 90)
- ISO/IEC 1539-1:1997(E) (Fortran 95)

HP FORTRAN

HP Fortran 95/90 fully supports the multivendor OpenMP Fortran Version 1.1 Specification, including support for directed parallel processing using OpenMP directives on shared memory multiprocessor systems.

HP Fortran 95/90 supports most statement, library, and directive extensions from the HPF-2 specification (High Performance FORTRAN Language Specification, Version 2.0, January 31, 1997).

HP Fortran supports extensions to the ISO and ANSI standards, including a number of extensions defined by HP Fortran for the various HP Fortran platforms (operating system/architecture pairs). In addition to HP Tru64 UNIX Alpha systems, HP Fortran platforms include:

- HP Fortran for OpenVMS Integrity Systems
- HP Fortran and HP Fortran 77 for OpenVMS Alpha systems
- Compaq Fortran 77 for OpenVMS VAX systems

Major additions to the FORTRAN 77 standard introduced by the Fortran 90 standard include:

- Array operations
- Improved facilities for numeric computation
- Parameterized intrinsic data types
- User-defined data types
- Facilities for modular data and procedure definitions
- Pointers
- The concept of language evolution

- Support for DATE_AND_TIME intrinsic for obtaining dates using a four-digit year format

HP Fortran contains full support for the Fortran 95 standard, including the following features:

- FORALL statement and construct
- Automatic deallocation of ALLOCATABLE arrays
- DIM argument to MAXLOC and MINLOC
- PURE user-defined subprograms
- ELEMENTAL user-defined subprograms (a restricted form of a pure procedure)
- Pointer initialization (initial value)
- The NULL intrinsic to nullify a pointer
- Derived-type structure initialization
- CPU_TIME intrinsic subroutine
- KIND argument to CEILING and FLOOR intrinsics
- Nested WHERE constructs, masked ELSEWHERE statement, and named WHERE constructs
- Comments allowed in namelist input
- Generic identifier in END INTERFACE statements
- Minimal width field editing using a numeric edit descriptor with 0 width
- Detection of Obsolescent and/or Deleted features listed in the Fortran 95 standard. HP Fortran flags these obsolescent and deleted features, but fully supports them.

HP Fortran includes the following features and enhancements:

- Full support for 64-bit address space, including 64-bit static space
- Support for linking against static and shared libraries
- Support for creating shareable code to be put into a shared library
- Support for stack-based storage
- Support for dynamic memory allocation
- Support for reading and writing binary data files in nonnative formats, including IEEE™ (little-endian and big-endian), VAX, IBM™ System\360, and CRAY™ integer and floating point formats
- User control over IEEE floating point exception handling, reporting, and resulting values
- Control for memory boundary alignment of items in COMMON and fields in structures and warnings for unaligned data
- Directives to control listing page titles and subtitles, object file identification field, COMMON and record field alignment, and some attributes of COMMON blocks
- Ability to CALL an external function subprogram
- 7200 Character Statement Length
- Free form unlimited line length
- Mixing Subroutines/Functions in Generic Interfaces
- Composite data declarations using STRUCTURE, END STRUCTURE, and RECORD statements, and access to record components through field references
- Explicit specification of storage allocation units for data types such as:

  INTEGER*4
  LOGICAL*4
  REAL*4
  REAL*8
  COMPLEX*8
- Support for 64-bit signed integers using INTEGER*8 and LOGICAL*8
- Support for 128-bit floating-point real numbers (reals) using REAL*16 and COMPLEX*32
- A set of data types:
  - — BYTE
  - — LOGICAL*1, LOGICAL*2, LOGICAL*4, LOGICAL*8
  - — INTEGER*1, INTEGER*2, INTEGER*4, INTEGER*8
  - — REAL*4, REAL*8, REAL*16
  - — COMPLEX*8, COMPLEX*16, DOUBLE COMPLEX, COMPLEX*32
  - — POINTER (CRAY style)
- Data statement style initialization in type declaration statements
- AUTOMATIC and STATIC statements
- Bit constants to initialize LOGICAL, REAL, and INTEGER values and participate in arithmetic and logical expressions
- Built-in functions %LOC, %REF, and %VAL
- VOLATILE statement
- Bit manipulation functions
- Binary, hexadecimal, and octal constants and Z and O format edit descriptors applicable to all data types
- I/O unit numbers that can be any nonnegative INTEGER*4 value

- Variable amounts of data can be read from and written to "STREAM" files, which contain no record delimiters
- ENCODE and DECODE statements
- ACCEPT, TYPE, and REWRITE input/output statements
- DEFINE FILE, UNLOCK, and DELETE statements
- USEROPEN subroutine invocation at file OPEN time
- Support for I/O record larger than 2.1 Gigabytes in variable-length unformatted files
- Support for reading nondelimited character strings as input for character NAMELIST items
- Debug statements in source
- Generation of a source listing file with optional machine code representation of the executable source
- Generation of a symbolic assembly language representation of the executable source that can be assembled
- Variable format expressions in a FORMAT statement
- Optional run-time bounds checking of array subscripts and character substrings
- 31-character identifiers that can include dollar sign ($) and underscore (_)
- Support for executing in-line assembler code using the ASM intrinsics
- Support for the supercomputer intrinsics POPCNT, POPPAR, LEADZ, TRAILZ, and MULT_HIGH
- Language elements that support the various extended range and extended precision floating point architectural features:
  - — 32-bit IEEE S_floating data type, with an 8-bit exponent and 24-bit mantissa, which provides a range of 1.17549435E-38 (normalized) to 3.40282347E38 (the IEEE denormalized limit is 1.40129846E-45) and a precision of typically 7 decimal digits
  - — 64-bit IEEE T_floating data type, with an 11-bit exponent and 53-bit mantissa, which provides a range of 2.2250738585072013D-308 (normalized) to 1.7976931348623158D308 (the IEEE denormalized limit is 4.94065645841246544D-324) and a precision of typically 15 decimal digits
  - — 128-bit IEEE extended Alpha X_floating data type, with a 15-bit exponent and a 113-bit mantissa, which provides a range of approximately 6.48Q-4966 to 1.18Q4932 and a precision of typically 33 decimal digits
- Command line control for:
  - — The size of default INTEGER, REAL, and DOUBLE PRECISION data items
  - — The levels and types of optimization to be applied to the program
  - — The directories to search for INCLUDE files
  - — Inclusion or suppression of various compile-time warnings
  - — Inclusion or suppression of run-time checking for various I/O and computational errors
  - — Control over whether compilation terminates after a specific number of errors has been found
  - — Choosing whether executing code will be thread-reentrant
- Internal procedures can be passed as actual arguments to procedures
- Kind types for all of the hardware-supported data types:
  - — For 1-, 2-, 4-, and 8-byte LOGICAL data:

    ```
    LOGICAL (KIND=1)
    LOGICAL (KIND=2)
    LOGICAL (KIND=4)
    LOGICAL (KIND=8)
    ```
  - — For 1-, 2-, 4-, and 8-byte INTEGER data:

    ```
    INTEGER (KIND=1)
    INTEGER (KIND=2)
    INTEGER (KIND=4)
    INTEGER (KIND=8)
    ```
  - — For 4-, 8-, and 16-byte REAL data:

    ```
    REAL (KIND=4)
    REAL (KIND=8)
    REAL (KIND=16)
    ```
  - — For single precision, double precision, and quad-precision COMPLEX data:

    ```
    COMPLEX (KIND=4)
    COMPLEX (KIND=8)
    COMPLEX (KIND=16)
    ```
- The following features found in Compaq Visual Fortran:
  - — # Constants–constants using other than base 10
  - — C Strings–NULL terminated strings
  - — Conditional Compilation And Metacommand Expressions ($define, $undefine, $if, $elseif, $else, $endif)
  - — $FREEFORM, $NOFREEFORM, $FIXEDFORM–source file format
  - — $INTEGER, $REAL–selects size

- $FIXEDFORMLINESIZE–line length for fixed form source
- $STRICT, $NOSTRICT–F90 conformance
- $PACK-structure packing
- $ATTRIBUTES ALIAS–external name for a subprogram or common block
- $ATTRIBUTES C, STDCALL–calling and naming conventions
- $ATTRIBUTES VALUE, REFERENCE–calling conventions
- \ Descriptor–prevents writing an end-of-record mark
- Ew.dDe and Gw.dDe Edit Descriptors–similar to Ew.dEe and Gw.dEe
- $DECLARE and $NODECLARE (same as IMPLICIT NONE)
- $ATTRIBUTES EXTERN–variable allocated in another source file
- $ATTRIBUTES VARYING–variable number of arguments
- $ATTRIBUTES ALLOCATABLE–allocatable array
- Mixing Subroutines/Functions in Generic Interfaces
- $MESSAGE–output message during compilation
- $LINE (same as C's #line)
- INT1 converts to one byte integer by truncating
- INT2 converts to two byte integer by truncating
- INT4 converts to four byte integer by truncating
- COTAN returns cotangent
- DCOTAN returns double precision cotangent
- IMAG returns the imaginary part of complex number
- IBCHNG reverses value of bit
- ISHA shifts arithmetically left or right
- ISHC performs a circular shift
- ISHL shifts logically left or right

• Support for directed decomposition for parallel processing on shared memory multiprocessor systems using source code directives from either OpenMP (!$OMP) or HP Fortran (!$PAR):

- PARALLEL and END PARALLEL directives to define parallel regions
- DO and END DO directives to define parallel work constructs
- PARALLEL and SECTIONS directives to define parallel work constructs
- PRIVATE and SHARED attributes to describe data local or global to the threads of execution
- CRITICAL section directive to define a guarded region where one thread executes at a time
- TASK COMMON or THREADPRIVATE directives to allow each thread to have a local copy of a COMMON block
- Environment variables to control resource utilization at run-time
- Library routines to query and adjust the run-time parallel environment
- Nested OpenMP parallel regions
- NUM_THREADS clause

• A number of High Performance Fortran (HPF) features, including:

- The data parallel FORALL statement and construct
- Execution model procedure prefixes:

  EXTRINSIC(HPF)
  EXTRINSIC(HPF_LOCAL)
  EXTRINSIC(HPF_SERIAL)

- PURE procedure prefix to specify a lack of procedure side effects
- The following HPF data alignment and distribution directives:

  ALIGN
  DISTRIBUTE
  INHERIT
  PROCESSORS
  TEMPLATE
  SHADOW
  ON (in conjunction with INDEPENDENT loops)

- Many HPF-2 approved extensions, including:
  * HPF_LOCAL routines, and all HPF_LOCAL_LIBRARY routines except LOCAL_BLKCNT, LOCAL_LINDEX, and LOCAL_LINDEX but none of the approved extensions to HPF_LOCAL_LIBRARY routines
  * HPF_SERIAL routines
  * ON directive within INDEPENDENT loops
  * RESIDENT directive used in conjunction with INDEPENDENT loops
  * Mapping of derived type components
  * Pointers to mapped objects

  * Shadow width declarations
- — HPF intrinsic procedures and library routines:
  * NUMBER_OF_PROCESSORS and PROCESSORS_SHAPE
  * Reduction functions
  * Combining scatter functions
  * Parallel prefix and suffix functions
  * Sorting functions
  * System inquiry intrinsics
  * Computational intrinsics
  * Mapping inquiry subroutines
  * Bit manipulation functions
  * Array reduction functions
  * Array combining scatter functions
  * Array parallel prefix and suffix functions
  * Array sorting functions GRADE_UP, GRADE_DOWN
- — HPF INDEPENDENT directive
- — HPF SEQUENCE and NOSEQUENCE directives

HP Fortran 77 contains the following extensions to the FORTRAN 77 standard:

- Support for recursive subprograms
- IMPLICIT NONE statements
- INCLUDE statement
- NAMELIST-directed I/O
- DO WHILE and ENDDO statements
- Use of exclamation point (!) for end of line comments
- Generation of Cross Reference Listings
- Support for NTT Technical Requirement TR550001, Multivendor Integration Architecture (MIA) Version 1.1, Division 2, Part 3-2, Programming Language FORTRAN
- Support for automatic arrays
- Support for the SELECT CASE - CASE - CASE DEFAULT - END SELECT statements
- Support for the EXIT and CYCLE statements and for construct names on DO - END DO statements
- Reporting of unused and uninitialized variables
- Support for DATE_AND_TIME intrinsic for obtaining dates using a four-digit year format

HP Fortran provides a multiphase optimizer that is capable of performing optimizations across entire programs. Specific optimizations performed by both HP Fortran 95/90 and HP Fortran 77 include:

- Constant folding
- Optimizations of arithmetic IF, logical IF, and block IF-THEN-ELSE
- Global common subexpression elimination
- Removal of invariant expressions from loops
- Global allocation of general registers across program units
- In-line expansion of statement functions and routines
- Optimization of array addressing in loops
- Value propagation
- Deletion of redundant and unreachable code
- Loop unrolling
- Thorough dependence analysis
- Software pipelining to rearrange instructions between different unrolled loop iterations
- Optimized interface to intrinsic functions
- Loop transformation optimizations that apply to array references within loops, including:
  - — Loop blocking
  - — Loop distribution
  - — Loop fusion
  - — Loop interchange
  - — Loop scalar replacement
  - — Outer loop unrolling
- Speculative code scheduling, where a conditionally executed instruction is moved to a position before a test instruction and executed unconditionally. This reduces instruction latency stalls. (Performance may be reduced somewhat, because the run-time system must dismiss exceptions caused by speculative instructions.)

Specific optimizations performed by HP Fortran 95/90 include:

- Array temporary elimination
- A number of HPF-specific optimizations, including:
  - — Message vectorization
  - — Nearest-neighbor optimizations for improved communication performance of constructs typically seen in PDE solvers
  - — Parallelism of reductions

— Run-time preprocessing of loops for improved performance of irregular data access code

— Many other communication-based optimizations

Both HP Fortran 95/90 and HP Fortran 77 are shareable, re-entrant compilers that operate under the HP Tru64 UNIX operating system. They globally optimize source programs while taking advantage of the native instruction set and the HP Tru64 UNIX virtual memory system.

COMPAQ EXTENDED MATH LIBRARY (CXML)

Compaq Extended Math Library (CXML) is a set of mathematical subprograms that are optimized for HP architectures. Included subprograms cover the areas of:

- Basic Linear Algebra
- Linear System and Eigenproblem Solvers
- Sparse Linear System Solvers
- Sorting
- Random Number Generation
- Signal Processing

The Basic Linear Algebra library includes the industry-standard Basic Linear Algebra Subprograms (BLAS) Level 1, Level 2, and Level 3. Also included are subprograms for BLAS Level 1 Extensions, Sparse BLAS Level 1, and Array Math Functions (VLIB).

The Linear System and Eigenproblem Solver library provides the complete LAPACK V3 package developed by a consortium of university and government laboratories. LAPACK is an industry-standard subprogram package offering an extensive set of linear system and eigenproblem solvers. LAPACK uses blocked algorithms that are better suited to most modern architectures, particularly ones with memory hierarchies. LAPACK will supersede LINPACK and EISPACK for most users.

The Sparse Linear System library provides both direct and iterative sparse linear system solvers. The direct solver package supports symmetric and symmetrically structured matrices with real or complex elements. The iterative solver package contains a basic set of storage schemes, preconditioners, and iterative solvers. The design of this package is modular and matrix-free, allowing future expansion and easy modification by users.

The Signal Processing library provides a basic set of signal processing functions. Included are one-, two-, and three-dimensional Fast Fourier Transforms (FFT), group FFTs, Cosine/Sine Transforms (FCT/FST), Convolution, Correlation, and Digital Filters.

Many CXML subprograms are optimized for the supported hardware platforms. Optimization techniques include traditional optimizations such as loop unrolling and loop reordering. CXML subprograms also provide efficient management of the hierarchical memory system, using techniques such as the following:

- Reuse of data within registers to minimize memory accesses
- Efficient cache management
- Use of blocked algorithms that minimize translation buffer misses and unnecessary paging

Since CXML routines can be called from all languages that support the HP Tru64 UNIX calling standard, the library provides optimized computation for applications written in these languages. Where appropriate, most subprograms are available in both real and complex versions, as well as in both single and double precision. CXML for HP Tru64 Alpha supports IEEE floating-point format.

Parallel Library Support for Symmetric Multiprocessing

CXML also supports symmetric multiprocessing (SMP) for improved performance. Key BLAS Level 2 and 3 routines have been modified to execute in parallel if run on SMP hardware. Also modified for this purpose are:

- LAPACK GETRF and POTRF routines
- Sparse iterative solvers
- Direct sparse solvers
- FFT routines

These parallel routines along with the other serial routines are supplied in an alternative library. The user can choose to link with either the parallel or the serial library, depending on whether SMP support is required, since each library contains the complete set of routines.

Basic Linear Algebra Subprograms

Linear algebra operations are fundamental to many mathematical applications, and several libraries of linear algebra subprograms exist throughout the computer industry. The CXML BLAS library contains the most commonly used linear algebra subprograms.

The CXML linear algebra library contains five groups of subprograms at three levels:

- Basic Linear Algebra Subprograms (BLAS) Level 1
- BLAS Level 1 Extensions
- BLAS Level 1 Sparse Extensions
- BLAS Level 2

- BLAS Level 3

BLAS Level 1 (Scalar/Vector and Vector/Vector Operations)

BLAS Level 1 provides a set of elementary vector functions, operating on one or two vectors. These are typically very small routines, and they make less efficient use of the computing resources of modern computer architectures than the Level 2 and 3 operations.

CXML provides the 15 standard BLAS Level 1 operations:

- The index of the element of a vector having maximum absolute value
- The sum of the absolute values of the elements of a vector
- Inner product of two real vectors
- Scalar plus the extended precision inner product of two real vectors
- Conjugated inner product of two complex vectors
- Unconjugated inner product of two complex vectors
- Square root of the sum of squares (norm) of the elements of a vector
- Scalar times a vector plus a vector
- Copy one vector to another
- Apply a Givens rotation
- Apply a modified Givens plane rotation
- Generate elements for a Givens plane rotation
- Generate elements for a modified Givens plane rotation
- Product of a vector times a scalar
- Swap the elements of two vectors

BLAS Level 1 Extensions (Vector/Vector Operations)

When developing mathematical algorithms using the BLAS Level 1, scientists and engineers found that several additional constructs were used on a regular basis. These constructs are well known throughout the computer industry as BLAS Level 1 Extensions.

CXML contains 13 BLAS Level 1 Extension operations:

- Index of element having the minimum absolute value
- Index of element having the maximum value
- Index of element having the minimum value
- Largest value of the elements of a vector
- Smallest value of the elements of a vector
- Largest absolute value of the elements of a vector
- Smallest absolute value of the elements of a vector
- Sum of the values of the elements of a vector
- Set all elements of a vector equal to a scalar
- Constant times a vector set to another vector ($y = a*x$)
- Euclidean norm with no intermediate scaling
- Sum of the squares of the elements of a vector
- Constant times a vector plus a vector set to another vector ($z = a*x + y$)

BLAS Level 1 Sparse Extensions (Vector/Vector Operations)

This group of operations is similar to the BLAS Level 1 routines, but is designed to work on sparse vectors (vectors in which most of the elements are zero). Six of the routines are from industry standard Sparse BLAS 1, and the remaining three are enhancements.

The nine sparse BLAS Level 1 operations are:

- Scalar times a sparse vector plus a vector
- Sum of a sparse vector and a full vector
- Inner product of a sparse vector and a full vector
- Gather a sparse vector from a full vector
- Gather a sparse vector from the scaled elements of a full vector
- Gather a sparse vector from a full vector and zero corresponding elements of full vector
- Apply Givens rotation to a sparse vector and a full vector
- Scatter a sparse vector into a full vector
- Scale and scatter a sparse vector into a full vector

BLAS Level 2 (Matrix/Vector Operations)

The BLAS Level 2 codes make more effective use of the data in the registers, reducing the number of register loads and stores required. In addition, loop unrolling techniques are used to minimize cache misses and page faults. The BLAS Level 2 subprograms use the following types of operations:

- Matrix/vector products
- Rank-1 and rank-2 matrix updates
- Solutions of triangular systems of equations

Six types of matrices are supported by these BLAS Level 2 routines:

- General
- General band
- Symmetric/Hermitian
- Symmetric/Hermitian band
- Triangular
- Triangular band

BLAS Level 3 (Matrix/Matrix Operations)

The BLAS Level 3 routines operate at a level that makes the most efficient use of machine resources. CXML optimizes these routines by partitioning matrices into blocks and computing matrix/matrix operations on each block. This approach avoids excessive memory accesses by providing full reuse of data while each block is in the cache or the registers. BLAS Level 3 routines provide this kind of blocking for three basic types of operations:

- Matrix/matrix products
- Rank-k and rank-2k updates of a symmetric matrix
- Solving triangular systems of equations with multiple right-hand sides

Three types of matrices are supported by these BLAS Level 3 routines:

- General
- Symmetric/Hermitian
- Triangular

A set of additional matrix-matrix routines is provided:

- Add two matrices
- Subtract one matrix from another
- Transpose a matrix, in-place or out-of-place

Array Math Functions

The Array Math Functions provide a set of basic math functions that operate on arrays of numbers rather than on scalars. On vector and superscalar architectures, such functions have a performance advantage over a loop of scalar operations. The library includes the following array functions for double precision numbers:

- Sine of array
- Cosine of array
- Cosine and sine of array
- Exponent of array
- Logarithm of array
- Square root of array
- Reciprocal of array

LAPACK Library Contents

LAPACK is a library of linear algebra subprograms intended to solve a wide range of problems, including dense systems of linear equations, linear least squares problems, eigenvalue problems, and singular value problems. It is also useful in doing computations such as matrix factorizations and estimations of condition numbers.

The CXML LAPACK library provides the complete LAPACK V3 package, compiled, tested, and ready to use. Combined with the optimized BLAS Level 3 routines, the CXML LAPACK provides optimal performance on all supported platforms. LAPACK should be used in place of LINPACK and EISPACK, because it is more efficient, accurate, and robust.

LAPACK supports both real and complex, single and double precision data. It operates on the following types of matrices:

- Bidiagonal
- General band
- General unsymmetric
- General tridiagonal
- Hermitian
- Hermitian, packed storage
- Upper Hessenberg, generalized problem
- Upper Hessenberg
- Orthogonal
- Orthogonal, packed storage
- Symmetric/Hermitian positive definite band
- Symmetric/Hermitian positive definite

- Symmetric/Hermitian positive definite, packed storage
- Symmetric/Hermitian positive definite tridiagonal
- Symmetric band
- Symmetric, packed storage
- Symmetric tridiagonal
- Symmetric
- Triangular band
- Triangular, generalized problem
- Triangular, packed storage
- Triangular
- Trapezoidal
- Unitary
- Unitary, packed storage

LAPACK provides the following operations:

- Triangular factorization
- Unblocked triangular factorization
- Solve a system of linear equations (based on triangular factorization)
- Compute the inverse (based on triangular factorization)
- Compute a split Cholesky factorization of a symmetric/Hermitian positive definite band matrix
- Unblocked computation of inverse
- Estimate condition number
- Refine initial solution returned by solver
- Perform QR factorization without pivoting
- Unblocked QR factorization
- Solve linear least squares problem (based on QR factorization)
- Solve the linear equality constrained least squares (LSE) problem
- Solve the Gauss-Markov linear model problem
- Perform LQ factorization without pivoting
- Unblocked LQ factorization
- Solve underdetermined linear system (based on LQ factorization)
- Generate a real orthogonal or complex unitary matrix as a product of Householder matrices
- Unblocked generation of real orthogonal or unitary matrix
- Multiply a matrix by a real orthogonal or complex unitary matrix by applying a product of Householder matrices
- Unblocked version of multiplication of a matrix by a real orthogonal or complex unitary matrix by applying a product of Householder matrices
- Reduce a square matrix to upper Hessenberg form
- Unblocked version of square matrix reduction
- Reduce a symmetric matrix to real symmetric tridiagonal form
- Reduce a band matrix to bidiagonal form
- Unblocked version of symmetric matrix reduction
- Reduce a rectangular matrix to bidiagonal form
- Reduce a band symmetric/Hermitian matrix to tridiagonal form
- Reduce a symmetric/Hermitian-definite banded generalized eigenproblem to standard form
- Compute various norms of a complex Hermitian tridiagonal matrix
- Compute eigenvalues and optional Schur factorization or eigenvectors using QR algorithm
- Compute selected eigenvectors by inverse iteration
- Compute eigenvectors from Schur factorization
- Compute eigenvectors using the Pal-Walker-Kahan variant of the QL or QR algorithm
- For a pair of N-by-N real nonsymmetric matrices, compute the generalized eigenvalues, the real Schur form, the left and/or right Schur vectors, and the left and/or right generalized eigenvectors
- Solve the generalized nonsymmetric eigenproblem Ax = lambda Bx
- Solve the generalized definite banded eigenproblem Ax = lambda Bx
- Solve the generalized symmetric/Hermitian-definite banded eigenproblem
- Solve the symmetric eigenproblem using divide-and-conquer algorithm
- Compute singular values and, optionally, singular vectors using the QR algorithm
- Compute the generalized (quotient) singular value decomposition
- Compute the generalized singular value decomposition (GSVD) on the M-by-N matrix A and P-by-N matrix B
- Solve a generalized linear regression model problem

Sparse System Solver Subprograms

The CXML Sparse System Solver library contains a set of subprograms that can be used to solve sparse linear systems of equations. Two packages providing direct and iterative methods are supported.

Direct Method Sparse Solver Package:

The direct solver package includes routines to solve symmetric and symmetrically structured matrices with real or complex elements. For symmetric matrices, these routines can solve both positive definite and indefinite systems.

The direct solver package routines use a row major, upper triangular storage format.

Routines are provided to do the following:

- Initialize the solver
- Define the structure of the matrix
- Reorder the matrix
- Factor the matrix
- Compute the solution vector of the system
- Return statistics about the phases of the solving process

The package permits the factorization of a matrix to be used to compute solutions for additional right-hand sides, and for the reordering of a matrix to be used to solve additional systems with the same structure but different values for the matrix elements.

Iterative Method Sparse Solver Package:

For the iterative method, the library provides a modular set of storage schemes, preconditioners, and solvers. These solvers and preconditioners are easily accessed through an integrated driver routine.

Six iterative sparse solvers for real, double precision data are supplied:

- Preconditioned conjugate gradient method
- Preconditioned least squares conjugate gradient method
- Preconditioned biconjugate method
- Preconditioned conjugate gradient squared method
- Preconditioned generalized minimum residual method
- Preconditioned transpose free QMR method

Routines for three storage schemes are provided, or the user can develop routines to employ a custom storage scheme. The supplied storage schemes include:

- Symmetric diagonal
- Unsymmetric diagonal
- General storage by rows

Three preconditioners are supplied, which can be selectively applied to the data. Users can also supply custom preconditioners. The preconditioners supplied include:

- Diagonal
- Polynomial (Neumann)
- Incomplete LU with zero diagonals added

Sorting Subprograms

Two sort subprograms using the Quicksort algorithm and two general purpose radix sort subprograms are provided, as follows:

- Sort elements of a vector using the Quicksort algorithm
- Sort an indexed vector of data using the Quicksort algorithm
- Sort data using a radix sort algorithm
- Sort an indexed vector of data using a radix sort algorithm

All of the above sorts operate on data stored in memory.

Random Number Subprograms

CXML provides four random number generator subprograms:

- Produce a vector of uniform [0,1], long-period random numbers using the L'Ecuyer multiplicative method. Two auxiliary input routines are provided to allow this subprogram to be called from within a parallel section of a program.
- Produce a vector of N(0,1), normally-distributed random numbers. Two auxiliary input routines are provided to allow this subprogram to be called from within a parallel section of a program.
- Produce single precision random numbers using a linear multiplicative algorithm
- Produce single precision random numbers using a Lehmer multiplicative generator

Signal Processing Subprograms

The CXML Signal Processing library contains a set of subprograms in four basic areas of signal processing:

- Fast Fourier Transforms (FFT)
- Fast Cosine and Fast Sine Transforms (FCT and FST)
- Convolution and correlation
- Digital filters

Fast Fourier Transforms and Cosine and Sine Transforms

CXML provides one-dimensional, two-dimensional, three-dimensional, and group FFT routines and one-dimensional FCT/FST routines. Each routine is supplied in two forms:

- The first form computes the transform in one unit operation. This is convenient for programs requiring speed on only one or a few operations.
- The second form is provided for programs requiring speed on repeated operations. With this form, each routine is subdivided into three routines. One routine builds the rotation factors, a second routine applies them to perform the transform, and a third routine deallocates any virtual memory allocated in the first routine. Thus, for repeated operations, the rotation factors need to be built only once.

Convolution and Correlation

CXML provides routines for computing one-dimensional discrete convolutions and correlations. These routines can process both periodic and nonperiodic data.

Digital Filters

CXML provides support for one-dimensional, nonrecursive digital filtering. Based on the Kaisers Sinh-Bessel algorithm, these routines allow programming of bandpass, bandstop, low-pass, and high-pass filters.

Cray SciLib Portability Support

SCIPORT is an HP implementation of V7 of the Cray Research scientific numerical library, SciLib. SCIPORT provides 64 bit single-precision and 64-bit integer interfaces to underlying CXML routines for Cray users porting programs to Alpha systems running HP Tru64 UNIX. SCIPORT also provides equivalent versions of almost all Cray Math Library and CF77 (Cray Fortran 77) Math intrinsic routines.

In order to be completely source code compatible with SciLib, the SCIPORT library calling sequence supports 64-bit integers passed by reference. However, internally, SCIPORT uses 32 bit integers. Consequently, some run-time uses of SciLib may not be supported by SCIPORT.

SCIPORT provides the following:

- 64-bit versions of all Cray SciLib single-precision BLAS Level 1, Level 2, and Level 3 routines
- All Cray SciLib LAPACK routines
- All Cray SciLib Special Linear System Solver routines
- All Cray SciLib Signal Processing routines
- All Cray SciLib Sorting and Searching routines

These routines are completely interchangeable with their Cray SciLib counterparts up to the runtime limit on integer size, and with the exception of the ORDERS routine, require no program changes to function correctly. Owing to endian differences of machine architecture, special considerations must be given when the ORDERS routine is used to sort multibyte character strings.

Run-Time Library Redistribution

The HP Fortran kit may include updated Run-Time Library shareable images. HP grants the user a nonexclusive royalty-free worldwide right to reproduce and distribute the executable version of the Run-Time Library (the “RTLs”), provided that the user does all of the following:

- Distributes the RTLs only in conjunction with and as a part of the user’s software application product that is designed to operate in the HP Tru64 UNIX environment.
- Does not use the name, logo, or trademarks of HP to market the user’s software application product.
- Includes the copyright notice of HP Fortran on the user’s product disk label and/or on the title page of the documentation for software application product.

- Agrees to indemnify, hold harmless, and defend HP from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of the software application product.

Except as expressly provided herein, HP grants no implied or express license under any of its patents, copyrights, trade secrets, trademarks, or any license or other proprietary interests and rights.

For HP Tru64 UNIX Alpha systems, the RTL images are designated as:

- libfor.a, libfor.so
- libUfor.a, libUfor.so
- libFutil.a, libFutil.so
- libshpf.a, libshpf.so (HP Fortran 95/90 only)
- libhpfcmpi.a, libhpfcmpi.so (HP Fortran 95/90 only)
- libhpfsmpi.a, libhpfsmpi.so (HP Fortran 95/90 only)
- for_msg.cat
- libcxml_ev4.a, libcxml_ev4.so
- libcxml_ev5.a, libcxml_ev5.so
- libcxml_ev6.a, libcxml_ev6.so
- libcxmlp_ev4.a, libcxmlp_ev4.so (HP Fortran 95/90 only)
- libcxmlp_ev5.a, libcxmlp_ev5.so (HP Fortran 95/90 only)
- libcxmlp_ev6.a, libcxmlp_ev6.so (HP Fortran 95/90 only)

HARDWARE REQUIREMENTS

Processors Supported:

Any Alpha system that is capable of running HP Tru64 UNIX Version 4.0F or newer.

Disk Space Requirements:

Disk space required for HP Fortran installation:

Root file system:	/ 0 MB
Other file systems:	/usr 71 MB /tmp 1 MB /var 0 MB

Disk space required for HP Fortran use (permanent):

Root file system:	/0 MB
Other file systems:	/usr 67 MB /var 0 MB

To install the Compaq Extended Math Library (CXML), an additional 75 MB (or more) is needed in the /usr file system.

These sizes are approximate; actual sizes may vary depending on the user's system environment, configuration, and software options.

Memory Requirements:

For systems used to compile a program for parallel execution with the *-hpf* flag, a minimum of 64 MB of physical memory is recommended.

SOFTWARE REQUIREMENTS

- HP Tru64 UNIX Operating System Version 4.0F (SPD 41.61.xx) or higher
- HP Tru64 UNIX Developers' Toolkit Version 4.0F or higher (SPD 44.36.xx) for HP Tru64 UNIX Version 4.0F or higher

SOFTWARE LICENSING INFORMATION

This software is furnished only under license. For more information about licensing terms and policies of HP, contact your local HP office.

LICENSE MANAGEMENT FACILITY SUPPORT

HP Fortran supports the License Management Facility of HP.

License units for HP Fortran are allocated on an Unlimited System Use plus Concurrent Use basis.

Each Concurrent Use license allows any one individual at a time to use the layered product.

For more information on the License Management Facility, refer to the HP Tru64 UNIX Operating System Software Product Description (SPD 41.61.xx) or to the HP Tru64 UNIX Operating System documentation set.

OPTIONAL SOFTWARE

- Compaq MPI is required to link and execute High Performance Fortran programs compiled for parallel execution using the *-hpf* option, for shared-memory or Memory Channel Cluster target machines. For AlphaServer SC target machines, no additional software is required.

See *http://hp.com/techservers/software/cmpi.html*.

- Compaq Kap Fortran/OpenMP V4.4 for HP Tru64 UNIX (for HP Fortran 95/90 compiler; see SPD 45.72.13).

GROWTH CONSIDERATIONS

The minimum hardware/software requirements for any future version of this product may be different from the requirements for the current version.

DISTRIBUTION MEDIA

Media and documentation for HP Fortran are available on the HP Software Product Library CD-ROM set for HP Tru64 UNIX Products (QA-054AA-H8) or a CD-ROM containing only the HP Fortran for HP Tru64 UNIX Alpha Systems product (QA-MV2AA-H8). Documentation in printed format can be ordered separately (see the HP Fortran "read before installing" cover letter or the online release notes).

SOFTWARE WARRANTY

This software is provided by HP with a 90-day conformance warranty in accordance with the HP warranty terms applicable to the license purchase.

The above information is valid at time of release. Please contact your local HP office for the most up-to-date information.

ORDERING INFORMATION

Software Licenses:

Unlimited System Use: QL-MV2A*-AA
Personal Use: QL-MV2AM-2B
Concurrent Use: QL-MV2AM-3B
Concurrent 5 Pack: QL-MV2AM-3C
Concurrent 10 Pack: QL-MV2AM-3D

Software Documentation:

HP Fortran 95/90 Documentation: QA-MV2AA-GZ
HP Fortran 77 Documentation: QA-MV2AB-GZ

Software Product Services: QT-MV2*-**

* Denotes variant fields. For additional information on available licenses, services, and media, refer to the appropriate price book.

A variety of service options are available from HP. For more information, contact your local HP office.

COPYRIGHT INFORMATION