LA-UR-03-4510
Approved for public release;
distribution is unlimited.

Title: EOSPAC 5 User Manual

Author(s): David A. Pimentel

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the University of California for the U.S. Department of Energy under contract W-7405-ENG-36. By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.
Form 836 (8/00)

Table of Contents

EOSPAC 5 User Manual

1. Overview 1

2. EOSPAC Architecture 2

3. MIXPAC 4

4. EOSPAC Setup and Interpolation Examples 4

5. Miscellaneous Definitions 4
5.1. Error Message Flags and Values 4
5.2. EOS Interpolation Types 5
5.3. EOS Derivative Types 7
5.4. Sesame Table Descriptions 7
5.5. EOS Data Types 8
5.6. Header and Module File Descriptions 12
5.7. Valid Sesame Data File Names and Search Paths 14
5.8. Expected Endianism of Sesame Data Files 15
5.9. EOS Data Type Dependencies Found Within MIXPAC 15

6. EOSPAC Subroutines/Functions Accessible to Host Code 16
6.1. Level-1 Subroutines/Functions 16
6.1.1. Subroutine ES1ERRMSG (errorCode, errorMessage) 16
6.1.2. Void ES1INFO (itype, iregn, ktabs, iname, llogs, xcnvt, ycnvt, fcnvt, matid, znbar, anbar, dens0, ifile, ierr1) 16
6.1.3. Subroutine ES1NAME (tableID, tableName) 17
6.1.4. Void ES1TABS (llog1, IOPT, lprnt, iprnt, mtyps, mregs, inams, ucons, imids, idtab, mtabs, ktabs, ierrs) 17
6.1.5. Subroutine ES1VALS (ieosf, idrvs, intrp, ftabs, mtabs, nzons, iregn, xvals, yvals, fvals, nvalsi, ierr1) 19
6.1.6. Subroutine mapToF90Structures (ftabs, ftabsSize, miscData, basicTableDataArray, tableDataArray, mtyps, mregs) 20
6.1.7. Subroutine mapToFTABS (ftabs, ftabsSize, miscData, basicTableDataArray, tableDataArray, inams, mtyps, mregs) 20
6.2. Level-2 Subroutines/Functions 20
6.3. Level-3 Subroutines/Functions 21
6.4. Level-4 Subroutines/Functions 21
6.4.1. Void ES4ALLOC (mtabs, ktabs, ierr4) 21
6.4.2. Void ES4XPND (mdata, itabs, mtabs, ktabs, ierr4) 21
6.5. MIXPAC Subroutines/Functions 22
6.5.1. Subroutine MX1VALS (ieosf, idrvs, intrp, ftabs, mtabs, nzons, mtyps, mregs, nmixr, imixr, cmixr, iregs, xvals, yvals, fvals, nvalsi, ierr1) 22

7. References 23

Appendix A – ISPLT, ISMTH, IMONO and IDERIV as functions of IOPT

Appendix B – Header Files
Appendix C – C++ Example
Appendix D – F77 Example

EOSPAC 5 User Manual

1. Overview

The EOSPAC utility package is a collection of C++ and Fortran-90 interface routines, which can be used to access the Sesame data library. The Sesame data library contains thermodynamic (e.g., equation of state) and transport coefficients (e.g., opacity and conductivity), and it is described in detail in Ref. 1. Note, for simplicity, the term EOS (equation of state) here includes both thermodynamic variables and transport coefficients. The EOSPAC utility package is designed to be used by physics codes (henceforth "host codes") written in multiple languages and on multiple platforms. Each routine contains detailed internal documentation that complements this user manual.

The host code is assumed to deal with a physical problem consisting of discrete elements called zones, which are organized into groups called regions. The zones and regions must both be independently indexed 1,2,3.... Each region must be associated with a single material whose identification number corresponds to available Sesame data. Additionally, each region must be associated with all of the EOSPAC data types (section 5.5) required by the aforementioned physical problem. The host code must specify, for each zone, a region index (zero denotes a zone to be ignored), a density, and a temperature (or pressure or internal energy for some applications) during the interpolation phase.

Use of the EOSPAC utility package consists of two phases, a setup phase controlled by routine ES1TABS and an interpolation phase controlled by routine ES1VALS. The setup phase loads designated EOS data tables, and the interpolation phase computes requested EOS values within each zone.

ES1TABS is called by the host code during problem setup. The input and output quantities are described in the list of arguments for ES1TABS. Before calling ES1TABS, the host code must have created a pointer to an area for storing the EOS data tables. As needed, the internal dynamic memory allocation routine named ES4XPND uses the C functions named *calloc* and *realloc* to create or expand/contract respectively the aforementioned EOS storage area. ES4XPND can be replaced with another routine given that ES1TABS is passed the appropriate identifiers for the EOS table area.

ES1VALS is called by the host code during physics computation. The input and output quantities are described in the list of arguments for ES1VALS. Before calling ES1VALS, the host code must have created an area for storing the computed EOS values. This EOS value area must be large enough to accommodate the requested EOS values since no size expansion is done. The EOSPAC package performs an interpolation in the EOS table area to compute the EOS values. The EOSPAC package can also return gathered EOS data table values instead of interpolating.

Historically, a separate package was designed to calculate EOS for equilibrium mixtures of materials: MIXPAC. The MIXPAC routines have been included within the EOSPAC 5 architecture, and the driver routine, MX1VALS, can be used in lieu of the aforementioned interpolation phase (ES1VALS).

The remainder of this document describes the EOSPAC architecture and routine interfaces, the MIXPAC routine interface, the distributed examples, and the EOSPAC flags and types. Functions of IOPT and examples of header files, C++, and F77 are contained in the appendices. The underlined words/section numbers are references (links) to other parts of the EOSPAC 5 manual where more information can be found.

2. EOSPAC Architecture

The EOSPAC subroutines are organized into four levels distinguished by information they possess. Level-l subroutines are executive controllers called by the host code, which contain sufficient information to produce a specific implementation of EOSPAC. Level-2 subroutines are utilities, which possess the information necessary to access or modify the data in the EOS table storage area. Level-3 subroutines are utilities, which possess the information necessary to access the data in Sesame-formatted EOS data library files. Level-4 subroutines are utilities, which possess only the information necessary to perform certain primitive operations on specified data sets.

It is important to note that the EOSPAC interface library defines and uses some primitive data types within its C++ routines: INTEGER, REAL and BOOLEAN. Each of these primitives is defined within es4types.h described in section 5.6 Typically, this means that BOOLEANs and INTEGERs are defined to be 32-bit primitives, and REALs are defined to be 64-bit primitives. The Fortran routines use the Kind specifier to specify the bit sizes of primitive data types, and all references within this document to said primitives are assumed to be default size unless defined with a Kind specifier (see defineKinds module in section 5.6).

The host code must provide a pointer to a REAL array to serve as the EOS table storage area. In EOSPAC this array is named FTABS and has length NTABS. Before loading data into FTABS, the setup subroutines call subroutine ES4XPND (section 6.4.2) to check if there is sufficient space available and, if not, to expand the size of FTABS. Any other expansion method can be implemented by forcing the loading of a different ES4XPND, but the subroutine arguments and subroutine object name must conform to those of the ES4XPND provided within the EOSPAC library.

The EOSPAC setup subroutines load into FTABS three general blocks of data: 1) miscellaneous information, 2) data table specific information, and 3) data tables. The first such block stored in FTABS contains six elements of miscellaneous information:

initWord	=	“es2init", the indicator that FTABS is initialized,
nTypes	=	the number of EOS data types loaded,
nRegions	=	the number of regions in the problem,
infoSize	=	the number of information words in FTABS,
currentIndex	=	the index of the next available word in FTABS,
ntabs	=	the number of words in FTABS.

The second block stored in FTABS contains nTypes*nRegions fifteen-element blocks of data table specific information. Each block contains the following sequential information where $0 \le i < nTypes$ and $0 \le j < nRegions$:

functionData	=	pointer to global storage of Sesame data type information including the character string name of the ith data type,
materialID	=	the INTEGER identification value of the jth region's material,
aveAtomicNumber	=	the REAL value of the jth region's mean atomic number,
aveAtomicWgt	=	the REAL value of the jth region's mean atomic mass,
refDensity	=	the REAL value of the jth region's normal density,
logAxes	=	the BOOLEAN common logarithm indicator for the ith data type of region j,
xCnvrtFactor	=	the REAL unit conversion factor for the primary independent variable (mass density) for the ith data type of region j,
yCnvrtFactor	=	the REAL unit conversion factor for the secondary independent variable (temperature, specific pressure, or specific energy) for the ith data type of region j,
fCnvrtFactor	=	the REAL unit conversion factor for the dependent variable for the ith data type of region j,
fileId	=	the INTEGER index corresponding to the name of the data library file from which the jth region's data was loaded (section 5.7),
nxTable	=	the INTEGER number of primary independent variables of the jth region,
nyTable	=	the INTEGER number of secondary independent variables of the jth region,
xAxisIndex	=	the INTEGER index of the first primary independent variable of the jth region,
yAxisIndex	=	the INTEGER index of the first secondary independent variable of the jth region,
tableIndex	=	the INTEGER index of the last secondary independent variable of the jth region.

Thus, the number of information words in FTABS is

infoSize = (1 + 6*intSize/8) + (9 + 6*intSize/8)*nTypes*nRegions,

where "intSize" is the integer size in bytes for a given platform defined as follows:

Platform	intSize
Cray	8
Other	4

The third and final general block of data stored in FTABS contains the actual data tables. These data tables are organized as specified by the nxTable, nyTable, xAxisIndex, yAxisIndex, and tableIndex values stored in the second general data block described above.

EOSPAC can load data tables from user-generated, Sesame-formatted, private files as well as from the Sesame EOS data library files [Ref. 1].

The host code must provide an array to serve as the EOS value storage area, to which the EOSPAC interpolation subroutines return the computed EOS values. In EOSPAC this 2-D array

is named FVALS and has the dimensions NZONS and NVALS, where NZONS is the number of zones in the problem and NVALS is the number of EOS values computed per zone (section 6.1.5). The interpolation subroutines do not expand the size of FVALS, so the host code must provide sufficient space for the EOS values it requires.

The EOSPAC interpolation subroutines use a correlated value search algorithm [Ref. 2] to locate the data table values nearest to the values of the independent variables for the zones associated with a given region. The appropriate data table values are then gathered and used to interpolate the required EOS values for the zones. The interpolation algorithms available in EOSPAC are a 4-point bilinear, a 6-point biquadratic and a 12-point birational function [Ref. 3]. For host codes that do their own interpolation, EOSPAC can return the gathered data table values without interpolating.

3. MIXPAC

MIXPAC is a subroutine package for calculating EOS for equilibrium mixtures of materials [Ref. 4]. These subroutines have been integrated into the EOSPAC 5 architecture; therefore, the interface of the MIXPAC driver, MX1VALS (section 6.5.1), has been changed since the referenced documentation's [Ref. 4] publication. These interface changes are indicated in the parameter descriptions of section 6.5.1. Additionally, some data types supported by MIXPAC are dependent upon other data types being loaded by ES1TABS during setup. These dependencies are described in section 5.9.

4. EOSPAC Setup and Interpolation Examples

Simple examples of how to implement the EOSPAC setup and interpolation phases have been distributed with the object library. Links to copies of these examples are accessible in the following table:

Description	Source File	GNU Makefile(s)	Standard Output
C++ example	example.cxx	Makefile, .make_depends	example.stdout
F77 example	example.f	Makefile	example.stdout

5. Miscellaneous Definitions

Before going further with this discussion, some nomenclature must be described. Several definitions used by the EOSPAC internal database are addressed in this section: 1) Error Message Flags and Values, 2) EOS Interpolation Types, 3) EOS Derivative Types, 4) Sesame Table Descriptions, 5) EOS Data Types, 6) Header and Module File Descriptions, 7) Valid Sesame Data File Names and Search Paths, 8) Expected Endianism of Sesame Data Files, and 9) EOS Data Type Dependencies Found Within MIXPAC.

5.1. Error Message Flags and Values

The following is a list of possible error messages and corresponding error codes that EOSPAC 5 may return to the host code.

Name	Integer Value	Error Message
ES5_OK	0	No errors
ES5_NOTINIT	1	EOS table area is not initialized
ES5_BADTYPE	2	Table type index is out of bounds
ES5_BADREGION	3	Region index is out of bounds
ES5_NOTABLE	4	Data table is not in EOS table area
ES5_BADDERIVTYPE	5	Derivative is not recognized
ES5_BADINTRPTYPE	6	Interpolation is not recognized
ES5_MATIDZERO	7	Material ID is zero
ES5_BADFVALSDIM	8	Not enough space for fvals
ES5_BADTABLETYPE	9	Data table type is not recognized
ES5_NOFILESFOUND	10	No data library files exist
ES5_MATNOTFOUND	11	Material ID is not in library
ES5_TYPENOTFOUND	12	Data table type is not in library
ES5_BADFILEID	13	File index outside of range
ES5_EXPANDFAILED	14	EOS table area cannot be expanded
ES5_OPENFAILED	15	Could not open data file
ES5_GETNMATSFAILED	16	Could not read number of materials
ES5_GETVERSNFAILED	17	Could not load version from data file
ES5_LDMASTERFAILED	18	Could not load master directory
ES5_LDMATDIRFAILED	19	Could not load material directory
ES5_LDTABLEFAILED	20	Could not load data table
ES5_UNKNOWNHDFTAG	21	Unknown HDF tag
ES5_NOCOLDCURVE	22	The prerequisite cold curve was not loaded
ES5_CONVERGEFAILED	23	Iteration did not converge while performing the mixing algorithm

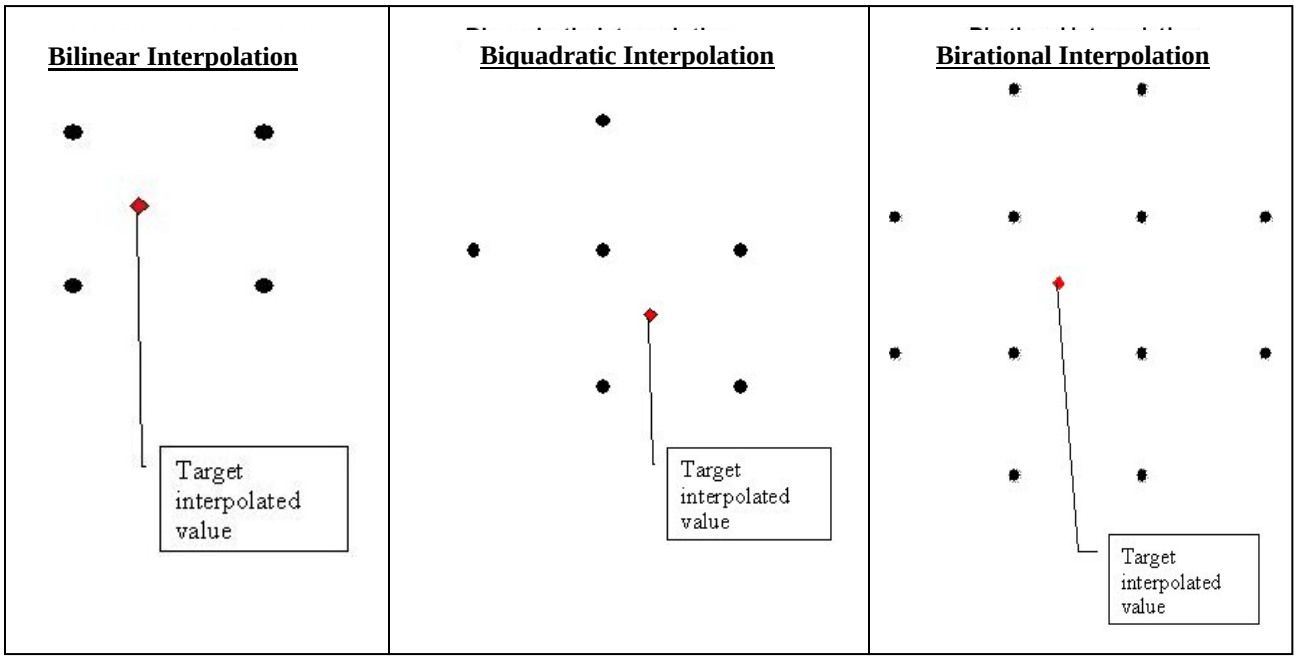
5.2. EOS Interpolation Types

The interpolation identifiers have been defined in terms of integer values as follows:

Label	Label's Integer Value	Description
ES4_BILINE	1	Bilinear (4-point) interpolation
ES4_BIQUAD	2	Biquadratic (6-point) interpolation
ES4_BIRATF	3	Birational (12-point) interpolation

The user must be aware of some obvious limitations of these interpolators. All of the interpolators use a bilinear method of extrapolation when calculating data outside the tabulated data ranges. The bilinear interpolator is only limited by the fact that it uses low-order finite-difference equations on a symmetric 4-point grid. The biquadratic interpolation uses an asymmetric 6-point grid that tends to be sensitive at the tabulated data range boundaries (specifically at low temperature and density). The birational interpolator uses a symmetric 12-point grid, and similar to the biquadratic interpolator, the birational interpolator has demonstrated sensitivities near the tabulated data range boundaries—especially in the partial

derivative values. The grids used by each of these interpolators are shown in the following figures:



5.3. EOS Derivative Types

The identifiers indicating what values are computed and returned have been defined in terms of integer values as follows:

Label	Label's Integer Value	Description
ES4_FUNC	1	Return EOS function f(x,y) only.
ES4_DERIVS	2	Return EOS function f(x,y) plus partial derivatives with respect to ln(x,y).
ES4_GATHER	3	Return the gathered EOS data table (x,y,f) values.

5.4. Sesame Table Descriptions

As described in the referenced documentation [Ref.1], the Sesame data library contains the following tables:

Sesame Table Number(s)	Description
101 - 199	Comments about the material
201	Mean Atomic Number, Mean Atomic Mass, Normal (solid) Density, Solid Bulk Modulus and Exchange Coefficient
301	Total EOS (304+305+306)
303	Ion EOS plus Cold Curve (305+306)
304	Electron EOS
305	Ion EOS (including zero point)
306	Cold Curve (no zero point)

Sesame Table Number(s)	Description
401	Vaporization Table
411	Solid Melt Table
412	Liquid Melt Table
431	Shear Modulus Table
501	Opacity Grid Boundary: Calculated vs. Interpolated
502	Rosseland Mean Opacity (cm^2 g^{-1})
503	Electron Conductive Opacity (cm^2 g^{-1}); Hubbard-Lampe Opacity Model
504	Mean Ion Charge (free electrons per atom); Hubbard-Lampe Opacity Model
505	Planck Mean Opacity (cm^2 g^{-1})
601	Mean Ion Charge (free electrons per atom); Ziman Conductivity Model
602	Electrical Conductivity (sec^{-1})
603	Thermal Conductivity (cm^{-1} sec^{-1})
604	Thermoelectric Coefficient (cm^{-1} sec^{-1})
605	Electron Conductive Opacity (cm^2 g^{-1}); Ziman Conductivity Model

5.5. EOS Data Types

Once again, some nomenclature must be described. In the earliest versions of EOSPAC, three data categories [Ref. 5] were defined: 1) Thermodynamic Properties, 2) Photon Transport Coefficients, and 3) Electron Transport Coefficients. Like EOSPAC Version 3, EOSPAC Version 5 makes reference to the original three categories and a fourth that accesses the 400-series Sesame catalogs. Additionally, due to a user request, an additional data type has been implemented within EOSPAC to return the pressure- and temperature-based total density: ES4_DPTTOT.

The following is a list of EOS data types accessible through EOSPAC 5:

Label	Label's Integer Value	Table Name	Sesame Table Number	Table Category	Description
ES4_NULLPTR	0	---	---	---	Null table entry
ES4_PRTOT	1	prtot	301	1	Temperature-based total pressure
ES4_ENTOT	2	entot	301	1	Temperature-based total internal energy/mass
ES4_TPTOT	3	tptot	301	1	Pressure-based total temperature
ES4_TNTOT	4	tntot	301	1	Energy-based total temperature
ES4_PNTOT	5	pntot	301	1	Energy-based total pressure
ES4_EPTOT	6	eptot	301	1	Pressure-based total internal energy/mass

Label	Label's Integer Value	Table Name	Sesame Table Number	Table Category	Description
ES4_PRION	7	prion	303	1	Temperature-based ion pressure
ES4_ENION	8	enion	303	1	Temperature-based ion internal energy/mass
ES4_TPION	9	tpion	303	1	Pressure-based ion temperature
ES4_TNION	10	tnion	303	1	Energy-based ion temperature
ES4_PNION	11	pnion	303	1	Energy-based ion pressure
ES4_EPION	12	epion	303	1	Pressure-based ion internal energy/mass
ES4_PRELC	13	prelc	304	1	Temperature-based electron pressure
ES4_ENELC	14	enelc	304	1	Temperature-based electron internal energy/mass
ES4_TPELC	15	tpelc	304	1	Pressure-based electron temperature
ES4_TNELC	16	tnelc	304	1	Energy-based electron temperature
ES4_PNELC	17	pnelc	304	1	Energy-based electron pressure
ES4_EPELC	18	epelc	304	1	Pressure-based electron internal energy/mass
ES4_PRCLD	19	prcld	306	1	Temperature-based cold curve pressure
ES4_ENCLD	20	encld	306	1	Temperature-based cold curve internal energy/mass
ES4_OPACR	21	opacr	502	2	Temperature-based Rosseland mean opacity
ES4_OPACC2	22	opacc2	503	2	Temperature-based electron conductive opacity
ES4_ZFREE2	23	zfree2	504	2	Temperature-based number of free electrons per ion
ES4_OPACP	24	opacp	505	2	Temperature-based Planck mean opacity
ES4_ZFREE3	25	zfree3	601	3	Temperature-based number of free electrons per ion
ES4_ECONDE	26	econde	602	3	Temperature-based electron electrical conductivity
ES4_TCONDE	27	tconde	603	3	Temperature-based electron thermal conductivity
ES4_THERME	28	therme	604	3	Temperature-based electron thermoelectric coefficient

Label	Label's Integer Value	Table Name	Sesame Table Number	Table Category	Description
ES4_OPACC3	29	opacc3	605	3	Temperature-based electron conductive opacity
ES4_TMELT	30	tmelt	411	4	Temperature-based melting temperature
ES4_PMELT	31	pmelt	411	4	Temperature-based melting pressure
ES4_EMELT	32	emelt	411	4	Temperature-based melting internal energy/mass
ES4_TFREEZ	33	tfreez	412	4	Temperature-based freezing temperature
ES4_PFREEZ	34	pfreez	412	4	Temperature-based freezing pressure
ES4_EFREEZ	35	efreez	412	4	Temperature-based freezing internal energy/mass
ES4_SHEARM	36	shearm	431	4	Temperature-based shear modulus
ES4_DPTTOT	37	dpttot	301	1	Pressure- and temperature-based total density

Category 2, 3 and 4 data types are loaded with logarithmic data by default. Category 1 data types may be loaded with logarithmic data by setting the llog1 parameter of ES1TABS, described in section 6.1.4, equal to TRUE.

Some terms are defined below:

1. An "inverted" table is one that is loaded by EOSPAC from a Sesame file, and then an independent variable is transposed with the dependent variable; i.e., ES4_PRTOT, P(rho,T) is loaded and then ES4_TPTOT, T(rho,P) is produced by inverting P with T.

2. A "merged" table is one that requires the inversion of one table to be merged with another.

Here is a complete list of table dependencies for "inverted" and "merged" tables. The "Required Cold Curve" is listed, because ES1VALS requires it to perform the necessary transformations of the user-supplied independent variable values prior to interpolating. The "logs?" column indicates if the value of the first ES1TAB parameter (llog1) is TRUE or FALSE, and it affects which cold curve is required by ES1VALS. Note that cold curve dependencies also exist for some "non-inverted" tables when logs is TRUE. Please remember that the llog1 parameter is an ordered array of BOOLEAN that contains values corresponding to the order of inams array (seventh parameter of ES1TABS.

User-Requested Table	logs?	Required Cold Curve
ES4_TPTOT	FALSE	ES4_PRCLD
ES4_EPTOT	FALSE	ES4_PRCLD
ES4_TPION	FALSE	ES4_PRCLD
ES4_EPION	FALSE	ES4_PRCLD
ES4_TNTOT	FALSE	ES4_ENCLD
ES4_PNTOT	FALSE	ES4_ENCLD
ES4_TNION	FALSE	ES4_ENCLD
ES4_PNION	FALSE	ES4_ENCLD
ES4_PRTOT	TRUE	ES4_PRCLD
ES4_PNTOT	TRUE	ES4_PRCLD and ES4_ENCLD
ES4_PRION	TRUE	ES4_PRCLD
ES4_PNION	TRUE	ES4_PRCLD and ES4_ENCLD
ES4_ENTOT	TRUE	ES4_ENCLD
ES4_EPTOT	TRUE	ES4_ENCLD
ES4_ENION	TRUE	ES4_ENCLD
ES4_EPION	TRUE	ES4_ENCLD

What this means to you, as a user, is ES1TABS and ES1VALS to date have always required you to request the cold curve tables for "inverted" and "merged" tables respectively. There are also some special cases when logs is TRUE.

Some common unit conversions for table categories 1 and 4 are listed below. The Original Units are actually stored within the Sesame data files, and the Converted Units are examples of units required by the host code:

Property	Original Units		Converted Units
Density	Mg/m^3	=	g/cm^3 * 1.0000E-00
Temperature	Kelvin	=	Kev * 8.6171E-08
Pressure	GJ/m^3	=	JK/cm^3 * 1.0000E-06
Internal Energy/Mass	GJ/Mg	=	JK/g * 1.0000E-06
Shear Modulus	GJ/m^3	=	JK/cm^3 * 1.0000E-06

Likewise, some common unit conversions for table categories 2 and 3 are listed below:

Property	Original Units		Converted Units
Density	g/cm^3	=	g/cm^3 * 1.0000E-00
Temperature	Ev	=	Kev * 1.0000E-03
Opacity	cm^2/g	=	cm^2/g * 1.0000E-00
Conductive Opacity	cm^2/g	=	cm^2/g * 1.0000E-00
Electrical Conductivity	1/s	=	1/sh * 1.0000E-08
Thermal Conductivity	1/s/cm	=	1/sh/cm * 1.0000E-08
Thermo-Electric Coefficient	1/s/cm	=	1/sh/cm * 1.0000E-08

5.6. Header and Module File Descriptions

The identifiers indicating what values are computed and returned have been defined in terms of integer values in the following table.

File Name	Description
eospac4.h	Fortran 90 header file for named parameter constants used to control the EOSPAC library via the calls to ES1TABS, ES1VALS and MX1VALS (see sections 5.2, 5.3 and 5.5).
eospac4c.h	C/C++ header file for named static constants used to control the EOSPAC library via the calls to ES1TABS, ES1VALS and MX1VALS (see sections 5.2, 5.3 and 5.5).
eospac4f77.h	Fortran 77 header file for named parameter constants used to control the EOSPAC library via the calls to ES1TABS, ES1VALS and MX1VALS (see sections 5.2, 5.3 and 5.5).
es4proto.h	C++ header file containing function prototypes of callable external entry points and internal functions. Includes wrapperDefines.h.
es4types.h	C/C++ header file for types used by the EOSPAC library. Includes es5err.h, <stdio.h>, <stdlib.h>, <string.h>, and <math.h>.
es5err.h	C/C++ header file containing static constant error flags for EOSPAC (see section 5.1).
es5errf.h	Fortran header file containing parameter constant error flags for EOSPAC (see section 5.1).
wrapperDefines.h	Header file containing preprocessor macros used to simplify cross-platform C-Fortran interfaces to the external entry points defined in es4proto.h.

For all of you die-hard Fortran-90 developers, EOSPAC contains some Fortran-90 modules in an attempt to simplify data storage. The modules are listed in the following table:

Module Name	Description
defineKinds	Fortran-90 Kind definitions used by other EOSPAC modules.
miscInfo	Module encapsulating the defineKinds module and the es5errf.h and eospac4.h header files.
dataInterface	The dataInterface module can be used by the host code to translate the loaded data tables and related information into Fortran-90 derived types.

The defineKinds module is defined as follows:

Variable Type and Name	Value
integer, parameter :: i_kind	SELECTED_INT_KIND(r=8)
integer, parameter :: i4kind	SELECTED_INT_KIND(r=8)
integer, parameter :: i8kind	SELECTED_INT_KIND(r=16)
integer, parameter :: i8kind	SELECTED_INT_KIND(r=16)
integer, parameter :: r8kind	SELECTED_REAL_KIND(p=12,r=250)
integer(i_kind), parameter :: idum	1_i_kind
integer(i_kind), parameter :: intSize	BIT_SIZE(idum)/8_i_kind

The miscInfo module uses the defineKinds module and it includes the es5errf.h and eospac4.h header files.

The dataInterface module uses the miscInfo module, and its derived TYPE definitions are as follows:

Variable Type and Name	Description
type, public :: eosMiscData	Derived type container to store Miscellaneous Data
sequence	Strict memory ordering enforced
character(len=8) initWord	String to show that memory block is initialized
integer(i_kind) nTypes	Number of table types actually used
integer(i_kind) nRegions	Number of regions of different materials
integer(i_kind) infoSize	Size of the header block in words
integer(i_kind) currentIndex	Current word index into memory block
integer(i_kind) maxIndex	Size in words of memory block
end type eosMiscData	

Variable Type and Name	Description
type, public :: eosBasicTableData	Derived type container to store basic information for each loaded data table
sequence	Strict memory ordering enforced
integer(i_kind) materialID	Sesame material ID number
real(r8kind) aveAtomicNumber	Material's atomic number
real(r8kind) aveAtomicWgt	Material's atomic mass
real(r8kind) refDensity	Material's reference density
logical(i_kind) logAxes	TRUE indicates the data table contains common logs
real(r8kind) xCnvrtFactor	X conversion factor
real(r8kind) yCnvrtFactor	Y conversion factor
real(r8kind) fCnvrtFactor	F conversion factor
integer(i_kind) fileId	Integer index of Sesame file
integer(i_kind) nxTable	Number of X values
integer(i_kind) nyTable	Number of Y values
integer(i_kind) xAxisIndex	Index of zeroth X value
integer(i_kind) yAxisIndex	Index of zeroth Y value
integer(i_kind) tableIndex	Index of last Y value

Variable Type and Name	Description
end type eosBasicTableData	

Variable Type and Name	Description
type, public :: eosTableData	Derived type container to store a data table
sequence	Strict memory ordering enforced
integer(i_kind) nx	Number of X data elements
integer(i_kind) ny	Number of Y data elements
real(r8kind) , pointer :: x(:)	Pointer to X data array
real(r8kind) , pointer :: y(:)	Pointer to Y data array
real(r8kind) , pointer :: F(:,:)	Pointer to F data array
end type eosTableData	

The dataInterface module also contains some subroutines to translate the data to and from Fortran-90 derived types described above: mapToF90Structures and mapToFTABS. These subroutines are described in sections 6.1.6 and 6.1.7 respectively.

5.7. Valid Sesame Data File Names and Search Paths

Historically, EOSPAC has used a predefined ordered list of file names in which it searches for requested data. This feature has been maintained to date, and the ordered list and corresponding fileId values are given in the following table:

fileId	File Name	fileId	File Name	fileId	File Name	fileId	File Name
1	sesameu	12	sesame1	23	sesou2	34	sescu3
2	sesameu1	13	sesame2	24	sesou3	35	sescu4
3	sesameu2	14	sesame3	25	sesou4	36	sescu9
4	sesameu3	15	sesame4	26	sesop	37	sescp
5	sesameu4	16	sesep	27	sesop1	38	sescp1
6	sesamea	17	sesep1	28	sesop2	39	sescp2
7	sesamea1	18	sesep2	29	sesop3	40	sescp3
8	sesamea2	19	sesep3	30	sesop4	41	sescp4
9	sesameb	20	sesep4	31	sescu		
10	sesamec	21	sesou	32	sescu1		
11	sesame	22	sesou1	33	sescu2		

EOSPAC searches for files with these names in the following ordered list of directories:

1. Current working directory
2. Unclassified data directory
3. Classified data directory

The unclassified data directory is one of the following by default:

/usr/projects/data/eos
/usr/local/codes/data/eos
/usr/local/udata/ses

The classified data directory is one of the following by default:

/usr/projects/data/eos
/usr/local/codes/data/eos
/usr/local/cdata

In lieu of the default search paths, EOSPAC uses the environment variables named SESPATHU and SESPATHC to define the unclassified and classified data directories respectively.

5.8. Expected Endianism of Sesame Data Files

EOSPAC was originally developed on the Cray platform, and the Sesame data files predate its development by many years. Consequently, the endianism (i.e., floating point number byte order) of the Sesame data files was historically Big Endian, and it continues to be Big Endian on the IEEE platforms. Therefore, EOSPAC expects the Sesame data files to contain Big Endian floating point data, and it performs appropriate dynamic byte swapping for a machine's native endianism.

5.9. EOS Data Type Dependencies Found Within MIXPAC

The MIXPAC routine named MX1VALS has an added requirement in that for some data types it needs other specific data to be loaded by ES1TABS; therefore the host code is required to request the loading of this additional data. These required data are in fact balance functions used to provide mixed data that is in temperature equilibrium as well as pressure equilibrium. These data type requirements are cross referenced in the following table:

Requested Data Type	Balance Function Data Type Required to be Loaded by ES1TABS
ES4_PNTOT	ES4_TNTOT
ES4_EPTOT	ES4_TPTOT
ES4_PNION	ES4_TNION
ES4_EPION	ES4_TPION
ES4_PNELC	ES4_TNELC
ES4_EPELC	ES4_TPELC

6. EOSPAC Subroutines/Functions Accessible to Host Code

The EOSPAC subroutines are organized into four levels distinguished by the information they possess. Level-l subroutines are executive controllers called by the host code with sufficient information to produce a specific implementation of EOSPAC. Level-2 subroutines are utilities, which possess the information necessary to access or modify the data in the EOS table storage area. Level-3 subroutines are utilities, which possess the information necessary to access the data in Sesame-formatted EOS data library files. Level-4 subroutines are utilities, which possess only the information necessary to perform certain primitive operations on specified data sets. The subroutine names have the form ESlxxxx, where ES indicates package membership, l=1,2,3,4 indicates the level of the subroutine, and xxxx is a mnemonic (not limited to four characters) indicating the operation performed. The level-1,2,3,4 subroutines are described in the following subsections, and some historical notes are shown in red to indicate interface changes since the original Cray versions of EOSPAC (i.e., EOSPAC 3).

6.1. Level-1 Subroutines/Functions

6.1.1. Subroutine ES1ERRMSG (errorCode, errorMessage)

Purpose: This routine takes an error number and returns a string with the corresponding error message.

Language: Fortran-90

Comments: This routine was added for the convenience of the host code programmer.

Refer to the es4proto.h and wrapperDefines.h header files described in section 5.6 for the appropriate routine name when called from a C/C++ host code.

Parameters:

- errorCode — Input integer error number.
- errorMessage — Output corresponding character*60 error message. The returned string value contains a "\0" (null) character for C/C++ string compatibility.

6.1.2. Void ES1INFO (itype, iregn, ktabs, iname, llogs, xcnvt, ycnvt, fcnvt, matid, znbar, anbar, dens0, ifile, ierr1)

Purpose: Get information about the EOS data table.

Language: C++

Comments: This function serves as a wrapper to the Level 2 routine named es2info, which is not accessible by the host code.
Refer to the es4proto.h and wrapperDefines.h header files described in section 5.6 for the appropriate routine name when called from a C/C++ host code.

Parameters:

- itype — Input INTEGER index of data table type.
- iregn — Input INTEGER index of region.
- ktabs — Input INTEGER pointer to EOS table area.
- iname — Output INTEGER offset into list of character names of data types.
- llogs — Output BOOLEAN log indicator for data table type.

xcnvt	Output REAL unit conversion factor for x-values.
ycnvt	Output REAL unit conversion factor for y-values.
fcnvt	Output REAL unit conversion factor for f-values.
matid	Output INTEGER material ID number for data table.
znbar	Output REAL mean atomic number for data table.
anbar	Output REAL mean atomic mass for data table.
dens0	Output REAL normal density for data table.
ifile	Output INTEGER offset into array of valid data library file names from which data table was loaded.
ierr1	Output INTEGER error indicator containing the following possible values: ES5_OK, ES5_NOTINIT, ES5_BADTYPE, or ES5_BADREGION.

6.1.3. Subroutine ES1NAME (tableID, tableName)

Purpose: Get the table name corresponding to a given integer ID.

Language: Fortran-90

Comments: This routine was added for the convenience of the host code programmer.

Refer to the es4proto.h and wrapperDefines.h header files described in section 5.6 for the appropriate routine name when called from a C/C++ host code.

Parameters:

tableID	Input the integer ID.
tableName	Output the corresponding table name padded to character*8 (see "Data Types" in section 5.5). Be certain the returned string value does NOT contain a "\0" (null) character for C/C++ string compatibility.

6.1.4. Void ES1TABS (llog1, IOPT, lprnt, iprnt, mtyps, mregs, inams, ucons, imids, idtab, mtabs, ktabs, ierrs)

Purpose: Controls EOSPAC utility package setup phase to set up and print designated EOS data tables.

Language: C++

Comments: Refer to the es4proto.h and wrapperDefines.h header files described in section 5.6 for the appropriate routine name when called from a C/C++ host code.

Parameters:

llog1 Input BOOLEAN array–each element is true if it corresponds to a Sesame category-1 table that the user wishes to contain common logs. This was a scalar logical value in Ref. 5.

Cold curve data never contains common logarithmic values; therefore, ES1TABS overrides llog1 with a false value if the host code requests a cold curve to contain common log data.

This is a 1-D array of length (mtyps*mregs), where each element corresponds to the ith data type and j^{th} region. The array is organized as follows: {[llog1(i+(j-1)*mtyps): i=1 to mtyps], j=1 to mregs}.

iopt	Input integer array of various options for Sesame data types. It additionally may contain values allowing the output of backward- and forward-finite-difference output for selected data types/regions and the forced-monotonicity of f with respect to x and/or y. See Appendix A for a summary of valid values and how each value is overloaded (i.e., how it is translated into other variable values). This was a scalar value named isplt in Ref. 5. This is a 1-D array of length (mtyps*mregs), where each element corresponds to the ith data type and j^{th} region. The array is organized as follows: {[iopt(i+(j-1)*mtyps): i=1 to mtyps], j=1 to mregs}.
lprnt	Input BOOLEAN true if tables are to be printed.
iprnt	Input INTEGER flag indicates overwrite or append data to EOS print file named "tablesloaded.dat". If iprnt=0, then "tablesloaded.dat" is overwritten; otherwise data is appended to "tablesloaded.dat." This was the Fortran unit number or name of the EOS print file in Ref. 5.
mtyps	Input INTEGER number of Sesame data types used.
mregs	Input INTEGER number of regions in problem.
inams	Input INTEGER array containing values corresponding to the names of Sesame data types used (section 5.5). Values no longer Hollerith as in Ref. 5. This is a 1-D array of length (mtyps), where each element corresponds to the ith data type.
ucons	Input REAL array containing unit conversion factors for all Sesame data types for each table variable (x,y,f). This is a 1-D array of length (mtyps*3), where each element corresponds to the ith data type and j^{th} variable x, y and f, respectively. The array is organized as follows: {[ucons(i+(j-1)*mtyps): i=1 to mtyps], j=1 to 3}.
imids	Input INTEGER array containing material ID numbers for all Sesame data types for each region. This is a 1-D array of length (mtyps*mregs), where each element corresponds to the ith data type and j^{th} region. The array is organized as follows: {[imids(i+(i+(j-1)*mtyps): i=1 to mtyps], j=1 to mregs}.
idtab	Input INTEGER containing identifier for EOS table area for use in user-supplied memory expansion routine. If the default routine is used, this argument is ignored.
mtabs	Input/Output INTEGER length of EOS table area.
ktabs	Input/Output Cray or C Pointer to EOS table area (REAL FTABS array previously described in section 2).
ierrs	Output INTEGER array containing error indicators for all Sesame data types for each region. This is a 1-D array of length (mtyps*mregs), where each element corresponds to the ith data type and j^{th} region. The array is organized as follows: {[ierrs(i+(j-1)*mtyps): i=1 to mtyps], j=1 to mregs}. Each element may contain one of the following possible values: ES5_OK, ES5_NOTINIT, ES5_BADTYPE, ES5_BADREGION, ES5_BADTABLETYPE, ES5_NOFILESFOUND, ES5_MATNOTFOUND, ES5_TYPENOTFOUND, or ES5_EXPANDFAILED.

6.1.5. Subroutine ES1VALS (ieosf, idrvs, intrp, ftabs, mtabs, nzons, iregn, xvals, yvals, fvals, nvalsi, ierr1)

Purpose: Controls EOSPAC utility package interpolation phase to compute values of EOS function f(x,y) and derivatives with respect to ln(x,y) or to gather EOS data table (x,y,f) values.

Language: Fortran-90

Comments: If the derivatives of f(x,y) with respect to ln(x,y) are returned, then the following formulae must be used by the host code to calculate the derivatives of f(x,y) with respect to (x,y):

$$\frac{\partial f(x,y)}{\partial x} = \frac{1}{x}\frac{\partial f(x,y)}{\partial(\ln x)} \quad \text{and/or} \quad \frac{\partial f(x,y)}{\partial y} = \frac{1}{y}\frac{\partial f(x,y)}{\partial(\ln y)}$$

Refer to the es4proto.h and wrapperDefines.h header files described in section 5.6 for the appropriate routine name when called from a C/C++ host code.

Parameters:

- ieosf — Input the integer(i_kind) corresponding to the chosen data type (section 5.1). Value no longer Hollerith as in Ref. 5.
- idrvs — Input the integer(i_kind) corresponding to the chosen derivative type (section 5.3). Value no longer Hollerith as in Ref. 5.
- intrp — Input the integer(i_kind) corresponding to the chosen interpolation type (section 5.2). Value no longer Hollerith as in Ref. 5.
- ftabs — Input the real(r8kind) EOS table area array. This was a pointer named ktabs in Ref. 5.
- mtabs — Input the integer(i_kind) size of the ftabs array. Not used in Ref. 5.
- nzons — Input the integer(i_kind) number of zones being processed.
- iregn — Input the integer(i_kind) scalar region to look up. This was an integer array in Ref. 5.
- xvals — Input the real(r8kind) array of nzons x-values.
- yvals — Input the real(r8kind) array of nzons y-values.
- fvals — Output the real(r8kind) EOS value area may contain the following:
  if idrvs=ES4_FUNC, then
  fvals(:,1) = nzons values of eos function f(x,y) .
  if idrvs= ES4_DERIVS, then
  fvals(:,1) = nzons values of eos function f(x,y) ,
  fvals(:,2) = nzons values of derivative of f with respect to ln(x),
  fvals(:,3) = nzons values of derivative of f with respect to ln(y).
  if idrvs=ES4_GATHER, then fvals contains nzons*nvals gathered eos data table x,y,f values, where
  if intrp = ES4_BILINE, then nvals=8,
  if intrp = ES4_BIQUAD, then nvals=12,
  if intrp = ES4_BIRATF, then nvals=20.
- nvalsi — Input integer(i_kind) length(fvals)/nzons. Not used in Ref. 5.

ierr1 Output integer(i_kind) error indicator containing one of the following possible values: ES5_OK, ES5_BADFVALSDIM, ES5_NOTINIT, ES5_BADTYPE, ES5_BADREGION, ES5_NOTABLE, ES5_BADDERIVTYPE, ES5_BADINTRPTYPE, ES5_MATIDZERO, or ES5_BADTABLETYPE.

6.1.6. Subroutine mapToF90Structures (ftabs, ftabsSize, miscData, basicTableDataArray, tableDataArray, mtyps, mregs)

Purpose: The purpose of mapToF90Structures is to transfer the 1-D array (ftabs) data that is returned from ES1TABS into Fortran-90 derived type object arrays.
Language: Fortran-90
Comments: This subroutine is only valid for a Fortran-90 (or later) host code.
Parameters:

ftabs	Input real(r8kind) EOS table area.
ftabsSize	Input integer(i_kind) size of ftabs.
miscData	Output eosMiscData derived type object.
basicTableDataArray	Output array containing (mtyps*mregs) eosBasicTableData derived type objects.
tableDataArray	Output array containing (mtyps*mregs) eosTableData derived type objects.
mtyps	Input integer(i_kind) number of sesame table types to transfer.
mregs	Input integer(i_kind) number of regions to transfer.

6.1.7. Subroutine mapToFTABS (ftabs, ftabsSize, miscData, basicTableDataArray, tableDataArray, inams, mtyps, mregs)

Purpose: The purpose of mapToF90Structures is to transfer the Fortran-90 derived type object array data into the 1-D array (ftabs).
Language: Fortran-90
Comments: This subroutine is only valid for a Fortran-90 (or later) host code.
Parameters:

ftabs	Output real(r8kind) EOS table area.
ftabsSize	Input integer(i_kind) size of ftabs.
miscData	Input eosMiscData derived type object.
basicTableDataArray	Input array containing (mtyps*mregs) eosBasicTableData derived type objects.
tableDataArray	Input array containing (mtyps*mregs) eosTableData derived type objects.
inams	Input INTEGER array containing values corresponding to ES1TABS the names of Sesame data types used (section 5.5). This is a 1-D array of length (mtyps), where each element corresponds to the ith data type. This array must be identical to that passed to ES1TABS.
mtyps	Input integer(i_kind) number of sesame table types to transfer.
mregs	Input integer(i_kind) number of regions to transfer.

6.2. Level-2 Subroutines/Functions

These routines are not described in this document since they are used by EOSPAC internally. Historically, EOSPAC allowed developers to create their own driver routines to control EOSPAC's lower level routines; however, this is no longer recommended or feasible due to significant architectural changes.

6.3. Level-3 Subroutines/Functions

All of these subroutines and functions have been converted to C++ and hidden from a Fortran-90 host code since they have not been included in an extern "C" function wrapper. Therefore, they will not be discussed here since they are not available to all host codes.

6.4. Level-4 Subroutines/Functions

6.4.1. Void ES4ALLOC (mtabs, ktabs, ierr4)

Purpose: Using the C routine name calloc, allocate memory to the EOS table area (FTABS). This is provided primarily for Fortran host code developers who do not wish to use the Fortran extension routine named malloc.

Language: C++

Comments: Refer to the es4proto.h and wrapperDefines.h header files described in section 5.6 for the appropriate routine name when called from a C/C++ host code.

Parameters:

- mtabs — Input INTEGER length of EOS table area.
- ktabs — Input/Output INTEGER pointer to EOS table area.
- ierr4 — Output INTEGER error indicator containing the following possible values: ES5_OK, and ES5_EXPANDFAILED.

6.4.2. Void ES4XPND (mdata, itabs, mtabs, ktabs, ierr4)

Purpose: Expand/Shrink EOS table area to hold additional/less data.

Language: C++

Comments: If the host code programmer wishes to do so, the expansion routine ES4XPND can be replaced with another one that also passes ES1TABS the appropriate identifiers for the EOS table area.
Refer to the es4proto.h and wrapperDefines.h header files described in section 5.6 for the appropriate routine name when called from a C/C++ host code.

Parameters:

- mdata — Input INTEGER number of words of additional data.
- idtab — This is an obsolete parameter in the current version of this routine. It is only mentioned here because it is described in Ref. 5.
- itabs — Input INTEGER of the current index of EOS table area (zero based).
- mtabs — Input/Output INTEGER length of the EOS table area.
- ktabs — Input/Output INTEGER pointer to the EOS table area.
- ierr4 — Output INTEGER error indicator containing the following possible values: ES5_OK, and ES5_EXPANDFAILED.

6.5. MIXPAC Subroutines/Functions

6.5.1. Subroutine MX1VALS (ieosf, idrvs, intrp, ftabs, mtabs, nzons, mtyps, mregs, nmixr, imixr, cmixr, iregs, xvals, yvals, fvals, nvalsi, ierr1)

Purpose: Control MIXPAC package to compute values of EOS function f(x,y) and derivatives with respect to ln(x,y) for equilibrium mixture of materials.

Language: Fortran-90

Comments: Refer to the es4proto.h and wrapperDefines.h header files described in section 5.6 for the appropriate routine name when called from a C/C++ host code.

Parameters:

- ieosf — Input the integer(i_kind) corresponding to the chosen data type (section 5.1). Value no longer Hollerith as in Ref. 4.
- idrvs — Input the integer(i_kind) corresponding to the chosen derivative type (section 5.3). Value no longer Hollerith as in Ref. 4.
- intrp — Input the integer(i_kind) corresponding to the chosen interpolation type (section 5.2). Value no longer Hollerith as in Ref. 4.
- ftabs — Input the real(r8kind) EOS table area array. This was a Cray pointer named ktabs as specified in Ref. 4.
- mtabs — Input the integer(i_kind) size of the ftabs array. Not used in Ref. 4.
- nzons — Input the integer(i_kind) number of zones being processed.
- mtyps — Input integer(i_kind) number of Sesame data types used. Not used in Ref. 4.
- mregs — Input integer(i_kind) number of regions in problem. Not used in Ref. 4.
- nmixr — Input integer(i_kind) maximum number of regions to be mixed.
- imixr — Input integer(i_kind) array containing indices of all regions to be mixed for each region. The array dimension is (nmixr,mregs), where mregs is the total number of regions defined by the host code–see ES1TABS specification. Given i varies fastest, this array is organized contiguously in memory as follows:
  {[imixr(i)(j): i=1 to nmixr], j=1 to mregs}.
- cmixr — Input real(r8kind) array containing concentrations in all zones for each region to be mixed. This array dimension is (nzons,nmixr). Given i varies fastest, this array is organized contiguously in memory as follows:
  {[cmixr(i)(j): i=1 to nzons], j=1 to nmixr}.
- iregs — Input integer(i_kind) array containing index of region for each zone.
- xvals — Input real(r8kind) array containing x-value for each zone.
- yvals — Input real(r8kind) array containing y-value for each zone.
- fvals — Output the real(r8kind) EOS value area may contain the following:
  if idrvs=ES4_FUNC, then
  fvals(:,1) = nzons values of eos function f(x,y) .
  if idrvs= ES4_DERIVS, then
  fvals(:,1) = nzons values of eos function f(x,y) ,
  fvals(:,2) = nzons values of derivative of f with respect to ln(x),
  fvals(:,3) = nzons values of derivative of f with respect to ln(y).
- nvalsi — Input integer(i_kind) length(fvals)/nzons. Not used in Ref. 4.

ierr1 Output integer(i_kind) error indicator containing the following possible values: ES5_OK, ES5_NOTINIT, ES5_BADREGION, ES5_NOTABLE, ES5_BADDERIVTYPE, ES5_BADINTRPTYPE, ES5_BADTABLETYPE, and ES5_CONVERGEFAILED.

7. References

1. S. P. Lyon and J. D. Johnson, "Sesame: The Los Alamos National Laboratory Equation of State Database," Los Alamos National Laboratory report LA-UR-92-3407 (1992).
2. W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes in Fortran 77 Second Edition: The Art of Scientific Computing*, Cambridge University Press, Cambridge, Massachusetts (1999).
3. G. I. Kerley, "Rational Function Method Of Interpolation," Los Alamos Scientific Laboratory report LA-6903-MS (August 1977).
4. C. W. Cranfill, "MIXPAC: A Subroutine Package For Calculating Equations Of State For Equilibrium Mixtures Of Materials," Los Alamos National Laboratory report LA-9861-M (August 1983).
5. C. W. Cranfill, "EOSPAC: A Subroutine Package For Accessing The Los Alamos Sesame EOS Data Library," Los Alamos National Laboratory report LA-9728-M (August 1983).

Appendix A

ISPLT, ISMTH, IMONO and IDERIV as functions of IOPT

The IOPT parameter is an array of integer values that specify the increase of a data table's grid resolution, smoothing and splitting options for Sesame category-1 data types, the output of backward- and forward-finite-difference output for all data types, and the forced-monotonicity of f(x,y) with respect to x and/or y. Each element of the IOPT array corresponds to a given data type and region, and it is assigned a value by the host code where each digit corresponds to a desired option flag:

Define the digits of IOPT = EDCBA where

E = numToInsert
D = ideriv
C = imono
B = ismth
A = isplt

and each option flag value is defined as follows:

Variable	Value(s)	Description
numToInsert	0-9	Number of grid points to insert between each original grid point with respect to all independent variables (i.e., increase grid resolution)
ideriv	0	Disable finite derivative output
	1	Enable finite derivative output
imono	0	Disable forced monotonicity
	1	Enable forced monotonicity with respect to x of F(x,y)
	2	Enable forced monotonicity with respect to x and y of F(x,y)
	3	Enable forced monotonicity with respect to y of F(x,y)
ismth	0	Disable data table smoothing
	1	Enable data table smoothing that imposes a linear floor on temperature dependence, forces linear temperature dependence for low temperature, and forces linear density dependence for low and high density
	2	Enable data table smoothing that forces quadratic temperature dependence for low temperature, and linear temperature dependence for high temperature
isplt	-3	Forces splitting for ion-electron data tables using the cold curve plus Cowan-nuclear model for ions

Variable	Value(s)	Description
	-2	Forces splitting for ion-electron data tables using the cold curve plus ideal gas model for ions
	-1	Forces splitting for ion-electron data tables using the cold curve plus number-proportional model for ions
	0	Disable ion-electron splitting
	1	Allows splitting for ion-electron data tables not found in the database using the cold curve plus number-proportional model for ions
	2	Allows splitting for ion-electron data tables not found in the database using the cold curve plus ideal gas model for ions
	3	Allows splitting for ion-electron data tables not found in the database using the cold curve plus Cowan-nuclear model for ions

Appendix B

Header Files

1. eospac4.h

```
!
!       header file for named parameter constants used to
!       control the EOSPAC library via the calls to es1tabs
!       and es1vals (specific for an F9x host code)
!
!
!       Parameters for min x and max log10(x) values
        real(r8kind), parameter :: tiny              = 1.0d-99
        real(r8kind), parameter :: tinylogm          = 99.0d+00
!
!       Parameters for the different table types
        integer(i_kind), parameter :: ES4_NULLPTR    = 0 !  null table entry
        integer(i_kind), parameter :: ES4_PRTOT      = 1 !  temperature-based total pressure.
        integer(i_kind), parameter :: ES4_ENTOT      = 2 !  temperature-based total internal
                                                            energy/mass.
        integer(i_kind), parameter :: ES4_TPTOT      = 3 !  pressure-based total temperature.
        integer(i_kind), parameter :: ES4_TNTOT      = 4 !  energy-based total temperature.
        integer(i_kind), parameter :: ES4_PNTOT      = 5 !  energy-based total pressure.
        integer(i_kind), parameter :: ES4_EPTOT      = 6 !  pressure-based total internal
                                                            energy/mass.
        integer(i_kind), parameter :: ES4_PRION      = 7 !  temperature-based ion pressure.
        integer(i_kind), parameter :: ES4_ENION      = 8 !  temperature-based ion internal
                                                            energy/mass.
        integer(i_kind), parameter :: ES4_TPION      = 9 !  pressure -based ion temperature.
        integer(i_kind), parameter :: ES4_TNION      = 10!  energy-based ion temperature.
        integer(i_kind), parameter :: ES4_PNION      = 11!  energy-based ion pressure.
        integer(i_kind), parameter :: ES4_EPION      = 12!  pressure-based ion internal energy/mass.
        integer(i_kind), parameter :: ES4_PRELC      = 13!  temperature-based electron pressure.
        integer(i_kind), parameter :: ES4_ENELC      = 14!  temperature-based electron internal
                                                            energy/mass.
        integer(i_kind), parameter :: ES4_TPELC      = 15!  pressure-based electron temperature.
        integer(i_kind), parameter :: ES4_TNELC      = 16!  energy-based electron temperature.
        integer(i_kind), parameter :: ES4_PNELC      = 17!  energy-based electron pressure.
        integer(i_kind), parameter :: ES4_EPELC      = 18!  pressure-based electron internal energy/
                                                            mass.
        integer(i_kind), parameter :: ES4_PRCLD      = 19!  temperature-based cold curve pressure.
        integer(i_kind), parameter :: ES4_ENCLD      = 20!  temperature-based cold curve internal
                                                            energy/mass.
        integer(i_kind), parameter :: ES4_OPACR      = 21!  temperature-based rosseland mean opacity.
        integer(i_kind), parameter :: ES4_OPACC2     = 22!  temperature-based electron conductive
                                                            opacity.
        integer(i_kind), parameter :: ES4_ZFREE2     = 23!  temperature-based number of free
                                                            electrons per ion.
        integer(i_kind), parameter :: ES4_OPACP      = 24!  temperature-based planck mean opacity.
        integer(i_kind), parameter :: ES4_ZFREE3     = 25!  temperature-based number of free
                                                            electrons per ion.
        integer(i_kind), parameter :: ES4_ECONDE     = 26!  temperature-based electron electrical
                                                            conductivity.
        integer(i_kind), parameter :: ES4_TCONDE     = 27!  temperature-based electron thermal
                                                            conductivity.
        integer(i_kind), parameter :: ES4_THERME     = 28!  temperature-based electron thermo-
                                                            electric coef.
        integer(i_kind), parameter :: ES4_OPACC3     = 29!  temperature-based electron conductive
                                                            opacity.
        integer(i_kind), parameter :: ES4_TMELT      = 30!  temperature-based melting temperature.
        integer(i_kind), parameter :: ES4_PMELT      = 31!  temperature-based melting pressure.
```

```
        integer(i_kind), parameter :: ES4_EMELT      = 32!  temperature-based melting internal
                                                            energy/mass.
        integer(i_kind), parameter :: ES4_TFREEZ     = 33!  temperature-based freezing temperature.
        integer(i_kind), parameter :: ES4_PFREEZ     = 34!  temperature-based freezing pressure.
        integer(i_kind), parameter :: ES4_EFREEZ     = 35!  temperature-based freezing internal
                                                            energy/mass.
        integer(i_kind), parameter :: ES4_SHEARM     = 36!  temperature-based shear modulus.
        integer(i_kind), parameter :: ES4_DPTTOT     = 37!  pressure- and temperature-based total
                                                            density.
!
!       Parameters control what is computed and returned corresponding to corrected
!       transformations within ES1VALS.
        integer(i_kind), parameter :: ES4_FUNC       = 1 !  eos function f(x,y) only.
        integer(i_kind), parameter :: ES4_DERIVS     = 2 !  eos function f(x,y) plus derivatives wrt
                                                            alog(x,y).
        integer(i_kind), parameter :: ES4_GATHER     = 3 !  gathered eos data table x,y,f values.
!
!       Parameters control what is computed and returned corresponding to original
!       transformations within ES1VALS.
        integer(i_kind), parameter :: ES4_FUNC0      = 4 !  eos function f(x,y) only.
        integer(i_kind), parameter :: ES4_DERIVS0    = 5 !  eos function f(x,y) plus derivatives wrt
                                                            alog(x,y).
!
!       Parameters for the different interpolation methods
        integer(i_kind), parameter :: ES4_BILINE     = 1 !  bi-linear (4-point) interpolation.
        integer(i_kind), parameter :: ES4_BIQUAD     = 2 !  bi-quadratic (6-point) interpolation.
        integer(i_kind), parameter :: ES4_BIRATF     = 3 !  bi-rational-function (12-point)
                                                            interpolation.
```

2. eospac4c.h

```
/*
*       header file for named parameter constants used to
*       control the EOSPAC library via the calls to es1tabs
*       and es1vals (specific to C/C++ host codes)
*/
#ifndef EOSPAC_EOSPAC4C_H
#define EOSPAC_EOSPAC4C_H

/* Parameters for the different table types                                          */
static const int ES4_NULLPTR  = 0;  /* null table entry                               */
static const int ES4_PRTOT    = 1;  /* temperature-based total pressure.              */
static const int ES4_ENTOT    = 2;  /* temperature-based total internal energy/mass.  */
static const int ES4_TPTOT    = 3;  /* pressure-based total temperature.              */
static const int ES4_TNTOT    = 4;  /* energy-based total temperature.                */
static const int ES4_PNTOT    = 5;  /* energy-based total pressure.                   */
static const int ES4_EPTOT    = 6;  /* pressure-based total internal energy/mass.     */
static const int ES4_PRION    = 7;  /* temperature-based ion pressure.                */
static const int ES4_ENION    = 8;  /* temperature-based ion internal energy/mass.    */
static const int ES4_TPION    = 9;  /* pressure-based ion temperature.                */
static const int ES4_TNION    = 10; /* energy-based ion temperature.                  */
static const int ES4_PNION    = 11; /* energy-based ion pressure.                     */
static const int ES4_EPION    = 12; /* pressure-based ion internal energy/mass.       */
static const int ES4_PRELC    = 13; /* temperature-based electron pressure.           */
static const int ES4_ENELC    = 14; /* temperature-based electron internal energy/mass. */
static const int ES4_TPELC    = 15; /* pressure-based electron temperature.           */
static const int ES4_TNELC    = 16; /* energy-based electron temperature.             */
static const int ES4_PNELC    = 17; /* energy-based electron pressure.                */
static const int ES4_EPELC    = 18; /* pressure-based electron internal energy/mass.  */
static const int ES4_PRCLD    = 19; /* temperature-based cold curve pressure.         */
static const int ES4_ENCLD    = 20; /* temperature-based cold curve internal energy/mass. */
static const int ES4_OPACR    = 21; /* temperature-based rosseland mean opacity.      */
static const int ES4_OPACC2   = 22; /* temperature-based electron conductive opacity. */
static const int ES4_ZFREE2   = 23; /* temperature-based number of free electrons per ion. */
```

```
static const int ES4_OPACP    = 24; /* temperature-based planck mean opacity.                  */
static const int ES4_ZFREE3   = 25; /* temperature-based number of free electrons per ion.     */
static const int ES4_ECONDE   = 26; /* temperature-based electron electrical conductivity.     */
static const int ES4_TCONDE   = 27; /* temperature-based electron thermal conductivity.        */
static const int ES4_THERME   = 28; /* temperature-based electron thermo-electric coef.        */
static const int ES4_OPACC3   = 29; /* temperature-based electron conductive opacity.          */
static const int ES4_TMELT    = 30; /* temperature-based melting temperature.                  */
static const int ES4_PMELT    = 31; /* temperature-based melting pressure.                     */
static const int ES4_EMELT    = 32; /* temperature-based melting internal energy/mass.         */
static const int ES4_TFREEZ   = 33; /* temperature-based freezing temperature.                 */
static const int ES4_PFREEZ   = 34; /* temperature-based freezing pressure.                    */
static const int ES4_EFREEZ   = 35; /* temperature-based freezing internal energy/mass.        */
static const int ES4_SHEARM   = 36; /* temperature-based shear modulus.                        */
static const int ES4_DPTTOT   = 37; /* pressure- and temperature-based total density.          */

/* Parameters control what is computed and returned corresponding to                            /*
/* corrected transformations within ES1VALS.                                                    /*
static const int ES4_FUNC     = 1;  /* eos function f(x,y) only.                               */
static const int ES4_DERIVS   = 2;  /* eos function f(x,y) plus derivatives wrt alog(x,y)      */
static const int ES4_GATHER   = 3;  /* gathered eos data table x,y,f values.                   */

/* Parameters control what is computed and returned corresponding to original                   */
/* transformations within ES1VALS.                                                              */
static const int ES4_FUNC0    = 4;  /* eos function f(x,y) only.                               */
static const int ES4_DERIVS0  = 5;  /* eos function f(x,y) plus derivatives wrt alog(x,y).     */

/* Parameters for the different interpolation methods                                           */
static const int ES4_BILINE   = 1;  /* bi-linear (4-point) interpolation.                      */
static const int ES4_BIQUAD   = 2;  /* bi-quadratic (6-point) interpolation.                   */
static const int ES4_BIRATF   = 3;  /* bi-rational-function (12-point) interpolation.          */

#endif /* EOSPAC_EOSPAC4C_H */
```

3. eospac4f77.h

```
c
c       header file for named parameter constants used to
c       control the EOSPAC4 library via the calls to es1tabs
c       and es1vals (specific for an F77 host code)
c
c
c       parameters for min x and max log10(x) values
        double precision tiny, tinylogm
        parameter (tiny      = 1.0d-99)
        parameter (tinylogm  = 99.0d+00)
c
c       Parameters for the different table types
        integer  ES4_NULLPTR,ES4_PRTOT, ES4_ENTOT,  ES4_TPTOT,
     &           ES4_TNTOT, ES4_PNTOT,  ES4_EPTOT,  ES4_PRION,
     &           ES4_ENION, ES4_TPION,  ES4_TNION,  ES4_PNION,
     &           ES4_EPION, ES4_PRELC,  ES4_ENELC,  ES4_TPELC,
     &           ES4_TNELC, ES4_PNELC,  ES4_EPELC,  ES4_PRCLD,
     &           ES4_ENCLD, ES4_OPACR,  ES4_OPACC2, ES4_ZFREE2,
     &           ES4_OPACP, ES4_ZFREE3, ES4_ECONDE, ES4_TCONDE,
     &           ES4_THERME,ES4_OPACC3, ES4_TMELT,  ES4_PMELT,
     &           ES4_EMELT, ES4_TFREEZ, ES4_PFREEZ, ES4_EFREEZ,
     &           ES4_SHEARM,ES4_DPTTOT

        parameter (ES4_NULLPTR   = 0)  !  null table entry
        parameter (ES4_PRTOT     = 1)  !  temperature-based total pressure.
        parameter (ES4_ENTOT     = 2)  !  temperature-based total internal energy/mass.
        parameter (ES4_TPTOT     = 3)  !  pressure-based total temperature.
        parameter (ES4_TNTOT     = 4)  !  energy-based total temperature.
        parameter (ES4_PNTOT     = 5)  !  energy-based total pressure.
        parameter (ES4_EPTOT     = 6)  !  pressure -based total internal energy/mass.
```

```
      parameter (ES4_PRION     = 7)  !  temperature-based ion pressure.
      parameter (ES4_ENION     = 8)  !  temperature-based ion internal energy/mass.
      parameter (ES4_TPION     = 9)  !  pressure-based ion temperature.
      parameter (ES4_TNION     = 10) !  energy-based ion temperature.
      parameter (ES4_PNION     = 11) !  energy-based ion pressure.
      parameter (ES4_EPION     = 12) !  pressure-based ion internal energy/mass.
      parameter (ES4_PRELC     = 13) !  temperature-based electron pressure.
      parameter (ES4_ENELC     = 14) !  temperature-based electron internal energy/mass.
      parameter (ES4_TPELC     = 15) !  pressure-based electron temperature.
      parameter (ES4_TNELC     = 16) !  energy-based electron temperature.
      parameter (ES4_PNELC     = 17) !  energy-based electron pressure.
      parameter (ES4_EPELC     = 18) !  pressure-based electron internal energy/mass.
      parameter (ES4_PRCLD     = 19) !  temperature-based cold curve pressure.
      parameter (ES4_ENCLD     = 20) !  temperature-based cold curve internal energy/mass.
      parameter (ES4_OPACR     = 21) !  temperature-based rosseland mean opacity.
      parameter (ES4_OPACC2    = 22) !  temperature-based electron conductive opacity.
      parameter (ES4_ZFREE2    = 23) !  temperature-based number of free electrons per ion.
      parameter (ES4_OPACP     = 24) !  temperature-based planck mean opacity.
      parameter (ES4_ZFREE3    = 25) !  temperature-based number of free electrons per ion.
      parameter (ES4_ECONDE    = 26) !  temperature-based electron electrical conductivity.
      parameter (ES4_TCONDE    = 27) !  temperature-based electron thermal conductivity.
      parameter (ES4_THERME    = 28) !  temperature-based electron thermo-electric coef.
      parameter (ES4_OPACC3    = 29) !  temperature-based electron conductive opacity.
      parameter (ES4_TMELT     = 30) !  temperature-based melting temperature.
      parameter (ES4_PMELT     = 31) !  temperature-based melting pressure.
      parameter (ES4_EMELT     = 32) !  temperature-based melting internal energy/mass.
      parameter (ES4_TFREEZ    = 33) !  temperature-based freezing temperature.
      parameter (ES4_PFREEZ    = 34) !  temperature-based freezing pressure.
      parameter (ES4_EFREEZ    = 35) !  temperature-based freezing internal energy/mass.
      parameter (ES4_SHEARM    = 36) !  temperature-based shear modulus.
      parameter (ES4_DPTTOT    = 37) !  pressure- and temperature-based total density.
c
c     Parameters control what is computed and returned corresponding to corrected
c     transformations within ES1VALS.
      integer ES4_FUNC, ES4_DERIVS, ES4_GATHER
      parameter (ES4_FUNC      = 1)  !  eos function f(x,y) only.
      parameter (ES4_DERIVS    = 2)  !  eos function f(x,y) plus derivatives wrt alog(x,y).
      parameter (ES4_GATHER    = 3)  !  gathered eos data table x,y,f values.
c
c     Parameters control what is computed and returned corresponding to original
c     transformations within ES1VALS.
      integer ES4_FUNC0, ES4_DERIVS0
      parameter (ES4_FUNC0     = 4)  !  eos function f(x,y) only.
      parameter (ES4_DERIVS0   = 5)  !  eos function f(x,y) plus derivatives wrt alog(x,y).
c
c     Parameters for the different interpolation methods
      integer ES4_BILINE, ES4_BIQUAD, ES4_BIRATF
      parameter (ES4_BILINE    = 1)  !  bi-linear (4-point) interpolation.
      parameter (ES4_BIQUAD    = 2)  !  bi-quadratic (6-point) interpolation.
      parameter (ES4_BIRATF    = 3)  !  bi-rational-function (12-point) interpolation.
```

4. es4proto.h

```
/*
 * C and C++ external function prototypes defined within EOSPAC.
 */

#ifndef EOSPAC_ES4PROTO_H
#define EOSPAC_ES4PROTO_H

#include "wrapperDefines.h"

#ifdef __cplusplus
/*
```

```
 * This sections contains the C++ prototypes for EOSPAC.
 */
extern "C" {

    void FUNC_INTERFACE es1errmsg
                        (INTEGER& errorCode, char errorMessage[]);

    void FUNC_INTERFACE es1tabs
            (           BOOLEAN   llog1[],  INTEGER   iopt[],   BOOLEAN& lprnt,      INTEGER&   iprnt,
                        INTEGER&  mtyps,    INTEGER&  mregs,    INTEGER   inams[],
                        REAL      ucons[],  INTEGER   imids[],  INTEGER& idtab,      INTEGER&   mtabs,
                        REAL**    ktabs,    INTEGER   ierrs[]);

    void FUNC_INTERFACE es1vals
            (const      INTEGER&  ieosf,    const     INTEGER&  idrvs,    const      INTEGER&   intrp,
             const      REAL      ftabs[],  const     INTEGER&  mtabs,    const      INTEGER&   nzons,
                        INTEGER&  iregn,    const     REAL      xvals[],  const      REAL       yvals[],
                        REAL      fvals[],  const     INTEGER&  nvalsi,              INTEGER&   ierr1);

    void FUNC_INTERFACE mx1vals
            (const      INTEGER&  ieosf,    const     INTEGER&  idrvs,    const      INTEGER&   intrp,
             const      REAL      ftabs[],  const     INTEGER&  mtabs,    const      INTEGER&   nzons,
             const      INTEGER&  mtyps,    const     INTEGER&  mregs,
             const      INTEGER&  nmixr,    const     INTEGER   imixr[],  const      REAL       cmixr[]
                        INTEGER   iregs[],  const     REAL      xvals[],  const      REAL       yvals[],
                        REAL      fvals[],  const     INTEGER&  nvalsi,              INTEGER&   ierr1);

    void FUNC_INTERFACE es1info
            (const      INTEGER&  itype,    const     INTEGER&  iregn,
                        REAL**    ktabs,              INTEGER&  iname,               BOOLEAN&   llogs,
                        REAL&     xcnvt,              REAL&     ycnvt,               REAL&      fcnvt,
                        INTEGER&  matid,              REAL&     znbar,               REAL&      anbar,
                        REAL&     dens0,              INTEGER&  ifile,               INTEGER&   ierr2);

    void FUNC_INTERFACE es2func
            (           REAL      ftabs[],  const     INTEGER&  iregn,
             const      INTEGER&  ieosf,    const     INTEGER&  idrvs,
             const      INTEGER&  intrp,              INTEGER&  itype,               INTEGER&   ifunc,
                        INTEGER&  icate,              INTEGER&  ibase,               BOOLEAN&   llogs,
                        INTEGER&  nvals,              INTEGER&  ierr2);

    void FUNC_INTERFACE es2grids
            (const      BOOLEAN&  llogs,    const     INTEGER*  inams,               REAL       ftabs[],
                        INTEGER&  mtabs,              INTEGER&  itabs,    const      INTEGER&   mtyps,
             const      INTEGER&  mregs,    const     INTEGER&  jtype,
             const      INTEGER&  jregn,    const     INTEGER&  numToInsert,         INTEGER&   ierr2);

    void FUNC_INTERFACE es2tbls
             (          INTEGER&  nxtbl,              INTEGER&  nytbl,               INTEGER&   ixtbl,
                        INTEGER&  iytbl,              INTEGER&  iftbl,
             const      INTEGER&  itype,    const     INTEGER&  iregn,               INTEGER&   ierr2);

    void FUNC_INTERFACE es4alloc
            (const      INTEGER&  mtabs,              REAL**    ktabs,               INTEGER&   ierr4);

    void FUNC_INTERFACE es4line
            (           INTEGER&  nvals,    const     INTEGER&  nzons,
             const      INTEGER&  nxtbl,    const     INTEGER&  nytbl,
             const      INTEGER&  ixtbl,    const     REAL      ytbls[],
             const      REAL      ftbls[],  const     REAL      yvalv[],
                        REAL      fvalv[],  const     INTEGER&  nvalv);

    void FUNC_INTERFACE es4xpnd
            (const      INTEGER&  mdata,    const     INTEGER&  itabs,
                        INTEGER&  mtabs,              REAL**    ktabs,               INTEGER&   ierr4);
```

```
   void FUNC_INTERFACE srch2d
           (const     INTEGER& MY,       const     REAL      Y[],      const       INTEGER&   IY,
            const     INTEGER& N,        const     REAL      X[],      const       INTEGER&   IX,
                      INTEGER  J[],                REAL      D[],                  REAL       F[]);

   void FUNC_INTERFACE srch3d
           (const     INTEGER& MY,       const     REAL      Y[],      const       INTEGER&   IY,
            const     INTEGER& N,        const     REAL      X[],      const       INTEGER&   IX,
                      INTEGER  J[],                REAL      D[],                  REAL       F[]);

   void FUNC_INTERFACE srchdf
           (const     INTEGER& MY,       const     REAL      Y[],      const       INTEGER&   IY,
            const     INTEGER& N,        const     REAL      X[],      const       INTEGER&   IX,
                      INTEGER  J[],                REAL      D[],                  REAL F[]);
}

#else /* ! __cplusplus */

/*
 * This sections contains the C prototypes for EOSPAC.
 */
   void FUNC_INTERFACE es1errmsg
            (         INTEGER* errorCode,         char      errorMessage[]);

   void FUNC_INTERFACE es1tabs
            (         BOOLEAN  llog1[],  INTEGER  iopt[],   BOOLEAN* lprnt,      INTEGER*   iprnt,
                      INTEGER* mtyps,    INTEGER* mregs,    INTEGER  inams[],
                      REAL     ucons[],  INTEGER  imids[],  INTEGER* idtab,      INTEGER*   mtabs,
                      REAL**   ktabs,    INTEGER  ierrs[]);

   void FUNC_INTERFACE es1vals
           (const     INTEGER* ieosf,    const     INTEGER* idrvs,    const INTEGER*         intrp,
            const     REAL     ftabs[],  const     INTEGER* mtabs,    const INTEGER*         nzons,
                      INTEGER* iregn,    const     REAL     xvals[],  const REAL             yvals[],
                      REAL     fvals[],  const     INTEGER* nvalsi,         INTEGER*         ierr1);

   void FUNC_INTERFACE mx1vals
           (const     INTEGER* ieosf,    const     INTEGER* idrvs,    const INTEGER*         intrp,
            const     REAL     ftabs[],  const     INTEGER* mtabs,    const INTEGER*         nzons,
            const     INTEGER* mtyps,    const     INTEGER* mregs,
            const     INTEGER* nmixr,    const     INTEGER  imixr[],  const REAL             cmixr[],
                      INTEGER  iregs[],  const     REAL     xvals[],  const REAL             yvals[],
                      REAL     fvals[],  const     INTEGER* nvalsi,         INTEGER*         ierr1);

   void FUNC_INTERFACE es1info
           (const     INTEGER* itype,    const     INTEGER* iregn,
                      REAL**   ktabs,              INTEGER* iname,          BOOLEAN*         llogs,
                      REAL*    xcnvt,              REAL*    ycnvt,          REAL*            fcnvt,
                      INTEGER* matid,              REAL*    znbar,          REAL*            anbar,
                      REAL*    dens0,              INTEGER* ifile,          INTEGER*         ierr2);

   void FUNC_INTERFACE es2func
            (         REAL     ftabs[],  const     INTEGER* iregn,
             const    INTEGER* ieosf,    const     INTEGER* idrvs,
             const    INTEGER* intrp,              INTEGER* itype,
                      INTEGER* ifunc,              INTEGER* icate,
                      INTEGER* ibase,              BOOLEAN* llogs,
                      INTEGER* nvals,              INTEGER* ierr2);

   void FUNC_INTERFACE es2grids
             (const   BOOLEAN* llogs,    const     INTEGER* inams,
                      REAL     ftabs[],            INTEGER* mtabs,
                      INTEGER* itabs,    const     INTEGER* mtyps,
             const    INTEGER* mregs,    const     INTEGER* jtype,
             const    INTEGER* jregn,    const     INTEGER* numToInsert,    INTEGER*         ierr2);
```

```
void FUNC_INTERFACE es2tbls
          (        INTEGER* nxtbl,             INTEGER* nytbl,         INTEGER*         ixtbl,
                   INTEGER* iytbl,             INTEGER* iftbl,
           const   INTEGER* itype,    const    INTEGER* iregn,         INTEGER*         ierr2);

void FUNC_INTERFACE es4alloc
          (const   INTEGER* mtabs,             REAL**   ktabs,         INTEGER*         ierr4);

void FUNC_INTERFACE es4line
          (        INTEGER* nvals,    const    INTEGER* nzons,
           const   INTEGER* nxtbl,    const    INTEGER* nytbl,
           const   INTEGER* ixtbl,    const    REAL     ytbls[],
           const   REAL     ftbls[],  const    REAL     yvalv[],
                   REAL     fvalv[],  const    INTEGER* nvalv);

void FUNC_INTERFACE es4xpnd
          (const   INTEGER* mdata,    const    INTEGER* itabs,
                   INTEGER* mtabs,             REAL**   ktabs,
                   INTEGER* ierr4);

void FUNC_INTERFACE srch2d
          (const   INTEGER* MY,       const    REAL     Y[],     const      INTEGER*   IY,
           const   INTEGER* N,        const    REAL     X[],     const      INTEGER*   IX,
                   INTEGER  J[],               REAL     D[],                REAL       F[]);

void FUNC_INTERFACE srch3d
          (const   INTEGER* MY,       const    REAL     Y[],     const      INTEGER*   IY,
           const   INTEGER* N,        const    REAL     X[],     const      INTEGER*   IX,
                   INTEGER  J[],               REAL     D[],                REAL       F[]);

void FUNC_INTERFACE srchdf
          (const   INTEGER* MY,       const    REAL     Y[],     const      INTEGER*   IY,
           const   INTEGER* N,        const    REAL     X[],     const      INTEGER*   IX,
                   INTEGER  J[],               REAL     D[],                REAL       F[]);

#endif /* __cplusplus */

#endif /* EOSPAC_ES4PROTO_H */
```

5. es4types.h

```
/*
 * C/C++ header file that defines the basic data types used by
 * the EOSPAC library.
 */

#ifndef EOSPAC_TYPES_H
#define EOSPAC_TYPES_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#ifndef USER_SPECIFIED_AUTOCONF_TARGET

#if defined(__sgi) && ! defined(SGI)
#define SGI
#endif

#if (defined(__alpha) && defined(__osf__)) && ! defined(ALPHA)
#define ALPHA
#endif
```

```
#endif /* USER_SPECIFIED_AUTOCONF_TARGET */

#if defined(SGI) || defined(ALPHA)

typedef int INTEGER;
typedef int BOOLEAN;

#else

typedef long INTEGER;
typedef long BOOLEAN;

#endif

#if ! defined(ALPHA) && ! defined(IBM)

static const BOOLEAN FALSE    = 0;

#ifndef UNICOS
static const BOOLEAN TRUE     = 1;  /* make true correspond to Fortran .true.                */
#else
static const BOOLEAN TRUE     = -1; /* make true correspond to Fortran .true.                */
#endif

#endif

typedef double REAL;

static const REAL tiny        = 1.0e-99;
static const REAL tinylogm    = 99.0;

struct eosFunctionData {
      int                     eosFunctionNumber;
      char                    eosFunctionName[16];
      int                     linkIndex, catalog, category, base;
      BOOLEAN                 logAxes;
      int                     functionNumber;
};

struct materialData {
      INTEGER                 materialID;
      REAL                    aveAtomicNumber, aveAtomicWgt;
      REAL                    refDensity;
};


struct eosTableData {
      eosFunctionData*        functionData;
      materialData            matData;
      BOOLEAN                 logAxes;
      REAL                    xCnvrtFactor, yCnvrtFactor, fCnvrtFactor;
      INTEGER                 fileId;
      INTEGER                 nxTable, nyTable,
                              xAxisIndex, yAxisIndex, tableIndex;
};


struct eosMiscData {
      char initWord[8];             /* string to show that memory block is initialized.       */
      INTEGER nTypes;               /* number of table types actually used                    */
      INTEGER nRegions;             /* number of regions of different materials               */
      INTEGER infoSize;             /* size of the header block in words                      */
      INTEGER currentIndex;         /* current word index into memory block                   */
      INTEGER maxIndex;             /* size in words of memory block                          */
};
```

```
struct NRmachPrecData {             /* global values used by es4invt or returned by es4mach   */
      BOOLEAN gotMachinePrecision;  /* Flag indicating if es4mach has been called             */
      REAL eps;                     /* The smallest positive floating-point number
                                       such that (1.0 + eps) > 1, is referred to as the
                                       "floating-point precision."                            */
      REAL epsneg;                  /* The smallest positive floating-point number
                                       such that (1.0 - eps) < 1, is another way of defining
                                       the "floating-point precision."                        */
      REAL maxErr;                  /* Newton convergence tolerence                           */
      INTEGER maxIter;              /* Newton maximum iterations                              */
};


#include "es5err.h"

#endif /* EOSPAC_TYPES_H */
```

6. es5err.h

```
/*
 * This header file contains the
 * C/C++ version of error flag constants definitions.
 * This is included within the es4types.h header file.
 */

#ifndef EOSPAC_ES5ERR_H
#define EOSPAC_ES5ERR_H

/*
 * The following are user-correctable errors
 */
static const int ES5_OK               = 0;   /* no errors                                      */
static const int ES5_NOTINIT          = 1;   /* eos table area is not initialized              */
static const int ES5_BADTYPE          = 2;   /* table type index is out of bounds              */
static const int ES5_BADREGION        = 3;   /* region index is out of bounds                  */
static const int ES5_NOTABLE          = 4;   /* data table is not in eos table area            */
static const int ES5_BADDERIVTYPE     = 5;   /* derivative is not recognized                   */
static const int ES5_BADINTRPTYPE     = 6;   /* interpolation is not recognized                */
static const int ES5_MATIDZERO        = 7;   /* matid is zero                                  */
static const int ES5_BADFVALSDIM      = 8;   /* not enough space for fvals                     */
static const int ES5_BADTABLETYPE     = 9;   /* data table type is not recognized              */
static const int ES5_NOFILESFOUND     = 10;  /* no data library files exist                    */
static const int ES5_MATNOTFOUND      = 11;  /* material id is not in library                  */
static const int ES5_TYPENOTFOUND     = 12;  /* data table type is not in library              */
static const int ES5_BADFILEID        = 13;  /* file index outside of range                    */
static const int ES5_EXPANDFAILED     = 14;  /* eos table area cannot be expanded              */
static const int ES5_OPENFAILED       = 15;  /* could not open data file                       */
static const int ES5_GETNMATSFAILED   = 16;  /* could not read number of materials             */
static const int ES5_GETVERSNFAILED   = 17;  /* could not load version from data file          */
static const int ES5_LDMASTERFAILED   = 18;  /* could not load master directory                */
static const int ES5_LDMATDIRFAILED   = 19;  /* could not load material directory              */
static const int ES5_LDTABLEFAILED    = 20;  /* could not load data table                      */
static const int ES5_UNKNOWNHDFTAG    = 21;  /* unknown HDF tag                                */
static const int ES5_NOCOLDCURVE      = 22;  /* host did not request needed cold curve for
                                                grid refinement                               */
static const int ES5_CONVERGEFAILED   = 23;  /* iteration did not converge                     */

/*
 * The following are EOSPAC internal error codes.
 * Please notify vendor if any of these are returned to host code.
 */
static const int ES5_BADFUNCTION      = -1;  /* improper function reference made               */
static const int ES5_ALLOCFAILED      = -2;  /* memory allocation failed                       */
```

```
#endif /* EOSPAC_ES5ERR_H */
```

7. es5errf.h

```
c
c       This header file contains the
c       Fortran version of error flag constants definitions.
c
c       The following are user-correctable errors
c
        integer ES5_OK                  ! no errors,
        integer ES5_NOTINIT             ! eos table area is not initialized,
        integer ES5_BADTYPE             ! table type index is out of bounds,
        integer ES5_BADREGION           ! region index is out of bounds,
        integer ES5_NOTABLE             ! data table is not in eos table area,
        integer ES5_BADDERIVTYPE        ! derivative is not recognized,
        integer ES5_BADINTRPTYPE        ! interpolation is not recognized,
        integer ES5_MATIDZERO           ! matid is zero,
        integer ES5_BADFVALSDIM         ! not enough space for fvals
        integer ES5_BADTABLETYPE        ! data table type is not recognized,
        integer ES5_NOFILESFOUND        ! no data library files exist,
        integer ES5_MATNOTFOUND         ! material id is not in library,
        integer ES5_TYPENOTFOUND        ! data table type is not in library,
        integer ES5_BADFILEID           ! file index outside of range
        integer ES5_EXPANDFAILED        ! eos table area cannot be expanded.
        integer ES5_OPENFAILED          ! could not open data file
        integer ES5_GETNMATSFAILED      ! could not read number of materials
        integer ES5_GETVERSNFAILED      ! could not load version from data file
        integer ES5_LDMASTERFAILED      ! could not load master directory
        integer ES5_LDMATDIRFAILED      ! could not load material directory
        integer ES5_LDTABLEFAILED       ! could not load data table
        integer ES5_UNKNOWNHDFTAG       ! unknown HDF tag
        integer ES5_NOCOLDCURVE         ! host did not request needed cold
                                        ! curve for grid refinement
        integer ES5_CONVERGEFAILED      ! iteration did not converge
        integer nErrCodes               ! Total number of pre-defined Error Codes
c
c       The following are EOSPAC internal error codes.
c       Please notify vendor if any of these are returned to host code.
        integer ES5_BADFUNCTION         ! improper function reference made
        integer ES5_ALLOCFAILED         ! memory allocation failed
        integer nInternalErrCodes       ! Total number of internal Error Codes

        parameter(ES5_OK                =  0)
        parameter(ES5_NOTINIT           =  1)
        parameter(ES5_BADTYPE           =  2)
        parameter(ES5_BADREGION         =  3)
        parameter(ES5_NOTABLE           =  4)
        parameter(ES5_BADDERIVTYPE      =  5)
        parameter(ES5_BADINTRPTYPE      =  6)
        parameter(ES5_MATIDZERO         =  7)
        parameter(ES5_BADFVALSDIM       =  8)
        parameter(ES5_BADTABLETYPE      =  9)
        parameter(ES5_NOFILESFOUND      = 10)
        parameter(ES5_MATNOTFOUND       = 11)
        parameter(ES5_TYPENOTFOUND      = 12)
        parameter(ES5_BADFILEID         = 13)
        parameter(ES5_EXPANDFAILED      = 14)
        parameter(ES5_OPENFAILED        = 15)
        parameter(ES5_GETNMATSFAILED    = 16)
        parameter(ES5_GETVERSNFAILED    = 17)
        parameter(ES5_LDMASTERFAILED    = 18)
        parameter(ES5_LDMATDIRFAILED    = 19)
```

```
      parameter(ES5_LDTABLEFAILED     = 20)
      parameter(ES5_UNKNOWNHDFTAG     = 21)
      parameter(ES5_NOCOLDCURVE       = 22)
      parameter(ES5_CONVERGEFAILED    = 23)
      parameter(nErrCodes             = 24)

      parameter(ES5_BADFUNCTION       = -1)
      parameter(ES5_ALLOCFAILED       = -2)
      parameter(nInternalErrCodes     =  2)
```

8. wrapperDefines.h

```
/*
 * The following are some preprocessor macros used to simplify
 * cross-platform C-FORTRAN interfaces. It is included within
 * the es4proto.h header file.
 */

#ifndef EOSPAC_WRAPPERDEFINES_H
#define EOSPAC_WRAPPERDEFINES_H

#ifndef USER_SPECIFIED_AUTOCONF_TARGET

#if defined(_X86_) && ! defined(PC)
#define PC
#endif

#if defined(__linux) && ! defined(LINUX)
#define LINUX
#endif

#if defined(_CRAY) && ! defined(UNICOS)
#define UNICOS
#endif

#endif /* ! USER_SPECIFIED_AUTOCONF_TARGET */

#if defined PC

#define FUNC_INTERFACE __stdcall

#else /* ! PC */

#define FUNC_INTERFACE

#endif /* PC */

#if defined PC || defined UNICOS

#define es1errmsg ES1ERRMSG
#define es1vals ES1VALS
#define mx1vals MX1VALS
#define es1tabs ES1TABS
#define es1info ES1INFO
#define es2func ES2FUNC
#define es2tbls ES2TBLS
#define es4alloc ES4ALLOC
#define es4line ES4LINE
#define es4xpnd ES4XPND
#define srch2d SRCH2D
#define srch3d SRCH3D
#define srchdf SRCHDF
#define es2grids ES2GRIDS
#define cnvt2ptr CNVT2PTR
#define getTableData GETTABLEDATA
```

```
#define putTableData PUTTABLEDATA

#else /* ! (PC or UNICOS) */

#if defined IBM

#define es1errmsg es1errmsg
#define es1vals es1vals
#define mx1vals mx1vals
#define es1tabs es1tabs
#define es1info es1info
#define es2func es2func
#define es2tbls es2tbls
#define es4alloc es4alloc
#define es4line es4line
#define es4xpnd es4xpnd
#define srch2d srch2d
#define srch3d srch3d
#define srchdf srchdf
#define es2grids es2grids
#define cnvt2ptr cnvt2ptr
#define getTableData gettabledata
#define putTableData puttabledata

#else /* ! IBM */

#define es1errmsg es1errmsg_
#define es1vals es1vals_
#define mx1vals mx1vals_
#define es1tabs es1tabs_
#define es1info es1info_
#define es2func es2func_
#define es2tbls es2tbls_
#define es4alloc es4alloc_
#define es4line es4line_
#define es4xpnd es4xpnd_
#define srch2d srch2d_
#define srch3d srch3d_
#define srchdf srchdf_
#define es2grids es2grids_
#define cnvt2ptr cnvt2ptr_
#define getTableData gettabledata_
#define putTableData puttabledata_

#endif /* IBM */

#endif /* PC or UNICOS */

#endif /* EOSPAC_WRAPPERDEFINES_H */
```

Appendix C

C++ Example

1. example.cxx

```
#include "es4types.h"
#include "es4proto.h"
#include "eospac4c.h"
#include "stdio.h"

void display(INTEGER inam,  INTEGER iregn, INTEGER nzons,
         REAL*   xvals, REAL*   yvals, REAL*   fvals) {
 INTEGER j;
  printf("\nDATA TYPE: %12d    REGION: %12d\n",inam,iregn);
  printf("           x           y           F       dF/dx       dF/dy\n");
  for(j=0;j<nzons;j++) {
    printf("%12.4e%12.4e%12.4e%12.4e%12.4e\n",
       xvals[j],yvals[j],fvals[j+0*nzons],fvals[j+1*nzons],fvals[j+2*nzons]);
  }
}

REAL max(REAL a, REAL b) {
  if(a<b) return b;
  return a;
}

main() {
  //
  // Demonstrate the usage of ES1TABS, ES1VALS, ES1ERRMSG and ES1INFO
  //     switch-to-other-buffer
  // ES1TABS subroutine parameters
  INTEGER mtyps=3; // number of tables to load
  INTEGER mregs=1; //number of problem regions
  BOOLEAN *llog1 = (BOOLEAN*) malloc(sizeof(BOOLEAN)*mtyps*mregs);
  INTEGER *iopt  = (INTEGER*) malloc(sizeof(INTEGER)*mtyps*mregs);
  BOOLEAN lprnt;
  INTEGER iprnt;
  INTEGER *inams = (INTEGER*) malloc(sizeof(INTEGER)*mtyps);
  REAL    *ucons = (REAL*)    malloc(sizeof(REAL)   *mtyps*3);
  INTEGER *imids  = (INTEGER*) malloc(sizeof(INTEGER)*mtyps*mregs);
  INTEGER idtab, mtabs;
  REAL    *ktabs;
  INTEGER *ierrs = (INTEGER*) malloc(sizeof(INTEGER)*mtyps*mregs);

  // ES1VALS subroutine parameters
  INTEGER idrvs, intrp, iregn, ierr1;
  INTEGER nzons=20; // number of interpolated F values to return (zones)
  REAL    *xvals = (REAL*)malloc(sizeof(REAL)*nzons);   // x values for interpolation
  REAL    *yvals = (REAL*)malloc(sizeof(REAL)*nzons);   // y values for interpolation
  REAL    *fvals = (REAL*)malloc(sizeof(REAL)*nzons*3); // F, dF/dx, and dF/dy in each zone

  // ES1INFO subroutine parameters
  INTEGER iname, matid, ifile;
  BOOLEAN llogs;
  REAL    xcnvt, ycnvt, fcnvt, znbar, anbar, dens0;

  // Local variables
  INTEGER i, j, k;
  INTEGER errorCode;
  char    errorMessage[60];
```

```
// Assign subroutine parameter values
mtabs = 20000;             // arbitrary intial words in ftabs
lprnt = TRUE;              // dump loaded tables to an ascii file
iprnt = 0;                 // arbitrary value -- no longer used
idtab = 0;                 // arbitrary value -- no longer used
inams[0] = ES4_PRCLD;      // pressure cold curve
inams[1] = ES4_ENCLD;      // energy cold curve
inams[2] = ES4_PNTOT;      // total pressure (energy-based)
for(j=0;j<mtyps;j++) {
  ucons[j        ] = 1.;   // table x units conversion factor
  ucons[j+  mtyps] = 1.;   // table y units conversion factor
  ucons[j+2*mtyps] = 1.;   // table F(x,y) units conversion factor
  for(i=0;i<mregs;i++) {
    k = j+i*mtyps;
    llog1[k] = FALSE;  // don't return log10 tables for 300-tables
    iopt[k]  = 0;      // perform no data smoothing
    imids[k] = 3717;   // SESAME material ID number
    ierrs[k] = ES5_OK; // initialize error flag
  }
}

// Allocate memory for tables to be loaded
ktabs = (REAL*)malloc(sizeof(REAL)*mtabs);

// Load desired data tables
es1tabs(&llog1[0], &iopt[0],  lprnt, iprnt, mtyps,
    mregs, &inams[0], &ucons[0], &imids[0],
    idtab, mtabs, &ktabs, ierrs);
for(j=0;j<mtyps;j++) {
  for(i=0;i<mregs;i++) {
    k = j+i*mtyps;
    if (ierrs[k]) {
  es1errmsg(ierrs[k], errorMessage);
  printf("LOAD ERROR %d, TABLE %d: %s\n", ierrs[k], inams[k], errorMessage);
    }
  }
}

// Assign xvals and yvals array values
for(i=0;i<nzons;i++) {
  xvals[i] = (REAL)((i+1)*1000.0);
  yvals[i] = (REAL)((i+1)*2000.0);
}

// Perform interpolation for each data type
idrvs = ES4_DERIVS; // return F, dF/dx, and dF/dy
intrp = ES4_BIRATF; // use bi-rational interpolation
iregn = 1;          // only 1 region has been used in this example
for(i=0;i<mtyps;i++) {
  es1info(i+1, iregn, &ktabs, iname, llogs, xcnvt, ycnvt, fcnvt,
      matid, znbar, anbar, dens0, ifile, ierr1);
  if (ierr1) {
    es1errmsg(ierr1, errorMessage);
    printf("INFO ERROR %d, TABLE %d: %s\n", ierr1, inams[i], errorMessage);
  }
  es1vals(inams[i], idrvs, intrp, ktabs, mtabs,
      nzons, iregn, xvals, yvals, fvals, 3, ierr1);
  if (ierr1) {
    es1errmsg(ierr1, errorMessage);
    printf("INTERPOLATION ERROR %d, TABLE %d: %s\n",ierr1, inams[i], errorMessage);
  }
  else {
    // convert partial derivatives from dF/dlnx to dFdx and dF/dlny to dFdy
    for(j=0;j<nzons;j++) {
  fvals[j+1*nzons] /= max(xvals[j],tiny);
```

```
    fvals[j+2*nzons] /= max(yvals[j],tiny);
      }
      display(inams[i],iregn,nzons,xvals,yvals,fvals);
    }
  }

  // Display all possible error codes and corresponding messages
  printf("\n%s\n","ES1ERRMSG subroutine test");
  for(i=0;i<=23;i++) {
    errorCode = i;
    es1errmsg(errorCode, errorMessage);
    printf("%d: %s\n",errorCode, errorMessage);
  }
}
```

2. Makefile

```
# * * * * * * * * * * * * * * * * * * * * * * * * * * *
#
# "C++ example program" GNU Make Makefile
#
# * * * * * * * * * * * * * * * * * * * * * * * * * * *

LIBRARY = eospac

.SUFFIXES:
.SUFFIXES: .F .f .cxx .c .o .a

.F.o :
    $(F90) $(FFLAGS) -c $<

.f.o :
    @rm -f $*.f
    $(CPP) $(INCLUDE_DIRS) $(CPPFLAGS) $(DEFINES) $< ./$*.f
    $(F90) $(FFLAGS) -c $*.f
    @rm -f $*.f

.cxx.o :
    ${CC} ${CFLAGS} ${DEFINES} ${INCLUDE_DIRS} -c $<

.c.o :
    ${CC} ${CFLAGS} ${DEFINES} ${INCLUDE_DIRS} -c $<

##############################################################################
# gmake function to search $(2) for lib$(1).a where $(1) is a
# space-delimited list of library name(s) to find, and $(2) is a
# space-delimited list of environment variable(s).
# Usage example:
# CLIBS = $(call libSearch,gcc stdc++,$(LD_LIBRARY_PATH) $(LD_LIBRARY64_PATH))
libSearch = $(foreach a, $(1), \
                      $(firstword $(shell find $(subst :, ,$(2)) -name lib$(a).a)))
##############################################################################

CPP                 = /usr/ccs/lib/cpp
DEFINES             = -DSUN
CPPFLAGS            = -P

F90                 = f90
DEBUG               = -g
FFLAGS              = -fpp $(DEFINES) $(DEBUG) $(INCLUDE_DIRS) $(ADDRESS)

CC                  = CC
CFLAGS              = -xlang=f90 $(DEFINES) $(DEBUG) $(ADDRESS)

AR                  = /usr/local/gnu/bin/ar
```

```
ARFLAGS            = cru

LD                 = $(CC)
LIB_PATH           = ../../../lib
LDFLAGS            = $(CFLAGS) -L${LIB_PATH}/sun

SRC_DIR            = examples/C++

VPATH :            = ../../../$(SRC_DIR)

INCLUDE_DIRS += -I../../../include -I$(VPATH)
MD_INCLUDE_DIRS += -I ../../include

BUILD_LIB = -l$(LIBRARY)
OPT_LIBS =
INSTALL_SRC_FILES = example.cxx example.readme .make_depends
INSTALL_BINBUILD_FILES = example
INSTALL_TXTBUILD_FILES = Makefile tablesloaded.dat example.stdout
INSTALL = /usr/lanl/bin/install -c -m 664

# * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

all: example

compileMsg:
    @echo ' '
    @echo '***' Compiling files for $(MSG_NAME).
    @echo ' '

test: all
    @echo ' '
    @echo '***' This is the C++ example output. '***'
    @example
    @echo '***' This is the C++ example output. '***' > example.stdout
    @example >> example.stdout

example: override MSG_NAME = example
example: clean example.o
    @cp $(VPATH)/.make_depends .
    $(LD) -o example example.o $(LDFLAGS) $(BUILD_LIB) $(OPT_LIBS)
remove_dot.o:
    @rm -f *.o

install: override install_subdir = $(strip sun$(ADDRESS))
install: override exec_prefix = $(strip ${prefix}/${install_subdir})
install: test createinstalldirs remove_dot.o
    @rm -f *.o
    @echo ' '
    @echo '***' Installing example C++ program.
    @echo ' '
    @for myFile in                    ${INSTALL_SRC_FILES}; do \
        ${INSTALL}     $(VPATH)/$$myFile \
                       ${exec_prefix}/examples/C++/$$myFile; \
        echo ${exec_prefix}/examples/C++/$$myFile; \
    done;
    @for myFile in                    ${INSTALL_TXTBUILD_FILES}; do \
        ${INSTALL}     $$myFile ${exec_prefix}/examples/C++/$$myFile; \
        echo ${exec_prefix}/examples/C++/$$myFile; \
    done;
    for myFile in             ${INSTALL_BINBUILD_FILES}; do \
        ${INSTALL} -m 775 $$myFile ${exec_prefix}/examples/C++/$$myFile; \
        echo ${exec_prefix}/examples/C++/$$myFile; \
    done;

createinstalldirs: ../../../config/mkinstalldirs
```

```
    @echo ' '
    @echo '***' Creating/Verifying examples installation directory.
    @echo ' '
    @../../../config/mkinstalldirs ${exec_prefix}/examples/C++

depends:
    (cd $(VPATH); \
    touch .make_depends; \
    ../../sun/make_depends $(MD_INCLUDE_DIRS) *.cxx > /dev/null)

clean: remove_dot.o
    @rm -f example

distclean: clean
    @rm -f Makefile config.*

.PHONY :           all test example remove_dot.o get_dot.o depends \
    clean distclean install createinstalldirs compileMsg

.NOTPARALLEL:
include .make_depends
```

3. .make_depends

```
example.o: example.cxx \
        ../../include/eospac4c.h \
        ../../include/es4proto.h \
        ../../include/es4types.h \
        ../../include/es5err.h \
        ../../include/wrapperDefines.h \
```

4. example.stdout

```
*** This is the C++ example output. ***

DATA TYPE: 19 REGION: 1
     x          y           F        dF/dx     dF/dy
1.0000e+03 2.0000e+03 1.8975e+07 3.4545e+04 0.0000e+00
2.0000e+03 4.0000e+03 6.6337e+07 5.9686e+04 0.0000e+00
3.0000e+03 6.0000e+03 1.3640e+08 8.0805e+04 0.0000e+00
4.0000e+03 8.0000e+03 2.2778e+08 1.0106e+05 0.0000e+00
5.0000e+03 1.0000e+04 3.3702e+08 1.1679e+05 0.0000e+00
6.0000e+03 1.2000e+04 4.6163e+08 1.3383e+05 0.0000e+00
7.0000e+03 1.4000e+04 6.0465e+08 1.5194e+05 0.0000e+00
8.0000e+03 1.6000e+04 7.6498e+08 1.6847e+05 0.0000e+00
9.0000e+03 1.8000e+04 9.4112e+08 1.8358e+05 0.0000e+00
1.0000e+04 2.0000e+04 1.1317e+09 1.9743e+05 0.0000e+00
1.1000e+04 2.2000e+04 1.3356e+09 2.1017e+05 0.0000e+00
1.2000e+04 2.4000e+04 1.5517e+09 2.2191e+05 0.0000e+00
1.3000e+04 2.6000e+04 1.7791e+09 2.3274e+05 0.0000e+00
1.4000e+04 2.8000e+04 2.0177e+09 2.4568e+05 0.0000e+00
1.5000e+04 3.0000e+04 2.2709e+09 2.6069e+05 0.0000e+00
1.6000e+04 3.2000e+04 2.5388e+09 2.7489e+05 0.0000e+00
1.7000e+04 3.4000e+04 2.8205e+09 2.8834e+05 0.0000e+00
1.8000e+04 3.6000e+04 3.1152e+09 3.0109e+05 0.0000e+00
1.9000e+04 3.8000e+04 3.4224e+09 3.1319e+05 0.0000e+00
2.0000e+04 4.0000e+04 3.7414e+09 3.2468e+05 0.0000e+00

DATA TYPE: 20 REGION: 1
     x          y           F        dF/dx     dF/dy
1.0000e+03 2.0000e+03 2.0958e+04 1.8985e+01 0.0000e+00
2.0000e+03 4.0000e+03 3.8589e+04 1.6478e+01 0.0000e+00
3.0000e+03 6.0000e+03 5.4441e+04 1.5105e+01 0.0000e+00
4.0000e+03 8.0000e+03 6.8938e+04 1.4067e+01 0.0000e+00
```

```
5.0000e+03 1.0000e+04 8.2771e+04 1.3652e+01 0.0000e+00
6.0000e+03 1.2000e+04 9.6148e+04 1.2935e+01 0.0000e+00
7.0000e+03 1.4000e+04 1.0870e+05 1.2224e+01 0.0000e+00
8.0000e+03 1.6000e+04 1.2068e+05 1.1756e+01 0.0000e+00
9.0000e+03 1.8000e+04 1.3226e+05 1.1431e+01 0.0000e+00
1.0000e+04 2.0000e+04 1.4357e+05 1.1196e+01 0.0000e+00
1.1000e+04 2.2000e+04 1.5467e+05 1.1021e+01 0.0000e+00
1.2000e+04 2.4000e+04 1.6562e+05 1.0887e+01 0.0000e+00
1.3000e+04 2.6000e+04 1.7645e+05 1.0782e+01 0.0000e+00
1.4000e+04 2.8000e+04 1.8714e+05 1.0509e+01 0.0000e+00
1.5000e+04 3.0000e+04 1.9746e+05 1.0142e+01 0.0000e+00
1.6000e+04 3.2000e+04 2.0746e+05 9.8639e+00 0.0000e+00
1.7000e+04 3.4000e+04 2.1721e+05 9.6484e+00 0.0000e+00
1.8000e+04 3.6000e+04 2.2677e+05 9.4780e+00 0.0000e+00
1.9000e+04 3.8000e+04 2.3618e+05 9.3409e+00 0.0000e+00
2.0000e+04 4.0000e+04 2.4546e+05 9.2290e+00 0.0000e+00

DATA TYPE: 5 REGION: 1
     x          y           F        dF/dx     dF/dy
1.0000e+03 2.0000e+03 1.8975e+07 2.3847e+04 5.9051e+03
2.0000e+03 4.0000e+03 6.6337e+07 4.0636e+04 1.1153e+04
3.0000e+03 6.0000e+03 1.3640e+08 5.4108e+04 1.6037e+04
4.0000e+03 8.0000e+03 2.2778e+08 6.7336e+04 2.0659e+04
5.0000e+03 1.0000e+04 3.3702e+08 7.5440e+04 2.5069e+04
6.0000e+03 1.2000e+04 4.6163e+08 8.5996e+04 2.9628e+04
7.0000e+03 1.4000e+04 6.0465e+08 9.8248e+04 3.4105e+04
8.0000e+03 1.6000e+04 7.6498e+08 1.0866e+05 3.8373e+04
9.0000e+03 1.8000e+04 9.4112e+08 1.1747e+05 4.2498e+04
1.0000e+04 2.0000e+04 1.1317e+09 1.2488e+05 4.6520e+04
1.1000e+04 2.2000e+04 1.3356e+09 1.3106e+05 5.0466e+04
1.2000e+04 2.4000e+04 1.5517e+09 1.3616e+05 5.4355e+04
1.3000e+04 2.6000e+04 1.7791e+09 1.4028e+05 5.8200e+04
1.4000e+04 2.8000e+04 2.0177e+09 1.4779e+05 6.2258e+04
1.5000e+04 3.0000e+04 2.2709e+09 1.5832e+05 6.6439e+04
1.6000e+04 3.2000e+04 2.5388e+09 1.6764e+05 7.0492e+04
1.7000e+04 3.4000e+04 2.8205e+09 1.7591e+05 7.4445e+04
1.8000e+04 3.6000e+04 3.1152e+09 1.8325e+05 7.8320e+04
1.9000e+04 3.8000e+04 3.4224e+09 1.8975e+05 8.2133e+04
2.0000e+04 4.0000e+04 3.7414e+09 1.9550e+05 8.5894e+04

ES1ERRMSG subroutine test
0:  No Error
1:  eos table area is not initialized
2:  table type index is out of bounds
3:  region index is out of bounds
4:  data table is not in eos table area
5:  derivative code is not recognized
6:  interpolation code is not recognized
7:  matid is zero
8:  second dimension of fvals is not large enough
9:  data table type is not recognized
10: no data library files exist
11: material id is not in library
12: data table type is not in library
13: file index outside of range
14: eos table area cannot be expanded
15: could not open data file
16: could not read number of materials
17: could not read version number from file
18: could not load master directory
19: could not load material directory
20: could not load data table
21: unknown HDF tag
22: host did not request needed cold curve for grid refinement
23: iteration did not converge
```

Appendix D

F77 Example

1. example.f

```
      program f77example
c
c     Demonstrate the usage of ES1TABS and ES1VALS
c
      implicit none
      include 'es5errf.h'
      include 'eospac4f77.h'

c     ES1TABS subroutine parameters
      integer mtyps, mregs
      parameter(mtyps=3)        ! number of tables to load
      parameter(mregs=1)        ! number of problem regions
      logical llog1(mtyps*mregs)
      integer iopt(mtyps*mregs)
      logical lprnt
      integer iprnt
      integer inams(mtyps)
      real*8  ucons(mtyps*3)
      integer imids(mtyps*mregs)
      integer idtab, mtabs
      real*8  ftabs
      pointer (ktabs,ftabs(1))
      integer ierrs(mtyps*mregs)

c     ES1VALS subroutine parameters
      integer idrvs, intrp, iregn, nzons, ierr1
      parameter(nzons=20)       ! number of interpolated F values
                                ! to return (zones)
      real*8  xvals(nzons)      ! x values for interpolation
      real*8  yvals(nzons)      ! y values for interpolation
      real*8  fvals(nzons,3)    ! F, dF/dx, and dF/dy in each zone

c     ES1INFO subroutine parameters
      integer iname, matid, ifile;
      logical llogs;
      real*8  xcnvt, ycnvt, fcnvt, znbar, anbar, dens0;

c     Local variables
      integer i, j, k
      integer errorCode
      character*60 errorMessage

c     Assign subroutine parameter values
      mtabs = 20000             ! arbitrary intial words in ftabs
      lprnt = .TRUE.            ! dump loaded tables to an ascii file
      iprnt = 0                 ! arbitrary value -- no longer used
      idtab = 0                 ! arbitrary value -- no longer used
      inams(1) = ES4_PRCLD      ! pressure cold curve
      inams(2) = ES4_ENCLD      ! energy cold curve
      inams(3) = ES4_PNTOT      ! total pressure (energy-based)
      do j = 1, mtyps
         ucons(j        ) = 1.  ! table x units conversion factor
         ucons(j+  mtyps) = 1.  ! table y units conversion factor
         ucons(j+2*mtyps) = 1.  ! table F(x,y) units conversion factor
         do i = 1, mregs
            k = j+(i-1)*mtyps
            llog1(k) = .FALSE.  ! don't return log10 tables for
```

```
                                 ! 300-tables
            iopt(k)   = 0        ! perform no data smoothing
            imids(k)  = 3717     ! SESAME material ID number
            ierrs(k)  = ES5_OK   ! initialize error flag
         enddo
      enddo

c     Allocate memory for tables to be loaded
      call es4alloc(mtabs, ktabs, ierr1)
      if (ierr1.NE.ES5_OK) then
         call es1errmsg(ierrs(k), errorMessage)
         write(*,'(a,i3,a,i3,a)') 'LOAD ERROR',ierrs(k),
     &        ', TABLE',inams(k),': '//errorMessage
      endif

c     Load desired data tables
      call es1tabs(llog1, iopt,  lprnt, iprnt, mtyps,
     &     mregs, inams, ucons, imids,
     &     idtab, mtabs, ktabs, ierrs)
      do j = 1, mtyps
         do i = 1, mregs
            k = j+(i-1)*mtyps
            if (ierrs(k).NE.ES5_OK) then
               call es1errmsg(ierrs(k), errorMessage)
               write(*,'(a,i3,a,i3,a)') 'LOAD ERROR',ierrs(k),
     &              ', TABLE',inams(k),': '//errorMessage
            endif
         enddo
      enddo

c     Assign xvals and yvals array values
      do i = 1, nzons
         xvals(i) = dble(i*1000.0)
         yvals(i) = dble(i*2000.0)
      enddo

c     Perform interpolation for each data type
      idrvs = ES4_DERIVS        ! return F, dF/dx, and dF/dy
      intrp = ES4_BIRATF        ! use bi-rational interpolation
      iregn = 1                 ! only 1 region has been used in
                                ! this example
      do i = 1, mtyps
         call es1info(i, iregn, ktabs, iname, llogs, xcnvt, ycnvt,
     &                fcnvt, matid, znbar, anbar, dens0, ifile, ierr1)
         if (ierr1.NE.ES5_OK) then
            call es1errmsg(ierr1, errorMessage)
            write(*,'(a,i3,a,i3,a)') 'INFO ERROR',ierr1,
     &           ', TABLE',inams(i),': '//errorMessage
         endif
         call es1vals(inams(i), idrvs, intrp, ftabs, mtabs,
     &                nzons, iregn, xvals, yvals, fvals, 3, ierr1)
         if (ierr1.NE.ES5_OK) then
            call es1errmsg(ierr1, errorMessage)
            write(*,'(a,i3,a,i3,a)') 'INTERPOLATION ERROR',ierr1,
     &           ', TABLE',inams(i),': '//errorMessage
         else
c           convert partial derivatives from dF/dlnx to dFdx and dF/dlny to dFdy
            do j = 1, nzons
               fvals(j,2) = fvals(j,2) / max(xvals(j),tiny)
               fvals(j,3) = fvals(j,3) / max(yvals(j),tiny)
            enddo
            call display(inams(i),iregn,nzons,xvals,yvals,fvals)
         endif
      enddo
```

```
c     Display all possible error codes and corresponding messages
      write(*,*)
      write(*,*) 'ES1ERRMSG subroutine test'
      do i=0,23
         errorCode = i
         call es1errmsg(errorCode, errorMessage)
         write(*,'(i2,a)') errorCode, ': '//errorMessage
      enddo
      end

      subroutine display(inam,iregn,nzons,xvals,yvals,fvals)
      integer inam, iregn, nzons
      real*8  xvals(nzons), yvals(nzons), fvals(nzons,3)
      integer j
      write(*,1) "DATA TYPE:",inam,"    REGION:",iregn
      write(*,2) "x","y","F","dF/dx","dF/dy"
      do j = 1, nzons
         write(*,3) xvals(j),yvals(j),fvals(j,1),fvals(j,2),fvals(j,3)
      enddo
 1    format(/2(a,i2))
 2    format(5a12)
 3    format(1p,5e12.4)
      end
```

2. Makefile

```
# * * * * * * * * * * * * * * * * * * * * * * * * * * *
#
# "F77 example program" GNU Make Makefile
#
# * * * * * * * * * * * * * * * * * * * * * * * * * * *

LIBRARY = eospac

.SUFFIXES:
.SUFFIXES: .F .f .cxx .c .o .a

.F.o :
       $(F90) $(FFLAGS) -c $<

.f.o :
       $(F90) $(FFLAGS) -c $<

.cxx.o :
       ${CC} ${CFLAGS} ${DEFINES} ${INCLUDE_DIRS} -c $<

.c.o :
       ${CC} ${CFLAGS} ${DEFINES} ${INCLUDE_DIRS} -c $<

##############################################################################
# gmake function to search $(2) for lib$(1).a where $(1) is a
# space-delimited list of library name(s) to find, and $(2) is a
# space-delimited list of environment variable(s).
# Usage example:
# CLIBS = $(call libSearch,gcc stdc++,$(LD_LIBRARY_PATH) $(LD_LIBRARY64_PATH))
libSearch = $(foreach a, $(1), \
       $(firstword $(shell find $(subst :, ,$(2)) -name lib$(a).a)))
##############################################################################

CPP            = /usr/ccs/lib/cpp
DEFINES        = -DSUN
CPPFLAGS       = -P

F90            = f90
DEBUG          = -g
```

```
FFLAGS          = -fpp $(DEFINES) $(DEBUG) $(INCLUDE_DIRS) $(ADDRESS)

CC              = CC
CFLAGS          = -xlang=f90 $(DEFINES) $(DEBUG) $(ADDRESS)

CLIBS           = $(call libSearch,,)

AR              = /usr/local/gnu/bin/ar
ARFLAGS         = cru

LD              = $(CC) $(CFLAGS) $(LDFLAGS)
LIB_PATH        = ../../../lib
LDFLAGS         = -L. -lc -lnsl -L${LIB_PATH}/sun

SRC_DIR         = examples/F77

VPATH :         = ../../../$(SRC_DIR)

INCLUDE_DIRS += -I../../../include -I$(VPATH)
MD_INCLUDE_DIRS += -I ../../include



BUILD_LIB       = -l$(LIBRARY)
OPT_LIBS        =
INSTALL_SRC_FILES = example.f example.readme
INSTALL_BINBUILD_FILES = example
INSTALL_TXTBUILD_FILES = example.stdout Makefile tablesloaded.dat
INSTALL         = /usr/lanl/bin/install -c -m 664

# * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

all: example

compileMsg:
        @echo ' '
        @echo '***' Compiling files for $(MSG_NAME).
        @echo ' '

test: all
        @echo ' '
        @echo '***' This is the F77 example output. '***'
        @example
        @echo '***' This is the F77 example output. '***' > example.stdout
        @example >> example.stdout

example: override MSG_NAME = example
example: clean example.o
        $(LD) -o example example.o $(LDFLAGS) $(BUILD_LIB) $(OPT_LIBS) $(CLIBS)

remove_dot.o:
        @rm -f *.o

install: override install_subdir = $(strip sun$(ADDRESS))
install: override exec_prefix = $(strip ${prefix}/${install_subdir})
install: test createinstalldirs remove_dot.o
        @rm -f *.o
        @echo ' '
        @echo '***' Installing example F77 program.
        @echo ' '
        @for myFile in         ${INSTALL_SRC_FILES}; do \
                {INSTALL}      $(VPATH)/$$myFile \
                        ${exec_prefix}/examples/F77/$$myFile; \
                echo ${exec_prefix}/examples/F77/$$myFile; \
        done;
```

```
        @for myFile in                ${INSTALL_TXTBUILD_FILES}; do \
                ${INSTALL}     $$myFile ${exec_prefix}/examples/F77/$$myFile; \
                echo ${exec_prefix}/examples/F77/$$myFile; \
        done;
        @for myFile in        ${INSTALL_BINBUILD_FILES}; do \
                ${INSTALL} -m 775 $$myFile ${exec_prefix}/examples/F77/$$myFile; \
                echo ${exec_prefix}/examples/F77/$$myFile; \
        done;

createinstalldirs: ../../../config/mkinstalldirs
        @echo ' '
        @echo '***' Creating/Verifying examples installation directory.
        @echo ' '
        @../../../config/mkinstalldirs ${exec_prefix}/examples/F77

clean: remove_dot.o
        @rm -f example*

distclean: clean
        @rm -f Makefile config.*

.NOTPARALLEL:

.PHONY :   all test example remove_dot.o get_dot.o \
           clean distclean install createinstalldirs compileMsg
```

3. example.stdout

```
*** This is the F77 example output. ***

DATA TYPE:19 REGION: 1
     x          y           F       dF/dx      dF/dy
1.0000E+03 2.0000E+03 1.8975E+07 3.4545E+04 0.0000E+00
2.0000E+03 4.0000E+03 6.6337E+07 5.9686E+04 0.0000E+00
3.0000E+03 6.0000E+03 1.3640E+08 8.0805E+04 0.0000E+00
4.0000E+03 8.0000E+03 2.2778E+08 1.0106E+05 0.0000E+00
5.0000E+03 1.0000E+04 3.3702E+08 1.1679E+05 0.0000E+00
6.0000E+03 1.2000E+04 4.6163E+08 1.3383E+05 0.0000E+00
7.0000E+03 1.4000E+04 6.0465E+08 1.5194E+05 0.0000E+00
8.0000E+03 1.6000E+04 7.6498E+08 1.6847E+05 0.0000E+00
9.0000E+03 1.8000E+04 9.4112E+08 1.8358E+05 0.0000E+00
1.0000E+04 2.0000E+04 1.1317E+09 1.9743E+05 0.0000E+00
1.1000E+04 2.2000E+04 1.3356E+09 2.1017E+05 0.0000E+00
1.2000E+04 2.4000E+04 1.5517E+09 2.2191E+05 0.0000E+00
1.3000E+04 2.6000E+04 1.7791E+09 2.3274E+05 0.0000E+00
1.4000E+04 2.8000E+04 2.0177E+09 2.4568E+05 0.0000E+00
1.5000E+04 3.0000E+04 2.2709E+09 2.6069E+05 0.0000E+00
1.6000E+04 3.2000E+04 2.5388E+09 2.7489E+05 0.0000E+00
1.7000E+04 3.4000E+04 2.8205E+09 2.8834E+05 0.0000E+00
1.8000E+04 3.6000E+04 3.1152E+09 3.0109E+05 0.0000E+00
1.9000E+04 3.8000E+04 3.4224E+09 3.1319E+05 0.0000E+00
2.0000E+04 4.0000E+04 3.7414E+09 3.2468E+05 0.0000E+00

DATA TYPE:20 REGION: 1
     x          y           F       dF/dx      dF/dy
1.0000E+03 2.0000E+03 2.0958E+04 1.8985E+01 0.0000E+00
2.0000E+03 4.0000E+03 3.8589E+04 1.6478E+01 0.0000E+00
3.0000E+03 6.0000E+03 5.4441E+04 1.5105E+01 0.0000E+00
4.0000E+03 8.0000E+03 6.8938E+04 1.4067E+01 0.0000E+00
5.0000E+03 1.0000E+04 8.2771E+04 1.3652E+01 0.0000E+00
6.0000E+03 1.2000E+04 9.6148E+04 1.2935E+01 0.0000E+00
7.0000E+03 1.4000E+04 1.0870E+05 1.2224E+01 0.0000E+00
8.0000E+03 1.6000E+04 1.2068E+05 1.1756E+01 0.0000E+00
9.0000E+03 1.8000E+04 1.3226E+05 1.1431E+01 0.0000E+00
1.0000E+04 2.0000E+04 1.4357E+05 1.1196E+01 0.0000E+00
```

```
1.1000E+04 2.2000E+04 1.5467E+05 1.1021E+01 0.0000E+00
1.2000E+04 2.4000E+04 1.6562E+05 1.0887E+01 0.0000E+00
1.3000E+04 2.6000E+04 1.7645E+05 1.0782E+01 0.0000E+00
1.4000E+04 2.8000E+04 1.8714E+05 1.0509E+01 0.0000E+00
1.5000E+04 3.0000E+04 1.9746E+05 1.0142E+01 0.0000E+00
1.6000E+04 3.2000E+04 2.0746E+05 9.8639E+00 0.0000E+00
1.7000E+04 3.4000E+04 2.1721E+05 9.6484E+00 0.0000E+00
1.8000E+04 3.6000E+04 2.2677E+05 9.4780E+00 0.0000E+00
1.9000E+04 3.8000E+04 2.3618E+05 9.3409E+00 0.0000E+00
2.0000E+04 4.0000E+04 2.4546E+05 9.2290E+00 0.0000E+00

DATA TYPE:5 REGION: 1
     x          y           F       dF/dx      dF/dy
1.0000E+03 2.0000E+03 1.8975E+07 2.3847E+04 5.9051E+03
2.0000E+03 4.0000E+03 6.6337E+07 4.0636E+04 1.1153E+04
3.0000E+03 6.0000E+03 1.3640E+08 5.4108E+04 1.6037E+04
4.0000E+03 8.0000E+03 2.2778E+08 6.7336E+04 2.0659E+04
5.0000E+03 1.0000E+04 3.3702E+08 7.5440E+04 2.5069E+04
6.0000E+03 1.2000E+04 4.6163E+08 8.5996E+04 2.9628E+04
7.0000E+03 1.4000E+04 6.0465E+08 9.8248E+04 3.4105E+04
8.0000E+03 1.6000E+04 7.6498E+08 1.0866E+05 3.8373E+04
9.0000E+03 1.8000E+04 9.4112E+08 1.1747E+05 4.2498E+04
1.0000E+04 2.0000E+04 1.1317E+09 1.2488E+05 4.6520E+04
1.1000E+04 2.2000E+04 1.3356E+09 1.3106E+05 5.0466E+04
1.2000E+04 2.4000E+04 1.5517E+09 1.3616E+05 5.4355E+04
1.3000E+04 2.6000E+04 1.7791E+09 1.4028E+05 5.8200E+04
1.4000E+04 2.8000E+04 2.0177E+09 1.4779E+05 6.2258E+04
1.5000E+04 3.0000E+04 2.2709E+09 1.5832E+05 6.6439E+04
1.6000E+04 3.2000E+04 2.5388E+09 1.6764E+05 7.0492E+04
1.7000E+04 3.4000E+04 2.8205E+09 1.7591E+05 7.4445E+04
1.8000E+04 3.6000E+04 3.1152E+09 1.8325E+05 7.8320E+04
1.9000E+04 3.8000E+04 3.4224E+09 1.8975E+05 8.2133E+04
2.0000E+04 4.0000E+04 3.7414E+09 1.9550E+05 8.5894E+04

ES1ERRMSG subroutine test
0:  No Error
1:  eos table area is not initialized
2:  table type index is out of bounds
3:  region index is out of bounds
4:  data table is not in eos table area
5:  derivative code is not recognized
6:  interpolation code is not recognized
7:  matid is zero
8:  second dimension of fvals is not large enough
9:  data table type is not recognized
10: no data library files exist
11: material id is not in library
12: data table type is not in library
13: file index outside of range
14: eos table area cannot be expanded
15: could not open data file
16: could not read number of materials
17: could not read version number from file
18: could not load master directory
19: could not load material directory
20: could not load data table
21: unknown HDF tag
22: host did not request needed cold curve for grid refinement
23: iteration did not converge
```