

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. Reference herein to any social initiative (including but not limited to Diversity, Equity, and Inclusion (DEI); Community Benefits Plans (CBP); Justice 40; etc.) is made by the Author independent of any current requirement by the United States Government and does not constitute or imply endorsement, recommendation, or support by the United States Government or any agency thereof.

LA-UR-23-29913

Approved for public release; distribution is unlimited.

Title: EOSPAC User's Manual: Version 6.5 Second Edition

Author(s): Pimentel, David A.

Intended for: Report

Issued: 2025-08-08 (rev.3)



Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by Triad National Security, LLC for the National Nuclear Security Administration of U.S. Department of Energy under contract 89233218CNA000001. By approving this article, the publisher recognizes that the U.S. Government retains 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.

EOSPAC User's Manual: Version 6.5

Second Edition

LA-UR-23-29913

Artifact ID: EOSPAC6-02-01

DAVID A. PIMENTEL, XCP-5, LOS ALAMOS NATIONAL LABORATORY

JULY 14, 2025

Revision 3



-

This program was prepared by Triad National Security, LLC at Los Alamos National Laboratory (LANL) under contract No. 89233218CNA000001 with the U.S. Department of Energy (DOE). All rights in the program are reserved by the DOE and Triad National Security, LLC. Permission is granted to the public to copy and use this software without charge, provided that this Notice and any statement of authorship are reproduced on all copies. Neither the U.S. Government nor Triad makes any warranty, express or implied, or assumes any liability or responsibility for the use of this software.

Contents

Title	i
Copyright	ii
List of Figures	vii
List of Tables	ix
1 INTRODUCTION	1
2 CONVENTIONS	3
1 DATA ORGANIZATION	3
2 ROUTINE NAMES	4
3 CONSTANT IDENTIFIER NAMES	4
4 DATA TYPES	5
5 ANNOTATED UPDATES	5
3 BASIC THEORY AND MODELS	7
1 Nomenclature	7
2 Entropy	8
3 Ion EOS Models	10
3.1 Ideal Gas Model	11
3.2 Cowan Model	12
3.3 Number Proportional Model	15
4 Additional Thermodynamic Quantities	17
4.1 Identities	17
4.2 Sound speed	18
4.3 Isentropic Compressibility	19
4.4 Isothermal Compressibility	19
4.5 Gruneisen Coefficient	20
4.6 Specific heats	20
4.7 Thermal expansion alpha	23
4.8 Equilibrium mixture quantities	23
4 GENERAL INTERFACE DESCRIPTION	27
1 USE CASES	28
1.1 Serial Case	28
1.2 Parallel Case	29
1.3 Parallel Case with Shared Memory	30
5 SETUP MATERIAL DATA	35

1	DATA LOCATIONS	35
1.1	Environment-variable-defined and default search paths	36
1.2	Ordered File Names List Creation	36
1.3	Index file	36
1.4	Default file name list	39
1.5	Ordered File Names List Example	40
2	DATA ORGANIZATION	42
3	ROUTINES AND PARAMETERS	44
3.1	eos_CreateTables	44
3.2	eos_DestroyAll	45
3.3	eos_DestroyTables	46
3.4	eos_GetMaxDataFileNameLength	47
3.5	eos_GetPackedTables	47
3.6	eos_GetPackedTablesSize	48
3.7	eos_GetSharedPackedTablesSize	49
3.8	eos_GpuOffloadData	50
3.9	eos_LoadTables	52
3.10	eos_SetDataFileName	53
3.11	eos_SetPackedTables	54
3.12	eos_SetSharedPackedTables	55
4	C/C++ LANGUAGE BINDINGS	57
5	FORTRAN LANGUAGE BINDINGS	59
6	INTERPOLATE MATERIAL DATA	61
1	DATA ORGANIZATION	61
2	ROUTINES AND PARAMETERS	61
2.1	eos_CheckExtrap	62
2.2	eos_Interpolate	64
2.3	eos_Mix	66
3	C/C++ LANGUAGE BINDINGS	68
4	FORTRAN LANGUAGE BINDINGS	69
7	MISCELLANEOUS INFORMATION ROUTINES	71
1	ROUTINES AND PARAMETERS	71
1.1	eos_ErrorCodesEqual	71
1.2	eos_GetErrorCode	72
1.3	eos_GetErrorMessage	72
1.4	eos_GetTableCmnts	73
1.5	eos_GetTableInfo	74
1.6	eos_GetMetaData	75
1.7	eos_GetTableMetaData	75
1.8	eos_GetVersion	76

1.9	eos_GetVersionLength	76
1.10	eos_ResetOption	77
1.11	eos_SetOption	78
2	C/C++ LANGUAGE BINDINGS	78
3	FORTRAN LANGUAGE BINDINGS	80
8	TOOLS	83
9	SELECTED NUMERIC DETAILS	87
1	CUSTOM SMOOTHING AND INTERPOLATION	87
2	FORCED DATA MONOTONICITY	88
3	EXTENDED PRECISION IS DISABLED	89
4	MASS FRACTION DATA INTERPOLATION	90
5	NUMERICAL INTEGRATION	91
6	LINEAR AND BILINEAR INTERPOLATION	92
7	INVERT AT SETUP	94
7.1	Data Transformations	94
7.2	Usage of EOS_INSERT_DATA	99
7.3	The EOS_X_CONVERT, EOS_Y_CONVERT and EOS_Y_CONVERT options usage	100
7.4	Usage of EOS_X_CONVERT, EOS_Y_CONVERT, EOS_XY_MODIFY, EOS_XY_PASSTHRU and EOS_USE_HOST_XY	102
10	USAGE EXAMPLES	107
1	C HOST CODE EXAMPLE	107
2	C++ HOST CODE EXAMPLE	113
3	FORTRAN 77 HOST CODE EXAMPLE	120
4	FORTRAN 90 HOST CODE EXAMPLE	127
5	SHARED MEMORY EXAMPLE	133
11	TECHNICAL SUPPORT INFORMATION	139
12	ACKNOWLEDGEMENTS	141
13	BIBLIOGRAPHY	143
14	APPENDIX	147
A	TABLE TYPES: <i>MNEMONIC CONVENTIONS</i>	149
B	TABLE TYPES: <i>GROUPED BY CATEGORY, SORTED BY NAME</i>	153
B.1	Category 1: Unrelated to SESAME data	154
B.2	Category 2: General information found in SESAME's 100- and 200-series tables	155
B.3	Category 3: Total EOS in SESAME's 301 tables	156
B.4	Category 4: Ion+Cold EOS in SESAME's 303 tables	161
B.5	Category 5: Electron EOS in SESAME's 304 tables	166
B.6	Category 6: Ion EOS in SESAME's 305 tables	170
B.7	Category 7: Cold curve EOS in SESAME's 306 tables	175
B.8	Category 8: Mass fraction EOS in SESAME's 321 tables	176

B.9	Category 9: Vaporization data in SESAME's 401 tables	177
B.10	Category 10: Melt data in SESAME's 411 and 412 tables	182
B.11	Category 11: Shear Modulus data in SESAME's 431 tables	185
B.12	Category 12: Opacity data in SESAME's 500-series tables	186
B.13	Category 13: Conductivity data in SESAME's 600-series tables	187
C	TABLE TYPES: <i>EOSPAC VERSION 5 CROSS REFERENCE</i>	189
D	OPTIONS: <i>SETUP PHASE</i>	195
E	DATA INFORMATION PARAMETERS	201
F	META-DATA INFORMATION PARAMETERS	211
G	OPTIONS: <i>INTERPOLATION PHASE</i>	215
H	ERROR CODES	219

List of Figures

3.1	Dimensionless parameters from the high-temperature solid entropy expression and the new low-temperature solid entropy expression.	15
4.1	General graphical description of the public user interface of EOSPAC.	28
4.2	Serial host code use of EOSPAC.	29
4.3	Parallel host code use of EOSPAC.	30
4.4	Parallel host code with shared memory use of EOSPAC.	31
5.1	Flowchart description of file search algorithms.	37
5.2	Example of sesameFilesDir.txt.	38
5.3	Input/output data organization.	43
5.4	Duplicate data organization.	43
5.5	Usage of eos_GpuOffloadData in relation to the various phases of EOSPAC operation.	51
9.1	General depiction of $P(\rho)$ forced to be monotonically-increasing.	89
9.2	Comparison of derivative values for three low temperature isotherms using the EOS_LINEAR interpolation option both with and without the EOS_DISCONTINUOUS_DERIVATIVES option enabled.	92
9.3	Comparison of derivative values for three low temperature isotherms using the EOS_LINEAR interpolation option both with and without the EOS_DISCONTINUOUS_DERIVATIVES option enabled. This demonstrates numerical issues with the current default bilinear interpolator's continuous derivatives at or near the data table boundary.	93
9.4	Rectangular SESAME grid of density (D) and temperature (T).	96
9.5	Non-rectangular SESAME grid of density (D) and internal energy (Ut).	96
9.6	Non-rectangular SESAME grid of density (D) and pressure (Pt).	97
9.7	Non-rectangular SESAME grid of pressure (Pt) and temperature (T).	97
9.8	Transformed, rectangular SESAME grid of internal energy (U^*) and temperature (T).	98
9.9	Transformed, rectangular SESAME grid of pressure (P^*) and temperature (T).	98
9.10	Color map of relative differences comparing the interpolation of $T(\rho, Pt)$, which was calculated using both of EOSPAC 6's default and pre-inverted (i.e., EOS_INVERT_AT_SETUP) inverse interpolation modes.	99

- 9.11 Color map of relative differences comparing the interpolation of $T(\rho, Pt)$, which was calculated using both of EOSPAC 6's default and pre-inverted (i.e., EOS_INVERT_AT_SETUP) inverse interpolation modes and the EOS_INSERT_DATA=2 option enabled. 100
- 9.12 Comparison of interpolated results using various combinations of the EOS_X_CONVERT, EOS_Y_CONVERT, EOS_XY_MODIFY, EOS_XY_PASSTHRU options and a convenient conversion factor, 2.0. 103
- 9.13 Comparison of interpolated results using various combinations of the EOS_X_CONVERT, EOS_Y_CONVERT, EOS_XY_MODIFY, EOS_USE_HOST_XY options and a convenient conversion factor, 2.0. 104
-

List of Tables

5.2	Ordered list of default SESAME file names.	39
5.2	Ordered list of default SESAME file names. <i>(Continued from previous page.)</i>	40
5.8	Some typical values of PATH_MAX.	47
6.3	Extrapolation return codes.	63
6.3	Extrapolation return codes. <i>(Continued from previous page.)</i>	64
E-1	Information parameter(s) related to SESAME's 100-series tables	201
E-2	Information parameter(s) related to SESAME's 201 tables	201
E-3	Information parameter(s) generally related to SESAME's tables except for SESAME's 100-series and 201 tables	202
E-3	Information parameter(s) generally related to SESAME's tables except for SESAME's 100-series and 201 tables	203
E-4	Information parameter(s) generally related to SESAME's tables	204
E-5	Information parameter(s) associated with non-inverted data tables	204
E-5	Information parameter(s) associated with non-inverted data tables	205
E-6	Information parameter(s) associated with species-specific stored data. For each option listed in this table, the extent of the returned array corresponds to the nXYPairs value supplied to the eos_Mix routine, and sufficient memory must be made available by the host code to store each requested array. See the EOS_SAVE_SPECIES_DATA option in APPENDIX D.	206
E-7	Internally-defined balance functions used by the eos_Mix routine to achieve the desired pressure and temperature equilibrium for the interpolated data corresponding to the defined EOSPAC 6 Constant (i.e., table type).	206
E-7	Internally-defined balance functions used by the eos_Mix routine to achieve the desired pressure and temperature equilibrium for the interpolated data corresponding to the defined EOSPAC 6 Constant (i.e., table type).	207
E-7	Internally-defined balance functions used by the eos_Mix routine to achieve the desired pressure and temperature equilibrium for the interpolated data corresponding to the defined EOSPAC 6 Constant (i.e., table type).	208

E-7	Internally-defined balance functions used by the eos_Mix routine to achieve the desired pressure and temperature equilibrium for the interpolated data corresponding to the defined EOSPAC 6 Constant (i.e., table type).	209
F-1	Information parameter(s) used for the first argument (infoItem) of the eos_GetMetaData routine	211
F-2	Information parameter(s) used for the second argument (infoItemCategory) of the eos_GetMetaData routine	211
F-2	Information parameter(s) used for the second argument (infoItemCategory) of the eos_GetMetaData routine	212
F-3	Information parameter(s) used for the second argument (infoItem) of the eos_GetTableMetaData routine	212
F-3	Information parameter(s) used for the second argument (infoItem) of the eos_GetTableMetaData routine	213

1 INTRODUCTION

“Begin at the beginning,” the King said, gravely, “and go on till you come to an end; then stop.”

– Lewis Carroll, *Alice in Wonderland*

The EOSPAC utility package is a collection of interface routines, which can be used to access the SESAME data library and perform various data adjustments and interpolations on the SESAME data. The SESAME data library[1] contains both thermodynamic (e.g., equation of state) and transport coefficients (e.g., opacity and conductivity). Note, for simplicity, the term EOS (equation of state) used herein 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. The remainder of this manual is organized into several sections. [Chapter 2](#) discusses conventions such as data organization and routine names. [Chapter 3](#) provides a general overview of basic theory and models implemented within EOSPAC. [Chapter 4](#) provides a general overview of how to use the EOSPAC interface library. [Chapters 5 to 7](#) describe the public interfaces of EOSPAC in detail. [Chapter 8](#) provides a brief introduction to some related tools, which may be of use to the user. [Chapter 9](#) provides details related to some selected numerical features of EOSPAC. [Chapter 10](#) gives examples for using the interface routines described in [chapters 5 to 7](#). [Chapter 11](#) provides technical support contact information. [Chapter 12](#) contains a brief set of acknowledgments. [Chapter 13](#) contains a list of referenced documents. Finally, [chapter 14](#) lists the “table types: *mnemonic conventions*”, “table types: *grouped by category, sorted by name*”, “table types: *eospac version 5 cross reference*”, “options: *setup phase*”, “data information parameters”, “meta-data information parameters”, “options: *interpolation phase*”, and the “error codes”.

2 CONVENTIONS

I'm a sworn enemy of convention. I despise the conventional in anything, even the arts.

– Hedy Lamarr

In spite of the opening quotation, several conventions are used throughout this document, and they are described in this chapter. These conventions are categorized as “data organization”, “routine names”, “constant identifier names”, and “data types”.

1 DATA ORGANIZATION

Conceptually EOSPAC is organized around data tables. A data table is specified by the material identification number, by the table type (e.g. pressure as a function of density and temperature), and the processing options (e.g. smoothed, monotonic, etc.). Two data tables differ if any option differs; thus, a smoothed data table is different than a monotonic data table. This is just common sense because the values returned for the two data tables will be different. The i -th data table will be referred to as T_i in the context of a routine’s argument descriptions.

A table handle is used to access the data table. The table handle is a language independent mechanism for a host code to access a specific instance of the data tables being managed by EOSPAC. Note that table handles are not implemented using native language pointers. The details of establishing a table handle is discussed in [chapter 5](#) and usage is shown in [chapters 6](#) and [7](#).

Multiple table handles are returned from the setup routine within a user-supplied array. The host code then uses the table handles to specify on which data tables EOSPAC is to operate. Typical operations are interpolating to get data at points desired by the host code, and to destroy the data tables.

2 ROUTINE NAMES

Routine name standardization is applied according to the following rules:

1. EOSPAC is a package of routines that provides a cohesive set of logically related functionality to host codes. The package name “eos_” (or the internal variant “_eos_”) is used as a prefix for all routine names in the package. This practically guarantees unique routine names when linked to the host codes. The prefix of a routine name allows users to instantly identify the physical package from which it came, and the prefix gives users a hint about functionality.
2. A routine name takes the form of ActionSubset where Action specifies a given operation and the optional Subset specifies a property, information, etc. The complete name will be eos_ActionSubset (or the internal variant _eos_ActionSubset).
3. The names of certain actions on tables have been standardized. The standardized action names are as follows:
 - “Create” will instantiate data object(s) to store a table or collection of tables
 - “Destroy” will destroy a table or collection of tables
 - “Get” retrieves information about a table
 - “Interpolate” performs interpolation using the table’s member data
 - “Load” will create a new table and fill the table’s members with appropriate information
 - “Reset” reasserts any default information to a table (i.e., option setting)
 - “Set” assigns information to a table (i.e., option setting)

To summarize, routine names are generally defined by eos_ActionSubset.

3 CONSTANT IDENTIFIER NAMES

Names of constant identifiers available to host codes are standardized by applying the following rules:

1. All identifiers begin with the following four characters: “EOS_”.
-

2. Either the underscore is used to separate words or camel case¹ is used if the name is comprised of multiple words.

4 DATA TYPES

Throughout this document language data types will be referred to generically. The actual definition is machine-, language-, and compiler-specific. The data types used by EOSPAC are:

- `EOS_INTEGER` a 32-bit signed integer data type
- `EOS_REAL` a 64-bit signed floating point data type
- `EOS_CHAR` an 8-bit character type.

Some parameters of data type `EOS_INTEGER` that are related to the data types are:

- `EOS_TRUE` a constant specifying a Boolean true
- `EOS_FALSE` a constant specifying a Boolean false
- `EOS_MaxErrMsgLen` a constant specifying the maximum character string length associated with an EOSPAC error message

5 ANNOTATED UPDATES

As this package evolves, so does its user manual. Littered throughout this document are annotations that highlight new features and/or details that have been implemented within the software. These annotations are marked with the following labels:

NEW for 6.2.1, **NEW for 6.2.2**, **NEW for 6.3**, **NEW for 6.3.1**, **NEW for 6.5**, **NEW for 6.5.7**
and **NEW for 6.5.8**, **NEW for 6.5.11**, **NEW for 6.5.13**

Of course, not all changes to this document are annotated with these labels, because many edits are

¹Camel case is the practice of writing compound words or phrases such that each word or abbreviation in the middle of the phrase begins with a capital letter, with no intervening spaces or punctuation. Common examples include "iPhone", "eBay", "FedEx", "DreamWorks", and "HarperCollins". It is also sometimes used in online usernames such as "JohnSmith", and to make multi-word domain names more legible, for example in advertisements.

required independently of any specific software release. Such changes to this document are intended to enhance or correct its content.

3 BASIC THEORY AND MODELS

In theory there is no difference between theory and practice. In practice there is.

– Yogi Berra

SESAME typically contains EOS and Vaporization data, Melt & Shear Modulus data, Opacity data and Conductivity data[1]. Where EOS data is missing from SESAME, EOSPAC will often attempt to calculate it. In some cases, the host code can determine the models used to calculate EOS data.

1 Nomenclature

α_{exp}	Thermal expansion alpha
a	Intrinsic Helmholtz free energy
C_{eVK}	Electron-volt to Kelvin conversion factor (11604.5221 K/eV)
c	Adiabatic (isentropic) sound speed
c_p	Constant-pressure specific heat
c_T	Isothermal sound speed
c_v	Constant-volume specific heat
η	Electron degeneracy parameter
F	Fermi integral
Γ	Gruneisen coefficient
h	Intrinsic enthalpy
\hbar	Reduced Planck constant
K_s	Isentropic compressibility
K_T	Isothermal compressibility
k	Boltzman constant

κ	Ratio of specific heats
M	Average atomic mass
m	Mass
p	Total pressure
p_i	Ion pressure
p_c	Cold curve pressure (at $T = 0$)
N_i	Ion number density
ρ	Density
s	Intrinsic Entropy
R	Universal Gas Constant (8.3144598e-03 kJ/K/mol)
T	Temperature
T_i	Ion temperature (eV)
T_D	Debye temperature (eV)
T_M	Lindemann melting temperature (eV)
u	Intrinsic Total internal energy
u_c	Cold curve internal energy, $u(\rho, T = 0)$
u_i	Intrinsic Ion internal energy
u^*	Thermal internal energy, $u - u_c$
v	Intrinsic volume ($v = \frac{1}{\rho}$)
Z	Free electrons per ion

It is important to note that the intrinsic variables used in this section are lowercase, but are typically uppercase throughout the remainder of this document (specifically in the appendices). The upper case variants are a nomenclature artifact used to improve the readability of the mnemonics in which they are used. If questions arise regarding the units of a given quantity, then one should assume they are consistent with the documented SESAME data units[1].

2 Entropy

Entropy is an example of data, while not stored within SESAME, is simple to calculate using [equation \(3.1\)](#) if both the internal energy and Helmholtz free energy data are available.

$$a = u - Ts \tag{3.1}$$

If only the internal energy data is available, as is the case with older EOS data, then some additional work must be done to achieve the desired end. The necessary $T ds$ expression is provided in [equation \(3.2\)](#).

$$du = Tds - Pdv \quad (3.2)$$

The differential of the function, $u = u(v, s)$ is

$$du = \left. \frac{\partial u}{\partial s} \right|_v + \left. \frac{\partial u}{\partial v} \right|_s . \quad (3.3)$$

Comparing [equations \(3.2\)](#) and [\(3.3\)](#), yields

$$T = \left. \frac{\partial u}{\partial s} \right|_v , \quad (3.4)$$

which may be rearranged into

$$\left. \partial s \right|_v = \left. \frac{\partial u}{T} \right|_v . \quad (3.5)$$

[Equation \(3.5\)](#) can be integrated and rearranged to yield the improper integral in [equation \(3.6\)](#).

$$s \Big|_v = \lim_{a \rightarrow 0} \int_a^T \left. \frac{\partial u}{a} \right|_v \quad (3.6)$$

[Equation \(3.6\)](#) may be rewritten after an integration by parts for a finite temperature, T , as

$$s = \lim_{a \rightarrow 0} \frac{u}{a} + \lim_{a \rightarrow 0} \int_a^T \frac{u}{T^2} dT . \quad (3.7)$$

The internal energy in SESAME is composed of a three-part decomposition [2] including a cold curve contribution (e.g., u_c). This cold curve energy contribution is effectively irrelevant in the calculation of entropy, since u_c is constant for each isochore (i.e., constant volume). This allows the approximation of entropy data using the thermalized internal energy (u^*); therefore, given

$u_c \equiv u(\rho, T = 0)$, [equation \(3.7\)](#) may be rewritten in terms of the cold curve internal energy as

$$s = \lim_{a \rightarrow 0} \frac{u - u_c}{a} + \lim_{a \rightarrow 0} \int_a^T \frac{u - u_c}{T^2} dT . \quad (3.8)$$

Finally, [equation \(3.8\)](#) may be transformed into a form that employs the thermal internal energy. The [equations \(3.9\)](#) and [\(3.10\)](#) are then used to calculate entropy, and [equation \(3.1\)](#) is subsequently used to calculate the Helmholtz free energy data.

$$s = \frac{u^*}{T} + \int_0^T \frac{u^*}{T^2} dT \quad (3.9)$$

$$s \Big|_{T=0} = u^* \Big|_{T=0} = \frac{u^*}{T} \Big|_{T=0} = 0 \quad (3.10)$$

This integral form avoids the numerical sensitivities of other differential forms, which are discussed further in [chapter 9 section 5](#).

3 Ion EOS Models

Other models are available to calculate EOS data corresponding to SESAME subtables[1]. These analytical models include the Ideal Gas Model, the Cowan Model and the Number Proportional Model. These models are used to create two-temperature¹ EOS data by subtracting the analytically-calculated data from SESAME's tabulated total EOS data. Due to cautionary guidance[3], experimentation with different ion EOS models is recommended if problems occur with two-temperature calculations.

¹Two-temperature EOS data allows a host code to perform calculations with temperature fields associated with ions and electrons separately.

3.1 Ideal Gas Model

The ideal gas law is a simple set of relationships describing the properties of a perfect monatomic gas.

$$p_i(\rho, T) = \frac{RT\rho}{M} \quad (3.11)$$

$$u_i(\rho, T) = \frac{3RT}{2M} \quad (3.12)$$

$$a_i(\rho, T) = -\frac{RT}{M} \left(-7.7072343 + \frac{3}{2} \ln(MT) + \ln\left(\frac{M}{\rho}\right) \right) \quad (3.13)$$

Equation (3.13) was taken directly from the OpenSesame software[4], which is used to generate SESAME EOS data. Equation (3.1) supplements equations (3.11) and (3.13) to calculate the entropy data. Curiosity drives the author to determine the origin of equation (3.13). The entropy differential (TdS equation) is defined as equation (3.14).

$$ds = \frac{du}{T} + \frac{p}{T} dv \quad (3.14)$$

Equation (3.14) may be rewritten as equation (3.15).

$$ds = \frac{du}{dT} \frac{dT}{T} + \frac{R}{M} \frac{dv}{v} \quad (3.15)$$

Given $\frac{du}{dT} = \frac{3R}{2M}$ from equation (3.12), equation (3.15) yields equation (3.16).

$$\int ds = \int \frac{3R}{2M} \frac{dT}{T} + \int \frac{R}{M} \frac{dv}{v} \quad (3.16)$$

Integrating by substitution ($f = MT, df = dT$ and $g = Mv, dg = dv$) equation (3.16) results in equation (3.17).

$$s = \frac{3R}{2M} \ln(MT) + \frac{R}{M} \ln(Mv) + s_0 \quad (3.17)$$

The Helmholtz free energy is determined by combining equations (3.1) and (3.17) to yield equation (3.18), where $v = \frac{1}{\rho}$.

$$a = \frac{3RT}{2M} - \frac{RT}{M} \left(s_0 + \frac{3}{2} \ln(MT) + \ln \left(\frac{M}{\rho} \right) \right) \quad (3.18)$$

Equation (2.11) can be rewritten as equation (2.12).

$$a = -\frac{RT}{M} \left(\left(s_0 - \frac{3}{2} \right) + \frac{3}{2} \ln(MT) + \ln \left(\frac{M}{\rho} \right) \right) \quad (3.19)$$

The general form of equations (3.13) and (3.19) are identical. The value of $(s_0 - \frac{3}{2})$ is the result of applying the ideal gas limit for a monatomic gas, and it is beyond the scope of this document.

3.2 Cowan Model

This section describes the simple analytical model developed by R. D. Cowan and documented for the IONEOS Fast, Analytic, Ion Equation-of-State Routine[5]. A normalized local mass density and a dimensionless constant are defined by equations (3.20) and (3.21) respectively.

$$\xi = \frac{9Z^{0.3}\rho}{M} \quad (3.20)$$

$$\beta = 0.6Z^{\frac{1}{9}} \quad (3.21)$$

It is convenient to define the specific heat relationship:

$$c_v = \left. \frac{\partial u}{\partial T} \right|_v = T \left. \frac{\partial s}{\partial T} \right|_v \quad (3.22)$$

The Debye temperature and the Lindemann melting temperature are defined by equations (3.23) and (3.24) respectively.

$$T_D = \frac{(1.68)\xi^{2+\beta}}{(Z+22)(1+\xi)^2} \quad (3.23)$$

$$T_M = \frac{(0.32)\xi^{4+2\beta-\frac{2}{3}}}{(1+\xi)^4} \quad (3.24)$$

The ion temperature variables (ϕ_F and ϕ_S) and Gruneisen parameters (γ_F and γ_S) are described in equations (3.25) to (3.30) for the fluid (F) and solid (S) phases.

$$\phi_F = \left(\frac{T_M}{T_i}\right)^{\frac{1}{3}} \quad (3.25)$$

$$\phi_S = \frac{T_D}{T_i} \quad (3.26)$$

$$\gamma_F = 3\beta - 1 + \frac{6}{(1+\xi)} \quad (3.27)$$

$$\gamma_S = \beta + \frac{2}{(1+\xi)} \quad (3.28)$$

$$\gamma'_F = \gamma_F + \frac{2\gamma_F^2}{9} + \frac{6\xi}{(1+\xi)^2} \quad (3.29)$$

$$\gamma'_S = \beta + \frac{2}{(1+\xi)^2} \quad (3.30)$$

Use equations (3.31) to (3.33) for the fluid region ($T_i > T_M$).

$$p_i = \frac{RT\rho}{M}(1 + \gamma_F\phi_F) \quad (3.31)$$

$$u_i = \frac{3RT}{2M}(1 + \phi_F) \quad (3.32)$$

$$s_i = \frac{R}{M} \left[7 - 3\phi_F + \frac{3}{2} \ln \left(\frac{0.02T_i}{\left(\frac{0.42}{22+Z}\right)} \right) - \ln(\xi) \right] \quad (3.33)$$

Equation (3.33) was taken directly from the OpenSesame software[4], and it can be shown to satisfy the specific heat relation of equation (3.22). Equation (3.1) supplements equations (3.31) to (3.33) to calculate the Helmholtz free energy data.

Use equations (3.34) to (3.36) for the high-temperature solid region ($T_i \leq T_M$ and $3T_i > T_D$).

$$p_i = \rho \gamma_S u_i \quad (3.34)$$

$$u_i = \frac{3RT}{M} \left(1 + \frac{\phi_S^2}{20} - \frac{\phi_S^4}{1680} \right) \quad (3.35)$$

$$s_i = \frac{R}{M} \left[4 + 3 \left(\phi_S^2 \left(\frac{1}{40} - \frac{\phi_S^2}{2240} \right) - \ln(\phi_S) \right) \right] \quad (3.36)$$

Equation (3.36) was taken directly from the OpenSesame software[4], and it can be shown to satisfy the specific heat relation of equation (3.22). Equation (3.1) supplements equations (3.34) to (3.36) to calculate the Helmholtz free energy data.

Use equations (3.37) to (3.39) for the low-temperature solid region ($T_i \leq T_M$ and $3T_i \leq T_D$).

$$p_i = \rho \phi_S u_i \quad (3.37)$$

$$u_i = \frac{3RT}{M} \left(\frac{3}{8} \phi_S + \frac{\pi^4}{5 \phi_S^3} - \left(3 + \frac{9}{\phi_S} + \frac{18}{\phi_S^2} + \frac{18}{\phi_S^3} \right) e^{-\phi_S} \right) \quad (3.38)$$

$$s_i = \frac{R}{M} \left[4 \left(\frac{\pi^4}{5 \phi_S^3} - \left(\frac{9}{4} + \frac{9}{\phi_S} + \frac{18}{\phi_S^2} + \frac{18}{\phi_S^3} \right) e^{-\phi_S} \right) \right] \quad (3.39)$$

Equation (3.39) was analytically derived using equations (3.9) and (3.10). Equation (3.1) supplements equations (3.37) to (3.39) to calculate the Helmholtz free energy data.

It is important to note that the Cowan Model may introduce unwanted pathologies due the fact that its functions are discontinuous at $\phi_S = 3$. Figure 3.1 demonstrates the aforementioned discontinuity between equation (3.36) and equation (3.39), and it is quantified to be approximately a three percent deviation.

$$f_{new} = \frac{e^{-\phi}(4e^{\phi}\pi^4 - 45(8+8\phi+4\phi^2+\phi^3))}{5\phi^3}, \phi \geq 3$$

$$f_{new} = 4 + 3\left(\phi^2\left(\frac{1}{40} - \frac{\phi^2}{2240}\right) - \ln(\phi)\right), \phi < 3$$

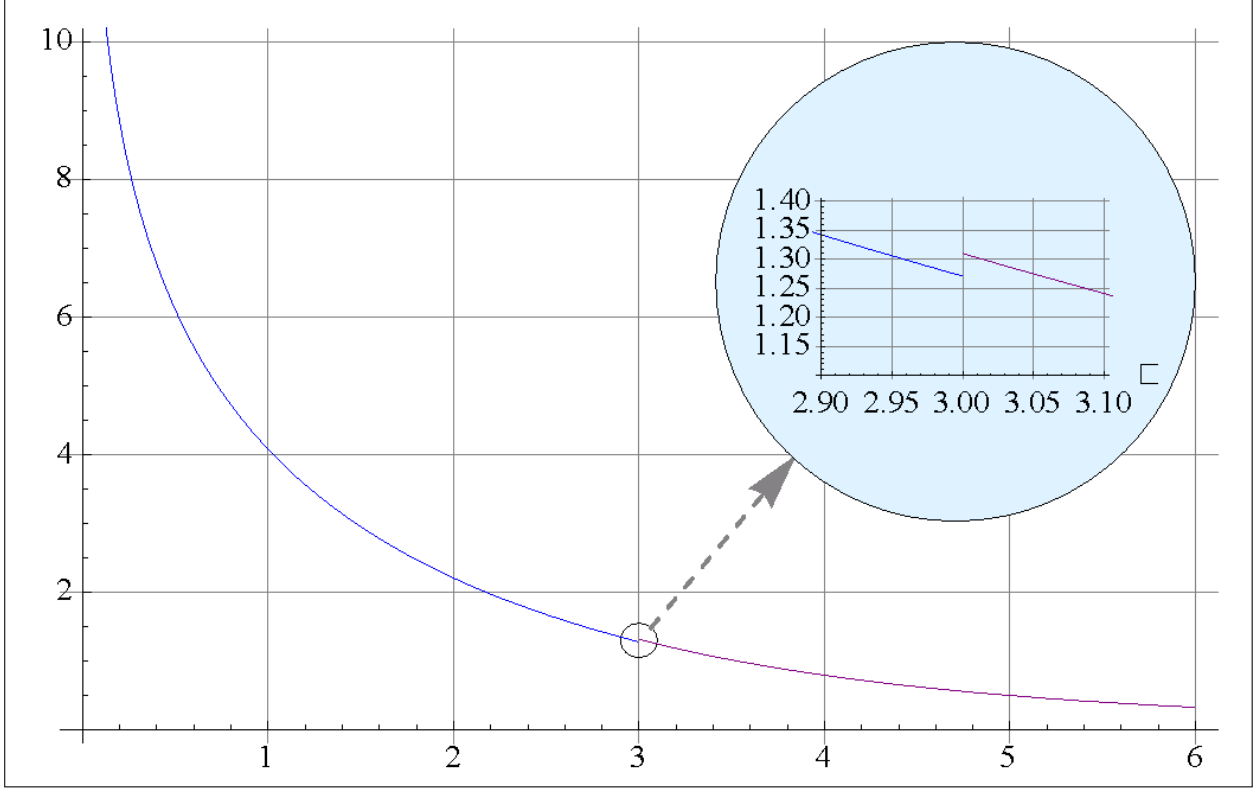


Figure 3.1: Dimensionless parameters from the high-temperature solid entropy expression and the new low-temperature solid entropy expression.

3.3 Number Proportional Model

Since a subtraction of the analytical model values from the tabulated total EOS data is performed to calculate an electron EOS, pathologies will typically exist within the resultant data at low temperatures and high densities due to the fact that the chosen ion EOS was not used to calculate the original EOS data. The number proportional model, in principle, albeit not always, mitigates such pathological data in that it uses simple ratio equations to model the ion EOS[3].

$$p_i(\rho, T) = \frac{p(\rho, T)}{1 + Z} \quad (3.40)$$

$$u_i(\rho, T) = \frac{u(\rho, T)}{1 + Z} \quad (3.41)$$

$$a_i(\rho, T) = \frac{u(\rho, T)}{1 + Z} \quad (3.42)$$

Equations (3.1) and (3.10) are used to calculate the entropy and, subsequently, the Helmholtz free energy in the event that no Helmholtz free energy data is tabulated; otherwise, equation (3.1) supplements equations (3.40) to (3.42) to calculate the entropy data.

The number of free electrons per ion is estimated by assuming the thermal electron EOS is determined using the Fermi-gas model.

$$Z = Z_1 F_{1/2}(\eta) \quad (3.43)$$

$$Z_1 = \frac{2}{N_i} \left(\frac{mkT}{2\pi\hbar^2} \right)^{\frac{3}{2}} \quad (3.44)$$

$$\frac{Z_0 Z}{1 + Z} = Z_1 F_{3/2}(\eta) \quad (3.45)$$

$$Z_0 = \frac{p(\rho, T) - p_c(\rho)}{N_i kT} \quad (3.46)$$

The Fermi integrals satisfy equation (3.47) to at least one-percent accuracy.

$$F_{3/2} = F_{1/2} \left(1 + (0.88388)F_{1/2} + (0.37208)F_{1/2}^2 + (0.02645)F_{1/2}^{10/3} \right)^{\frac{1}{5}} \quad (3.47)$$

Upon substituting equations (3.43) and (3.45) into equation (3.47), equation (3.48) is produced.

$$Z + 1 = Z_0 \left(1 + (0.88388) \left(\frac{Z}{Z_1} \right) + (0.37208) \left(\frac{Z}{Z_1} \right)^2 + (0.02645) \left(\frac{Z}{Z_1} \right)^{10/3} \right)^{-\frac{1}{5}} \quad (3.48)$$

Equation (3.48) can be solved iteratively, and it is constrained by equations (3.49) and (3.50).

$$Z \geq 0 \tag{3.49}$$

$$Z_0 \geq 1 \tag{3.50}$$

4 Additional Thermodynamic Quantities

Often users of EOSPAC are interested in calculating quantities, which are not directly provided by the EOSPAC interface. Distributed with EOSPAC is a utility named `get_sesame_data` (see [chapter 8](#)), which provides a command line interface to various EOSPAC capabilities like querying the content of SESAME data file(s). Additionally, `get_sesame_data` can calculate various derived thermodynamic values, which are described in this section.

Given density (ρ) and temperature (T), calculate the following: pressure (p), specific internal energy (u), specific Helmholtz free energy (a), specific entropy (s), sound speed (c), adiabatic bulk modulus (β), Gruneisen Coefficient (Γ), isothermal bulk modulus ($\beta_T = \rho c_T^2$), and specific heats (c_v and c_p). The pressure, specific internal energy, specific Helmholtz free energy, and specific entropy are simply calculated by interpolating the respective SESAME data at the given density and temperature. The other quantities require more effort as described in the following sections.

4.1 Identities

$$\left. \frac{\partial y}{\partial z} \right|_x \left. \frac{\partial z}{\partial x} \right|_y \left. \frac{\partial x}{\partial y} \right|_z = -1 \tag{3.51}$$

$$\left. \frac{\partial y}{\partial x} \right|_z \left. \frac{\partial x}{\partial y} \right|_z = 1 \tag{3.52}$$

$$\frac{\partial f}{\partial v} = -\rho^2 \frac{\partial f}{\partial \rho} \text{ where } v \equiv \frac{1}{\rho} \text{ and } \partial v \equiv -\frac{\partial \rho}{\rho^2} \tag{3.53}$$

4.2 Sound speed

The adiabatic (isentropic) sound speed is defined by [equation \(3.54\)](#).

$$c^2 = -v^2 \frac{\partial p}{\partial v} \Big|_s = \frac{\partial p}{\partial \rho} \Big|_s \quad (3.54)$$

Using [equation \(3.51\)](#), [equation \(3.54\)](#) can be rewritten as [equation \(3.55\)](#).

$$c^2 = \frac{-\frac{\partial s}{\partial \rho} \Big|_p}{\frac{\partial s}{\partial p} \Big|_\rho} \quad (3.55)$$

[Equation \(3.55\)](#) is a simple means to validate the adiabatic sound speed calculation.

Other variations, which accommodate the interpolation features of EOSPAC, may be derived, and they are listed in [equations \(3.56\) to \(3.58\)](#).

$$c^2 = \frac{\partial P}{\partial \rho} \Big|_T + \frac{T}{\rho^2 \frac{\partial U}{\partial T} \Big|_\rho} \left(\frac{\partial P}{\partial T} \Big|_\rho \right)^2 \quad (3.56)$$

$$c^2 = \frac{\partial P}{\partial \rho} \Big|_U + \frac{P}{\rho^2} \frac{\partial P}{\partial U} \Big|_\rho \quad (3.57)$$

$$c^2 = \frac{\partial P}{\partial \rho} \Big|_T + \frac{\partial P}{\partial T} \Big|_\rho \left(\frac{\partial T}{\partial \rho} \Big|_U + \frac{P}{\rho^2} \frac{\partial T}{\partial U} \Big|_\rho \right) \quad (3.58)$$

It is useful for the reader to note that [equation \(3.56\)](#) may be calculated using the interpolated values associated with the following non-inverted data table types, which are described in [APPENDIX B: EOS_Pt_DT](#) and [EOS_Ut_DT](#). The advantage of using non-inverted data table types is that the numerical errors are minimized – especially for the partial derivative values. Therefore, [equation \(3.56\)](#) is the recommended method to calculate the adiabatic sound speed using SESAME data.

Similarly, [equation \(3.57\)](#) can be evaluated using the interpolated values associated with the EOS_Pt_DUt inverted data table type. [Equation \(3.58\)](#) can be evaluated using the interpolated values associated with the inverted data table type, EOS_T_DUt, and the non-inverted data table type, EOS_Pt_DT. Finally, [equations \(3.54\)](#) and [\(3.55\)](#) can be evaluated using the interpolated partial derivatives associated with the inverted data table types, EOS_Pt_DSt and EOS_St_DPt, respectively. An assessment [6] has been performed to compare the numerical sensitivities of these various adiabatic sound speed calculation methods, [equations \(3.54\)](#) to [\(3.58\)](#), and it has been determined that [equation \(3.56\)](#) produces interpolated results using SESAME data with the minimal numerical noise. This is concluded to be the result of no required tabular inversions/transforms. As a result of the assessment, all table types that are dependent upon the isentropic sound speed (i.e., EOS_BSt_DT, EOS_BSt_DUt, etc.) use data that is calculated using [equation \(3.56\)](#).

4.3 Isentropic Compressibility

The adiabatic bulk modulus is defined by [equation \(3.59\)](#).

$$\beta = \rho c^2 \tag{3.59}$$

The isentropic compressibility is subsequently defined by [equation \(3.60\)](#).

$$K_s = \frac{1}{\beta} = \left(\rho \frac{\partial p}{\partial \rho} \Big|_s \right)^{-1} = - \frac{1}{v} \frac{\partial v}{\partial p} \Big|_s \tag{3.60}$$

4.4 Isothermal Compressibility

The isothermal bulk modulus is defined by [equation \(3.61\)](#).

$$\beta_T = \rho c_T^2 \tag{3.61}$$

This isothermal compressibility is defined by [equation \(3.62\)](#).

$$K_T = \frac{1}{\beta_T} = \left(\rho \frac{\partial p}{\partial \rho} \Big|_T \right)^{-1} = - \frac{1}{v} \frac{\partial v}{\partial p} \Big|_T \tag{3.62}$$

The $\left. \frac{\partial p}{\partial \rho} \right|_T$ partial derivative is a calculated side effect of interpolating the tabulated data for $p = p(\rho, T)$.

It may be of interest to the user to empirically verify the constraint of [equation \(3.63\)](#), which compares isothermal and adiabatic entropy.

$$c_T^2 \leq c^2 \quad (3.63)$$

4.5 Gruneisen Coefficient

The Gruneisen Coefficient is defined by [equation \(3.64\)](#).

$$\Gamma = \left. \frac{1}{\rho} \frac{\partial p}{\partial u} \right|_{\rho} \quad (3.64)$$

The $\left. \frac{\partial p}{\partial u} \right|_{\rho}$ partial derivative is a calculated side effect of interpolating the tabulated data for $p = p(\rho, u)$.

Alternatively,

$$\Gamma = \frac{\left. \frac{\partial p}{\partial T} \right|_{\rho}}{\left. \rho \frac{\partial u}{\partial T} \right|_{\rho}} = \frac{1}{c_v \rho} \left. \frac{\partial p}{\partial T} \right|_{\rho} \quad (3.65)$$

4.6 Specific heats

The constant volume specific heat is defined by [equation \(3.66\)](#).

$$c_v = \left. \frac{\partial u}{\partial T} \right|_v = \left. \frac{\partial u}{\partial T} \right|_{\rho} \quad (3.66)$$

The $\left. \frac{\partial u}{\partial T} \right|_{\rho}$ partial derivative is a calculated side effect of interpolating the tabulated data for $u = u(\rho, T)$.

The constant pressure specific heat is defined by [equation \(3.67\)](#).

$$c_p = T \left. \frac{\partial s}{\partial T} \right|_p = \left. \frac{\partial h}{\partial T} \right|_p \quad (3.67)$$

Unfortunately, at this point, we find that the constant pressure specific heat cannot be calculated using EOSPAC 6's interpolation results. This is due to the fact that the $\left. \frac{\partial s}{\partial T} \right|_p$ is not available since EOSPAC 6 does not calculate $s = s(p, T)$. In an attempt to derive an alternative equation, the following derivation is performed. The specific enthalpy is defined by [equation \(3.68\)](#).

$$h = u + pv \quad (3.68)$$

Using [equations \(3.67\)](#) and [\(3.68\)](#), the constant pressure specific heat is derived in [equation \(3.69\)](#).

$$c_p = \left. \frac{\partial u}{\partial T} \right|_p + v \left. \frac{\partial p}{\partial T} \right|_v + p \left. \frac{\partial v}{\partial T} \right|_p = \left. \frac{\partial u}{\partial T} \right|_p + p \left. \frac{\partial v}{\partial T} \right|_p \quad (3.69)$$

Given [equation \(3.51\)](#), the $\left. \frac{\partial v}{\partial T} \right|_p$ partial derivative is alternatively defined by [equation \(3.70\)](#).

$$\left. \frac{\partial v}{\partial T} \right|_p = - \frac{\left. \frac{\partial p}{\partial T} \right|_v}{\left. \frac{\partial p}{\partial v} \right|_T} \quad (3.70)$$

Using [equations \(3.53\)](#) and [\(3.70\)](#), [equation \(3.69\)](#) can be rewritten as [equation \(3.71\)](#).

$$c_p = \left. \frac{\partial u}{\partial T} \right|_p + \frac{p}{\rho^2} \left. \frac{\partial p}{\partial T} \right|_\rho \quad (3.71)$$

Unfortunately, the $\left. \frac{\partial u}{\partial T} \right|_p$ partial derivative is generally-unavailable using EOSPAC 6's interpolation methods on the SESAME data; therefore, an alternative form is required. Consider the ratio of

specific heats as defined in [equation \(3.72\)](#).

$$\kappa = \frac{c_p}{c_v} \quad (3.72)$$

[equation \(3.72\)](#) can be rewritten as [equation \(3.73\)](#).

$$\kappa = \frac{T \left. \frac{\partial s}{\partial T} \right|_p}{T \left. \frac{\partial s}{\partial T} \right|_v} \quad (3.73)$$

Using [equation \(3.51\)](#), both $\left. \frac{\partial s}{\partial T} \right|_p = \frac{-1}{\left(\left. \frac{\partial T}{\partial p} \right|_s \right) \left(\left. \frac{\partial p}{\partial s} \right|_T \right)}$ and $\left. \frac{\partial s}{\partial T} \right|_v = \frac{-1}{\left(\left. \frac{\partial T}{\partial v} \right|_s \right) \left(\left. \frac{\partial v}{\partial s} \right|_T \right)}$ are derived.

It follows that [equation \(3.73\)](#) can be rewritten as [equation \(3.74\)](#).

$$\kappa = \frac{\left. \frac{\partial v}{\partial s} \right|_T \left. \frac{\partial T}{\partial v} \right|_s}{\left. \frac{\partial p}{\partial s} \right|_T \left. \frac{\partial T}{\partial p} \right|_s} = \left(\left. \frac{\partial v}{\partial s} \right|_T \left. \frac{\partial s}{\partial p} \right|_T \right) \left(\left. \frac{\partial T}{\partial v} \right|_s \left. \frac{\partial p}{\partial T} \right|_s \right) \quad (3.74)$$

Using the Chain Rule, [equations \(3.75\)](#) and [\(3.76\)](#) are derived.

$$\left. \frac{\partial v}{\partial p} \right|_T = \left. \frac{\partial v}{\partial s} \right|_T \left. \frac{\partial s}{\partial p} \right|_T \quad (3.75)$$

$$\left. \frac{\partial p}{\partial v} \right|_s = \left. \frac{\partial p}{\partial T} \right|_s \left. \frac{\partial T}{\partial v} \right|_s \quad (3.76)$$

Applying [equations \(3.75\)](#) and [\(3.76\)](#) to [equation \(3.74\)](#) yields [equation \(3.77\)](#).

$$\kappa = \left. \frac{\partial v}{\partial p} \right|_T \left. \frac{\partial p}{\partial v} \right|_s \quad (3.77)$$

From equations (3.52), (3.59) to (3.62) and (3.72), equation (3.77) can be rewritten as equation (3.78).

$$c_p = \frac{K_T}{K_s} c_v = \frac{c^2}{c_T^2} c_v \quad (3.78)$$

4.7 Thermal expansion alpha

The thermal expansion alpha is another derived quantity of interest, which is defined in equation (3.79).

$$\alpha_{exp} = \frac{1}{\beta_T} \frac{\partial p}{\partial T} \Big|_{\rho} = K_T \frac{\partial p}{\partial T} \Big|_{\rho} \quad (3.79)$$

4.8 Equilibrium mixture quantities

There are two fundamental mechanisms used to define the composition of a material mixture. A definition based on mass fraction intrinsically combines individual components according to their mass (m_i) to get the desired mixture.

$$\lambda_i = \frac{m_i}{\sum_j m_j} \quad (3.80)$$

Defining a mixture based on mass fraction is a convenient convention for making alloys in a lab frame, because mass can be directly measured. Sometimes stoichiometric mixtures are desired, and one will naturally use each individual component's number [of atoms] fraction, which may be referred to as mole fraction (ϕ). Both number fraction and mole fraction may be used interchangeably, because Avogadro's number cancels in the unit-less fraction.

A number fraction is defined in equation (3.81).

$$\phi_i = \frac{n_i}{\sum_j n_j} \quad (3.81)$$

The number of moles for a given mass of material, i , is $n_i = \frac{m_i}{M_i}$ where M_i is the species' atomic mass or molecular weight.

Substituting n_i into [equation \(3.81\)](#) yields [equation \(3.82\)](#).

$$\phi_i = \frac{\frac{m_i}{M_i}}{\sum_j \frac{m_j}{M_j}} \quad (3.82)$$

Then both the numerator and denominator of [equation \(3.82\)](#) are multiplied by the reciprocal of the sum of the components' masses, and [equation \(3.80\)](#) is substituted to produce an expression of the number fraction as a function of mass fraction in [equation \(3.83\)](#).

$$\phi_i = \frac{\frac{m_i}{M_i}}{\sum_j \frac{m_j}{M_j}} \left[\frac{1}{\sum_k m_k} \right] = \frac{\frac{1}{M_i} \frac{m_i}{\sum_k m_k}}{\sum_j \frac{1}{M_j} \frac{m_j}{\sum_k m_k}} = \frac{\frac{\lambda_i}{M_i}}{\sum_j \frac{\lambda_j}{M_j}} \quad (3.83)$$

The number fractions should be normalized to be compliant with [equation \(3.84\)](#).

$$\sum_i \phi_i = 1 \quad (3.84)$$

SESAME models materials as a single atom or a composition-representative pseudo-particle, so the resulting atomic mass is a weighted average ([equation \(3.85\)](#)) of the components that were mixed.

$$\bar{M} = \sum_i \phi_i M_i \quad (3.85)$$

Given a normalized number fraction, ϕ_i , the normalized mass fraction is defined by [equation \(3.86\)](#).

$$\lambda_i = \frac{\phi_i M_i}{\bar{M}} \quad (3.86)$$

Conversely, given a normalized mass fraction, λ_i , the corresponding normalized number fraction, ϕ_i , is defined by [equation \(3.83\)](#).

Finally, one may calculate a mixture's average normal density from the constituents' normal densi-

ties as defined in [equation \(3.87\)](#).

$$\bar{\rho}_o = \left(\sum_i \frac{1}{(\rho_o)_i} \frac{\phi_i M_i}{\bar{M}} \right)^{-1} = \bar{M} \left(\sum_i \frac{\phi_i M_i}{(\rho_o)_i} \right)^{-1} = \left(\sum_i \frac{\lambda_i}{(\rho_o)_i} \right)^{-1} \quad (3.87)$$

4 GENERAL INTERFACE DESCRIPTION

In many cases, the user interface to a program is the most important part for a commercial company: whether the programs works correctly or not seems to be secondary.

– Linus Torvalds

This section describes, in general, the EOSPAC interface library and how a host code will use it. [Figure 4.1](#) shows how the EOSPAC public interface will interact with host codes written in various languages.

Five host code languages are specifically targeted by the public interface of EOSPAC: C++, C, FORTRAN 77, Fortran 90, and Fortran 2003. As shown in [Figure 4.1](#), EOSPAC provides a flat¹ public interface with unmangled² procedure definitions. The Fortran 2003 interface is the sole exception to the flat interface paradigm; host codes written in Fortran 2003 may leverage a language-specific interface, which uses the more modern mixed-language features of the Fortran 2003 specification. The current interface definitions have the distinct advantage of providing the user with consistent data types and procedure interfaces regardless of the host code’s language and working platform. To ensure language interoperability and platform portability EOSPAC Version 6 is written using the POSIX[7, 8] subset of C.

¹Procedure arguments are reduced to a set of basic data types common to all applicable programming languages.

²Procedure names are ensured to be visible, unique and sensible across the multiple-programming-language interface. In software compilation, name mangling (sometimes called name decoration) is a technique used to solve various problems caused by the need to resolve unique names for programming entities in many modern programming languages.

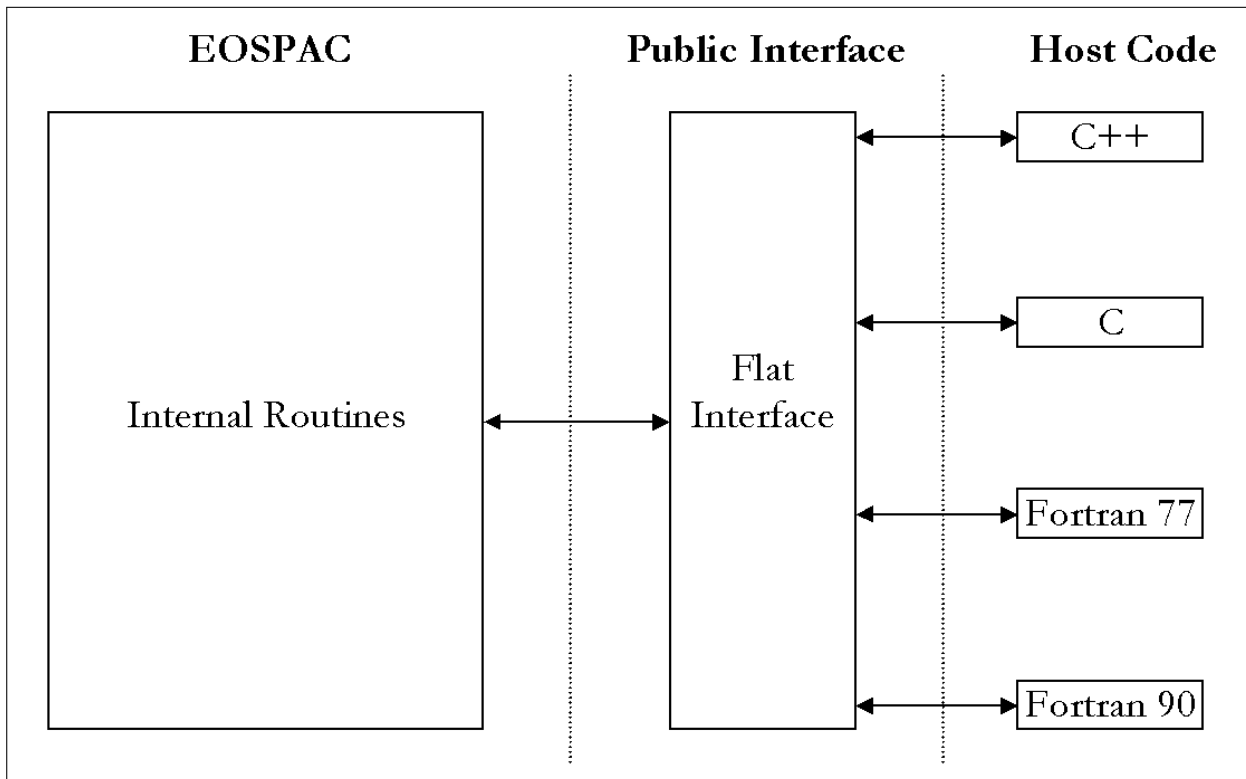


Figure 4.1: General graphical description of the public user interface of EOSPAC.

1 USE CASES

The use cases give an overview of typical user interactions with EOSPAC. There are only two such cases, which may be used in various ways by a host code, a serial host code case and a parallel host code case.

1.1 Serial Case

The serial case is shown in Figure 4.2. During the host code's setup phase the data tables are loaded, and setup options may be set or reset prior to and/or after the data is actually loaded into memory. During the host code's calculation phase the data of selected tables is accessed using either interpolation or mixing, and interpolation/mixing options may be set or reset prior to and/or

after the data is actually accessed. This is done N times where N is problem dependent, but should include at least one evaluation per data table so memory consumption is not frivolously wasted. An optional step is to get information and comments about the loaded data tables, for example, for debugging/informational purposes. This is done M times where M can vary from zero to the number of data tables or a multiple thereof. The data tables are destroyed when the host code is done using them.

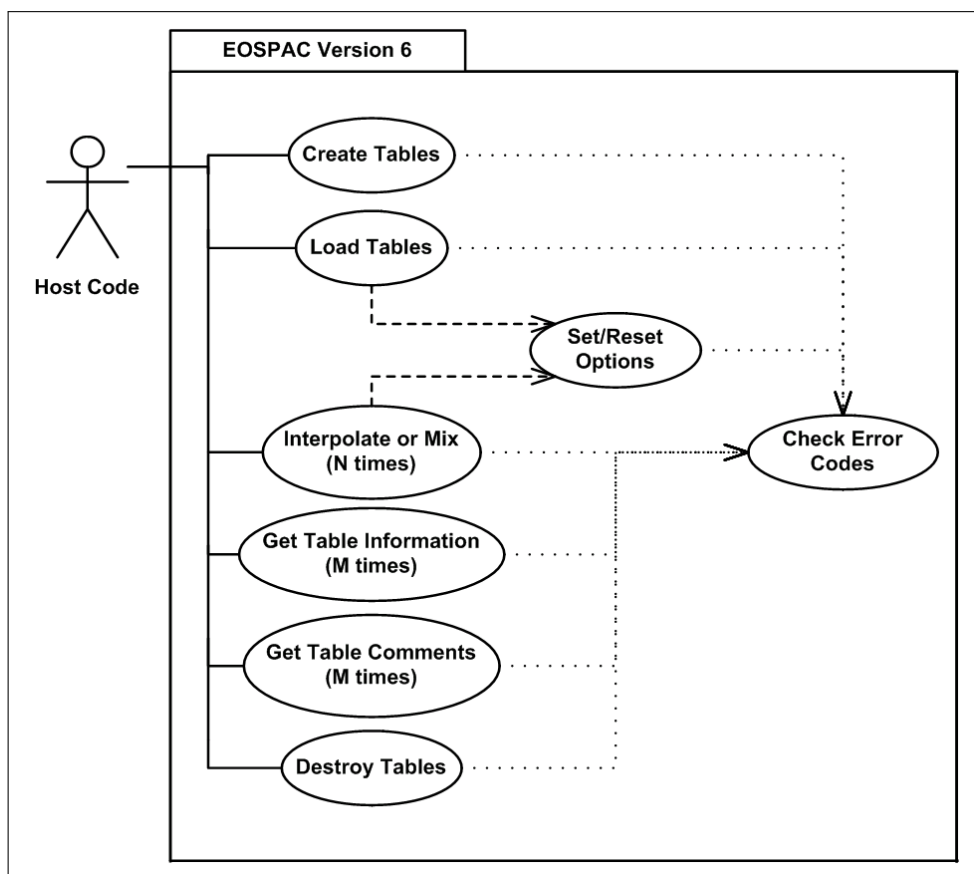


Figure 4.2: Serial host code use of EOSPAC.

1.2 Parallel Case

The parallel case is shown in Figure 4.3. The “Load Tables” occurs on a single process (P_0) and is identical to the serial case. The same process, P_0 , then queries the size of the packed tables and allocates storage to hold them. The P_0 process then extracts the packed tables from EOSPAC. The packed tables are then distributed to all child processes. Each child process then loads its packed tables into EOSPAC. The data is then accessed on each process just as if it was a serial run. Each

process then destroys its data tables when it is done using them.

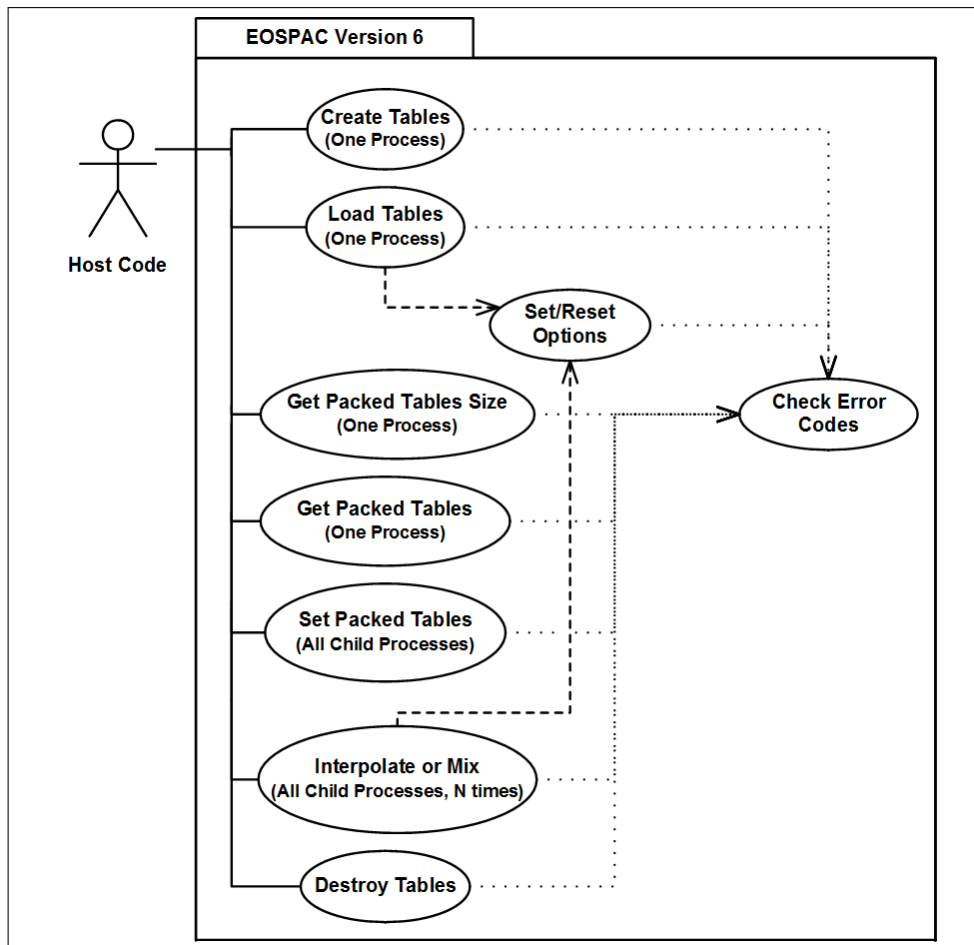


Figure 4.3: Parallel host code use of EOSPAC.

1.3 Parallel Case with Shared Memory

NEW for 6.5.7 Shared memory is used on some new parallel compute architectures to reduce the memory footprint of an application and its associated data on any given compute node. To accommodate this shared memory implementation, two new operations are supplied: “Get Shared Packed Tables Size” and “Set Shared Packed Tables”. Both if these operations are intended as replacements for the original “Get Packed Tables Size” and “Set Packed Tables” operations (Figure 4.3). The “Get Shared Packed Tables Size” operation retrieves both the original packed tables size, which may be used by the host code to allocate the requisite memory for parallel data broadcast as described in the previous subsection (chapter 4 section 1.2), and the new shared tables size, which is

used by the host code to allocate the requisite memory by the compute node's root process. Note that the "Get Shared Packed Tables Size" operation supercedes any need to use the original "Get Packed Tables Size" operation. The "Set Shared Packed Tables" operation is used to populate the shared memory allocation with the requisite data; however, given the necessary input parameter values, the "Set Shared Packed Tables" operation may be a potential surrogate/replacement for the original "Set Packed Tables" operation. It is important to note that, like with the usage of the original "Set Packed Tables Size" operation, the destruction of the memory populated by the "Set Shared Packed Tables" operation is the sole responsibility of the host code.

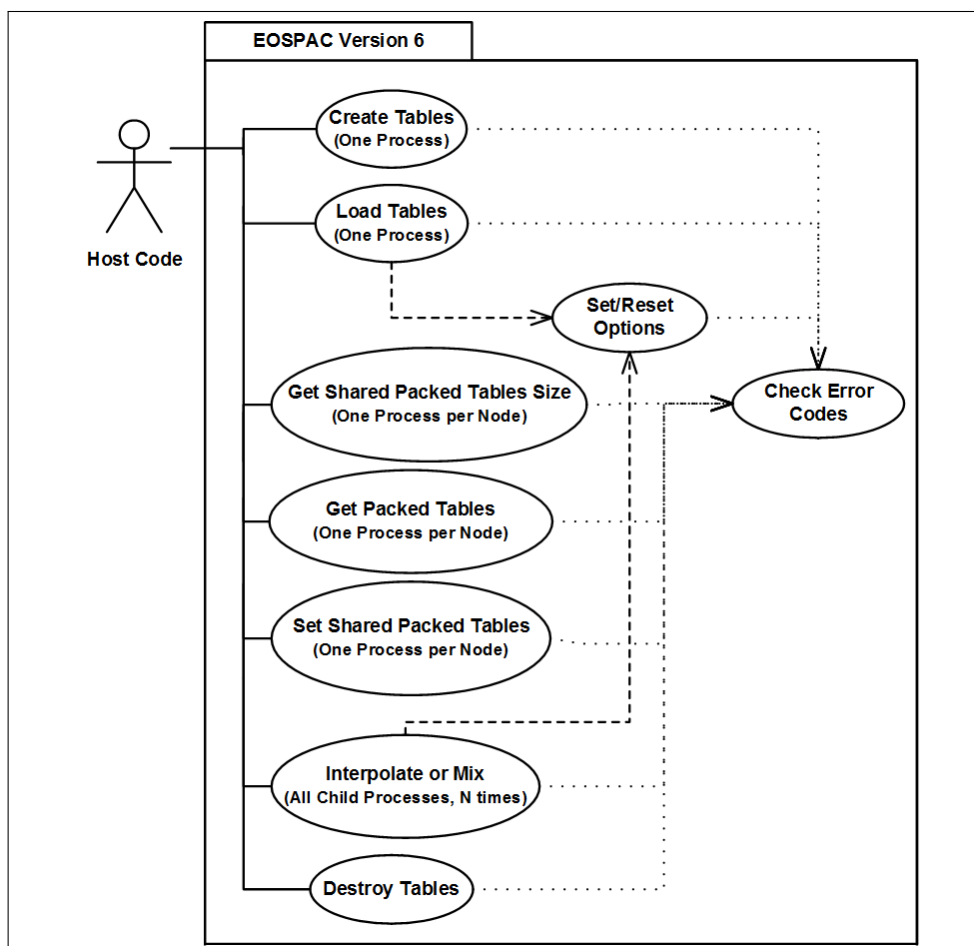


Figure 4.4: Parallel host code with shared memory use of EOSPAC.

It is recommended that the host's implementation of this feature should load all needed data prior to using the "Set Shared Packed Tables" operation. Subsequent attempts to "Load Data" is expected to fail unless the host first destroys everything already loaded. Failure to comply with

this recommendation will likely produce unexpected failures and/or errors.

The parallel case with shared memory case is shown in [Figure 4.4](#), and only the “Get Shared Packed Tables Size” and “Set Shared Packed Tables” operations are different from that described in the previous subsection ([chapter 4 section 1.2](#)). For a many-node job, a combination of the logic in [Figures 4.3](#) and [4.4](#) can be employed to both minimize the SESAME data file I/O to a single node and minimize the shared memory population to a single process per node. Aside from this brief mention, the details of such complex use cases are beyond this document’s scope.

An implemetation example is provided in [chapter 10 section 5](#), which uses the necessary functions to pack and unpack the loaded data prior to interpolation. This example is a similar implementation to that which would be used for the “Set Packed Tables” operation described in [chapter 4 section 1.2](#). The aforementioned example simply uses a simple `malloc` to allocate the “shared memory” that is used by the “Set Shared Packed Tables” operation; however, such a simple `malloc` may be replaced by a `MPI`^{3,4,5,6} code snippet like the following, which is far from complete:

```

1  EOS_INTEGER packedTablesSize, sharedTablesSize;
2  EOS_INTEGER tableHandle[N_TABLES];
3  EOS_INTEGER errorCode, tableHandleErrorCode;
4  EOS_CHAR *packedData = NULL, *sharedData = NULL;
5  EOS_INTEGER nTables;
6  EOS_BOOLEAN useShared;
7  int rank, islandRank, disp_unit;
8  MPI_Aint query_size;
9  MPI_Comm shmcomm;
10 MPI_Win win;
11 EOS_CHAR *base;
12 /*
13  * ... more user code including Load Data operations
14  */
15 /* Broadcast the packed size and data to all ranks */
16 MPI_Bcast (&sharedTablesSize, 1, MPI_INT, 0, MPI_COMM_WORLD);
17 MPI_Bcast (&packedTablesSize, 1, MPI_INT, 0, MPI_COMM_WORLD);
18 if (rank != 0)

```

³<https://www.open-mpi.org/>

⁴<https://www.open-mpi.org/faq/?category=sm>

⁵<https://www.open-mpi.org/doc/v4.1/man3/MPI.Win.shared.query.3.php>

⁶<https://www.open-mpi.org/doc/v3.0/man3/MPI.Win.allocate.shared.3.php>

```
19 {
20     packedData = malloc (sizeof (*packedData) * packedTablesSize);
21 }
22 MPI_Bcast (packedData, packedTablesSize, MPI_BYTE, 0, MPI_COMM_WORLD);
23
24 if (useShared)
25 {
26     printf ("\n*** Creating shared window and setting packed tables\n");
27     /* Create the window and get the pointer */
28     MPI_Win_allocate_shared ((islandRank ==
29                             0 ? sharedTablesSize : 0) * sizeof (*base),
30                             sizeof (*base), MPI_INFO_NULL, shmcomm, &base,
31                             &win);
32     MPI_Win_shared_query (win, MPI_PROC_NULL, &query_size, &disp_unit,
33                          &sharedData);
34     /* Lock the window and store the packed tables in shared memory */
35     MPI_Win_lock_all (MPI_MODE_NOCHECK, win);
36     if (rank == 0)
37     {
38         printf ("\n*** Calling eos_SetSharedPackedTables() on all ranks\n");
39     }
40     eos_SetSharedPackedTables (&nTables, &packedTablesSize, packedData,
41                              sharedData, islandRank == 0, tableHandle,
42                              &errorCode);
43     /* check eospac errorCode here */
44     MPI_Win_unlock_all (win);
45     MPI_Barrier (shmcomm);
46     free (packedData);
47 }
48 else
49 {
50     printf ("\n*** Setting packed tables (no shared memory)\n");
51     /* Don't share memory, each rank has a copy of the table */
52     eos_SetPackedTables (&nTables, &packedTablesSize, packedData,
53                        tableHandle, &errorCode);
54     /* check eospac errorCode here */
55     free (packedData);
56 }
```

```
57  /*  
58  * ... more user code  
59  */
```

5 SETUP MATERIAL DATA

You can have data without information, but you cannot have information without data.

– Daniel Keys Moran

The setup phase consists of calls to interface routines that establish EOSPAC data tables, which are associated with unique identifiers called table handles, and loads them with appropriate data. In addition to this setup routine, there exist routines to destroy data tables, pack their member data into a portable array, and unpack such an array into data tables. The packed array features allow parallel host codes to share data between processes if necessary.

1 DATA LOCATIONS

Before any description of how data is loaded, discarded, packed or unpacked within memory, it is vital to know how EOSPAC is able to find the SESAME data files desired. To do so, three algorithms are used to build a list of file names: 1) Environment-variable-defined and default search paths, 2) Index file, and 3) Default file name list. Once all of these algorithms are completed, the result is an ordered list of absolute-referenced file names that is subsequently edited to remove all duplicate file references. File attributes and, if necessary, bitwise file comparisons are made to eliminate any duplication of files. It is important to note here that two files are not considered duplicates if only part of the contained data is identical. The ordered list of file names is written to the TablesLoaded.dat file when either the EOS_APPEND_DATA or EOS_DUMP_DATA option is set (see [APPENDIX D](#)).

1.1 Environment-variable-defined and default search paths

Initially, the current working directory is put at the top of an ordered list of search paths. If EOSPAC detects that the current environment has set the variable named SESAMEPATH, it parses it for a list of search paths. Within the UNIX and Windows environments, this environment variable is delimited by colons and semicolons respectively. These path names are appended to the ordered list of paths. Finally, a default list of search paths is appended to the ordered list of paths:

DESCRIPTION	PATH NAME
LANL Production data path	/usr/projects/data/eos
LANL X-Div LAN data paths	/usr/local/codes/data/eos /opt/local/codes/data/eos
LANL Cray unclassified data path	/usr/local/udata/ses
LANL Cray classified data path	/usr/local/cdata
LLNL Production data path	/usr/gapps/lanl-data/eos
SANDIA Production data path	/projects/lanl-data/eos

1.2 Ordered File Names List Creation

For each of the search paths found by the algorithm described in [chapter 5 section 1.1](#), the two remaining algorithms are executed in order. These two remaining algorithms are described in [chapter 5 sections 1.3](#) and [1.4](#) respectively, and [Figure 5.1](#) contains a flowchart description of how they are implemented.

NEW for 6.2.2 As of version 6.2.2, EOSPAC will parse a “sesameFilesDir.txt” found in the current working directory every time the eos_CreateTables routine is called. This modification allows the host code to dynamically incorporate changes to the ordered files list.

1.3 Index file

EOSPAC tests for the existence of an index file, a text file named “sesameFilesDir.txt” ([Figure 5.2](#)), within the specified search path found by the algorithm described in [chapter 5 section 1.1](#).

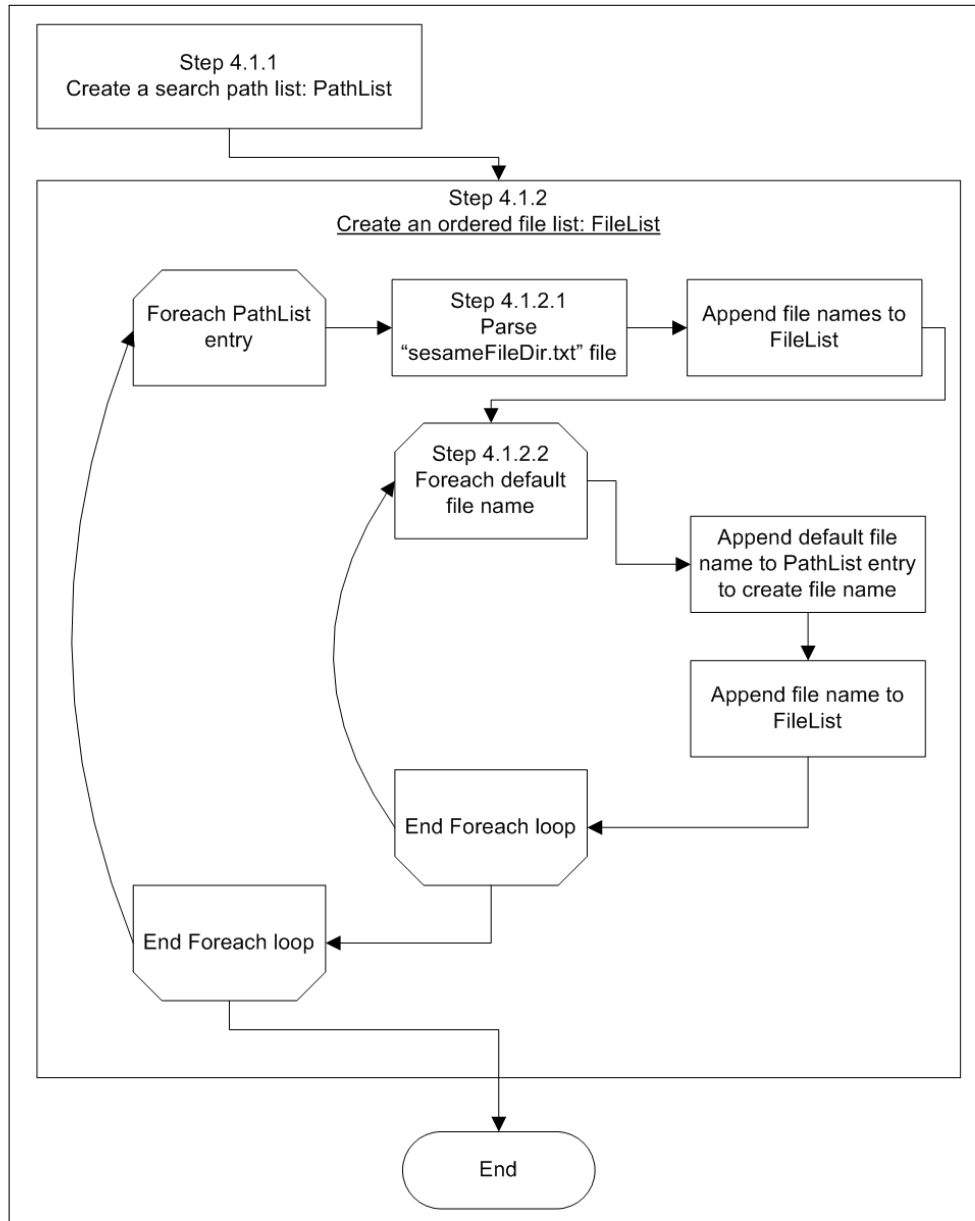


Figure 5.1: Flowchart description of file search algorithms.

If the index file is found, it is parsed according to the following rules to find references to SESAME data files:

- Delimiters include linefeed, carriage return, and semicolon.
- Comments are ignored and begin with #.
- Leading white space is ignored.

```
# Distributed Sesame file list

# Unix absolute reference (two file names per line)
/usr/local/codes/data/eos/sesame;/usr/local/codes/data/eos/sescu

# Unix absolute reference (one file name per line)
/usr/local/codes/data/eos/sescu1
/usr/local/codes/data/eos/sescu9
/usr/local/codes/data/eos/sesou

# DOS/Windows absolute reference (two file names per line)
I:\data\eos\sesame;I:\data\eos\sescu

# alternative DOS/Windows absolute references
\\xfiles\codes\data\eos\sescu1
\\xfiles\codes\data\eos\sescu9
\\xfiles\codes\data\eos\sesou

# relative references with respect to this index file's location
export-controlled/ieee64/sesame;export-controlled/ieee64/sescu
export-controlled/ieee64/sescu1
export-controlled/ieee64/sescu9
export-controlled/ieee64/sesou

# associate material id and Sesame file
MATID 9001 sesame3
MATID 9002 ../../../../sesame3
MATID 9003 /usr/local/codes/data/eos/sesame3
```

Figure 5.2: Example of sesameFilesDir.txt.

- Paths that are relative to the opened index file are converted to absolute paths.
- Invalid file names are silently ignored. A file name is invalid if it doesn't exist or it exceeds

the maximum number of characters (PATH_MAX) for the current file system. The value of PATH_MAX is discussed further in [chapter 5 section 3.4](#).

- **NEW for 6.2.1** If the the case-sensitive token, END, is found as the first non-whitespace characters on a line in the index file, then no other files will be added to the ordered file list, which is defined in [chapter 5 section 1.2](#). This feature is available as of version 6.2.1.
- **NEW for 6.2.2** If the case-sensitive token, MATID, is found as the first non-whitespace characters on a line in the index file, then the remainder of the line shall contain a material ID (integer) and the associated SESAME file name. A file association to a material ID supersedes any previous associations (e.g., associating 9001 to sesame3 and then to sesame2 will retain the last association). See [Figure 5.2](#) for examples. *It is important to note that once a material ID is associated with a specific SESAME file, the association will remain until either the code terminates or another explicit association is provided – there exists no mechanism to reset to the default data search algorithm.* This feature is available as of version 6.2.2.
- The MATID and END tokens constrain the data loaded for all table handles (i.e., it is a global effect). To set table handle-specific constraints, see the eos_GetMaxDataFileNameLength and eos_SetDataFileName functions described in [chapter 5 sections 3.4](#) and [3.10](#) respectively.

Once parsed, the list of file names found in “sesameFilesDir.txt” is appended to the list of SESAME data file names to be searched.

1.4 Default file name list

For compatibility with earlier versions of EOSPAC and old distributions of SESAME files, a default list of file names has been preserved. This ordered list of file names is provided in [Table 5.2](#). This list of file names, if found within the specified search path found by the algorithm described in [section chapter 5 section 1.1](#), is appended to the ordered list of files that will be searched for any requested data.

Table 5.2: Ordered list of default SESAME file names.

	File Name	File Name	File Name
1	sesameu	2 sesameu1	3 sesameu2
4	sesameu3	5 sesameu4	6 sesamea

Continued on next page

Table 5.2: Ordered list of default SESAME file names. (Continued from previous page.)

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

1.5 Ordered File Names List Example

Assume that the current working directory is defined as follows:

```
~/FILES/eospac6.00branch/Source/tests
```

Assume that the following SESAME data files exist for the machine being used:

```
~/FILES/tmp/tests/data/sesame1
~/FILES/tmp/tests/data/sesame3
~/FILES/sesame/081105/sesame_bin_081105
~/FILES/sesame/081105/sesame_bin_081105_sgi
~/FILES/sesame/081105/sesame
~/FILES/code/bll/test/sesame/sesame
~/FILES/code/bll/test/sesame/sescresu
~/FILES/code/bll/test/sesame/sescu
~/FILES/code/bll/test/sesame/sescu1
~/FILES/code/bll/test/sesame/sescu9
```

```
~/FILES/code/b11/test/sesame/sesou
~/FILES/eospac6_mainbranch/Source/tests/data/sesame1
~/FILES/eospac6_mainbranch/Source/tests/data/sesame3
~/FILES/eospac6_mainbranch/Source/tests/alt_tests/sesameFilesDir.txt
~/FILES/eospac6_mainbranch/Source/tests/alt_tests/lambda_parallel/sesame3
~/FILES/eospac6_automake_tests/Source/tests/data/sesame1
~/FILES/eospac6_automake_tests/Source/tests/data/sesame3
~/FILES/eospac6.00branch/Source/tests/data/sesame1
~/FILES/eospac6.00branch/Source/tests/data/sesame3
~/FILES/eospac6.00branch/Source/tests/sesameFilesDir.txt
~/FILES/eospac6.10alpha.7/Source/tests/data/sesame1
~/FILES/eospac6.10alpha.7/Source/tests/data/sesame3
~/FILES/eospac6.10alpha.7/Source/tests/sesameFilesDir.txt
~/FILES/eospac6.10alpha.7/Source/tests/alt_tests/sesameFilesDir.txt
~/FILES/eospac6.10alpha.7/Source/tests/alt_tests/lambda_parallel/sesame3
```

Assume that “sesameFilesDir.txt” in the current working directory that contains the following information:

```
# Sesame3 test data file list
./data/sesame3

# Sesame1 test data file list
./data/sesame1
```

Assume the value of the SESAMEPATH environment variable to contain

```
"/usr/projects/data/eos/export-controlled/ieee64:\${HOME}/FILES
/eospac6.10alpha.7/Source/tests/data:\${HOME}/FILES/code/b11/t
est/sesame"
```

Given all of the above assumptions, the ordered list of files names would be as follows:

```
0. ././data/sesame3
```

1. `././data/sesame1`
2. `/usr/projects/data/eos/export-controlled/ieee64/sesame`
3. `/usr/projects/data/eos/export-controlled/ieee64/sesou`
4. `/usr/projects/data/eos/export-controlled/ieee64/sescu`
5. `/usr/projects/data/eos/export-controlled/ieee64/sescu1`
6. `/usr/projects/data/eos/export-controlled/ieee64/sescu9`
7. `/users/myhome/./FILES/eospac6.10alpha.7/Source/tests/data/sesame1`
8. `/users/myhome/./FILES/eospac6.10alpha.7/Source/tests/data/sesame3`
9. `/users/myhome/FILES/code/bll/test/sesame/sesame`
10. `/users/myhome/FILES/code/bll/test/sesame/sesou`
11. `/users/myhome/FILES/code/bll/test/sesame/sescu`
12. `/users/myhome/FILES/code/bll/test/sesame/sescu1`
13. `/users/myhome/FILES/code/bll/test/sesame/sescu9`
14. `/usr/projects/data/eos/sesame`
15. `/usr/projects/data/eos/sesou`
16. `/usr/projects/data/eos/sescu`
17. `/usr/projects/data/eos/sescu1`
18. `/usr/projects/data/eos/sescu9`

2 DATA ORGANIZATION

As briefly described in this chapter's introduction, the loaded data is referenced by unique table handles. The arguments of the interface routines are organized into a set of ordered arrays such that each array element corresponds to a data table.

For example, the table types (see [APPENDICES B](#) and [C](#)), SESAME material ID numbers, table options (see [APPENDIX A](#)) and error codes (see [APPENDIX H](#)) are stored within identically dimensioned arrays (see [Figure 5.3](#)). Each row of [Figure 5.3](#) specifies a data table that is referenced by the table handle. This conceptual organization is used for all the setup routine arguments that are arrays.

<u>Table Handle</u>	<u>Table Type</u>	<u>Material ID</u>	<u>Table Options</u>	<u>Error Codes</u>
tableHandle ₁	tableType ₁	matID ₁	tableOptions ₁	errorCode ₁
tableHandle ₂	tableType ₂	matID ₂	tableOptions ₂	errorCode ₂
.
.
.
tableHandle _N	tableType _N	matID _N	tableOptions _N	errorCode _N

Figure 5.3: Input/output data organization.

If the host specifies the loading of identical data for multiple table handles (inadvertently or otherwise), then EOSPAC will share the identical data between the two table handles (Figure 5.4). In other words, the two unique handle values will point to the same data object within EOSPAC's internal data structures. This practice is not recommended because it unnecessarily complicates the loaded data's organization.

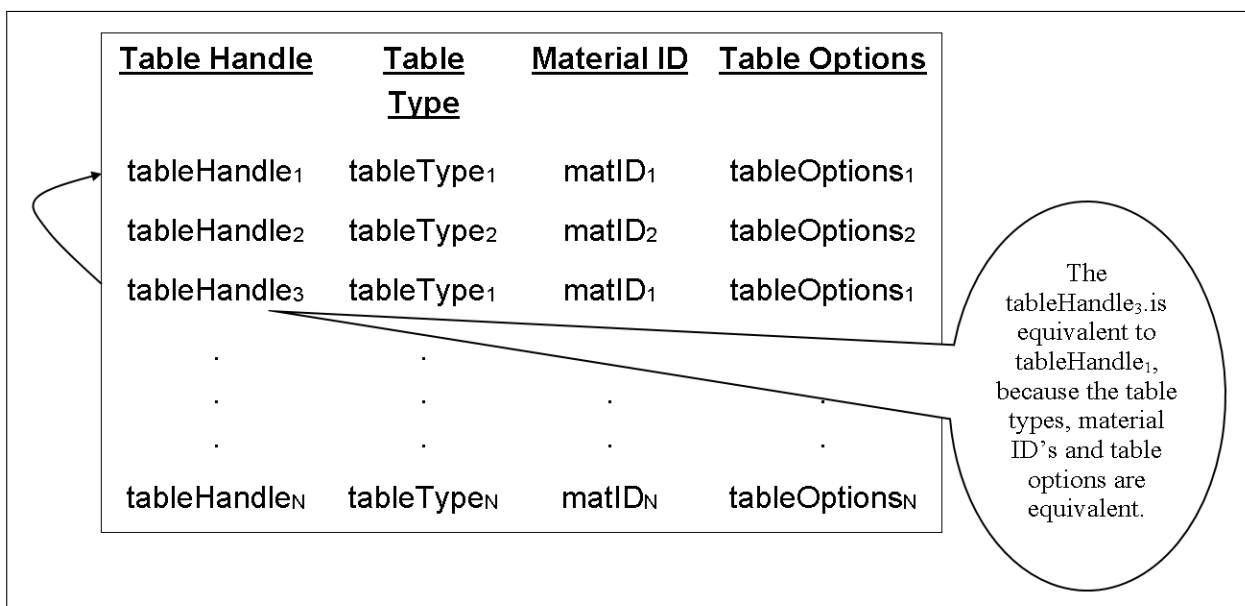


Figure 5.4: Duplicate data organization.

3 ROUTINES AND PARAMETERS

The routines and their associated parameters for setting up the material data are discussed in this section.

NEW for 6.3.1 The default EOSPAC behavior is to delay the inversion of tables (i.e., transform tabulated data to achieve new dependent and independent variables) until the EOSPAC interpolation phase as necessary, according to the requirements of the specified data table type. Since the release of version 6.3.1, the `EOS_INVERT_AT_SETUP` option allows the host code to force EOSPAC to create inverted tables for each specified table handle during the setup phase. The resulting inverted tables are then used during interpolation, and no iterative search algorithm is required, which improves interpolation performance. Of course, it must be understood that this will likely produce different interpolation results than the default behavior, because the inverted table grid may be of insufficient resolution. The quantification of such numeric differences are beyond the scope of this manual – see [chapter 9 section 7](#) and [APPENDIX D](#) for additional details.

3.1 eos_CreateTables

The `eos_CreateTables` routine allocates all memory to store the specified data tables. After calling `eos_CreateTables`, the host code may need to call `eos_SetOption` so the desired set up options can be changed from the documented defaults.

The input arguments are:

<code>nTables</code>	This is the scalar <code>EOS_INTEGER</code> total number of data tables on which to operate.
<code>tableType</code>	This is an <code>EOS_INTEGER</code> array containing the list of table types corresponding to each member data table, T_i , where $i = 1 \dots nTables$. See APPENDICES A and C for table type details.
<code>matID</code>	This is an <code>EOS_INTEGER</code> array containing the SESAME material identification numbers corresponding to each member data table, T_i , where $i = 1 \dots nTables$.

The output arguments are:

tableHandles	<p>This is an array of EOS_INTEGER handles to particular data tables. Each handle corresponds to a member data table, T_i, where $i = 1 \dots nTables$. The host code is responsible for managing this array of table handles.</p> <p>WARNING: If the host code changes any of the tableHandle values, then the logical effect may be likened to a memory leak – unpredictable and potentially-catastrophic behavior is to be expected. This is particularly true if negative values are used in lieu of the valid tableHandle values.</p>
errorCode	<p>This is a scalar EOS_INTEGER variable to contain an error code that may indicate one or more of the tables could not be created. The host code must call <code>eos_GetErrorCode</code> and <code>eos_GetErrorMessage</code> to retrieve error details for a specified tableHandle. See APPENDIX H for error code details.</p> <p>NOTE: <i>As of version 6.3, comparison of two error codes now requires the usage of the <code>eos_ErrorCodesEqual</code> routine described in chapter 7, section 1.1.</i></p>

3.2 eos_DestroyAll

The `eos_DestroyAll` routine releases all memory associated with any remaining data tables and temporary cached data used by EOSPAC routines internally. It is strongly recommended that this routine be used when the currently defined set of SESAME data files is no longer used (i.e., just prior to the end of the host code's execution).

There are no input arguments.

The output argument is:

`errorCode` This is a scalar `EOS_INTEGER` variable to contain an error code that may indicate failure to release all memory associated with temporary cached data. The host code must call `eos.GetErrorMessage` to retrieve error details. See [APPENDIX H](#) for error code details.

NOTE: *As of version 6.3, comparison of two error codes now requires the usage of the `eos_ErrorCodesEqual` routine described in [chapter 7 section 1.1](#).*

3.3 `eos_DestroyTables`

The `eos_DestroyTables` routine releases all memory associated with the specified data tables.

The input arguments are:

`nTables` This is the scalar `EOS_INTEGER` total number of data tables on which to operate.

`tableHandles` This is an array of `EOS_INTEGER` handles to particular data tables. Each handle corresponds to a member data table, T_i , where $i = 1 \dots nTables$. The host code is responsible for managing this array of table handles.

The output argument is:

`errorCode` This is a scalar `EOS_INTEGER` variable to contain an error code that may indicate one or more of the tables could not be destroyed. The host code must call `eos_GetErrorCode` and `eos_GetErrorMessage` to retrieve error details for a specified `tableHandle`. See [APPENDIX H](#) for error code details.

NOTE: *As of version 6.3, comparison of two error codes now requires the usage of the `eos_ErrorCodesEqual` routine described in [chapter 7 section 1.1](#).*

3.4 eos_GetMaxDataFileNameLength

NEW for 6.2.2 The `eos_GetMaxDataFileNameLength` routine is used to query the maximum number of characters (`PATH_MAX`) for the current file system. This is a file system-dependent value with typical values like those shown in [Table 5.8](#).

Table 5.8: Some typical values of `PATH_MAX`.

File System	Length (bytes)
Mac OSX (i386 and PPC)	1024
Modern Linux (i686 and x86_64)	4096
Solaris (Sparc)	1024
Windows/Cygwin	260

There are no input arguments.

The output argument is:

<code>max_length</code>	This is a scalar <code>EOS_INTEGER</code> to contain the definition of <code>PATH_MAX</code> .
-------------------------	--

3.5 eos_GetPackedTables

The `eos_GetPackedTables` routine fills a character array with the specified data table's data. The `eos_GetPackedTables` routine is used to extract the data tables from EOSPAC to allow multithreaded codes to share the data. This routine is also useful for preparing data tables to be written to a host code's binary restart file.

Before calling this routine the host code must call [eos_GetPackedTablesSize](#) to determine `packedTablesSize`, the total number of bytes required to contain the data associated with the specified data tables, allowing the host code to allocate adequate storage.

The input arguments are:

<code>nTables</code>	This is the scalar <code>EOS_INTEGER</code> total number of data tables on which to operate.
----------------------	--

`tableHandles` This is an array of EOS_INTEGER handles to particular data tables. Each handle corresponds to a member data table, T_i , where $i = 1 \dots nTables$. The host code is responsible for managing this array of table handles.

The output arguments are:

`packedTables` This is an array of EOS_CHAR that is used to store all of the member data of specified data tables. This array is designed to allow the host code to share data between multiple processors. If dynamic memory allocation for arrays is not possible, then this routine will prove difficult to use since it is to be allocated to hold `packedTablesSize` characters, where `packedTablesSize` is returned from the `eos_GetPackedTablesSize` routine.

`errorCode` This is a scalar EOS_INTEGER variable to contain an error code. See [APPENDIX H](#) for error code details.

NOTE: *As of version 6.3, comparison of two error codes now requires the usage of the `eos_ErrorCodesEqual` routine described in [chapter 7 section 1.1](#).*

3.6 eos_GetPackedTablesSize

The `eos_GetPackedTablesSize` routine calculates the total number of bytes required to contain the data associated with the specified data tables. The `eos_GetPackedTablesSize` routine is used with the `eos_GetPackedTables` routine.

The input arguments are:

`nTables` This is the scalar EOS_INTEGER total number of data tables on which to operate.

`tableHandles` This is an array of EOS_INTEGER handles to particular data tables. Each handle corresponds to a member data table, T_i , where $i = 1 \dots nTables$. The host code is responsible for managing this array of table handles.

The output arguments are:

packedTablesSize	This is the scalar EOS_INTEGER number of bytes required to hold a specified list of data tables' member data -- size of packedTables.
errorCode	This is a scalar EOS_INTEGER variable to contain an error code. See APPENDIX H for error code details. NOTE: As of version 6.3, comparison of two error codes now requires the usage of the eos_ErrorCodesEqual routine described in chapter 7 section 1.1 .

3.7 eos_GetSharedPackedTablesSize

NEW for 6.5.7 The `eos_GetSharedPackedTablesSize` routine calculates the total number of bytes required to contain the data associated with the specified data tables. The `eos_GetSharedPackedTablesSize` routine is used with the [eos_GetPackedTables](#) and the [eos_SetSharedPackedTables](#) routines.

The input arguments are:

nTables	This is the scalar EOS_INTEGER total number of data tables on which to operate.
tableHandles	This is an array of EOS_INTEGER handles to particular data tables. Each handle corresponds to a member data table, T_i , where $i = 1 \dots nTables$. The host code is responsible for managing this array of table handles.

The output arguments are:

packedTablesSize	This is the scalar EOS_INTEGER number of bytes required to hold a specified list of data tables' member data -- size of packedTables.
sharedTablesSize	This is the scalar EOS_INTEGER number of bytes required to hold a specified list of data tables' member data that is to be stored in shared memory -- size of sharedTables.

errorCode This is a scalar EOS_INTEGER variable to contain an error code. See [APPENDIX H](#) for error code details.

NOTE: *As of version 6.3, comparison of two error codes now requires the usage of the `eos_ErrorCodesEqual` routine described in [chapter 7 section 1.1](#).*

3.8 eos_GpuOffloadData

NEW for 6.5 In relation to porting the EOSPAC interpolation functionality to the GPU¹, version 6.5 introduces a new API² function that is used to offload all currently-loaded data onto the first available GPU device. This new function is only available if the EOSPAC library is compiled with the requisite `DO_OFFLOAD` preprocessor macro. The OpenMP³ 4.5 framework is used to offload data and kernels to the GPU device.

There are no input arguments.

The output arguments are:

errorCode This is a scalar EOS_INTEGER variable to contain an error code. See [APPENDIX H](#) for error code details.

NOTE: *As of version 6.3, comparison of two error codes now requires the usage of the `eos_ErrorCodesEqual` routine described in [chapter 7 section 1.1](#).*

This new API function's purpose is to finish the setup phase of EOSPAC's operation by offloading the loaded data to the available GPU and preventing the loading of additional data before the host code calls `eos_DestroyAll`. [Figure 5.5](#) augments [figure 4.2](#) by graphically-describing the usage of `eos_GpuOffloadData` in relation to the various phases of EOSPAC operation. Each instance (e.g., MPI rank) is responsible for calling `eos_GpuOffloadData` after the data have been distributed to all ranks.

¹Graphics Processing Unit

²Application Programming Interface

³<https://www.openmp.org/>

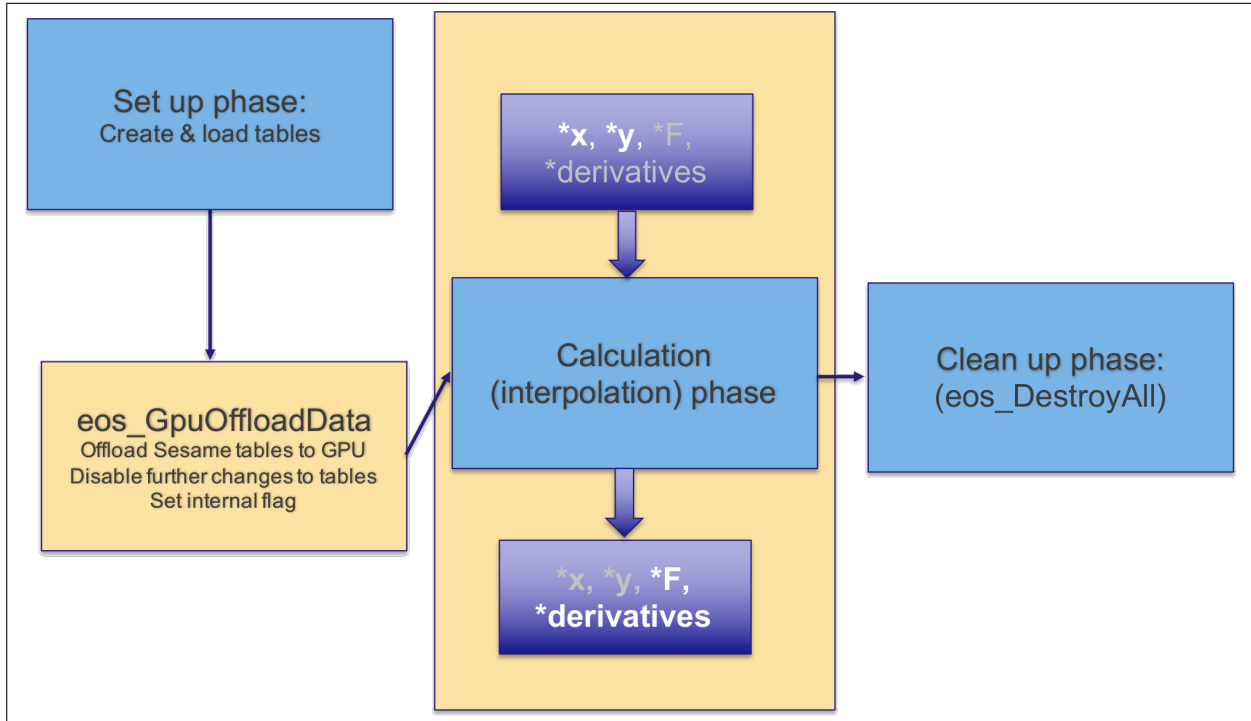


Figure 5.5: Usage of `eos_GpuOffloadData` in relation to the various phases of EOSPAC operation.

If the user's code is C/C++ and uses the `#include "eos_Interface.h"`, then it must either define the `DO_OFFLOAD` preprocessor macro to enable the inclusion of the prototype listed above or explicitly copy the prototype into his code. In order to simplify usage, this requirement will likely change in future releases.

When the host code uses the GPU-enabled version of EOSPAC and offloads the loaded data using the `eos.GpuOffloadData` function, the host code is responsible for providing device pointers to the input and output arrays of `eos.Interpolate`. If this requirement is not met, then a code execution failure is expected. Of course, if the data is not offloaded using `eos.GpuOffloadData`, then the standard host pointers are to be used. As of the release of version 6.5, the interpolation functionality is limited on the GPU as follows:

- Usage of `EOS_INVERT_AT_SETUP`— is required for all inverted data types. If this option is not set for inverted data table types during the setup phase, then unpredictable code execution failures will occur during the interpolation phase.
- A code assertion will occur during interpolation if any unsupported data table type is used.

Ssee [APPENDIX B](#) for a comprehensive list of GPU-compatible data table types.

- Extrapolation checking is disabled during interpolation; however, the host code retains the option to call `eos.CheckExtrap` to determine if `xVals` and `yVals` cause extrapolation.
- No internal copies of `xVals` and `yVals` are created; instead the `EOS_USE_HOST_XY` is enabled (see [APPENDIX D](#)).

3.9 eos_LoadTables

The `eos_LoadTables` routine fills a collection of data tables with the requested data tables from SESAME. Before calling this routine the host code must call `eos_CreateTables` to initialize memory for data tables and retrieve valid table handles. The host code may also need to call `eos_SetOption`, prior to calling `eos_LoadTables`, so the desired set up options can be changed from the documented defaults (see [APPENDIX D](#)).

The input arguments are:

<code>nTables</code>	This is the scalar <code>EOS_INTEGER</code> total number of data tables on which to operate.
<code>tableHandles</code>	This is an array of <code>EOS_INTEGER</code> handles to particular data tables. Each handle corresponds to a member data table, T_i , where $i = 1 \dots nTables$. The host code is responsible for managing this array of table handles.

The output argument is:

<code>errorCode</code>	This is a scalar <code>EOS_INTEGER</code> variable to contain an error code that may indicate failure to load the data. The host code must call <code>eos_GetErrorCode</code> and <code>eos_GetErrorMessage</code> to retrieve error details for a specified <code>tableHandle</code> . See APPENDIX H for error code details.
------------------------	--

NOTE: *As of version 6.3, comparison of two error codes now requires the usage of the `eos_ErrorCodesEqual` routine described in [chapter 7 section 1.1](#).*

3.10 eos_SetDataFileName

NEW for 6.2.2 The `eos_SetDataFileName` routine is used to set the file name for a specified table handle. This will constrain EOSPAC to searching for applicable data within the specified file, if it's a valid file. See [chapter 5 section 3.4](#) for details concerning the maximum length of the file name. This routine will fix an invalid table handle if it was invalidated by a previous call to `eos_CreateTables`. This routine must be called prior to `eos_LoadTables` for the specified table handle; otherwise an error will be returned. This routine should be used in conjunction with `eos_GetMaxDataFileNameLength`.

NEW for 6.5.8 This function has recently been modified internally so that it is readily-adaptable, as described herein, to host code languages other than C/C++ and Fortran (e.g., Python). Therefore, if the host code's language is Fortran 90 (or an older standard), then any custom Fortran compiler option that enforces case sensitivity cannot be used with the function named `eos_SetDataFileName` – attempts to do so may produce unexpected behavior and/or results. Additional knowledge of the named parameters for the various compiled forms of this function, which are packed into the static library, is required. Specifically, to accommodate different Fortran compilers, this API function's compiled objects are manually named as `eos_setdatafilename`, `eos_setdatafilename_`, `eos_setdatafilename__`, `_eos_setdatafilename`, `__eos_setdatafilename` and `EOS_SETDATAFILENAME`, which have been selected based upon the various known default name mangling schemes employed by older Fortran compilers. The variant named in this section's title, unlike other public EOSPAC functions, is reserved for naturally-case sensitive languages, and it is exclusively-defined with the first five parameters documented below. The other six variations define a sixth parameter that is a `long int`, and it exists for Fortran callers that implicitly pass the value, which is the byte length of the string parameter named `fileName`. This naming convention requirement does not apply to host codes that use the Fortran 2003 compatible module, `eos_Interface2003`.

The input arguments are:

<code>tableHandle</code>	This is a scalar <code>EOS_INTEGER</code> handle to a particular data table. The host code is responsible for managing this table handle.
<code>matID</code>	This is a scalar <code>EOS_INTEGER</code> containing the SESAME material identification number corresponding to the member data table.
<code>tableType</code>	This is a scalar <code>EOS_INTEGER</code> containing the table type corresponding to the member data table. See APPENDICES B and C for table type details.

fileName	This is a character string, of a maximum length defined by the constant named PATH_MAX, which is to contain the specified file name.
fileName_len	NEW for 6.5.8 This is a <code>long int</code> , that specifies the length of the specified fileName. This argument is newly-documented for version 6.5.8, and it is only included in the definitions of the following implicitly-defined functions (see inset above for details), which are maintained for a semblance of backward compatibility with older Fortran compilers: <code>eos_setdatafilename</code> , <code>eos_setdatafilename_</code> , <code>eos_setdatafilename__</code> , <code>_eos_setdatafilename</code> , <code>__eos_setdatafilename</code> and <code>EOS_SETDATAFILENAME</code> .

The output argument is:

errorCode	This is a scalar <code>EOS_INTEGER</code> variable to contain an error code. See APPENDIX H for error code details. NOTE: <i>As of version 6.3, comparison of two error codes now requires the usage of the <code>eos_ErrorCodesEqual</code> routine described in chapter 7 section 1.1.</i>
-----------	--

3.11 eos_SetPackedTables

The `eos_SetPackedTables` routine fills the specified data tables with data tables stored as a character array. Typically this is used to insert the data tables into the EOSPAC data structures after a multithreaded code has shared the data tables extracted by `eos_GetPackedTables`. The `eos_SetPackedTables` routine can also be used to unpack data tables recovered from a host code's binary restart file.

The input arguments are:

nTables	This is the scalar <code>EOS_INTEGER</code> total number of data tables on which to operate.
---------	--

packedTablesSize	This is the scalar EOS_INTEGER number of bytes required to hold a specified list of data tables' member data – size of packedTables. This parameter value is used for internal testing and diagnostics.
packedTables	This is an array of EOS_CHAR that is used to store all of the member data of specified data tables. This array is designed to allow the host code to share data between multiple processors. If dynamic memory allocation for arrays is not possible, then eos_SetPackedTables will prove difficult to use since packedTables must hold packedTablesSize characters, where packedTablesSize is returned from either the eos_GetPackedTablesSize or the eos_GetSharedPackedTablesSize routine.

The output arguments are:

tableHandles	This is an array of EOS_INTEGER handles to particular data tables. Each handle corresponds to a member data table, T_i , where $i = 1 \dots nTables$. The host code is responsible for managing this array of table handles. WARNING: The actual table handle value returned to the host code for any specific table, T_i , is not guaranteed to be consistent with the value generated by eos_CreateTables ; this is a behavior likened to an address returned by C malloc.
errorCode	This is a scalar EOS_INTEGER variable to contain an error code that may indicate failure to unpack the data. See APPENDIX H for error code details. NOTE: <i>As of version 6.3, comparison of two error codes now requires the usage of the eos_ErrorCodesEqual routine described in chapter 7 section 1.1.</i>

3.12 eos_SetSharedPackedTables

NEW for 6.5.7

The eos_SetSharedPackedTables is similar to eos_SetPackedTables, except that it

is primarily-designed to store the table data in shared memory, passed through the `sharedTables` argument. If implemented correctly, this routine may be used in lieu of `eos_SetPackedTables` (see descriptions of the `sharedTables` and `nodeRoot` arguments below). It is strongly recommended that the host should load all needed data prior to using this routine. Subsequent attempts to use “`eos_CreateTables`” and/or “`eos_LoadTables`” are expected to fail unless the host first destroys everything already loaded. Failure to comply with this recommendation will likely produce implementation-dependent failures and/or errors.

The input arguments are:

<code>nTables</code>	This is the scalar <code>EOS_INTEGER</code> total number of data tables on which to operate.
<code>packedTablesSize</code>	This is the scalar <code>EOS_INTEGER</code> number of bytes required to hold a specified list of data tables’ member data – size of <code>packedTables</code> . This parameter value is used for internal testing and diagnostics.
<code>packedTables</code>	This is an array of <code>EOS_CHAR</code> that is used to store all of the member data of specified data tables. This array is designed to allow the host code to share data between multiple nodes and/or ranks. If dynamic memory allocation for arrays is not possible, then <code>eos_SetSharedPackedTables</code> will prove difficult to use since <code>packedTables</code> must hold <code>packedTablesSize</code> characters, where <code>packedTablesSize</code> is returned from either the <code>eos_GetPackedTablesSize</code> or the <code>eos_GetSharedPackedTablesSize</code> routine.
<code>sharedTables</code>	This is an array of <code>EOS_CHAR</code> that is used to store selected member data of specified data tables within the referenced shared memory allocation. If this argument is null, then this routine acts like <code>eos_SetPackedTables</code> . If dynamic memory allocation for arrays is not possible, then <code>eos_SetSharedPackedTables</code> will prove difficult to use since <code>sharedTables</code> must hold the specified <code>sharedTablesSize</code> characters, where <code>sharedTablesSize</code> is returned from the <code>eos_GetSharedPackedTablesSize</code> routine.

`nodeRoot` This is the scalar `EOS_BOOLEAN` that indicates whether or not this routine is currently executed on the root of a memory island (i.e., compute node root process). This routine will write into the memory pointed to by `sharedTables`, only if `nodeRoot` is `EOS_TRUE`

The output arguments are:

`tableHandles` This is an array of `EOS_INTEGER` handles to particular data tables. Each handle corresponds to a member data table, T_i , where $i = 1 \dots nTables$. The host code is responsible for managing this array of table handles.

WARNING: The actual table handle value returned to the host code for any specific table, T_i , is not guaranteed to be consistent with the value generated by `eos.CreateTables`; this is a behavior likened to an address returned by C `malloc`.

`errorCode` This is a scalar `EOS_INTEGER` variable to contain an error code that may indicate failure to unpack the data. See [APPENDIX H](#) for error code details.

NOTE: As of version 6.3, comparison of two error codes now requires the usage of the `eos.ErrorCodesEqual` routine described in [chapter 7 section 1.1](#).

4 C/C++ LANGUAGE BINDINGS

```
void eos_CreateTables      (EOS_INTEGER *nTables,
                           EOS_INTEGER tableType[],
                           EOS_INTEGER matID[],
                           EOS_INTEGER tableHandles[],
                           EOS_INTEGER *errorCode);
void eos_DestroyAll      (EOS_INTEGER *errorCode);
```

subroutine eos_GetSharedPackedTablesSize	(EOS_INTEGER nTables, EOS_INTEGER tableHandles(*), EOS_INTEGER packedTablesSize, EOS_INTEGER sharedTablesSize, EOS_INTEGER errorCode)
subroutine eos_GpuOffloadData	(EOS_INTEGER errorCode)
subroutine eos_LoadTables	(EOS_INTEGER nTables, EOS_INTEGER tableHandles(*), EOS_INTEGER errorCode)
subroutine eos_SetDataFileName	(EOS_INTEGER tableHandle, EOS_INTEGER matID, EOS_INTEGER tableType, EOS_CHAR fileName, EOS_INTEGER errorCode)
subroutine eos_SetPackedTables	(EOS_INTEGER nTables, EOS_INTEGER packedTablesSize, EOS_CHAR packedTables, EOS_INTEGER tableHandles(*), EOS_INTEGER errorCode)
subroutine eos_SetSharedPackedTables	(EOS_INTEGER nTables, EOS_INTEGER packedTablesSize, EOS_CHAR packedTables, EOS_CHAR sharedTables, EOS_BOOLEAN nodeRoot, EOS_INTEGER tableHandles(*), EOS_INTEGER errorCode)

Within a Fortran 77 host code, use the header file named “eos_Interface.fi” to define the necessary constants used by EOSPAC. See [chapter 10](#) for Fortran 77 host code examples of using these routines.

Within a Fortran 90 host code, use the Fortran module named “eos_Interface” to define the necessary constants used by EOSPAC. See [chapter 10](#) for Fortran 90 host code examples of using these routines.

6 INTERPOLATE MATERIAL DATA

Those who rule data will rule the entire world.

– Masayoshi Son

The interpolation phase consists of calls to interface routines that use an established EOSPAC data table and return interpolated data requested by the host code. These routines are the most common way to use the data tables.

1 DATA ORGANIZATION

Unlike the setup routines, the interpolation routines perform their function on data associated with a single table handle.

2 ROUTINES AND PARAMETERS

NEW for 6.3 For each of the routines described in this section, all of the interpolation options defined in [APPENDIX F](#) are applicable; however, two new interpolation phase options have been introduced that are of particular interest to users wanting to improve performance: `EOS_XY_MODIFY` and `EOS_XY_PASSTHRU`.

NEW for 6.3.1 A new setup option has been introduced: `EOS_INVERT_AT_SETUP` — see [chapter 5 section 3](#), [chapter 9 section 7](#) and [APPENDIX D](#) for additional details.

Normally, EOSPAC will create temporary internal copies of the `xVals` and `yVals` arrays passed from

the host code into the following routines. These temporary arrays are then modified according to the conversion factors that may have been previously set using the [eos_SetOption](#) routine. The EOS_XY_MODIFY and EOS_XY_PASSTHRU options disable the creation of the temporary copies of xVals and yVals. The EOS_XY_MODIFY option instructs EOSPAC to directly change the values in the host code's xVals and yVals arrays into SESAME-compatible units using the conversion factors that may have been previously set using the [eos_SetOption](#) routine. The EOS_XY_PASSTHRU option instructs EOSPAC to make no changes to the xVals and yVals arrays – rather the values in the host code's xVals and yVals arrays are assumed to already be in SESAME-compatible units.

NEW for 6.5 A new interpolation option, EOS_USE_HOST_XY has been added. This new option enables the EOS_XY_MODIFY option's associated logic, and then it enables new logic to revert the modified xVals and yVals inputs after interpolation is completed. Be aware that the application of the host-supplied conversion factors may not identically-reproduce the original xVals and yVals input values. Another interpolation option, EOS_SKIP_EXTRAP_CHECK, disables all extrapolation checks except when the host code calls the [eos_CheckExtrap](#) function, which is described below. Both of these new options are defined in [APPENDIX G](#), and they are implemented to provide use cases to the end user that maximize the performance of [eos_Interpolate](#) on either the CPU¹ or the GPU².

Using the OpenMP 4.5 `target offload` features, GPU kernels have been created for selected API functions as described below.

2.1 eos_CheckExtrap

If the EOS_INTERP_EXTRAPOLATED error code is returned by either [eos_Interpolate](#) or [eos_Mix](#), then the [eos_CheckExtrap](#) routine allows the user to determine which (x, y) pairs caused extrapolation and in which direction (high or low), it occurred. The units of the xVals, and yVals arguments listed below are determined by the units listed for each tableType in [APPENDICES B](#) and [C](#).

The input arguments are:

tableHandle	This is a scalar EOS_INTEGER handle to a particular data table. The host code is responsible for managing this table handle.
nXYPairs	This is the total number of pairs of independent variable values provided for interpolation for the specified table.

¹Central Processing Unit

²Graphics Processing Unit

xVals	This is an array of the primary independent variable values to use during interpolation. There are nXYPairs elements in xVals.
yVals	This is an array of the secondary independent variable values to use during interpolation. There are nXYPairs elements in yVals.

The output arguments are:

xyBounds	This is an array of size nXYPairs elements that returns EOS_OK if extrapolation did <u>not</u> occur. If extrapolation occurred the variable and direction are determined from Table 3.
errorCode	This is a scalar EOS_INTEGER variable to contain an error code. See APPENDIX H for error code details.

NOTE: *As of version 6.3, comparison of two error codes now requires the usage of the [eos_ErrorCodesEqual](#) routine described in [chapter 7, section 1.1](#).*

In the case that [eos_Mix](#) returned EOS_INTERP_EXTRAPOLATED as an error code, an additional series of steps must be performed to determine which table handles correspond to the extrapolation error:

1. For each tableHandle sent to [eos_Mix](#), call [eos_GetErrorCode](#) and, optionally, [eos_GetErrorMessage](#).
2. For each of these tableHandles, call [eos_CheckExtrap](#) to determine one of codes listed in [Table 6.3](#).

Table 6.3: Extrapolation return codes.

Code	Definition
EOS_OK	No extrapolation occurred.
EOS_xHi_yHi	Both the x and y arguments were high.
EOS_xHi_yOk	The x argument was high, the y argument was OK.
EOS_xHi_yLo	The x argument was high, the y argument was low.
EOS_xOk_yLo	The x argument is OK and the y argument is low.

Continued on next page

Table 6.3: Extrapolation return codes. (Continued from previous page.)

Code	Definition
EOS_xLo_yLo	Both the x and y arguments were low.
EOS_xLo_yOk	The x argument was low, the y argument was OK.
EOS_xLo_yHi	The x argument was low, the y argument was OK.
EOS_xOk_yHi	The x argument is OK and the y argument is high.
EOS_CANT_INVERT_DATA	Can't invert with respect to the required independent variable.
EOS_CONVERGENCE_FAILED	Iterative algorithm did not converge during inverse interpolation.
EOS_UNDEFINED	The result is undefined.

Some additional details regarding the error codes listed in Table 6.3 are listed as follows:

1. If the y argument corresponds to a temperature value, then a zero temperature was used for interpolation rather than the value supplied by the host code.
2. If the x argument corresponds to a density value, then a zero density was used for interpolation rather than the value supplied by the host code.

2.2 eos_Interpolate

The eos.Interpolate routine provides interpolated values for a single material using a table handle associated with data stored within a data table. Before calling eos_Interpolate, the host code may need to call eos.SetOption so the desired interpolation options can be changed from the documented defaults. The units of the xVals, yVals, fVals, dFx and dFy arguments listed below are determined by the units listed for each tableType in APPENDICES B and C.

NEW for 6.5 If the eos.GpuOffloadData function has been used by the host code, then it is assumed that all pointers passed into and out of this eos_Interpolate function are GPU device pointers that reference memory on the GPU itself rather than the traditional CPU memory heap. Specifically, for C/C++ hosts, the OpenMP³ method named omp_target_alloc (or its equivalent) is assumed to have been used by the host to allocate memory on the GPU device. Similarly, a Fortran

³<https://www.openmp.org/>

host is assumed to use a compatible allocation method and a corresponding device attribute (i.e., CUDA⁴) to define and allocate device memory for the input/output arrays of `eos_Interpolate`. To leverage the interpolation GPU kernel it is important for the user to ensure that the table type(s) used is compatible for offload to the GPU; [APPENDIX B](#) indicates which table types are actually compatible with the GPU offload.

The input arguments are:

<code>tableHandle</code>	This is a scalar <code>EOS_INTEGER</code> handle to a particular data. The host code is responsible for managing this table handle.
<code>nXYPairs</code>	This is the scalar <code>EOS_INTEGER</code> total number of pairs of independent variable values provided for interpolation.
<code>xVals</code>	This is an <code>EOS_REAL</code> array of the primary independent variable values to use during interpolation. There are <code>nXYPairs</code> elements in <code>xVals</code> .
<code>yVals</code>	This is an <code>EOS_REAL</code> array of the secondary independent variable values to use during interpolation. There are <code>nXYPairs</code> elements in <code>yVals</code> .

The output arguments are:

<code>fVals</code>	This is an <code>EOS_REAL</code> array of the interpolated data corresponding to the given independent variable data (x and y). There are <code>nXYPairs</code> elements in <code>fVals</code> , unless the <code>tableHandle</code> is associated with the <code>EOS_M_DT</code> table type (see chapter 9, section 4 for details).
<code>dFx</code>	This is an <code>EOS_REAL</code> array of the interpolated partial derivatives of <code>fVals</code> with respect to x. There are <code>nXYPairs</code> elements in <code>dFx</code> .
<code>dFy</code>	This is an <code>EOS_REAL</code> array of the interpolated partial derivatives of <code>fVals</code> with respect to y. There are <code>nXYPairs</code> elements in <code>dFy</code> .

⁴<https://developer.nvidia.com/cuda-fortran>

errorCode This is a scalar EOS_INTEGER variable to contain an error code. See [APPENDIX H](#) for error code details.

NOTE: As of version 6.3, comparison of two error codes now requires the usage of the `eos_ErrorCodesEqual` routine described in [chapter 7, section 1.1](#).

2.3 eos_Mix

The mixed material interpolation uses established EOSPAC data tables and returns interpolated data of mixed materials requested by the host code. The `eos_Mix` routine is the typical way to generate mixed material data using the data tables' member data tables. The data tables to be mixed must be of the same table type. An error code is returned if the table type is not valid for mixing (EOS_NullTable, EOS_Info, etc.). The table types that are valid for `eos_Mix` were originally limited to the following short list⁵ of 29 table types:

EOS_B_DT	EOS_Ktc_DT	EOS_Pt_DT	EOS_T_DUic	EOS_Uic_DT
EOS_Kc_DT	EOS_Pc_D	EOS_Pt_DUt	EOS_T_DUt	EOS_Ut_DPt
EOS_Kec_DT	EOS_Pe_DT	EOS_T_DPe	EOS_Uc_D	EOS_Ut_DT
EOS_Keo_DT	EOS_Pe_DUe	EOS_T_DPic	EOS_Ue_DPe	EOS_Zfc_DT
EOS_Kp_DT	EOS_Pic_DT	EOS_T_DPt	EOS_Ue_DT	EOS_Zfo_DT
EOS_Kr_DT	EOS_Pic_DUic	EOS_T_DUe	EOS_Uic_DPic	

NEW for 6.5.13 To support the requirements of the new `mix_sesame_data` utility (see [chapter 8](#)), four additional table types are now supported by the `eos_Mix` routine's logic:

EOS_At_DT	EOS_Aic_DT	EOS_Ae_DT	EOS_Ac_D
-----------	------------	-----------	----------

The `eos_Mix` routine will provide interpolated values corresponding to mixtures of materials in pressure and temperature equilibrium, and the algorithm was derived from the original MIXPAC[11] package. Additional information concerning the EOS mixing algorithm is found in reference[12]. Before calling `eos_Mix`, the host code may need to call `eos_SetOption` so the desired interpolation and/or mixing options can be changed from the documented defaults. The units of the `xVals`, `yVals`, `fVals`, `dFx` and `dFy` arguments listed below are determined by the units listed for each `tableType` in

⁵These are cross-referenced to those of EOSPAC 5[9],[10] within [APPENDIX C](#).

APPENDICES B and C. Additional details about the associated pressure and temperature balance functions, which are used internally to achieve pressure and temperature equilibrium, are found in table E-7 in APPENDIX E.

The input arguments are:

nTables	This is the total number of data tables on which to operate.
tableHandles	This is an array of EOS_INTEGER handles to the tables to be mixed.
nXYPairs	This is the total number of pairs of independent variable values provided for interpolation for each table.
concInMix	This is an EOS_REAL array containing the number fraction concentration corresponding to each independent variable value pair and to each tableHandle of the desired data to mix. There are nTables*nXYPairs elements in concInMix, and it is stored sequentially in memory as follows: $[\text{concInMix}(i+(j-1)*\text{nXYPairs}): i=1 \text{ to } \text{nXYPairs}], j=1 \text{ to } \text{nTables}$ Note that the index, i, varies fastest as memory addresses increase incrementally.
xVals	This is an EOS_REAL array of the primary independent variable values to use during interpolation. There are nXYPairs elements in xVals.
yVals	This is an EOS_REAL array of the secondary independent variable values to use during interpolation. There are nXYPairs elements in yVals.

The output arguments are:

fVals	This is an EOS_REAL array of the interpolated data corresponding to the given independent variable data (x and y). There are nXYPairs elements in fVals.
dFx	This is an EOS_REAL array of the interpolated partial derivatives of fVals with respect to x. There are nXYPairs elements in dFx.

dFy	This is an EOS_REAL array of the interpolated partial derivatives of fVals with respect to y. There are nXYPairs elements in dFy.
errorCode	This is a scalar EOS_INTEGER variable to contain an error code. See APPENDIX H for error code details. NOTE: <i>As of version 6.3, comparison of two error codes now requires the usage of the <code>eos_ErrorCodesEqual</code> routine described in chapter 7, section 1.1.</i>

3 C/C++ LANGUAGE BINDINGS

```

void eos_CheckExtrap (EOS_INTEGER *tableHandle,
                    EOS_INTEGER *nXYPairs,
                    EOS_REAL *xVals,
                    EOS_REAL *yVals,
                    EOS_INTEGER *xyBounds,
                    EOS_INTEGER *errorCode);

void eos_Interpolate (EOS_INTEGER *tableHandle,
                    EOS_INTEGER *nXYPairs,
                    EOS_REAL *xVals,
                    EOS_REAL *yVals,
                    EOS_REAL *fVals,
                    EOS_REAL *dFx,
                    EOS_REAL *dFy,
                    EOS_INTEGER *errorCode);

```

```
void eos_Mix      (EOS_INTEGER *nTables,  
                  EOS_INTEGER *tableHandles,  
                  EOS_INTEGER *nXYPairs,  
                  EOS_REAL *concInMix,  
                  EOS_REAL *xVals,  
                  EOS_REAL *yVals,  
                  EOS_REAL *fVals,  
                  EOS_REAL *dFx,  
                  EOS_REAL *dFy,  
                  EOS_INTEGER *errorCode);
```

Use the header file named “eos_Interface.h” to define both the function prototypes listed above and the necessary constants used by EOSPAC. See [chapter 10](#) for usage examples of these routines.

4 FORTRAN LANGUAGE BINDINGS

```
subroutine eos_CheckExtrap (EOS_INTEGER tableHandle,  
                           EOS_INTEGER nXYPairs,  
                           EOS_REAL xVals(*),  
                           EOS_REAL yVals(*),  
                           EOS_INTEGER xyBounds(*),  
                           EOS_INTEGER errorCode)  
subroutine eos_Interpolate (EOS_INTEGER tableHandle,  
                            EOS_INTEGER nXYPairs,  
                            EOS_REAL xVals(*),  
                            EOS_REAL yVals(*),  
                            EOS_REAL fVals(*),  
                            EOS_REAL dFx(*),  
                            EOS_REAL dFy(*),  
                            EOS_INTEGER errorCode)
```

```
subroutine eos_Mix      (EOS_INTEGER nTables,  
                       EOS_INTEGER tableHandles(*),  
                       EOS_INTEGER nXYPairs,  
                       EOS_REAL concInMix(*),  
                       EOS_REAL xVals(*),  
                       EOS_REAL yVals(*),  
                       EOS_REAL fVals(*),  
                       EOS_REAL dFx(*),  
                       EOS_REAL dFy(*),  
                       EOS_INTEGER errorCode)
```

Within a Fortran 77 host code, use the header file named “eos_Interface.fi” to define the necessary constants used by EOSPAC. See [chapter 10](#) for Fortran 77 host code examples of using these routines.

Within a Fortran 90 host code, use the Fortran module named “eos_Interface” to define the necessary constants used by EOSPAC. See [chapter 10](#) for Fortran 90 host code examples of using these routines.

7 MISCELLANEOUS INFORMATION ROUTINES

Data is a tool for enhancing intuition.

– Hilary Mason

This section provides descriptions of some routines that submit or return miscellaneous information about or related to a data table or its contents. These routines are the only way to set or retrieve this information.

1 ROUTINES AND PARAMETERS

The routines and parameters that provide miscellaneous information are shown below.

1.1 eos_ErrorCodesEqual

NEW for 6.3 The eos_ErrorCodesEqual routine is used to determine if the provided EOSPAC error code corresponds to a specified standard error code. This routine is required because the error codes returned by most EOSPAC 6 routines are now encoded with an associated table handle, which means their values are dynamic. Only the EOS_OK error code is exempt from using this routine to test equivalence.

The input arguments are:

err1	This is a scalar EOS_INTEGER that corresponds to either the error code in question or a standard error code defined in APPENDIX H .
err2	This is a scalar EOS_INTEGER that corresponds to either the error code in question or a standard error code defined in APPENDIX H .

The output arguments are:

result	This is a scalar EOS_BOOLEAN to contain the true/false equivalence status of err1 and err2.
--------	---

1.2 eos_GetErrorCode

The eos_GetErrorCode routine is used to the most recent EOSPAC error code that corresponds to a specific table handle.

The input argument is:

tableHandle	This is a scalar EOS_INTEGER handle to a particular data table. The host code is responsible for managing this table handle.
-------------	--

The output argument is:

errorCode	This is a scalar EOS_INTEGER to contain the requested error code. See APPENDIX H for error code details.
-----------	--

NOTE: *As of version 6.3, comparison of two error codes now requires the usage of the [eos_ErrorCodesEqual](#) routine described in [chapter 7 section 1.1](#).*

1.3 eos_GetErrorMessage

The input argument is:

errorCode	This is a scalar EOS_INTEGER to contain an error code.
-----------	--

The output argument is:

errorMsg	This is a character string of a maximum length defined by the constant named EOS_MaxErrMsgLen.
----------	--

1.4 eos_GetTableCmnts

The eos_GetTableCmnts routine returns the comments available about the requested data table. The eos_GetTableCmnts routine operates on a single data table corresponding to a valid table handle.

Before calling eos_GetTableCmnts, the host code must call [eos_GetTableInfo](#) to find out the length of the comments, lenCmnts, allowing the host code to allocate adequate storage.

The input argument is:

tableHandle	This is the scalar EOS_INTEGER handle to particular data table.
-------------	---

The output arguments are:

cmntStr	This is a string of EOS_CHAR, of length lenCmnts, containing the requested comments. The value of lenCmnts for each tableHandle can be obtained by calling eos_GetTableInfo using the constant named EOS_Cmnt_Len (see APPENDIX E for details). If dynamic memory allocation for strings is not possible, then eos_GetTableCmnts will prove difficult to use.
---------	---

`errorCode` This is a scalar `EOS_INTEGER` variable to contain an error code that may indicate the comment table(s) could not be loaded. The host code must call `eos_GetErrorCode` and `eos_GetErrorMessage` to retrieve error details for a specified `tableHandle`. See [APPENDIX H](#) for error code details.

NOTE: *As of version 6.3, comparison of two error codes now requires the usage of the `eos_ErrorCodesEqual` routine described in [chapter 7 section 1.1](#).*

1.5 eos_GetTableInfo

The `eos_GetTableInfo` routine returns the values of requested information about data table members. This routine operates on a single data table corresponding to a valid table handle. Information is requested by passing a list of parameters to the routine that returns the requested information in the same order. The information that can be requested is in [APPENDIX E](#).

The input arguments are:

<code>tableHandle</code>	This is the <code>EOS_INTEGER</code> handle to particular data table.
<code>numInfoItems</code>	<code>EOS_INTEGER</code> scalar number of information items requested.
<code>infoItems</code>	This is an <code>EOS_INTEGER</code> array of information items requested. The allowed values are in APPENDIX E .

The output arguments are:

<code>infoVals</code>	This is an <code>EOS_REAL</code> array containing the information items requested. It contains <code>numInfoItems</code> values. The values are in the same order as requested in the <code>infoItems</code> array.
<code>errorCode</code>	This is a scalar <code>EOS_INTEGER</code> to contain the error code. The host code must call <code>eos_GetErrorCode</code> and <code>eos_GetErrorMessage</code> to retrieve error details for a specified <code>tableHandle</code> . See APPENDIX H for error code details.

NOTE: *As of version 6.3, comparison of two error codes now requires the usage of the `eos_ErrorCodesEqual` routine described in [chapter 7 section 1.1](#).*

1.6 eos_GetMetaData

NEW for 6.3 The `eos_GetMetaData` routine returns the value of requested meta information corresponding to a pair of constants, which are supplied to the routine by the host code. This routine reveals meta-data that is used internally by EOSPAC; therefore, no valid table handle is required prior to its use. The information that can be requested is defined in [APPENDIX F](#).

The input arguments are:

<code>infoItem</code>	This is a scalar <code>EOS_INTEGER</code> used to specify the desired information item. The allowed values are in APPENDIX F .
<code>infoItemCategory</code>	This is a scalar <code>EOS_INTEGER</code> used to specify the category of the desired information item. The allowed values are in APPENDIX F .

The output arguments are:

<code>infoStr</code>	This is a character string containing the information item requested. This string must be allocated by the host code, and it is required to be the minimum length of <code>EOS_META_DATA_STRLEN</code> characters.
<code>errorCode</code>	This is a scalar <code>EOS_INTEGER</code> to contain the error code. The host code must call <code>eos_GetErrorCode</code> and <code>eos_GetErrorMessage</code> to retrieve error details for a specified <code>tableHandle</code> . See APPENDIX H for error code details.

NOTE: *As of version 6.3, comparison of two error codes now requires the usage of the `eos_ErrorCodesEqual` routine described in [chapter 7 section 1.1](#).*

1.7 eos_GetTableMetaData

NEW for 6.3 The `eos_GetTableMetaData` routine returns the value of requested meta information corresponding to a valid table handle and a constant, which is defined in [APPENDIX F](#).

The input arguments are:

tableHandle	This is the EOS_INTEGER handle to particular data table.
infoItem	This is a scalar EOS_INTEGER specifying the desired information item. The allowed values are in APPENDIX F .

The output arguments are:

infoStr	This is a character string containing the information item requested. This string must be allocated by the host code, and it is required to be the minimum length of EOS_META_DATA_STRLEN characters.
errorCode	This is a scalar EOS_INTEGER to contain the error code. The host code must call eos_GetErrorCode and eos_GetErrorMessage to retrieve error details for a specified tableHandle. See APPENDIX H for error code details.

NOTE: *As of version 6.3, comparison of two error codes now requires the usage of the [eos_ErrorCodesEqual](#) routine described in [chapter 7 section 1.1](#).*

1.8 eos_GetVersion

The [eos_GetVersion](#) routine is used to retrieve a character string defining the current version of EOSPAC.

There are no input arguments.

The output argument is:

Version	This is a character string of a maximum length defined by the value returned by eos_GetVersionLength .
---------	--

1.9 eos_GetVersionLength

The [eos_GetVersionLength](#) routine is used to retrieve the length of the string returned by [eos_GetVersion](#).

There are no input arguments.

The output argument is:

Length	This is a scalar EOS_INTEGER defining the length of the string returned by eos_GetVersion. This length includes the null ('\0') terminating character, which is used in the “C” programming language.
--------	---

1.10 eos_ResetOption

The eos_ResetOption routine is used to reset an option related to a specified table handle to its default state (see APPENDICES D and F for default settings). The eos_ResetOption routine is used prior to calling eos_LoadTables, eos_Interpolate, and/or eos_Mix to specify applicable options for each table handle.

The input arguments are:

tableHandle	This is a scalar EOS_INTEGER handle to a particular data table. The host code is responsible for managing this table handle.
tableOption	This is a scalar EOS_INTEGER containing the option flag indicating what option to set corresponding to the tableHandle. See APPENDICES D and F for table option details.

The output argument is:

errorCode	This is a scalar EOS_INTEGER to contain the error code. The host code must call eos_GetErrorCode and eos_GetErrorMessage to retrieve error details for a specified tableHandle. See APPENDIX H for error code details.
-----------	--

NOTE: As of version 6.3, comparison of two error codes now requires the usage of the eos_ErrorCodesEqual routine described in chapter 7 section 1.1.

1.11 eos_SetOption

The `eos_SetOption` routine is used to set an option related to a specified table handle. The `eos_SetOption` routine is used prior to calling `eos_LoadTables`, `eos_Interpolate`, and/or `eos_Mix` to specify applicable options for each table handle.

The input arguments are:

<code>tableHandle</code>	This is a scalar <code>EOS_INTEGER</code> handle to a particular data table. The host code is responsible for managing this table handle.
<code>tableOption</code>	This is a scalar <code>EOS_INTEGER</code> containing the option flag indicating what option to set corresponding to the <code>tableHandle</code> . See APPENDICES D and F for table option details.
<code>tableOptionVal</code>	This is a scalar <code>EOS_REAL</code> containing the option value to be assigned to the <code>tableHandle</code> . Note that not all of the option flags defined in APPENDICES D and F use this value; however, a variable or literal is required when calling <code>eos_SetOption</code> due to the limitations of a flat public interface.

The output argument is:

<code>errorCode</code>	This is a scalar <code>EOS_INTEGER</code> to contain the error code. The host code must call <code>eos_GetErrorCode</code> and <code>eos_GetErrorMessage</code> to retrieve error details for a specified <code>tableHandle</code> . See APPENDIX H for error code details.
------------------------	---

NOTE: *As of version 6.3, comparison of two error codes now requires the usage of the `eos_ErrorCodesEqual` routine described in [chapter 7 section 1.1](#).*

2 C/C++ LANGUAGE BINDINGS

```
void eos_ErrorCodesEqual (EOS_INTEGER *err1,
                        EOS_INTEGER *err2,
                        EOS_BOOLEAN *result);
```

```
void eos_GetErrorCode      (EOS_INTEGER *tableHandle,
                           EOS_INTEGER *errorCode);
void eos_GetErrorMessage  (EOS_INTEGER *errorCode,
                           EOS_CHAR errorMsg[EOS_MaxErrMsgLen]);
void eos_GetMetaData      (EOS_INTEGER *infoItem,
                           EOS_INTEGER *infoItemCategory,
                           EOS_CHAR *infoStr,
                           EOS_INTEGER *errorCode);
void eos_GetTableMetaData (EOS_INTEGER *tableHandle,
                           EOS_INTEGER *infoItem,
                           EOS_CHAR *infoStr,
                           EOS_INTEGER *errorCode);
void eos_GetTableCmnts    (EOS_INTEGER *tableHandle,
                           EOS_CHAR *cmntStr,
                           EOS_INTEGER *errorCode);
void eos_GetTableInfo     (EOS_INTEGER *tableHandle,
                           EOS_INTEGER *numInfoItems,
                           EOS_INTEGER infoItems[],
                           EOS_REAL infoVals[],
                           EOS_INTEGER *errorCode);
void eos_GetVersion       (EOS_CHAR *version);
void eos_GetVersionLength (EOS_INTEGER *length);
void eos_ResetOption      (EOS_INTEGER *tableHandle, const
                           EOS_INTEGER *tableOption,
                           EOS_INTEGER *errorCode);
void eos_SetOption        (EOS_INTEGER *tableHandle, const
                           EOS_INTEGER *tableOption, const
                           EOS_REAL *tableOptionVal,
                           EOS_INTEGER *errorCode);
```

Use the header file named “eos_Interface.h” to define both the function prototypes listed above and the necessary constants used by EOSPAC. See [chapter 10](#) for usage examples of these routines.

3 FORTRAN LANGUAGE BINDINGS

subroutine eos_ErrorCodesEqual (EOS_INTEGER err1,
EOS_INTEGER err2,
EOS_BOOLEAN result)

subroutine eos_GetErrorCode (EOS_INTEGER tableHandle,
EOS_INTEGER errorCode)

subroutine eos_GetErrorMessage (EOS_INTEGER errorCode,
EOS_CHAR errorMsg(EOS_MaxErrMsgLen))

subroutine eos_GetMetaData (EOS_INTEGER infoItem,
EOS_INTEGER infoItemCategory,
EOS_CHAR infoStr,
EOS_INTEGER errorCode)

subroutine eos_GetTableMetaData (EOS_INTEGER tableHandle,
EOS_INTEGER infoItem,
EOS_CHAR infoStr,
EOS_INTEGER errorCode)

subroutine eos_GetTableInfo (EOS_INTEGER tableHandle,
EOS_INTEGER numInfoItems,
EOS_INTEGER infoItems,
EOS_REAL infoVals,
EOS_INTEGER errorCode)

subroutine eos_GetTableCmnts (EOS_INTEGER tableHandle,
EOS_CHAR cmntStr,
EOS_INTEGER errorCode)

subroutine eos_GetVersion (EOS_CHAR version)

subroutine eos_GetVersionLength (EOS_INTEGER length)

subroutine eos_ResetOption (EOS_INTEGER tableHandle,
EOS_INTEGER tableOption,
EOS_INTEGER errorCode)

subroutine eos_SetOption (EOS_INTEGER tableHandle,
EOS_INTEGER tableOption,
EOS_REAL tableOptionVal,
EOS_INTEGER errorCode)

Within a Fortran 77 host code, use the header file named “eos_Interface.fi” to define the necessary constants used by EOSPAC. See [chapter 10](#) for Fortran 77 host code examples of using these routines.

Within a Fortran 90 host code, use the Fortran module named “eos_Interface” to define the necessary constants used by EOSPAC. See [chapter 10](#) for Fortran 90 host code examples of using these routines.

8 TOOLS

I can make just such ones if I had tools, and I could make tools if I had tools to make them with.

– Eli Whitney

If one is interested in calculating quantities without the need to write a host code to use the EOSPAC interface, then there are some utilities, which are distributed with EOSPAC, to accomplish various tasks. One such utility was previously mentioned (see [section 4](#)), `get_sesame_data`. Given a Sesame material ID and table number, `get_sesame_data` will extract data from Sesame database and send it to stdout in a format compatible with GNUPLOT's input requirements for a 2-D plot. There are several command line variations for `get_sesame_data`:

1. `get_sesame_data [OPTIONS] <sesMaterialNum> <sesTableNum> [<sesSubtableIndex>]`
2. `get_sesame_data [OPTIONS] id [<file>]`
3. `get_sesame_data [OPTIONS] tables <sesMaterialNum> [, <sesMaterialNum> [, ...]]`
4. `get_sesame_data [OPTIONS] comments <sesMaterialNum> [, <sesMaterialNum> [, ...]]`
5. `get_sesame_data [OPTIONS] <sesMaterialNum>`

`<sesMaterialNum>` Sesame material ID number
`<sesTableNum>` Sesame table number
`<sesSubtableIndex>` Optional Sesame subtable number (default=1).

Another utility named `interp_sesame_data` is also distributed with EOSPAC that allows a user to perform various data interpolations from the command line. There are multiple command line variations for `interp_sesame_data`:

1. `interp_sesame_data` [`<OPTIONS>`] `<sesMaterialNum>` `<tableType>` `<x>[:<x1>]`
`[<y>[:<y1>]]`
2. `interp_sesame_data` [`<OPTIONS>`] `<sesMaterialNum>` `<tableType>` `-i <file>`

`<sesMaterialNum>`

Sesame material ID number

`<tableType>`

EOSPAC 6 table type (case insensitive)

`<x>`

First independent variable value of the table type (64-bit floating point)
 This argument is required unless either the `'-i'` or the `'-x'` option is used.
 The optional `:<x1>` defines an upper bound for a randomly-sampled range of values between `<x>` and `<x1>`.

`<y>`

Second independent variable value of the table type (64-bit floating point)
 This argument is required unless either the `'-i'` or the `'-y'` option is used.
 The optional `:<y1>` defines an upper bound for a randomly-sampled range of values between `<y>` and `<y1>`.

NEW for 6.5.11

A third utility, named `convert_sesame_data`, was completed for the 6.5.11 release of the EOSPAC interface library. This tool provides a simple mechanism to convert between supported SESAME file formats (e.g., IEEE64 binary^[1], legacy ASCII^[13], ASCII2^[14] and Helios¹).

USAGE: `convert_sesame_data` [`<OPTIONS>`] `<SESAME_FILE>` [`<OUTPUT_FILE>`]

`<SESAME_FILE>` - Source SESAME file name (required)

`<OUTPUT_FILE>` - Output SESAME file name (optional)

NEW for 6.5.13

Finally, a fourth utility was introduced for the 6.5.13 release, and it is named `mix_sesame_data`.² This tool allows users to create arbitrary mixed equations of state using an input file and the `eos_Mix` function, which is described in [section 2.3](#), and these are approximations of pressure-temperature equilibrium mixtures. While care has been taken to provide stable mixture

¹The Helios format is a single-precision variant of the legacy ASCII format^[13], which has been historically used to distribute SESAME data beyond the confines of Los Alamos National Laboratory.

²Yes, there is definitely an unimaginative naming pattern that has emerged in this chapter.

equations of state, it is important to note that not all possible combinations of SESAME data tables will be successfully mixed. The `mix_sesame_data` tool provides a simple command line interface for `mix_sesame_data`, and its documentation provides extensive details and examples for the `<INPUT_FILE>` and the small set of command line `<OPTIONS>`:

```
USAGE: mix_sesame_data [<OPTIONS>] <INPUT_FILE>
```

```
<INPUT_FILE>
```

```
Specify the required <INPUT_FILE>, which contains information as described  
in the INPUT FILE PARSER section of the included help.
```

[Chapter 3 section 4.8](#) of this document provides some expressions that the user may find to be useful when creating new mixture equations of state from SESAME data.

All of the utilities described in this section include individual documentation, which can be viewed by either using the desired tool's “-h” option or reading its associated “readme” file that is collocated with the compiled executable (e.g., `mix_sesame_data.readme`).

9 SELECTED NUMERIC DETAILS

The reason is not to glorify “bit chasing”; a more fundamental issue is at stake here: Numerical subroutines should deliver results that satisfy simple, useful mathematical laws whenever possible. [...] Without any underlying symmetry properties, the job of proving interesting results becomes extremely unpleasant. The enjoyment of one’s tools is an essential ingredient of successful work.

– Donald Knuth, *Vol. II, Seminumerical Algorithms, Section 4.2.2 part A, final paragraph*

This section provides additional descriptions of some complex EOSPAC features, which are implemented to address some numeric issue or other.

1 CUSTOM SMOOTHING AND INTERPOLATION

At the request of the user community, some very specific data smoothing capabilities of SAGE¹ have been added to EOSPAC. These features correspond to the setup and interpolation options EOS_PT_SMOOTHING, EOS_ADJUST_VAP_PRES, and EOS_USE_CUSTOM_INTERP. When the setup option, EOS_PT_SMOOTHING, is enabled for a table handle, the loaded equation of state data is smoothed in preparation for using the EOS_USE_CUSTOM_INTERP interpolation option. The setup option, EOS_ADJUST_VAP_PRES, is provided as a mechanism to shift the vapor pressure data according to

$$P_i = P_i - P_{shift} \left(1 - \frac{P_i}{P_{i-1}} \right) \quad (9.1)$$

¹SAGE is a one-, two-, and three-dimensional, multi-material Eulerian hydrodynamics code (LA-UR-04-2959).

where P_{shift} is the user-provided pressure value (GPa) that is used to ensure the ambient conditions of the tabulated data are acceptable. This vapor pressure adjustment has often been deemed necessary, and it is material-dependent. Once the data has been loaded and smoothed according to the rules associated with `EOS_PT_SMOOTHING` and `EOS_ADJUST_VAP_PRES`, interpolation may be performed with the `EOS_USE_CUSTOM_INTERP` option set. This interpolation option is limited to use with the `EOS_Ut_PtT` and `EOS_V_PtT` data types (pressure- and temperature-dependent internal-energy and specific volume respectively). The `EOS_USE_CUSTOM_INTERP` interpolation option uses linear interpolation to calculate the desired values from isotherms, which contain data made to conform to Maxwell's relations (Maxwell Construction, or Equal Area Construction). Any basic thermodynamics textbook should contain a description of Maxwell's relations.

2 FORCED DATA MONOTONICITY

Much of the data in the SESAME database is not monotonic with respect to one or both of the tabulated independent variables. This is a problem when a data table is to be inverted with respect to one of the tabulated independent variables. To ensure either global increasing- or decreasing-monotonicity, a simple algorithm is used, in which the average of a function's local minimum and local maximum is determined and then used to replace the local tabulated function values. Once that is done monotonicity is achieved, but a small slope is then imposed over the localized region so that either global increasing- or decreasing-monotonicity is imposed. The aforementioned small slope is calculated to be three-orders-of-magnitude larger than the machine's floating point precision (i.e., 10^{-12} on a 64-bit IEEE machine). It is important to note that this forced data monotonicity algorithm is not an "equal-area" calculation, which is used to impose Maxwell constructions on an EOS. [Figure 9.1](#) graphically describes the result (orange line) of this algorithm when applied to an isotherm (blue line) of an arbitrary pressure function. Although it cannot be seen due to the plot's pressure range, the orange line actually has an artificial slope of approximately 10^{-12} in the nearly-horizontal region, where the left-most pressure value of said region is the average of P_{min} and P_{max} . The aforementioned monotonicity-enforcing algorithm is imposed from the independent variable's minimum to maximum values.

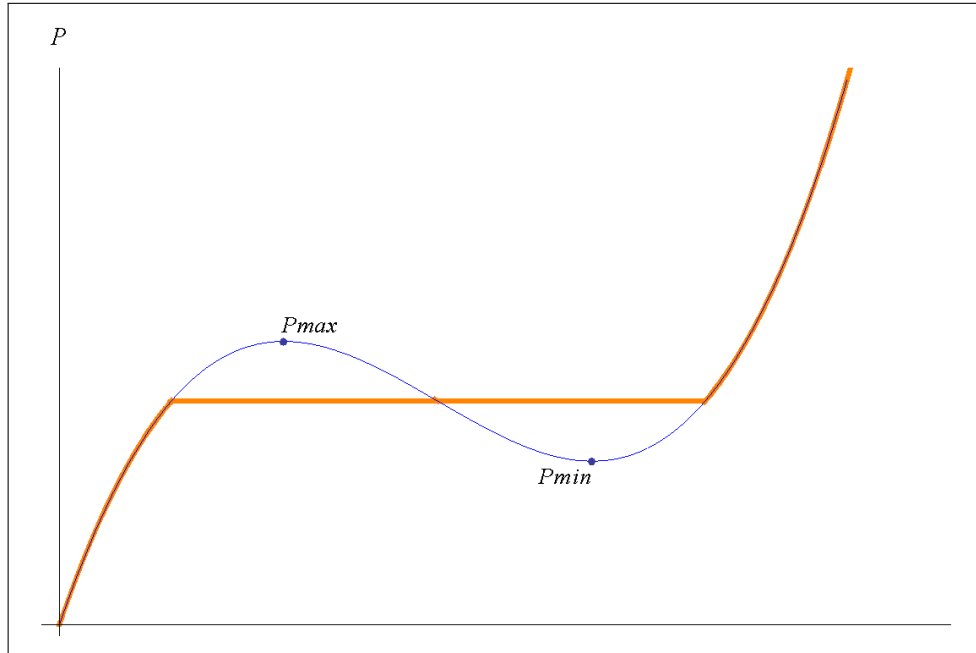


Figure 9.1: General depiction of $P(\rho)$ forced to be monotonically-increasing.

3 EXTENDED PRECISION IS DISABLED

In an effort to improve the portability of EOSPAC, the extended precision features of some machine architectures are disabled upon entry to any of EOSPAC's public routines, and then the extended precision is re-enabled prior to exiting said public routines. The problem of extended precision is described as follows[15]:

The IEEE-754 standard defines the bit-level behavior of floating-point arithmetic operations on all modern processors. This allows numerical programs to be ported between different platforms with identical results, in principle. In practice, there are often minor variations caused by differences in the order of operations (depending on the compiler and optimization level) but these are generally not significant.

However, more noticeable discrepancies can be seen when porting numerical programs between x86 systems and other platforms, because the the x87 floating point unit (FPU) on x86 processors computes results using extended precision internally (the values being

converted to double precision only when they are stored to memory). In contrast, processors such as SPARC, PA-RISC, Alpha, MIPS and POWER/PowerPC work with native double-precision values throughout. The differences between these implementations lead to changes in rounding and underflow/overflow behavior, because intermediate values have a greater relative precision and exponent range when computed in extended precision. In particular, comparisons involving extended precision values may fail where the equivalent double precision values would compare equal.

To avoid these incompatibilities, the x87 FPU also offers a hardware double-precision rounding mode. In this mode the results of each extended-precision floating-point operation are rounded to double precision in the floating-point registers by the FPU. It is important to note that the rounding only affects the precision, not the exponent range, so the result is a hybrid double-precision format with an extended range of exponents. On BSD systems such as FreeBSD, NetBSD and OpenBSD, the hardware double-precision rounding mode is the default, giving the greatest compatibility with native double precision platforms. On x86 GNU/Linux systems the default mode is extended precision (with the aim of providing increased accuracy). To enable the double-precision rounding mode it is necessary to override the default setting on per-process basis using the FLDCW "floating-point load control-word" machine instruction.

As a result of the problem described above, every effort is made to disable extended precision arithmetic on x86 machines.

4 MASS FRACTION DATA INTERPOLATION

For selected materials, Sesame contains mass fraction data tables, which tabulate phase-specific (i.e., beta, gamma, liquid, etc.) mass fraction data. EOSPAC has the capability to access and interpolate this mass fraction data if it's available. To implement this capability while minimizing changes to the public interface specification, the `eos_GetTableInfo` function ([chapter 7 section 1.5](#)) is used with `eos_Interpolate` ([chapter 6 section 2.2](#)) in an unusual way. Once the material data is loaded into memory using the `EOS_M_DT` data type option, the host code must call `eos_GetTableInfo` to obtain the total number of tabulated phases (see the `EOS_NUM_PHASES` parameter in APPENDIX E). Then the `eos_Interpolate` output array (`fVals`) must be allocated so that all of the material's phases can be interpolated at once; however, allocation of the derivative arrays (`dFx` and `dFy`) is not required since they are ignored within EOSPAC. For example, if `nXYPairs` is set to ten and the

number of phases is three, then the output array for eos_Interpolate are each allocated to hold thirty values; whereas, the input arrays (xVals and yVals) are only allocated to hold ten values. The interpolated output is organized so that each phase's interpolated mass fractions are stored in turn. The following "C" code snippet demonstrates how the input and output arrays are organized:

```
xVals = (EOS_REAL *) malloc (sizeof (EOS_REAL) * nXYPairs);
yVals = (EOS_REAL *) malloc (sizeof (EOS_REAL) * nXYPairs);
fVals = (EOS_REAL *) malloc (sizeof (EOS_REAL) * num_phases * nXYPairs);
for (j = 0; j < num_phases; j++) {
    for (k = 0; k < nXYPairs; k++) {
        printf ("%23.15e %23.15e %23.15e\n",
                xVals[k], yVals[k], fVals[nXYPairs*j + k]);
    }
}
```

To maintain data integrity, the interpolation is limited to use only the bilinear (EOS_LINEAR) interpolator for the EOS_M_DT data type.

5 NUMERICAL INTEGRATION

The capability to calculate entropy data is implemented with multiple algorithms. One such algorithm depends upon the numerical integration of the tabulated internal energy data with respect to temperature. To perform the numerical integration, a simple trapezoid rule is implemented. A specific note of interest is that the trapezoid integration equally-divides each tabulated temperature interval into ninety-nine sub-intervals prior to interpolation. The hard-wired number of sub-intervals was chosen arbitrarily because it seemed adequate. Another, but more subtle, item to note is that the form of [equation \(3.9\)](#) is implemented within EOSPAC so that the integrand values, $\frac{u}{T^2}$, for all applicable tabulated data are passed to the interpolator within the trapezoid integration algorithm instead of interpolating the internal energy, $u = u(\rho, T)$, prior to calculating the integrand. This smoothes the calculated results by damping incurred numerical errors. The $\frac{u}{T}$ and $\frac{u}{T^2}$ quantities are presumed to be zero when $T = 0$, so that the evaluated entropy data are consistent with the third law of thermodynamics (i.e., $s|_{T=0} = 0$ as shown in [equation \(3.10\)](#)). Of course, this last assumption depends upon the inherent data features, which are produced by the models used to generate the SESAME data, that require the internal energy to approach the zero limit faster than the associated temperature.

6 LINEAR AND BILINEAR INTERPOLATION

As of EOSPAC 6.2, the default linear/bilinear interpolators were replaced with a new algorithm that calculates continuous derivatives at the tabulated grid points. This feature was a departure from the discontinuous derivatives calculated by all previous versions of EOSPAC.

NEW for 6.3 A new interpolation option is introduced to allow a user to mitigate some unforeseen side effects of the continuous derivatives. The option name is `EOS_DISCONTINUOUS_DERIVATIVES`, because it reintroduces the original linear/bilinear interpolator logic that existed before EOSPAC 6.2.

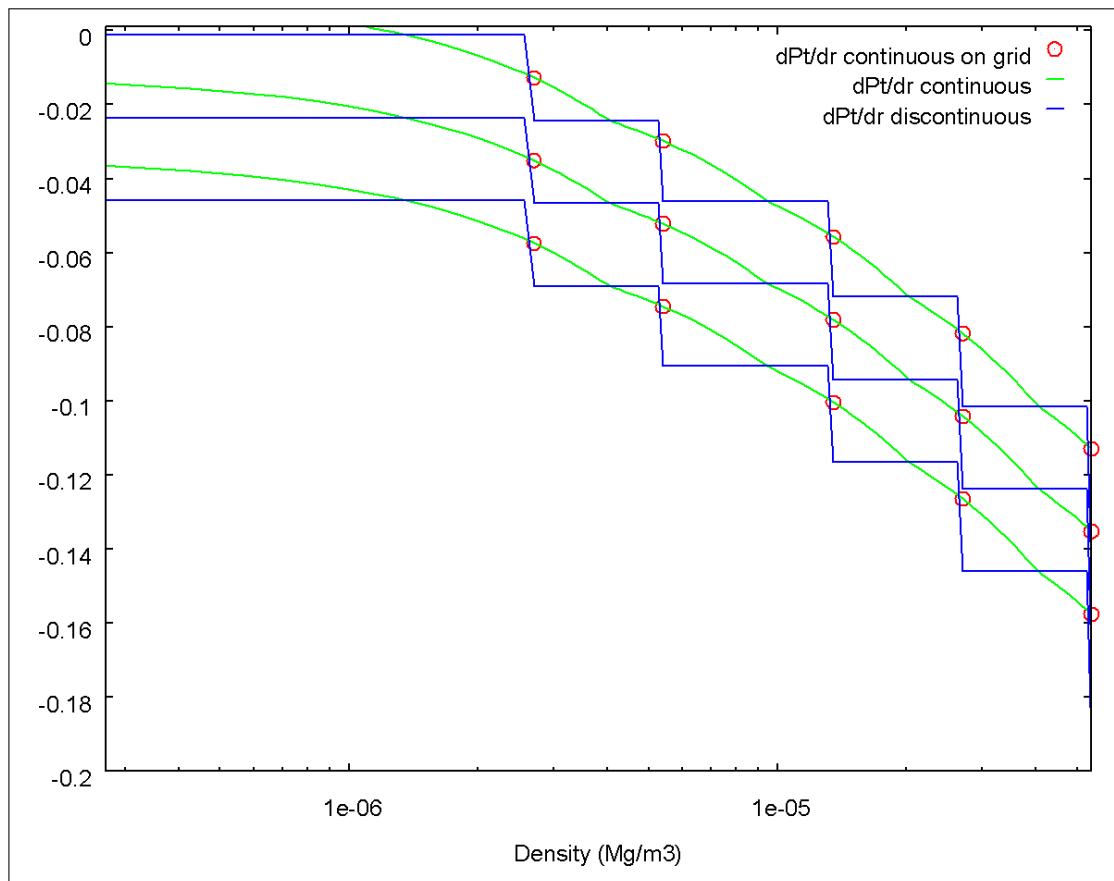


Figure 9.2: Comparison of derivative values for three low temperature isotherms using the `EOS_LINEAR` interpolation option both with and without the `EOS_DISCONTINUOUS_DERIVATIVES` option enabled.

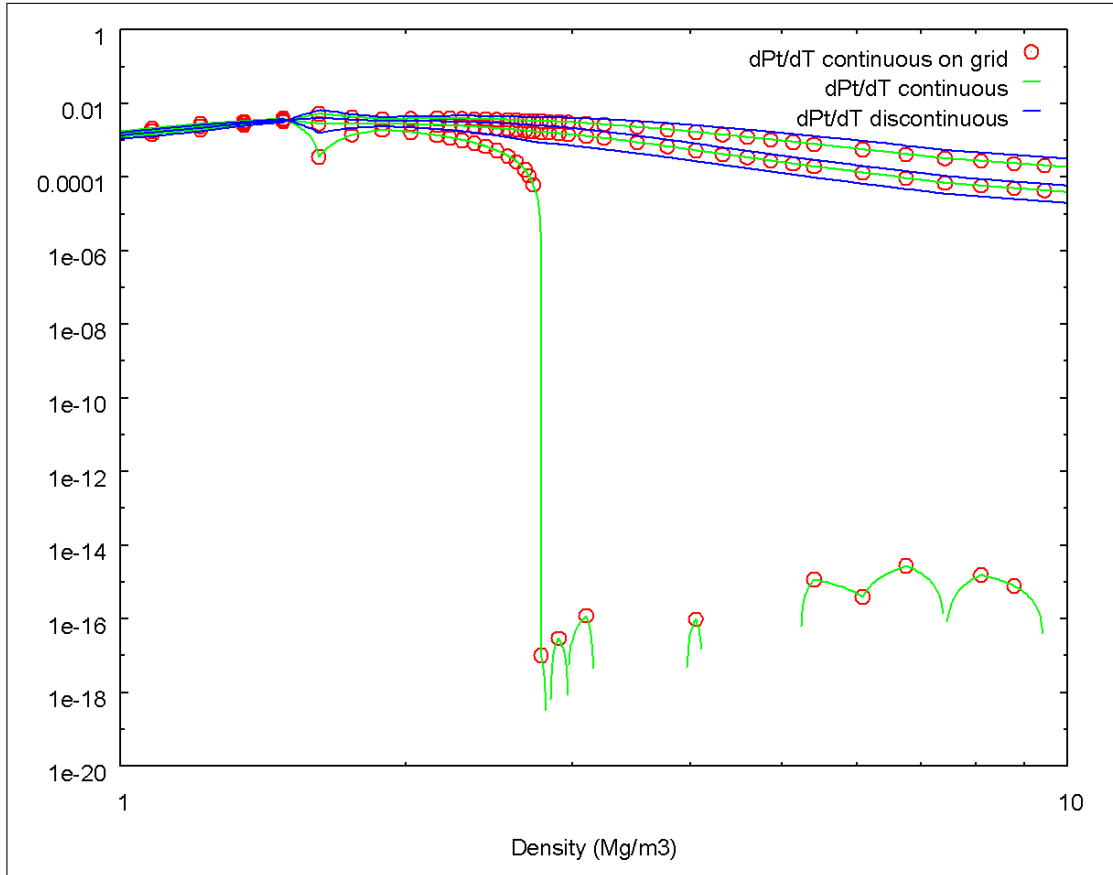


Figure 9.3: Comparison of derivative values for three low temperature isotherms using the EOS_LINEAR interpolation option both with and without the EOS_DISCONTINUOUS_DERIVATIVES option enabled. This demonstrates numerical issues with the current default bilinear interpolator’s continuous derivatives at or near the data table boundary.

Figures 9.2 and 9.3 the differences between the calculated derivatives when using the bilinear interpolator both with and without the EOS_DISCONTINUOUS_DERIVATIVES option enabled. On one hand, Figure 9.2 demonstrates an assumed advantage the continuous derivatives provide.

Unfortunately, Figure 9.3 demonstrates an example of some unforeseen numerical noise introduced by the same continuous derivative calculations — particularly near the data table boundaries where the interpolated values are small. Such numerical noise has been observed away from the tabulated table boundaries where the interpolated values are small, and this behavior can violate the ex-

pected monotonicity-preserving characteristics of the linear/bilinear interpolator for the calculated derivatives.

7 INVERT AT SETUP

NEW for 6.3.1 The `EOS_INVERT_AT_SETUP` option, which was previously described in [chapter 5 section 3](#), allows the host code to force EOSPAC to create inverted tables for each specified table handle during the setup phase. This, of course improves interpolation performance for the affected table(s). The downside to improved performance is that one should expect degraded accuracy for the interpolated results, because the inverted table grid may be of insufficient resolution. It was declared that the quantification of such numeric differences are beyond the scope of this manual; however, it is useful for the user to be aware that additional documentation is available that describes in detail both numerical and performance results associated with the `EOS_INVERT_AT_SETUP` option's usage. [\[16\]](#), [\[17\]](#), [\[18\]](#), [\[19\]](#)

7.1 Data Transformations

In order to highlight how EOSPAC 6 transforms selected data when it is loaded in conjunction with the `EOS_INVERT_AT_SETUP` option, first consider that historical versions of EOSPAC [\[9\]](#), [\[10\]](#), [\[20\]](#) used the following data transforms to create the grids of varied inverted tables:

$$P^*(\rho, T) = \frac{Pt(\rho, T) - Pc(\rho)}{\rho} \quad (9.2)$$

$$U^*(\rho, T) = Ut(\rho, T) - Uc(\rho) \quad (9.3)$$

$$A^*(\rho, T) = At(\rho, T) - Ac(\rho) \quad (9.4)$$

In addition to using those historical transforms, EOSPAC 6 now eliminates the isochore at $\rho = 0$ prior to table inversion, because it causes $P^*(\rho, T) \rightarrow \infty$ and it is not a physically-meaningful state of matter.

One can easily recognize that the transforms defined by [equations \(9.2\) to \(9.4\)](#), eliminate much

of the dynamic range of the Pt , Ut and At by subtracting their associated cold curve data. Additionally, it is apparent from the ideal gas law that internal energy is directly proportional to temperature. Where $v = 1/\rho$, the differential for internal energy is dependent upon (v, T)

$$dU = \left(\frac{\partial U}{\partial T} \right)_v TdT + \left(\frac{\partial U}{\partial v} \right)_T Tdv \quad (9.5)$$

One of the important features of an ideal gas is that its internal energy depends only upon its temperature, so [equation \(9.5\)](#) becomes

$$dU = \left(\frac{\partial U}{\partial T} \right)_v TdT \quad (9.6)$$

From [equations \(9.3\)](#) and [\(9.6\)](#), it is concluded that

$$U^* \propto T \quad (9.7)$$

Similar reasoning is applied to conclude that

$$A^* \propto T \quad (9.8)$$

Similarly, the ideal gas law states that the ratio of pressure and density is directly proportional to temperature:

$$Pv = RT \quad (9.9)$$

The R of [equation \(9.9\)](#) is the Universal Gas Constant.

Given $v = 1/\rho$, [equation \(9.9\)](#) can be rewritten as

$$\frac{P}{\rho} = RT \quad (9.10)$$

From equations (9.2) and (9.10), it is concluded that

$$P^* \propto T \quad (9.11)$$

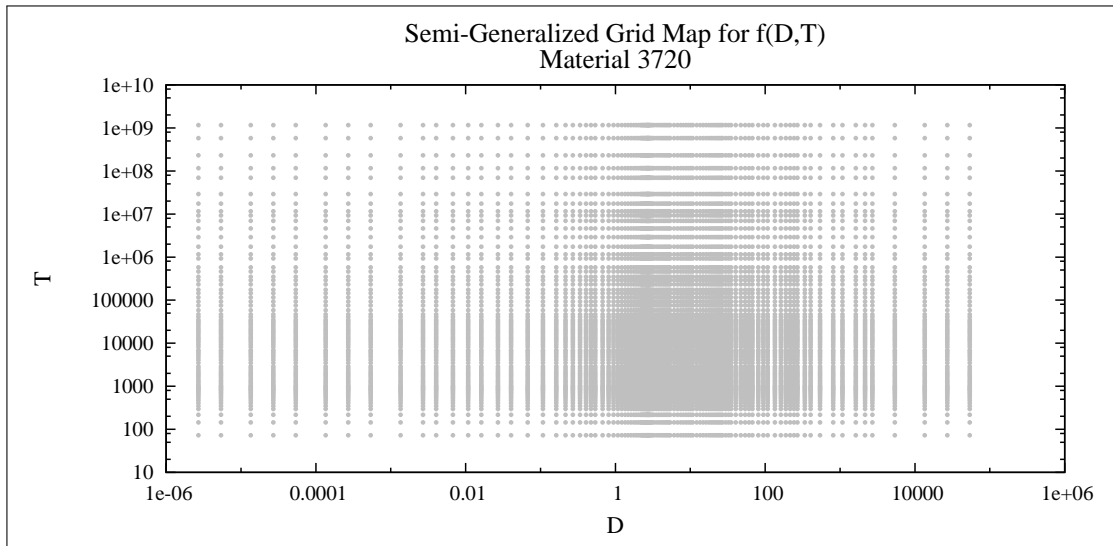


Figure 9.4: Rectangular SESAME grid of density (D) and temperature (T).

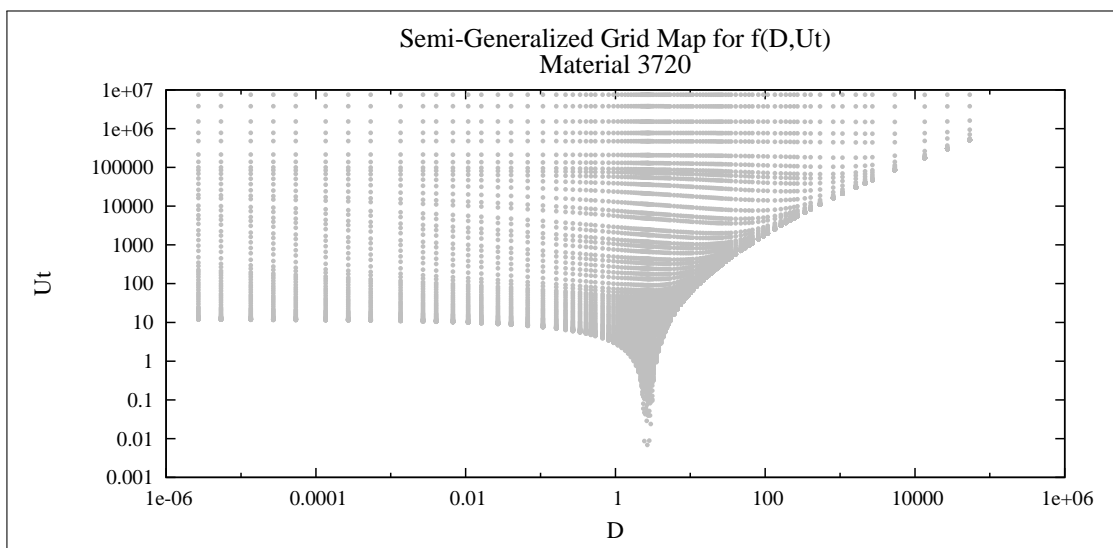


Figure 9.5: Non-rectangular SESAME grid of density (D) and internal energy (Ut).

Given the fact that SESAME data is tabulated with density and temperature as independent variables, it is reasonable to conclude that the transforms of [equations \(9.2\) to \(9.4\)](#) create data that are “temperature-like” quantities, and the non-rectangular grids represented in

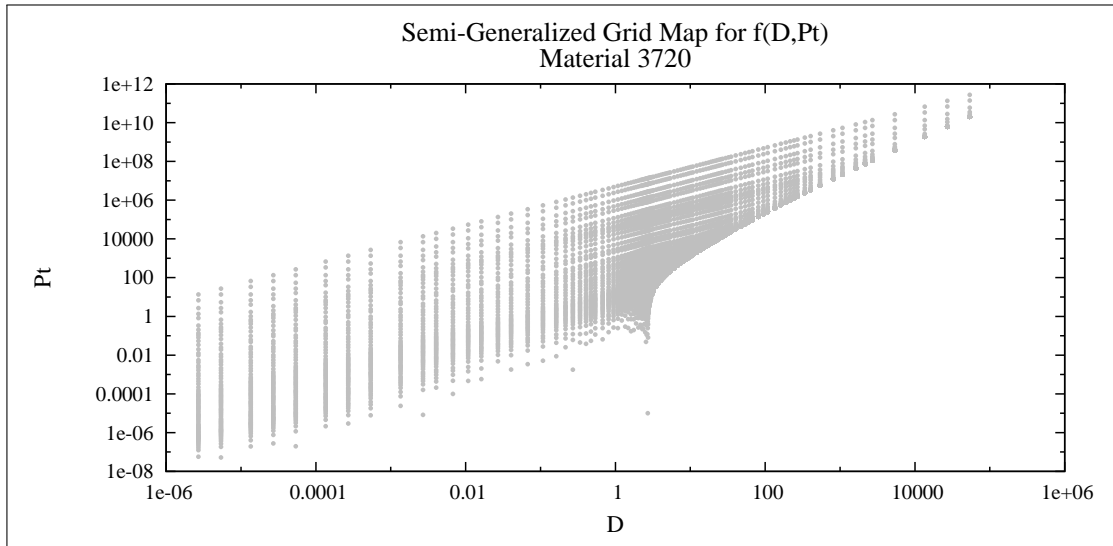


Figure 9.6: Non-rectangular SESAME grid of density (D) and pressure (Pt).

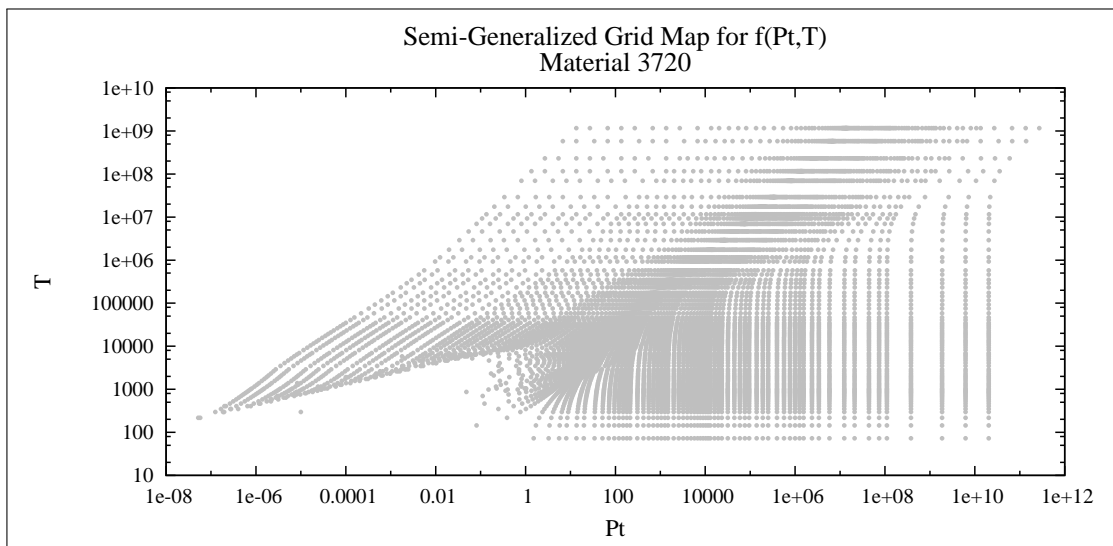


Figure 9.7: Non-rectangular SESAME grid of pressure (Pt) and temperature (T).

figures 9.5 to 9.7 are transformed into rectangular grid of density (ρ) and a “temperature-like” quantity like the representation in figure 9.4. The transformed grids of figures 9.5 and 9.6 are shown in figures 9.8 and 9.9 respectively. While the transformed grids shown in figures 9.8 and 9.9 are

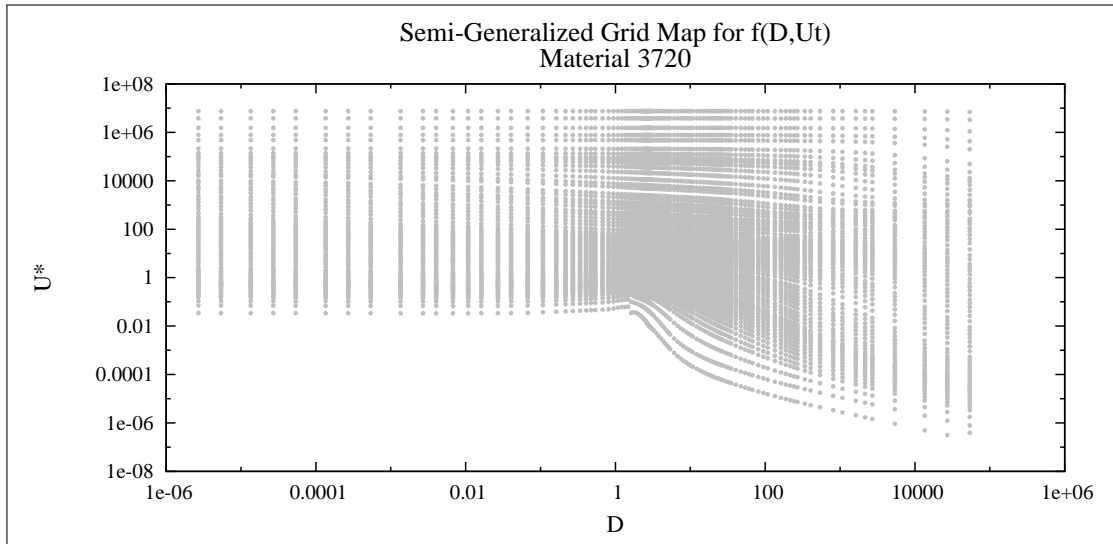


Figure 9.8: Transformed, rectangular SESAME grid of internal energy (U^*) and temperature (T).

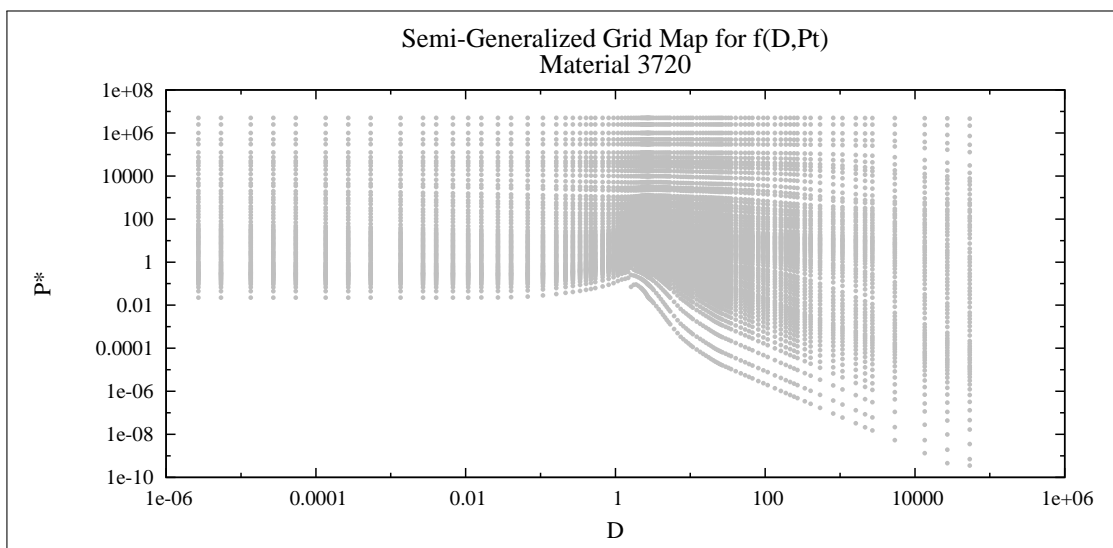


Figure 9.9: Transformed, rectangular SESAME grid of pressure (P^*) and temperature (T).

not perfectly-rectangular like that of [figure 9.4](#), the distributions are similar enough to stabilize the interpolated results over the entire table ranges. No transforms are applied to data associated with (P, T) grid like that in [figure 9.7](#).

7.2 Usage of EOS_INSERT_DATA

It has been shown[17] that using the EOS_INSERT_DATA option can improve the numerical accuracy of interpolation associated with the EOS_INVERT_AT_SETUP option; however, the benefit of this enhancement is dependent upon the selected table type, the chosen SESAME material ID and the number of points inserted (i.e., the EOS_INSERT_DATA option's tableOptionVal argument). Therefore, some trial and error may be required to achieve the desired accuracy when compared to the default interpolation mode without the EOS_INVERT_AT_SETUP option enabled.

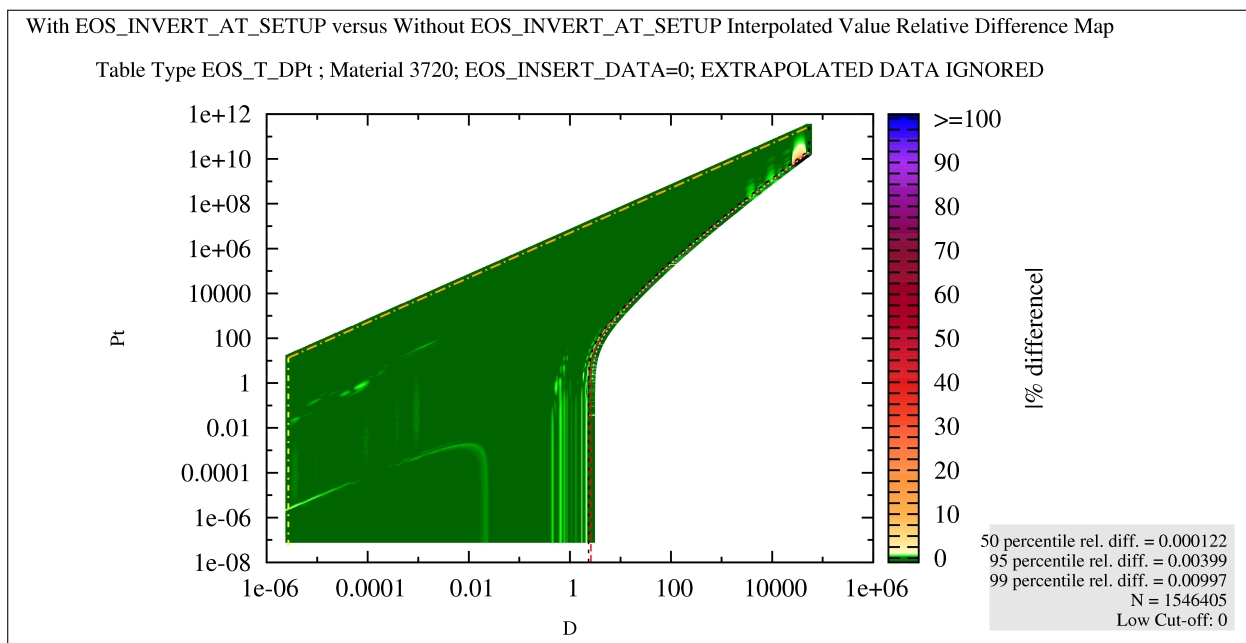


Figure 9.10: Color map of relative differences comparing the interpolation of $T(\rho, Pt)$, which was calculated using both of EOSPAC 6's default and pre-inverted (i.e., EOS_INVERT_AT_SETUP) inverse interpolation modes.

For example, [figures 9.10](#) and [9.11](#) show the birational interpolation accuracy improvement when the EOS_INSERT_DATA option's tableOptionVal=2, which increases the table's associated memory

usage by an approximate factor of 9. The figures show the relative differences between the interpolation results, both with and without the `EOS_INVERT_AT_SETUP` option enabled, are compared. This demonstrates that there is a price to be paid for improved interpolation performance.

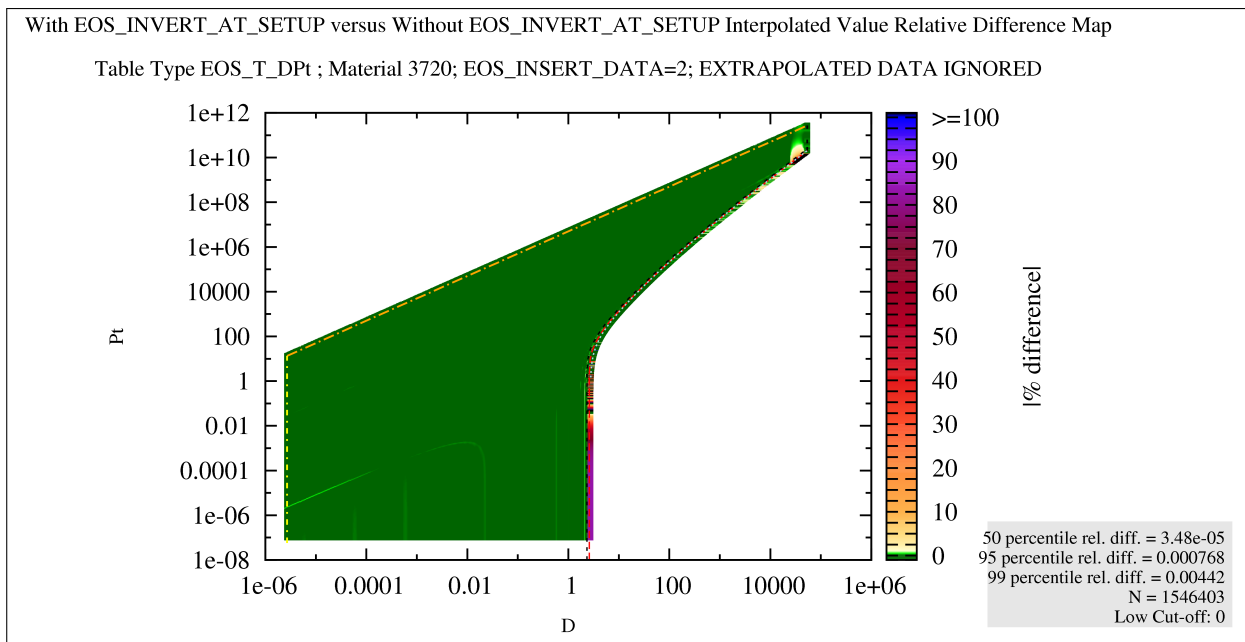


Figure 9.11: Color map of relative differences comparing the interpolation of $T(\rho, P_t)$, which was calculated using both of EOSPAC 6’s default and pre-inverted (i.e., `EOS_INVERT_AT_SETUP`) inverse interpolation modes and the `EOS_INSERT_DATA=2` option enabled.

7.3 The `EOS_X_CONVERT`, `EOS_Y_CONVERT` and `EOS_Z_CONVERT` options usage

Long overdue is this section describing how the various conversion factor options (i.e., `EOS_X_CONVERT`, `EOS_Y_CONVERT` and `EOS_F_CONVERT`) are both used within the EOSPAC interface library’s internal logic and affect the output of the interpolation functions (i.e., `eos.Interpolate` and `eos.Mix`). The interpolation functions assume that the input `xVals` and `yVals` arrays require conversion to SESAME data units, because all SESAME data is loaded in those native SESAME units (see [APPENDIX B](#) for details). For clarity, define the following:

$$X \equiv \text{xVals array data in SESAME units}$$

Y	\equiv	yVals array data in SESAME units
F	\equiv	fVals array data in SESAME units
x	\equiv	xVals array data in host code units
y	\equiv	yVals array data in host code units
f	\equiv	fVals array data in host code units
C_X	\equiv	conversion factor supplied with the EOS_X_CONVERT option
C_Y	\equiv	conversion factor supplied with the EOS_Y_CONVERT option
C_F	\equiv	conversion factor supplied with the EOS_F_CONVERT option

The various conversion factor options are provided for the host code developer's convenience to transform the interpolation function inputs into these native SESAME units, and similarly convert the interpolated results into the host code's native units. This is accomplished using the supplied conversion factors as defined in [equations \(9.12\)](#) and [\(9.13\)](#) upon entry to the interpolation functions.

$$X = \frac{x}{C_X} \quad (9.12)$$

$$Y = \frac{y}{C_Y} \quad (9.13)$$

Upon output, the interpolated values are converted into the host code's desired units using the supplied conversion factors as defined in [equations \(9.14\)](#) to [\(9.16\)](#).

$$f = C_F F \quad (9.14)$$

$$\frac{\partial f}{\partial x} = \left(\frac{C_F}{C_X} \right) \frac{\partial F}{\partial X} \quad (9.15)$$

$$\frac{\partial f}{\partial y} = \left(\frac{C_F}{C_Y} \right) \frac{\partial F}{\partial Y} \quad (9.16)$$

Now one must remember that X , Y and F used in [equations \(9.12\)](#) and [\(9.16\)](#) are alias variables for the variables for the specified table type used during interpolation. For example, for the EOS_Pt_DT table type the $X \equiv D$, the $Y \equiv T$ and the $F \equiv \text{Pt}$. Of course, other options can change how the

host code's values for X , Y and F are possibly changed. See the next [section 7.4](#) for some additional clarification of effects caused by other options.

7.4 Usage of EOS_X_CONVERT, EOS_Y_CONVERT, EOS_XY_MODIFY, EOS_XY_PASSTHRU and EOS_USE_HOST_XY

[Figure 9.12](#) displays various interpolation results when using different combinations of the current options of interest. These options are of interest due to their potential to create confusion for you, dear reader. To provide some clarification, a simple series of tests were created to interpolate an arbitrary SESAME table with common input and the various combinations of options defined in the following table:

Test	Defined Options and Values	Figure Reference
A	none	figure 9.12-A
B	EOS_XY_MODIFY=true	figure 9.12-B
C	EOS_XY_PASSTHRU=true	figure 9.12-C
D	EOS_X_CONVERT=2.0	figure 9.12-D
E	EOS_X_CONVERT=2.0, EOS_XY_MODIFY=1 (true)	figure 9.12-E
F	EOS_X_CONVERT=2.0, EOS_XY_PASSTHRU=1 (true)	figure 9.12-F
G	EOS_Y_CONVERT=2.0	figure 9.12-G
H	EOS_Y_CONVERT=2.0, EOS_XY_MODIFY=1 (true)	figure 9.12-H
I	EOS_Y_CONVERT=2.0, EOS_XY_PASSTHRU=1 (true)	figure 9.12-I

The default behavior of the [eos.Interpolate](#) function is to create internal copies of the supplied inputs arrays (xVals and yVals). Looking at the displayed results in [figure 9.12](#)'s A, B, C, F and I, one observes that the interpolation results (curves) and effects upon the input xVals and yVals arrays's data (abscissas and legends) are identical. Further inspection recognizes that the results shown in [figure 9.12](#)'s D and E have identical interpolated results (curves) that are different from the rest; however, the [figure 9.12-E](#)'s abscissal values stored in the host code's xVals data are changed by the usage of the EOS_X_CONVERT option with a value of 2.0 since the EOS_XY_MODIFY option was enabled. Similarly, the results shown in [figure 9.12](#)'s G and H have identical interpolated results (curves) that are different from the rest, and the legend values are changed by the usage of the EOS_Y_CONVERT option with a value of 2.0.

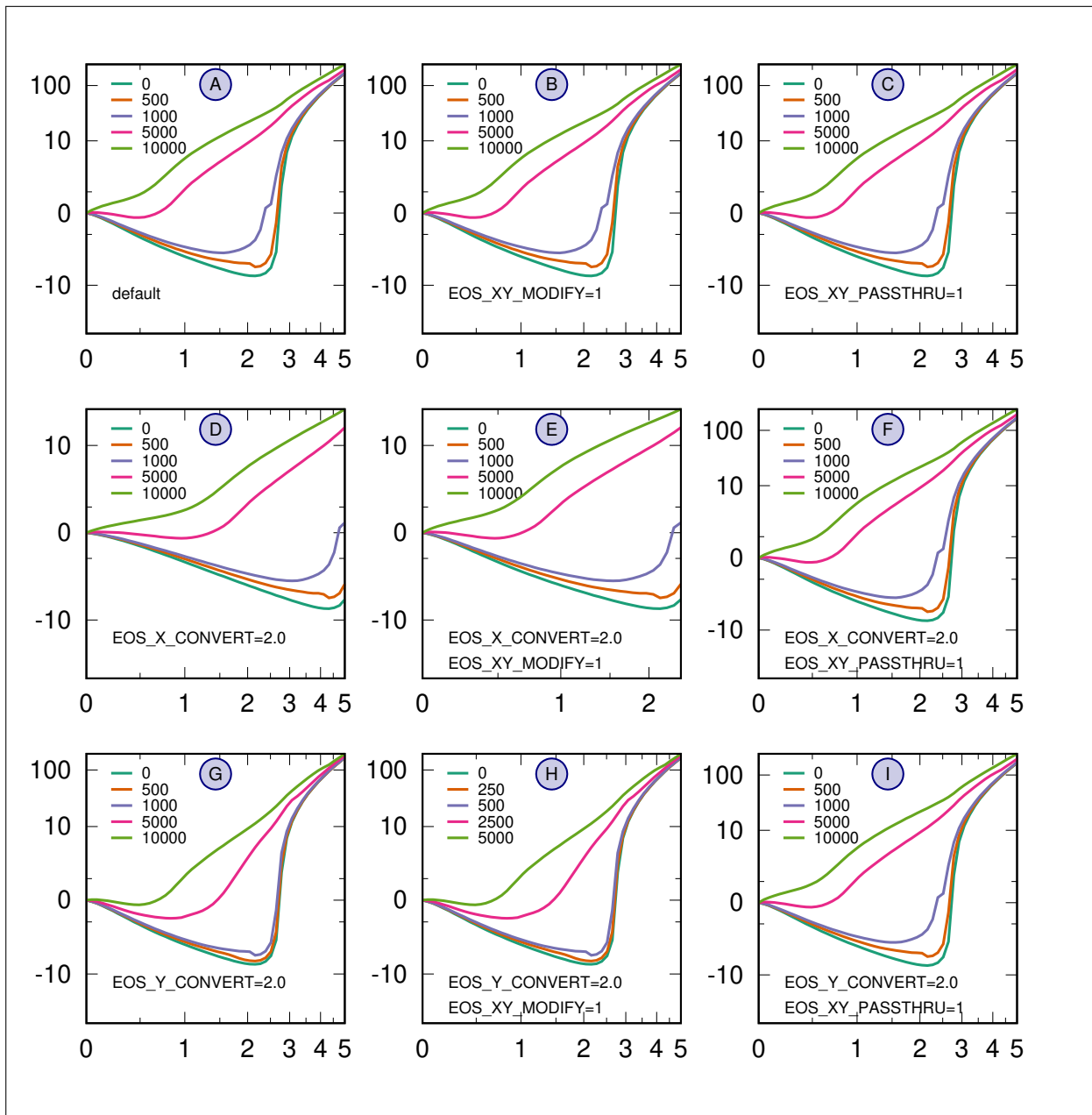


Figure 9.12: Comparison of interpolated results using various combinations of the `EOS_X_CONVERT`, `EOS_Y_CONVERT`, `EOS_XY_MODIFY`, `EOS_XY_PASSTHRU` options and a convenient conversion factor, 2.0.

“Now, what about that other pesky option, `EOS_USE_HOST_XY`? Is it redundant, or does it correspond to some other implementation subtlety?” you ask. The previously-described tests were performed while using the `EOS_USE_HOST_XY` option in lieu of the `EOS_XY_PASSTHRU`

option, and the results are shown in [figure 9.13](#). At first glance, the results appear to confirm that the `EOS_USE_HOST_XY` option is an alias for the `EOS_XY_MODIFY` option. This conclusion is obvious when one compares the [figure 9.13](#)'s E and F. Unfortunately, not all is well with that conclusion when one compares the legends of the [figure 9.13](#)'s H and I.

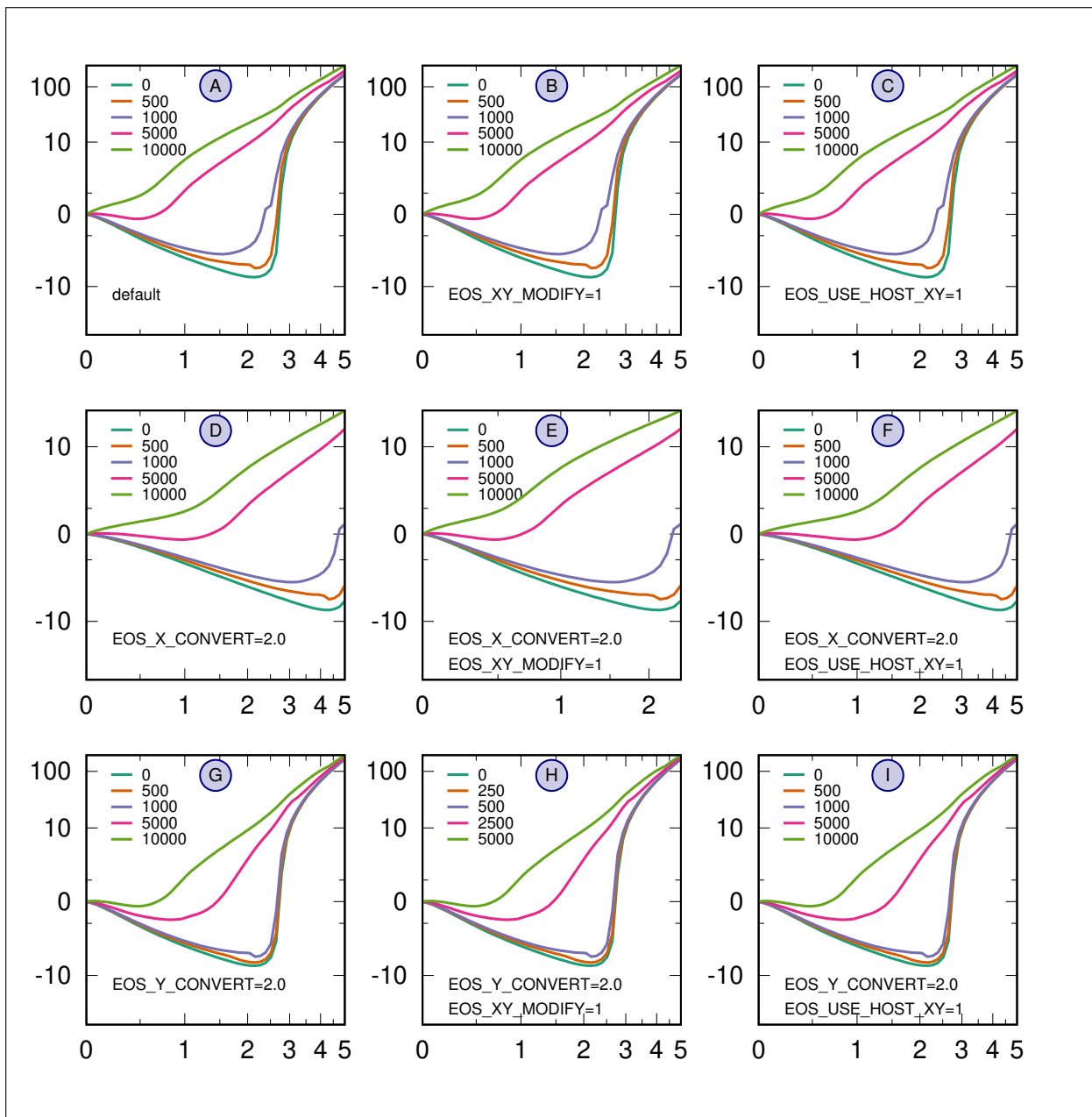


Figure 9.13: Comparison of interpolated results using various combinations of the `EOS_X_CONVERT`, `EOS_Y_CONVERT`, `EOS_XY_MODIFY`, `EOS_USE_HOST_XY` options and a convenient conversion factor, 2.0.

Due to this incongruity, the `EOS_USE_HOST_XY` option is likely destined for the dustbin of obsolescence sometime in the future ... possibly ... maybe Who knows? Perhaps an imaginative individual will decide an alternative purpose for the `EOS_USE_HOST_XY` option. Until then, this author recommends that it should not be used; however, if dear reader determines it has values as implemented, then please provide the appropriate feedback using the contact information in [chapter 11](#).

10 USAGE EXAMPLES

Reading computer manuals without the hardware is as frustrating as reading sex manuals without the software.

– Arthur C. Clarke

It's time to use the software on the available hardware. This section contains various examples showing the usage of the interface routines defined in [chapters 5 to 7](#).

1 C HOST CODE EXAMPLE

```
1 /*****
2  * Example Program
3  * -----
4  * Filetype: (SOURCE)
5  *
6  * Copyright -- see file named COPYRIGHTNOTICE
7  *
8  *****/
9
10 /*! \file
11  * \ingroup examples
12  * \brief This is a simple C example of how to use EOSPAC6 interface.
13  */
14
15 #include <stdio.h>
16 #include <stdlib.h>
```

```
17 #include "eos_Interface.h"
18
19 int main ()
20 {
21     enum
22     { nTablesE = 5 };
23     enum
24     { nXYPairsE = 4 };
25     enum
26     { nInfoItemsE = 12 };
27
28     EOS_INTEGER i, j;
29     EOS_REAL X[nXYPairsE], Y[nXYPairsE], F[nXYPairsE], dFx[nXYPairsE],
30         dFy[nXYPairsE];
31     EOS_INTEGER tableType[nTablesE], numIndVars[nTablesE];
32     EOS_INTEGER matID[nTablesE];
33     EOS_INTEGER tableHandle[nTablesE];
34     EOS_INTEGER errorCode;
35     EOS_INTEGER tableHandleErrorCode;
36     EOS_INTEGER nTables;
37     EOS_INTEGER nXYPairs;
38     EOS_REAL infoVals[nInfoItemsE];
39     EOS_INTEGER nInfoItems;
40     EOS_INTEGER infoItems[nInfoItemsE] = {
41         EOS_Cmnt_Len,
42         EOS_Exchange_Coeff,
43         EOS_F_Convert_Factor,
44         EOS_Log_Val,
45         EOS_Material_ID,
46         EOS_Mean_Atomic_Mass,
47         EOS_Mean_Atomic_Num,
48         EOS_Modulus,
49         EOS_Normal_Density,
50         EOS_Table_Type,
51         EOS_X_Convert_Factor,
52         EOS_Y_Convert_Factor
53     };
54     EOS_CHAR *infoItemDescriptions[nInfoItemsE] = {
```

```
55     "The length in characters of the comments available for the specified data table",
56     "The exchange coefficient",
57     "The conversion factor corresponding to the dependent variable, F(x,y)",
58     "Non-zero if the data table is in a log10 format",
59     "The SESAME material identification number",
60     "The mean atomic mass",
61     "The mean atomic number",
62     "The solid bulk modulus",
63     "The normal density",
64     "The type of data table. Corresponds to the parameters in APPENDIX B and APPENDIX C",
65     "The conversion factor corresponding to the primary independent variable, x",
66     "The conversion factor corresponding to the secondary independent variable, y"
67 };
68 EOS_CHAR *tableTypeLabel[nTablesE] = {
69     "EOS_Pt_DT",
70     "EOS_Dv_T",
71     "EOS_Ogb",
72     "EOS_Comment",
73     "EOS_Info"
74 };
75 EOS_CHAR errorMessage[EOS_MaxErrMsgLen];
76
77 EOS_INTEGER one = 1;
78
79 nTables = nTablesE;
80 nXYPairs = nXYPairsE;
81 nInfoItems = nInfoItemsE;
82
83 /*
84  * EOS_Pt_DT, material 2140 works for Sesame table 301 (record type 1)
85  * EOS_Dv_T, material 2140 works for Sesame table 401 (record type 2)
86  * EOS_Ogb, material 12140 works for Sesame table 501 (record type 3)
87  * EOS_Comment, material 2140 works for Sesame tables 101-199 (record type 4)
88  * EOS_Info, material 2140 works for Sesame table 201 (record type 5)
89  */
90 tableType[0] = EOS_Pt_DT;
91 tableType[1] = EOS_Dv_T;
92 tableType[2] = EOS_Ogb;
```

```
93  tableType[3] = EOS_Comment;
94  tableType[4] = EOS_Info;
95
96  numIndVars[0] = 2;
97  numIndVars[1] = 1;
98  numIndVars[2] = 0;
99  numIndVars[3] = 0;
100 numIndVars[4] = 0;
101
102 matID[0] = 2140;
103 matID[1] = 2140;
104 matID[2] = 12140;
105 matID[3] = 2140;
106 matID[4] = 2140;
107
108 errorCode = EOS_OK;
109 for (i = 0; i < nTables; i++) {
110     tableHandle[i] = 0;
111 }
112
113 /*
114  * initialize table data objects
115  */
116
117 eos_CreateTables (&nTables, tableType, matID, tableHandle, &errorCode);
118 if (errorCode != EOS_OK) {
119     for (i = 0; i < nTables; i++) {
120         tableHandleErrorCode = EOS_OK;
121         eos_GetErrorCode (&tableHandle[i], &tableHandleErrorCode);
122         eos_GetErrorMessage (&tableHandleErrorCode, errorMessage);
123         printf ("eos_CreateTables ERROR %i: %s\n", tableHandleErrorCode,
124               errorMessage);
125     }
126 }
127
128 /*
129  * set some options
130  */
```

```
131
132 for (i = 0; i < nTables; i++) {
133     /* enable smoothing */
134     eos_SetOption (&tableHandle[i], &EOS_SMOOTH, EOS_NullPtr, &errorCode);
135     if (errorCode != EOS_OK) {
136         eos_GetErrorMessage (&errorCode, errorMessage);
137         printf ("eos_SetOption ERROR %i: %s\n", errorCode, errorMessage);
138     }
139 }
140
141 /*
142  * load data into table data objects
143  */
144
145 eos_LoadTables (&nTables, tableHandle, &errorCode);
146 if (errorCode != EOS_OK) {
147     eos_GetErrorMessage (&errorCode, errorMessage);
148     printf ("eos_LoadTables ERROR %i: %s\n", errorCode, errorMessage);
149     for (i = 0; i < nTables; i++) {
150         tableHandleErrorCode = EOS_OK;
151         eos_GetErrorCode (&tableHandle[i], &tableHandleErrorCode);
152         eos_GetErrorMessage (&tableHandleErrorCode, errorMessage);
153         printf ("eos_LoadTables ERROR %i (TH=%i): %s\n", tableHandleErrorCode,
154             tableHandle[i], errorMessage);
155     }
156 }
157
158 /*
159  * interpolate -- errors codes are intentionally produced
160  */
161 X[0] = 3000.;
162 X[1] = 6000.;
163 X[2] = 8200.;
164 X[3] = 8300.;
165
166 Y[0] = 20000.0;
167 Y[1] = 620000.0;
168 Y[2] = 4000000.0;
```

```

169 Y[3] = 200000000.0;
170
171 for (i = 0; i < nTables; i++) {
172     printf ("\n--- Interpolate using tableType %s ---\n", tableTypeLabel[i]);
173     eos_Interpolate (&tableHandle[i], &nXYPairs, X, Y, F, dFx, dFy,
174                     &errorCode);
175     printf ("%s Interpolation Results:\n", tableTypeLabel[i]);
176     if (errorCode != EOS_OK) {
177         eos_GetErrorMessage (&errorCode, errorMessage);
178         printf ("eos_Interpolate ERROR %i (TH=%i): %s\n", errorCode,
179               tableHandle[i], errorMessage);
180     }
181     else {
182         for (j = 0; j < nXYPairs; j++) {
183             if (numIndVars[i] == 1)
184                 printf ("\ti=%i\tX = %e, F = %e, dFx = %e, errorCode: %d\n",
185                       j, X[j], F[j], dFx[j], errorCode);
186             if (numIndVars[i] == 2)
187                 printf
188                     ("\ti=%i\tX = %e, Y = %e, F = %e, dFx = %e, dFy = %e, errorCode: %d\n",
189                       j, X[j], Y[j], F[j], dFx[j], dFy[j], errorCode);
190         }
191     }
192 }
193
194 /*
195  * retrieve table info -- errors codes are intentionally produced
196  */
197
198 for (i = 0; i < nTables; i++) {
199     printf ("\n--- Table information for tableType %s , tableHandle=%i ---\n",
200           tableTypeLabel[i], tableHandle[i]);
201     for (j = 0; j < nInfoItems; j++) {
202         EOS_BOOLEAN equal;
203         eos_GetTableInfo (&(tableHandle[i]), &one, &(infoItems[j]),
204                           &(infoVals[j]), &errorCode);
205         eos_ErrorCodesEqual((EOS_INTEGER*)&EOS_INVALID_INFO_FLAG, &errorCode, &equal);
206         if (errorCode == EOS_OK) {

```

```

207     printf ("%2i. %-82s: %13.6f\n", j + 1, infoItemDescriptions[j],
208             infoVals[j]);
209 }
210 else if (! equal) {
211     /* Ignore EOS_INVALID_INFO_FLAG since not all infoItems are currently
212        applicable to a specific tableHandle. */
213     eos_GetErrorMessage (&errorCode, errorMessage);
214     printf ("eos_GetTableInfo ERROR %i: %s\n", errorCode, errorMessage);
215 }
216 }
217 }
218
219 /*
220  * Destroy all data objects
221  */
222
223 eos_DestroyAll (&errorCode);
224 if (errorCode != EOS_OK) {
225     for (i = 0; i < nTables; i++) {
226         tableHandleErrorCode = EOS_OK;
227         eos_GetErrorCode (&tableHandle[i], &tableHandleErrorCode);
228         eos_GetErrorMessage (&tableHandleErrorCode, errorMessage);
229         printf ("eos_DestroyAll ERROR %i: %s\n", tableHandleErrorCode,
230               errorMessage);
231     }
232 }
233
234 return 0;
235
236 }

```

2 C++ HOST CODE EXAMPLE

```

1 /*****
2  * Example Program
3  * -----

```

```
4 * Filetype: (SOURCE)
5 *
6 * Copyright -- see file named COPYRIGHTNOTICE
7 *
8 *****/
9
10 /*! \file
11 * \ingroup examples
12 * \brief This is a simple C++ example of how to use EOSPAC6 interface.
13 */
14
15 #include <iostream>
16 #include <iomanip>
17 #include "eos_Interface.h"
18
19 using namespace std;
20
21 int main ()
22 {
23
24     const EOS_INTEGER nTablesE = 5;
25     const EOS_INTEGER nXYPairsE = 4;
26     const EOS_INTEGER nInfoItemsE = 12;
27
28     EOS_INTEGER i, j;
29     EOS_REAL X[nXYPairsE], Y[nXYPairsE], F[nXYPairsE], dFx[nXYPairsE],
30         dFy[nXYPairsE];
31     EOS_INTEGER tableType[nTablesE], numIndVars[nTablesE];
32     EOS_INTEGER matID[nTablesE];
33     EOS_INTEGER tableHandle[nTablesE];
34     EOS_INTEGER errorCode;
35     EOS_INTEGER tableHandleErrorCode;
36     EOS_INTEGER nTables;
37     EOS_INTEGER nXYPairs;
38     EOS_REAL infoVals[nInfoItemsE];
39     EOS_INTEGER nInfoItems;
40     EOS_INTEGER infoItems[nInfoItemsE] = {
41         EOS_Cmnt_Len,
```

```
42     EOS_Exchange_Coeff,
43     EOS_F_Convert_Factor,
44     EOS_Log_Val,
45     EOS_Material_ID,
46     EOS_Mean_Atomic_Mass,
47     EOS_Mean_Atomic_Num,
48     EOS_Modulus,
49     EOS_Normal_Density,
50     EOS_Table_Type,
51     EOS_X_Convert_Factor,
52     EOS_Y_Convert_Factor
53 };
54 const EOS_CHAR *infoItemDescriptions[nInfoItemsE] = {
55     "The length in characters of the comments available for the specified data table",
56     "The exchange coefficient",
57     "The conversion factor corresponding to the dependent variable, F(x,y)",
58     "Non-zero if the data table is in a log10 format",
59     "The SESAME material identification number",
60     "The mean atomic mass",
61     "The mean atomic number",
62     "The solid bulk modulus",
63     "The normal density",
64     "The type of data table. Corresponds to the parameters in APPENDIX B and APPENDIX C",
65     "The conversion factor corresponding to the primary independent variable, x",
66     "The conversion factor corresponding to the secondary independent variable, y"
67 };
68 const EOS_CHAR *tableTypeLabel[nTablesE] = {
69     "EOS_Pt_DT",
70     "EOS_Dv_T",
71     "EOS_Ogb",
72     "EOS_Comment",
73     "EOS_Info"
74 };
75 EOS_CHAR errorMessage[EOS_MaxErrMsgLen];
76
77 EOS_INTEGER one = 1;
78
79 nTables = nTablesE;
```

```
80  nXYPairs = nXYPairsE;
81  nInfoItems = nInfoItemsE;
82
83  /*
84   * EOS_Pt_DT, material 2140 works for Sesame table 301 (record type 1)
85   * EOS_Dv_T, material 2140 works for Sesame table 401 (record type 2)
86   * EOS_Ogb, material 12140 works for Sesame table 501 (record type 3)
87   * EOS_Comment, material 2140 works for Sesame tables 101-199 (record type 4)
88   * EOS_Info, material 2140 works for Sesame table 201 (record type 5)
89   */
90  tableType[0] = EOS_Pt_DT;
91  tableType[1] = EOS_Dv_T;
92  tableType[2] = EOS_Ogb;
93  tableType[3] = EOS_Comment;
94  tableType[4] = EOS_Info;
95
96  numIndVars[0] = 2;
97  numIndVars[1] = 1;
98  numIndVars[2] = 0;
99  numIndVars[3] = 0;
100 numIndVars[4] = 0;
101
102 matID[0] = 2140;
103 matID[1] = 2140;
104 matID[2] = 12140;
105 matID[3] = 2140;
106 matID[4] = 2140;
107
108 errorCode = EOS_OK;
109 for (i = 0; i < nTables; i++) {
110     tableHandle[i] = 0;
111 }
112
113 /*
114  * initialize table data objects
115  */
116
117 eos_CreateTables (&nTables, tableType, matID, tableHandle, &errorCode);
```

```
118  if (errorCode != EOS_OK) {
119      for (i = 0; i < nTables; i++) {
120          tableHandleErrorCode = EOS_OK;
121          eos_GetErrorCode (&tableHandle[i], &tableHandleErrorCode);
122          eos_GetErrorMessage (&tableHandleErrorCode, errorMessage);
123          cout << "eos_CreateTables ERROR " << tableHandleErrorCode
124              << ": " << errorMessage << '\n';
125      }
126  }
127
128  /*
129   * set some options
130   */
131
132  for (i = 0; i < nTables; i++) {
133      /* enable smoothing */
134      eos_SetOption (&tableHandle[i], &EOS_SMOOTH, EOS_NullPtr, &errorCode);
135      if (errorCode != EOS_OK) {
136          eos_GetErrorMessage (&errorCode, errorMessage);
137          cout << "eos_SetOption ERROR " << errorCode << ": " << errorMessage << '\n';
138      }
139  }
140
141  /*
142   * load data into table data objects
143   */
144
145  eos_LoadTables (&nTables, tableHandle, &errorCode);
146  if (errorCode != EOS_OK) {
147      eos_GetErrorMessage (&errorCode, errorMessage);
148      cout << "eos_LoadTables ERROR " << errorCode << ": " << errorMessage << '\n';
149      for (i = 0; i < nTables; i++) {
150          tableHandleErrorCode = EOS_OK;
151          eos_GetErrorCode (&tableHandle[i], &tableHandleErrorCode);
152          eos_GetErrorMessage (&tableHandleErrorCode, errorMessage);
153          cout << "eos_LoadTables ERROR " << tableHandleErrorCode << "(TH="
154              << tableHandle[i] << "): " << errorMessage << '\n';
155      }

```

```
156 }
157
158 /*
159  * interpolate -- errors codes are intentionally produced
160  */
161 X[0] = 3000.;
162 X[1] = 6000.;
163 X[2] = 8200.;
164 X[3] = 8300.;
165
166 Y[0] = 20000.0;
167 Y[1] = 620000.0;
168 Y[2] = 4000000.0;
169 Y[3] = 200000000.0;
170
171 for (i = 0; i < nTables; i++) {
172     cout << "\n--- Interpolate using tableType " << tableTypeLabel[i] << " ---\n";
173     eos_Interpolate (&tableHandle[i], &nXYPairs, X, Y, F, dFx, dFy,
174                     &errorCode);
175     cout << tableTypeLabel[i] << " Interpolation Results:\n";
176     if (errorCode != EOS_OK) {
177         eos_GetErrorMessage (&errorCode, errorMessage);
178         cout << "eos_Interpolate ERROR " << errorCode << "(TH="
179             << tableHandle[i] << "): " << errorMessage << '\n';
180     }
181     else {
182         for (j = 0; j < nXYPairs; j++) {
183             if (numIndVars[i] == 1)
184                 cout << "\ti=" << j
185                     << "\tX = " << scientific << X[j]
186                     << ", F = " << scientific << F[j]
187                     << ", dFx = " << scientific << dFx[j]
188                     << ", errorCode: " << errorCode << '\n';
189             if (numIndVars[i] == 2)
190                 cout << "\ti=" << j
191                     << "\tX = " << scientific << X[j]
192                     << ", Y = " << scientific << Y[j]
193                     << ", F = " << scientific << F[j]
```

```

194         << " , dFx = " << scientific << dFx[j]
195         << " , dFy = " << scientific << dFy[j]
196         << " , errorCode: " << errorCode << '\n';
197     }
198 }
199 }
200
201 /*
202  * retrieve table info -- errors codes are intentionally produced
203  */
204
205 for (i = 0; i < nTables; i++) {
206     cout << "\n--- Table information for tableType " << tableTypeLabel[i]
207         << " , tableHandle=" << tableHandle[i]
208         << " ---\n";
209     for (j = 0; j < nInfoItems; j++) {
210         EOS_BOOLEAN equal;
211         eos_GetTableInfo (&(tableHandle[i]), &one, &(infoItems[j]),
212                         &(infoVals[j]), &errorCode);
213         eos_ErrorCodesEqual((EOS_INTEGER*)&EOS_INVALID_INFO_FLAG, &errorCode, &equal);
214         if (errorCode == EOS_OK) {
215             cout.setf(ios::fixed,ios::floatfield);
216             cout << setprecision(2) << setiosflags(ios::fixed)
217                 << setw(2) << right << j + 1 << ". "
218                 << setw(82) << left << infoItemDescriptions[j] << ": "
219                 << setprecision(6) << setiosflags(ios::fixed)
220                 << setw(13) << right << infoVals[j] << '\n';
221         }
222         else if (! equal) {
223             /* Ignore EOS_INVALID_INFO_FLAG since not all infoItems are currently
224              applicable to a specific tableHandle. */
225             eos_GetErrorMessage (&errorCode, errorMessage);
226             cout << "eos_GetTableInfo ERROR " << errorCode
227                 << ": " << errorMessage << '\n';
228         }
229     }
230 }
231

```

```

232  /*
233  * Destroy all data objects
234  */
235
236  eos_DestroyAll (&errorCode);
237  if (errorCode != EOS_OK) {
238      for (i = 0; i < nTables; i++) {
239          tableHandleErrorCode = EOS_OK;
240          eos_GetErrorCode (&tableHandle[i], &tableHandleErrorCode);
241          eos_GetErrorMessage (&tableHandleErrorCode, errorMessage);
242          cout << "eos_DestroyAll ERROR " << tableHandleErrorCode
243              << ": " << errorMessage << '\n';
244      }
245  }
246
247  return 0;
248
249 }

```

3 FORTRAN 77 HOST CODE EXAMPLE

```

1 C*****
2 c  Example F77 Program
3 c  -----
4 c  Filetype: (SOURCE)
5 c
6 c  Copyright -- see file named COPYRIGHTNOTICE
7 c
8 C*****
9
10 c> \file
11 c> \ingroup examples
12 c> \brief This is a simple F77 example of how to use EOSPAC6 interface.
13
14     program TestF77
15

```

```
16      implicit none
17
18      include 'eos_Interface.fi'
19
20      integer*4 nTables, nXYPairs, nInfoItems
21      parameter (nTables = 5)
22      parameter (nXYPairs = 4)
23      parameter (nInfoItems = 12)
24
25      integer*4 i, j
26      real*8 X(nXYPairs), Y(nXYPairs), F(nXYPairs), dFx(nXYPairs),
27      &      dFy(nXYPairs)
28      integer*4 tableType(nTables), numIndVars(nTables)
29      integer*4 matID(nTables)
30      integer*4 tableHandle(nTables)
31      integer*4 errorCode
32      integer*4 tableHandleErrorCode
33      real*8 infoVals(nInfoItems)
34      integer*4 infoItems(nInfoItems)
35      character*82 infoItemDescriptions(nInfoItems)
36      character*20 tableTypeLabel(nTables)
37      character*(EOS_MaxErrMsgLen) errorMessage
38      integer k
39
40      data infoItems /
41      &      EOS_Cmnt_Len,
42      &      EOS_Exchange_Coeff,
43      &      EOS_F_Convert_Factor,
44      &      EOS_Log_Val,
45      &      EOS_Material_ID,
46      &      EOS_Mean_Atomic_Mass,
47      &      EOS_Mean_Atomic_Num,
48      &      EOS_Modulus,
49      &      EOS_Normal_Density,
50      &      EOS_Table_Type,
51      &      EOS_X_Convert_Factor,
52      &      EOS_Y_Convert_Factor
53      &      /
```

```

54     data infoItemDescriptions /
55     &'The length in characters of the comments available for the specif
56     &ied data table',
57     &'The exchange coefficient',
58     &'The conversion factor corresponding to the dependent variable, F(
59     &x,y)',
60     &'Non-zero if the data table is in a log10 format',
61     &'The SESAME material identification number',
62     &'The mean atomic mass',
63     &'The mean atomic number',
64     &'The solid bulk modulus',
65     &'The normal density',
66     &'The type of data table. Corresponds to the parameters in APPENDIX
67     & B and APPENDIX C',
68     &'The conversion factor corresponding to the primary independent va
69     &riable, x',
70     &'The conversion factor corresponding to the secondary independent
71     &variable, y'
72     &/
73     data tableTypeLabel /
74     &     'EOS_Pt_DT',
75     &     'EOS_Dv_T',
76     &     'EOS_Ogb',
77     &     'EOS_Comment',
78     &     'EOS_Info'
79     &     /
80
81     logical equal
82
83 c     EOS_Pt_DT, material 2140 works for Sesame table 301 (record type 1)
84 c     EOS_Dv_T, material 2140 works for Sesame table 401 (record type 2)
85 c     EOS_Ogb, material 12140 works for Sesame table 501 (record type 3)
86 c     EOS_Comment, material 2140 works for Sesame tables 101-199 (record type 4)
87 c     EOS_Info, material 2140 works for Sesame table 201 (record type 5)
88     tableType(1) = EOS_Pt_DT
89     tableType(2) = EOS_Dv_T
90     tableType(3) = EOS_Ogb
91     tableType(4) = EOS_Comment

```

```
92     tableType(5) = EOS_Info
93
94     numIndVars(1) = 2
95     numIndVars(2) = 1
96     numIndVars(3) = 0
97     numIndVars(4) = 0
98     numIndVars(5) = 0
99
100    matID(1) = 2140
101    matID(2) = 2140
102    matID(3) = 12140
103    matID(4) = 2140
104    matID(5) = 2140
105
106    errorCode = EOS_OK
107    do 10 i=1, nTables
108        tableHandle(i) = 0
109 10  continue
110
111 C
112 C     initialize table data objects
113 C
114    call eos_CreateTables ( nTables, tableType, matID,
115 &                          tableHandle, errorCode)
116    if (errorCode.NE.EOS_OK) then
117        do 15 i=1, nTables
118            tableHandleErrorCode = EOS_OK
119            call eos_GetErrorCode
120 &                ( tableHandle(i), tableHandleErrorCode )
121            call eos_GetErrorMessage
122 &                ( tableHandleErrorCode, errorMessage )
123            call strLength(errorMessage, EOS_MaxErrMsgLen, k)
124            write(*,998) 'eos_CreateTables ERROR ',tableHandleErrorCode,
125 &                ': ',errorMessage(1:k)
126 15  continue
127    endif
128
129 C
```

```

130 C      set some options
131 C
132      do 20 i=1, nTables
133 C          enable smoothing
134          call eos_SetOption ( tableHandle(i), EOS_SMOOTH,
135 &                          EOS_NullVal, errorCode )
136          if (errorCode.NE.EOS_OK) then
137              call eos_GetErrorMessage ( errorCode, errorMessage )
138              call strLength(errorMessage, EOS_MaxErrMsgLen, k)
139              write(*,998) 'eos_SetOption ERROR ', errorCode,
140 &                          ': ', errorMessage(1:k)
141          endif
142 20      continue
143
144 C
145 C      load data into table data objects
146 C
147      call eos_LoadTables ( nTables, tableHandle, errorCode)
148      if (errorCode.NE.EOS_OK) then
149          call eos_GetErrorMessage ( errorCode, errorMessage )
150          call strLength(errorMessage, EOS_MaxErrMsgLen, k)
151          write(*,998) 'eos_LoadTables ERROR ', errorCode, ': ',
152 &                      errorMessage(1:k)
153          do 25 i=1, nTables
154              tableHandleErrorCode = EOS_OK
155              call eos_GetErrorCode
156 &                  ( tableHandle(i), tableHandleErrorCode )
157              call eos_GetErrorMessage
158 &                  ( tableHandleErrorCode, errorMessage )
159              call strLength(errorMessage, EOS_MaxErrMsgLen, k)
160              write(*,994) 'eos_LoadTables ERROR ', tableHandleErrorCode,
161 &                          ' (TH=', tableHandle(i), '): ',
162 &                          errorMessage(1:k)
163 25      continue
164      endif
165
166 C
167 C      interpolate -- errors codes are intentionally produced

```

```
168 C
169     X(1) = 3000.d0
170     X(2) = 6000.d0
171     X(3) = 8200.d0
172     X(4) = 8300.d0
173
174     Y(1) = 20000.0d0
175     Y(2) = 620000.0d0
176     Y(3) = 4000000.0d0
177     Y(4) = 200000000.0d0
178
179     do 30 i=1, nTables
180         write(*,*) ' '
181         write(*,997) '--- Interpolate using tableType ',
182 &             tableTypeLabel(i),' ---'
183         call eos_Interpolate ( tableHandle(i), nXYPairs, X, Y, F,
184 &             dFx, dFy, errorCode)
185         write(*,997) tableTypeLabel(i), ' Interpolation Results:'
186         if (errorCode.NE.EOS_OK) then
187             call eos_GetErrorMessage ( errorCode, errorMessage )
188             call strLength(errorMessage, EOS_MaxErrMsgLen, k)
189             write(*,994) 'eos_Interpolate ERROR ', errorCode,
190 &             ' (TH=', tableHandle(i), '): ',
191 &             errorMessage(1:k)
192         else
193             do 40 j=1, nXYPairs
194                 if (numIndVars(i).EQ.1) then
195                     write(*,996) j-1,X(j),F(j),dFx(j),errorCode
196                 endif
197                 if (numIndVars(i).EQ.2) then
198                     write(*,999) j-1,X(j),Y(j),F(j),dFx(j),dFy(j),errorCode
199                 endif
200             40 continue
201         endif
202     30 continue
203
204 C
205 c Retrieve all miscellaneous table info
```

```

206 C
207     do 45 i=1, nTables
208         write(*,*) ' '
209         write(*,997) '--- Table information for tableType ',
210 &         tableTypeLabel(i), ', tableHandle=', tableHandle(i),
211 &         ' ---'
212         do 50 j=1, nInfoItems
213             call eos_GetTableInfo (tableHandle(i), 1,
214 &                 infoItems(j), infoVals(j), errorCode)
215             call eos_ErrorCodesEqual(EOS_INVALID_INFO_FLAG, errorCode,
216 &                 equal)
217             if (errorCode.EQ.EOS_OK) then
218                 write(*,995) j, '. ', infoItemDescriptions(j), ': ',
219 &                 infoVals(j)
220             else if (.NOT.equal) then
221 C                 Ignore EOS_INVALID_INFO_FLAG since not all infoItems are currently
222 C                 applicable to a specific tableHandle.
223                 call eos_GetErrorMessage ( errorCode, errorMessage )
224                 call strLength(errorMessage, EOS_MaxErrMsgLen, k)
225                 write(*,998) 'eos_LoadTables ERROR ', errorCode,
226 &                 ': ', errorMessage(1:k)
227             endif
228     50     continue
229     45     continue
230
231 C
232 C     Destroy all data objects
233 C
234     call eos_DestroyAll (errorCode)
235     if (errorCode.NE.EOS_OK) then
236         do 35 i=1, nTables
237             tableHandleErrorCode = EOS_OK
238             call eos_GetErrorCode (
239 &                 tableHandle(i), tableHandleErrorCode )
240             call eos_GetErrorMessage (
241 &                 tableHandleErrorCode, errorMessage )
242             call strLength(errorMessage, EOS_MaxErrMsgLen, k)
243             write(*,998) 'eos_DestroyAll ERROR ', tableHandleErrorCode,

```

```

244      &          ' : ', errorMessage(1:k)
245 35      continue
246      endif
247
248 994 format (a,i5,a,i1,2a)
249 995 format (i2,a,a,a,f13.6)
250 996 format ('   i=',i2,'   X =',1pe13.6,
251      &       ', F =',1pe13.6,', dFx =',1pe13.6,', errorCode: ',i5)
252 997 format (a,:,a,:,2(a,:,i2))
253 998 format (a,i5,2a)
254 999 format ('   i=',i2,'   X =',1pe13.6,', Y =',1pe13.6,
255      &       ', F =',1pe13.6,', dFx =',1pe13.6,', dFy =',
256      &       1pe13.6,', errorCode: ',i5)
257
258      end
259
260      subroutine strLength(str, length, trimmedLength)
261      integer i, length, trimmedLength
262      character*(*) str
263      trimmedLength = 0
264      do 5 i=length, 1, -1
265          if (trimmedLength.EQ.0 .AND. str(i:i).NE.' ' .AND.
266      &      str(i:i).NE.char(0)) then
267              trimmedLength = i
268          endif
269      5      continue
270      end

```

4 FORTRAN 90 HOST CODE EXAMPLE

```

1 !*****
2 ! Example F90 program
3 ! -----
4 ! Filetype: (HEADER)
5 !
6 ! Copyright -- see file named COPYRIGHTNOTICE

```

```
7 !
8 !*****
9
10 !> @file
11 !! @ingroup examples
12 !! @brief This is a simple F90 example of how to use EOSPAC6 interface.
13
14 program TestF90
15
16   use eos_Interface
17
18   implicit none
19
20   integer(EOS_INTEGER),parameter :: nTables = 5
21   integer(EOS_INTEGER),parameter :: nXYPairs = 4
22   integer(EOS_INTEGER),parameter :: nInfoItems = 12
23
24   integer(EOS_INTEGER) :: i, j
25   real(EOS_REAL) :: X(nXYPairs), Y(nXYPairs), F(nXYPairs), dFx(nXYPairs), dFy(nXYPairs)
26   integer(EOS_INTEGER) :: tableType(nTables), numIndVars(nTables)
27   integer(EOS_INTEGER) :: matID(nTables)
28   integer(EOS_INTEGER) :: tableHandle(nTables)
29   integer(EOS_INTEGER) :: errorCode
30   integer(EOS_INTEGER) :: tableHandleErrorCode
31   real(EOS_REAL) :: infoVals(nInfoItems)
32   integer(EOS_INTEGER) :: infoItems(nInfoItems) = (/ &
33     EOS_Cmnt_Len, &
34     EOS_Exchange_Coeff, &
35     EOS_F_Convert_Factor, &
36     EOS_Log_Val, &
37     EOS_Material_ID, &
38     EOS_Mean_Atomic_Mass, &
39     EOS_Mean_Atomic_Num, &
40     EOS_Modulus, &
41     EOS_Normal_Density, &
42     EOS_Table_Type, &
43     EOS_X_Convert_Factor, &
44     EOS_Y_Convert_Factor &
```

```
45      /)
46  character(82) :: infoItemDescriptions(nInfoItems) = (/ &
47      'The length in characters of the comments available for the specified data table ', &
48      'The exchange coefficient ', &
49      'The conversion factor corresponding to the dependent variable, F(x,y) ', &
50      'Non-zero if the data table is in a log10 format ', &
51      'The SESAME material identification number ', &
52      'The mean atomic mass ', &
53      'The mean atomic number ', &
54      'The solid bulk modulus ', &
55      'The normal density ', &
56      'The type of data table. Corresponds to the parameters in APPENDIX B and APPENDIX C', &
57      'The conversion factor corresponding to the primary independent variable, x ', &
58      'The conversion factor corresponding to the secondary independent variable, y ' &
59      /)
60  character(11) :: tableTypeLabel(nTables) = (/ &
61      'EOS_Pt_DT ', &
62      'EOS_Dv_T ', &
63      'EOS_Ogb ', &
64      'EOS_Comment', &
65      'EOS_Info ' &
66      /)
67  character(EOS_MaxErrMsgLen) :: errorMessage
68
69  logical equal
70
71  ! EOS_Pt_DT, material 2140 works for Sesame table 301 (record type 1)
72  ! EOS_Dv_T, material 2140 works for Sesame table 401 (record type 2)
73  ! EOS_Ogb, material 12140 works for Sesame table 501 (record type 3)
74  ! EOS_Comment, material 2140 works for Sesame tables 101-199 (record type 4)
75  ! EOS_Info, material 2140 works for Sesame table 201 (record type 5)
76  tableType(1) = EOS_Pt_DT
77  tableType(2) = EOS_Dv_T
78  tableType(3) = EOS_Ogb
79  tableType(4) = EOS_Comment
80  tableType(5) = EOS_Info
81
82  numIndVars(1) = 2
```

```
83  numIndVars(2) = 1
84  numIndVars(3) = 0
85  numIndVars(4) = 0
86  numIndVars(5) = 0
87
88  matID(1) = 2140
89  matID(2) = 2140
90  matID(3) = 12140
91  matID(4) = 2140
92  matID(5) = 2140
93
94  errorCode = EOS_OK
95  do i=1, nTables
96      tableHandle(i) = 0
97  enddo
98
99  !
100 !    initialize table data objects
101 !
102 call eos_CreateTables ( nTables, tableType, matID, tableHandle, errorCode)
103 if (errorCode.NE.EOS_OK) then
104     do i=1, nTables
105         tableHandleErrorCode = EOS_OK
106         call eos_GetErrorCode ( tableHandle(i), tableHandleErrorCode )
107         call eos_GetErrorMessage ( tableHandleErrorCode, errorMessage )
108         write(*,998) 'eos_CreateTables ERROR ', tableHandleErrorCode, ': ', &
109                 errorMessage(1:(len_trim(errorMessage)-1))
110     enddo
111 endif
112
113 !
114 !    set some options
115 !
116 do i=1, nTables
117     !    enable smoothing
118     call eos_SetOption ( tableHandle(i), EOS_SMOOTH, EOS_NullVal, errorCode )
119     if (errorCode.NE.EOS_OK) then
120         call eos_GetErrorMessage ( errorCode, errorMessage )
```

```
121     write(*,998) 'eos_SetOption ERROR ', errorCode, ': ', &
122               errorMessage(1:(len_trim(errorMessage)-1))
123   endif
124 enddo
125
126 !
127 !   load data into table data objects
128 !
129 call eos_LoadTables ( nTables, tableHandle, errorCode)
130 if (errorCode.NE.EOS_OK) then
131   call eos_GetErrorMessage ( errorCode, errorMessage )
132   write(*,998) 'eos_LoadTables ERROR ', errorCode, ': ', &
133             errorMessage(1:(len_trim(errorMessage)-1))
134   do i=1, nTables
135     tableHandleErrorCode = EOS_OK
136     call eos_GetErrorCode ( tableHandle(i), tableHandleErrorCode )
137     call eos_GetErrorMessage ( tableHandleErrorCode, errorMessage )
138     write(*,994) 'eos_LoadTables ERROR ', tableHandleErrorCode, ' (TH=', &
139               tableHandle(i), '): ', &
140               errorMessage(1:(len_trim(errorMessage)-1))
141   enddo
142 endif
143
144 !
145 !   interpolate -- errors codes are intentionally produced
146 !
147 X(1) = 3000._EOS_REAL
148 X(2) = 6000._EOS_REAL
149 X(3) = 8200._EOS_REAL
150 X(4) = 8300._EOS_REAL
151
152 Y(1) = 20000.0_EOS_REAL
153 Y(2) = 620000.0_EOS_REAL
154 Y(3) = 4000000.0_EOS_REAL
155 Y(4) = 200000000.0_EOS_REAL
156
157 do i=1, nTables
158   write(*,*) ' '
```

```

159 write(*,997) '--- Interpolate using tableType ', tableTypeLabel(i),' ---'
160 call eos_Interpolate ( tableHandle(i), nXYPairs, X, Y, F, dFx, dFy, errorCode)
161 write(*,997) tableTypeLabel(i), ' Interpolation Results:'
162 if (errorCode.NE.EOS_OK) then
163     call eos_GetErrorMessage ( errorCode, errorMessage )
164     write(*,994) 'eos_Interpolate ERROR ', errorCode, ' (TH=', &
165         tableHandle(i), '): ', &
166         errorMessage(1:(len_trim(errorMessage)-1))
167 else
168     do j=1, nXYPairs
169         if (numIndVars(i).EQ.1) then
170             write(*,996) j-1,X(j),F(j),dFx(j),errorCode
171         endif
172         if (numIndVars(i).EQ.2) then
173             write(*,999) j-1,X(j),Y(j),F(j),dFx(j),dFy(j),errorCode
174         endif
175     enddo
176 endif
177 enddo
178
179 !
180 !     Retrieve all miscellaneous table info
181 !
182 do i=1, nTables
183     write(*,*) ' '
184     write(*,997) '--- Table information for tableType ', tableTypeLabel(i), &
185         ', tableHandle=', tableHandle(i), ' ---'
186     do j=1, nInfoItems
187         call eos_GetTableInfo ( tableHandle(i), 1_EOS_INTEGER, infoItems(j), &
188             infoVals(j), errorCode )
189         call eos_ErrorCodesEqual(EOS_INVALID_INFO_FLAG, errorCode, equal)
190         if (errorCode.EQ.EOS_OK) then
191             write(*,995) j, '. ', infoItemDescriptions(j), ': ', infoVals(j)
192         else if (.NOT.equal) then
193             ! Ignore EOS_INVALID_INFO_FLAG since not all infoItems are currently
194             ! applicable to a specific tableHandle.
195             call eos_GetErrorMessage ( errorCode, errorMessage )
196             write(*,998) 'eos_GetTableInfo ERROR ', errorCode, ': ', &

```

```

197         errorMessage(1:(len_trim(errorMessage)-1))
198     endif
199 enddo
200 enddo
201
202 !
203 !     Destroy all data objects
204 !
205 call eos_DestroyAll (errorCode)
206 if (errorCode.NE.EOS_OK) then
207     do i=1, nTables
208         tableHandleErrorCode = EOS_OK
209         call eos_GetErrorCode ( tableHandle(i), tableHandleErrorCode )
210         call eos_GetErrorMessage ( tableHandleErrorCode, errorMessage )
211         write(*,998) 'eos_DestroyAll ERROR ', tableHandleErrorCode, ': ', &
212             errorMessage(1:(len_trim(errorMessage)-1))
213     enddo
214 endif
215
216 994 format (a,i5,a,i1,2a)
217 995 format (i2,a,a,a,f13.6)
218 996 format ('    i=',i2,'    X =',1pe13.6, &
219           ', F =',1pe13.6,', dFx =',1pe13.6,', errorCode: ',i5)
220 997 format (a,:,a,:,2(a,:,i2))
221 998 format (a,i5,2a)
222 999 format ('    i=',i2,'    X =',1pe13.6,', Y =',1pe13.6, &
223           ', F =',1pe13.6,', dFx =',1pe13.6,', dFy =', &
224           1pe13.6,', errorCode: ',i5)
225
226 end program TestF90

```

5 SHARED MEMORY EXAMPLE

```

1 /*****
2 * Test Program
3 * -----

```

```

4  * Filetype: (SOURCE)
5  *
6  * Copyright -- see file named COPYRIGHTNOTICE
7  *
8  *****/
9  /*! \file
10 * \ingroup C tests eos_GetPackedTables eos_SetSharedPackedTables shared
11 * \brief Perform the following tests:
12 *     -# Ensure tables can be created, loaded, packed, unpacked and destroyed.
13 *     -# Then test interpolation using unpacked data.
14 *     -# Ensure the following functions work as expected:
15 *         eos_GetSharedPackedTablesSize, eos_GetPackedTables, eos_SetSharedPackedTables
16 *         See Gitlab issue eos/eospac-dev/eospac6#771 for details.
17 *
18 * \note
19 * MATIDS TO TEST: 3720
20 */
21
22 #include <stdio.h>
23 #include <stdlib.h>
24 #include "eos_Interface.h"
25
26 int main ()
27 {
28     int i, j;
29     EOS_INTEGER one = 1, packedTablesSize, sharedTablesSize;
30     EOS_INTEGER tableType[5] = { EOS_Uc_D, EOS_Pt_DUt, EOS_T_DUt, EOS_Pc_D, EOS_Ut_DPt };
31     EOS_CHAR *tableTypeStr[5] =
32         { "EOS_Uc_D", "EOS_Pt_DUt", "EOS_T_DUt", "EOS_Pc_D", "EOS_Ut_DPt" };
33     EOS_INTEGER matID[5] = { 3720, 3720, 3720, 3720, 3720 };
34     EOS_INTEGER tableHandle[5] = { -2, -2, -2, -2, -2 };
35     EOS_INTEGER errorCode, tableHandleErrorCode;
36     EOS_CHAR errorMessage[EOS_MaxErrMsgLen], *packedData = NULL, *sharedData = NULL;
37     EOS_INTEGER nTables = sizeof(tableType)/sizeof(tableType[0]);
38
39     errorCode = EOS_OK;
40     printf ("TH\n--\n");
41     for (i = 0; i < nTables; i++)

```

```
42 {
43     tableHandle[i] = 0;
44     printf ("%2i. creating table %s (%i)\n", i, tableTypeStr[i], tableType[i]);
45     eos_CreateTables (&one, &tableType[i], &matID[i], &tableHandle[i], &errorCode);
46     if (errorCode != EOS_OK)
47     {
48         eos_GetErrorMessage (&errorCode, errorMessage);
49         printf ("    eos_CreateTables ERROR %i: %s\n", errorCode, errorMessage);
50         return -1;
51     }
52 }
53
54 /* Load data */
55 printf ("\nTH\n--\n");
56 for (i = 0; i < nTables; i++)
57 {
58     printf ("%2i. loading table %s (%i)\n", tableHandle[i], tableTypeStr[i], tableType[i]);
59     eos_LoadTables (&one, &tableHandle[i], &errorCode);
60     if (errorCode != EOS_OK)
61     {
62         eos_GetErrorMessage (&errorCode, errorMessage);
63         printf ("    eos_LoadTables ERROR %i: %s\n", errorCode, errorMessage);
64         tableHandleErrorCode = EOS_OK;
65         eos_GetErrorCode (&tableHandle[i], &tableHandleErrorCode);
66         if (tableHandleErrorCode != EOS_OK)
67         {
68             eos_GetErrorMessage (&tableHandleErrorCode, errorMessage);
69             printf ("    eos_LoadTables ERROR %i: %s\n",
70                 tableHandleErrorCode, errorMessage);
71         }
72     }
73 }
74
75 /* packed all tables, delete all tables and then set them again with packed data */
76 printf ("\n*** call Get/SetSharedPackedTables AFTER calling eos_LoadTables()\n");
77 eos_GetSharedPackedTablesSize (&nTables, tableHandle, &packedTablesSize,
78                                 &sharedTablesSize, &errorCode);
79 printf ("fcmp_ignore: packedTablesSize = %d\n", packedTablesSize);
```

```

80  printf ("fcmp_ignore: sharedTablesSize = %d\n", sharedTablesSize);
81  packedData = (EOS_CHAR *) malloc (packedTablesSize);
82  eos_GetPackedTables (&nTables, tableHandle, packedData, &errorCode);
83  eos_DestroyAll (&errorCode);
84
85  /* The shareData array is used here to emulate an actual shared memory allocation, */
86  /* which is dependent upon the framework and architecture */
87  sharedData = (EOS_CHAR *) malloc (sharedTablesSize);
88  eos_SetSharedPackedTables (&nTables, &packedTablesSize, packedData,
89                          sharedData, EOS_TRUE, tableHandle, &errorCode);
90  free (packedData);
91
92  /* Interpolate some data */
93  {
94      EOS_INTEGER nXYPairs;
95      EOS_INTEGER xyBounds[4] = { 0, 0, 0, 0 };
96      EOS_REAL    xVals[4]     = { 1.0, 2.0, 3.0, 4.0 };
97      EOS_REAL    yVals[4]     = { 100.0, 200.0, 300.0, 400.0 };
98      EOS_REAL    fVals[4], dFx[4], dFy[4];
99
100     nXYPairs = sizeof(xVals)/sizeof(xVals[0]);
101
102     for (i = 0; i < nTables; i++)
103     {
104         if (i == 0 || i == 3)
105             continue;
106
107         /* Initialize output arrays */
108         for (j = 0; j < nXYPairs; j++)
109             fVals[j] = dFx[j] = dFy[j] = 0.0;
110
111         printf ("\n%d. Interpolating %s data for matid %d\n", i, tableTypeStr[i], matID[i]);
112
113         printf ("BEFORE eos_Interpolate\n");
114         for (j = 0; j < nXYPairs; j++)
115             printf ("\tX = %.15e, Y = %.15e, F = %.15e\n", xVals[j], yVals[j], fVals[j]);
116
117         eos_Interpolate (&tableHandle[i], &nXYPairs, xVals, yVals, fVals, dFx, dFy, &errorCode);

```

```
118
119     printf ("AFTER eos_Interpolate\n");
120     if (errorCode != EOS_OK)
121     {
122         EOS_BOOLEAN equal;
123         eos_ErrorCodesEqual ((EOS_INTEGER *) & EOS_INTERP_EXTRAPOLATED, &errorCode, &equal);
124         if (equal)
125         {
126             eos_CheckExtrap (&tableHandle[i], &nXYPairs, xVals, yVals,
127                             xyBounds, &errorCode);
128             if (errorCode != EOS_OK)
129             {
130                 eos_GetErrorMessage (&errorCode, errorMessage);
131                 printf ("eos_CheckExtrap ERROR: %s\n", errorMessage);
132                 return 1;
133             }
134             for (j = 0; j < nXYPairs; j++)
135                 printf
136                 ("\tX = %.15e, Y = %.15e, F = %.15e, extrapCode = %d\n",
137                  xVals[j], yVals[j], fVals[j], xyBounds[j]);
138         }
139         else
140         {
141             eos_GetErrorMessage (&errorCode, errorMessage);
142             printf ("eos_Interpolate ERROR(%d): %s\n", errorCode, errorMessage);
143         }
144     }
145     else
146     {
147         for (j = 0; j < nXYPairs; j++)
148             printf ("\tX = %.15e, Y = %.15e, F = %.15e\n", xVals[j], yVals[j], fVals[j]);
149     }
150 }
151 }
152
153 eos_DestroyAll (&errorCode);
154 free (sharedData);
155
```

```
156 | return 0;  
157 | }
```

11 TECHNICAL SUPPORT INFORMATION

We've stepped in a pile of should.

– Anonymous

Online documentation and references related to EOSPAC are provided at the following URL on both the open and secure networks at Los Alamos National Laboratory:

<https://xweb.lanl.gov/projects/data/eos/>

If you find that you are in need of technical support, bug reports and feature requests for EOSPAC version 6 can be obtained or submitted by contacting the Data Team via the EOSPAC version 6 mailing list, which is available on both the open and secure networks:

eospac-help@lanl.gov

The developer responsible for the EOSPAC code base is listed as follows:

David A. Pimentel
Los Alamos National Laboratory
MS F663
XCP-5, TA-03-1400, Rm. 4116
Los Alamos, New Mexico 87545
davidp@lanl.gov
(505) 665-1255

12 ACKNOWLEDGEMENTS

Knowledge is in the end based on acknowledgement.

– Ludwig Wittgenstein

- Bill Archer, formerly of LANL CCN-12, deserves many thanks for his large contributions to the initial planning and documentation of the EOSPAC 6 development project.
- Olga Chotinun, formerly of LANL HPC-1, was instrumental in the development of the software package.
- Angela Herring, formerly of LANL X-1-NAD, contributed many significant fixes and enhancements, and she deserves many thanks.
- Ben Mastriplito, formerly of LANL XCP-5, contributed many significant fixes and enhancements, and he deserves many thanks.
- Anna Pietarila Graham, LANL HPC-ENV, contributed many significant fixes and enhancements, and she deserves many thanks.
- Daniel Sheppard, LANL XCP-5, has provided advice concerning content of this document and various features in EOSPAC 6 and associated tools.
- Jacob Daniel Tronge, formerly of LANL HPC-ENV, contributed the shared memory enhancements, and he deserves many thanks.
- Ginger Young, formerly of LANL HPC-ENV, has provided advice concerning content of this document and various features in EOSPAC 6 and associated tools.

13 BIBLIOGRAPHY

If you steal from one author it's plagiarism; if you steal from many it's research.

– Wilson Mizner

- [1] Stanford P. Lyon and James D. Johnson. *SESAME: The Los Alamos National Laboratory Equation of State Database*. Technical Report LA-UR-92-3407, Los Alamos National Laboratory (USDOE), 1992.
- [2] Daniel G. Sheppard and David A. Pimentel. *SESAME: The Los Alamos National Laboratory Equation of State Database: extending SESAME for Multi-phase Representations*. Technical Report LA-UR-25-26258, Los Alamos National Laboratory (USDOE), 6 2025.
- [3] Charles W. Cranfill. *The EOSPAC utility package for accessing the Sesame data library*. Memorandum X7-87-U53, March 1987.
- [4] Denise C. George. *OPENSESAME*. Technical Report LA-CC-03-087, Los Alamos National Laboratory (USDOE), 2003.
- [5] Charles W. Cranfill and Richard More. *IONEOS. A Fast, Analytic, Ion Equation-of-State Routine*. Technical Report LA-07313-MS, Los Alamos National Laboratory (USDOE), 1978.
- [6] Daniel Glen Sheppard. Sound speed calculations in eospac6. Technical Report LA-UR-20-30467, Los Alamos National Laboratory (USDOE), 2020. Revision History: 2020-12-23 (Rev.0) ; Repository ID: info:lanl-repo/lareport/LA-UR-20-30467 ; Sponsor: USDOE National Nuclear Security Administration (NNSA) ; Intended For: Report ; Dataset: RASSTI.
- [7] *Information Technology – Portable Operating System Interface (POSIX), Part 1: System Application Program Interface (API) (C language)*, September 1994.

-
- [8] Donald Lewine. *POSIX Programmer's Guide*. O'Reilly & Associates, Inc. Sebastopol, CA, 1994.
- [9] David A Pimentel. *EOSPAC 5 Users Document, Version 5.34, Revision 0*. Technical Report LA-UR-03-4510, Los Alamos National Laboratory (USDOE), 2003.
- [10] David A Pimentel. *EOSPAC 5 Users Document, Version 5.34, Revision 1*. Technical Report LA-UR-03-4510, Los Alamos National Laboratory (USDOE), February 2007. (<https://xweb.lanl.gov/projects/data/eos/eospacReleaseDocs/ES5-UserManual.pdf>).
- [11] Charles W. Cranfill. *MIXPAC - A Subroutine Package for Calculating Equations of State for Equilibrium Mixtures of Materials*. Technical Report LA-09861-M, Los Alamos National Laboratory (USDOE), 1983.
- [12] Charles W. Cranfill. *EOS of a Material Mixture in Pressure Equilibrium*. Technical report, Los Alamos National Laboratory (USDOE), 1999.
- [13] Ginger Ann Young. *Sesame ASCII File Format*. Technical Report LA-UR-19-24891, Los Alamos National Laboratory (USDOE), 2019. Sponsor: USDOE National Nuclear Security Administration (NNSA); Version: 2; Repository ID: perma:lanl/rl-dv-0000357141; Dataset: rassti.
- [14] David A. Pimentel and Daniel G. Sheppard. *SESAME: ASCII2 File Format*. Technical Report LA-UR-25-26256, Los Alamos National Laboratory (USDOE), 6 2025. Sponsor: USDOE National Nuclear Security Administration (NNSA) ; Version: 2 ; Repository ID: perma:lanl/rl-dv-0000491063 ; Dataset: rassti.
- [15] Brian J. Gough. Foreword by Richard M. Stallman. *An Introduction to GCC - for the GNU compilers gcc and g++*. Network Theory Ltd, August 2005. (<http://www.network-theory.co.uk/docs/gccintro/index.html>).
- [16] David A. Pimentel, Daniel Glen Sheppard, Miles Allen Buechler, and Ann Elisabet Mattsson. *(U) Pre-inversion and interpolation: investigating the pros and cons of a performance enhancement "pill" in EOSPAC 6*. Technical Report LA-UR-18-24923, Los Alamos National Laboratory (USDOE), 2018. Extended abstract.
- [17] David A. Pimentel and Daniel Glen Sheppard. *Pre-inverted SESAME data table construction enhancements to correct unexpected inverse interpolation pathologies in EOSPAC 6*. Technical Report LA-UR-18-20858, Los Alamos National Laboratory (USDOE), 2018. Revision History: 2018-02-05 (Rev.0) ; 2018-02-07 (Rev.1).
-

-
- [18] David A. Pimentel and Daniel G. Sheppard. *(U)Table construction enhancements to minimize edge effects during interpolation of pre-inverted SESAME data*. Memorandum XCP-5:17-015(U), February 2017.
- [19] David A. Pimentel and Daniel G. Sheppard. *(U)A recommendation regarding pre-inversion using EOSPAC6*. Memorandum WRS:18-008(U), September 2018.
- [20] Charles W. Cranfill. *EOSPAC - A Subroutine Package for Accessing the Los Alamos SESAME EOS Data Library*. Technical Report LA-09728-M, Los Alamos National Laboratory (USDOE), 1983.
-

14 APPENDIX

**Appendix usually means “small outgrowth from large intestine,” but in this case it means “additional information accompanying main text.” Or are those really the same things? Think carefully before you insult this book.*

– Pseudonymous Bosch, *The Name of This Book Is Secret*

A TABLE TYPES: MNEMONIC CONVENTIONS

Below is an alphabetized list of mnemonics used to create the EOSPAC table type identifier names that are defined in both [APPENDICES B](#) and [C](#). These mnemonics are combined as follows to create the aforementioned identifier names:

EOS_#_@\$

where # is the mnemonic of the dependent function, $F(x,y)$

@ is the mnemonic of the primary independent variable, x

\$ is the mnemonic of the secondary independent variable, y.

Mnemonic	Description
ALPHAt	Total Thermal Expansion Alpha
Ac	Specific-Helmholtz-Free-Energy Cold Curve
Ae	Electron Specific-Helmholtz-Free-Energy
Af	Freeze Specific-Helmholtz-Free-Energy
Aic	Ion Specific-Helmholtz-Free-Energy plus Cold Curve Specific-Helmholtz-Free-Energy
Aiz	Ion Specific-Helmholtz-Free-Energy Including Zero Point
Als	Liquid or Solid Specific-Helmholtz-Free-Energy
Am	Melt Specific-Helmholtz-Free-Energy
At	Total Specific-Helmholtz-Free-Energy
Av	Vapor Specific-Helmholtz-Free-Energy
B	Thermoelectric Coefficient
BSt	Total Isentropic (adiabatic) Bulk Modulus
BTt	Total Isothermal Bulk Modulus
Comment	Descriptive Comments
CPt	Total Specific heat (constant pressure)
CVt	Total Specific heat (constant volume)
D	Density
Dls	Liquid or Solid Density on coexistence line
Dv	Vapor Density on coexistence line
GAMMAAt	Total Gruneisen Coefficient
Gc	Specific-Gibbs-Free-Energy Cold Curve
Ge	Electron Specific-Gibbs-Free-Energy

Continued on next page

Mnemonic	Description
Gic	Ion Specific-Gibbs-Free-Energy plus Cold Curve Specific-Gibbs-Free-Energy
Giz	Ion Specific-Gibbs-Free-Energy Including Zero Point
Gs	Shear Modulus
Gt	Total Specific-Gibbs-Free-Energy
Hc	Specific-Enthalpy Cold Curve
He	Electron Specific-Enthalpy
Hic	Ion Specific-Enthalpy plus Cold Curve Specific-Enthalpy
Hiz	Ion Specific-Enthalpy Including Zero Point
Ht	Total Specific-Enthalpy
Info	Atomic Number, Atomic Mass, Normal Density, Solid Bulk Modulus, Exchange Coefficient
Kc	Electron Conductive Opacity (Conductivity Model)
Kec	Electrical Conductivity
Keo	Electron Conductive Opacity (Opacity Model)
Kp	Planck Mean Opacity
Kr	Rosseland Mean Opacity
Ktc	Thermal Conductivity
M	Mass fraction
NullTable	null table
Ogb	Calculated versus Interpolated Opacity Grid Boundary
Pc	Pressure Cold Curve
Pe	Electron Pressure
Pf	Freeze Pressure
Pic	Ion Pressure plus Cold Curve Pressure
Piz	Ion Pressure Including Zero Point
Pm	Melt Pressure
Pt	Total Pressure
Pv	Vapor Pressure
Se	Electron Specific-Entropy
Sic	Ion Specific-Entropy plus Cold Curve Specific-Entropy
Siz	Ion Specific-Entropy Including Zero Specific-Entropy
St	Total Specific-Entropy
T	Temperature

Continued on next page

Mnemonic	Description
Tf	Freeze Temperature
Tm	Melt Temperature
Uc	Specific-Internal-Energy Cold Curve
Ue	Electron Specific-Internal-Energy
Uf	Freeze Specific-Internal-Energy
Uic	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy
Uiz	Ion Specific-Internal-Energy Including Zero Point
Uls	Liquid or Solid Specific-Internal-Energy
Um	Melt Specific-Internal-Energy
Ut	Total Specific-Internal-Energy
Uv	Vapor Specific-Internal-Energy
V	Specific Volume
Zfc	Mean Ion Charge (Conductivity Model)
Zfo	Mean Ion Charge (Opacity Model)

B TABLE TYPES: *GROUPED BY CATEGORY, SORTED BY NAME*

Below is a list of defined constants corresponding to the 292 data table types available within EOSPAC. These defined constants have been grouped into several data categories, alphabetized according to the defined constant names, and cross-referenced to the applicable EOSPAC 5[9],[10] defined constants. The constant names have been created using the mnemonics defined in [APPENDIX A](#). The data categories are listed below and referenced to pages within this appendix.

It is important to note that the actual values of these constants may change without notice; therefore, use the constant names – do not hardwire the values into the host code. The EOSPAC 6 Constants are color coded as follows:

- **I** indicates the table is inverted with respect to the first independent variable.
- **II** indicates the table is inverted with respect to the second independent variable.
- **M** indicates the table is a combination of two other tables.
- **GPU** indicates the table can be offloaded to an available GPU and used for interpolation within the appropriate GPU kernel(s). There are currently 195 data table types that are compatible with the GPU interpolation kernel(s). The EOS_INVERT_AT_SETUP option must be enabled for the inverted data table types if they are to be interpolated on the GPU.

B.1 Category 1: Unrelated to SESAME data

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_NullTable	null table	n/a

B.2 Category 2: General information found in SESAME's 100- and 200-series tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Comment	Descriptive Comments	101-199
EOS_Info	Atomic Number, Atomic Mass, Normal Density, Solid Bulk Modulus, Exchange Coefficient	201

B.3 Category 3: Total EOS in SESAME's 301 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_ALPHAt_DT GPU	Total Thermal Expansion Alpha ($1/K$) (Density (Mg/m^3)- and Temperature (K)-dependent)	301
EOS_ALPHAt_DUt M GPU	Total Thermal Expansion Alpha ($1/K$) (Density (Mg/m^3)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_At_DGt M GPU	Total Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Total Specific-Gibbs-Free-Energy (MJ/kg)-dependent)	301
EOS_At_DHt M GPU	Total Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Total Specific-Enthalpy (MJ/kg)-dependent)	301
EOS_At_DPt M GPU	Total Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Total Pressure (GPa)-dependent)	301
EOS_At_DSt M GPU	Total Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Total Specific-Entropy ($MJ/kg/K$)-dependent)	301
EOS_At_DT GPU	Total Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	301
EOS_At_DUt M GPU	Total Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_BSt_DT GPU	Total Isentropic (adiabatic) Bulk Modulus (GPa) (Density (Mg/m^3)- and Temperature (K)-dependent)	301
EOS_BSt_DUt M GPU	Total Isentropic (adiabatic) Bulk Modulus (GPa) (Density (Mg/m^3)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_BTt_DT GPU	Total Isothermal Bulk Modulus (GPa) (Density (Mg/m^3)- and Temperature (K)-dependent)	301

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_BTt_DUt M GPU	Total Isothermal Bulk Modulus (GPa) (Density (Mg/m^3)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_CPt_DT GPU	Total Specific heat (constant pressure) ($MJ/kg/K$) (Density (Mg/m^3)- and Temperature (K)-dependent)	301
EOS_CPt_DUt M GPU	Total Specific heat (constant pressure) ($MJ/kg/K$) (Density (Mg/m^3)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_CVt_DT GPU	Total Specific heat (constant volume) ($MJ/kg/K$) (Density (Mg/m^3)- and Temperature (K)-dependent)	301
EOS_CVt_DUt M GPU	Total Specific heat (constant volume) ($MJ/kg/K$) (Density (Mg/m^3)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_D_PtT I	Density (Mg/m^3) (Total Pressure (GPa)- and Temperature (K)-dependent)	301
EOS_GAMMAat_DT GPU	Total Gruneisen Coefficient (dimensionless) (Density (Mg/m^3)- and Temperature (K)-dependent)	301
EOS_GAMMAat_DUt M GPU	Total Gruneisen Coefficient (dimensionless) (Density (Mg/m^3)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_Gt_DAt M GPU	Total Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Total Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	301
EOS_Gt_DPt M GPU	Total Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Total Pressure (GPa)-dependent)	301
EOS_Gt_DSt M GPU	Total Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Total Specific-Entropy ($MJ/kg/K$)-dependent)	301
EOS_Gt_DT GPU	Total Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	301











Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Gt_DUt M GPU	Total Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_Ht_DAt M GPU	Total Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Total Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	301
EOS_Ht_DPt M GPU	Total Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Total Pressure (GPa)-dependent)	301
EOS_Ht_DSt M GPU	Total Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Total Specific-Entropy ($MJ/kg/K$)-dependent)	301
EOS_Ht_DT GPU	Total Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	301
EOS_Ht_DUt M GPU	Total Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_Pt_DAt M GPU	Total Pressure (GPa) (Density (Mg/m^3)- and Total Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	301
EOS_Pt_DGt M GPU	Total Pressure (GPa) (Density (Mg/m^3)- and Total Specific-Gibbs-Free-Energy (MJ/kg)-dependent)	301
EOS_Pt_DHt M GPU	Total Pressure (GPa) (Density (Mg/m^3)- and Total Specific-Enthalpy (MJ/kg)-dependent)	301
EOS_Pt_DSt M GPU	Total Pressure (GPa) (Density (Mg/m^3)- and Total Specific-Entropy ($MJ/kg/K$)-dependent)	301
EOS_Pt_DT GPU	Total Pressure (GPa) (Density (Mg/m^3)- and Temperature (K)-dependent)	301

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Pt_DUt M GPU	Total Pressure (<i>GPa</i>) (Density (<i>Mg/m³</i>)- and Total Specific-Internal-Energy (<i>MJ/kg</i>)-dependent)	301
EOS_St_DAt M GPU	Total Specific-Entropy (<i>MJ/kg/K</i>) (Density (<i>Mg/m³</i>)- and Total Specific-Helmholtz-Free-Energy (<i>MJ/kg</i>)-dependent)	301
EOS_St_DGt M GPU	Total Specific-Entropy (<i>MJ/kg/K</i>) (Density (<i>Mg/m³</i>)- and Total Specific-Gibbs-Free-Energy (<i>MJ/kg</i>)-dependent)	301
EOS_St_DHt M GPU	Total Specific-Entropy (<i>MJ/kg/K</i>) (Density (<i>Mg/m³</i>)- and Total Specific-Enthalpy (<i>MJ/kg</i>)-dependent)	301
EOS_St_DPt M GPU	Total Specific-Entropy (<i>MJ/kg/K</i>) (Density (<i>Mg/m³</i>)- and Total Pressure (<i>GPa</i>)-dependent)	301
EOS_St_DT GPU	Total Specific-Entropy (<i>MJ/kg/K</i>) (Density (<i>Mg/m³</i>)- and Temperature (<i>K</i>)-dependent)	301
EOS_St_DUt M GPU	Total Specific-Entropy (<i>MJ/kg/K</i>) (Density (<i>Mg/m³</i>)- and Total Specific-Internal-Energy (<i>MJ/kg</i>)-dependent)	301
EOS_T_DAt II GPU	Temperature (<i>K</i>) (Density (<i>Mg/m³</i>)- and Total Specific-Helmholtz-Free-Energy (<i>MJ/kg</i>)-dependent)	301
EOS_T_DGt II GPU	Temperature (<i>K</i>) (Density (<i>Mg/m³</i>)- and Total Specific-Gibbs-Free-Energy (<i>MJ/kg</i>)-dependent)	301
EOS_T_DHt II GPU	Temperature (<i>K</i>) (Density (<i>Mg/m³</i>)- and Total Specific-Enthalpy (<i>MJ/kg</i>)-dependent)	301
EOS_T_DPt II GPU	Temperature (<i>K</i>) (Density (<i>Mg/m³</i>)- and Total Pressure (<i>GPa</i>)-dependent)	301

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_T_DSt 	Temperature (K) (Density (Mg/m^3)- and Total Specific-Entropy ($MJ/kg/K$)-dependent)	301
EOS_T_DUt 	Temperature (K) (Density (Mg/m^3)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_Ut_DAt 	Total Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Total Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	301
EOS_Ut_DGt 	Total Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Total Specific-Gibbs-Free-Energy (MJ/kg)-dependent)	301
EOS_Ut_DHt 	Total Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Total Specific-Enthalpy (MJ/kg)-dependent)	301
EOS_Ut_DPt 	Total Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Total Pressure (GPa)-dependent)	301
EOS_Ut_DSt 	Total Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Total Specific-Entropy ($MJ/kg/K$)-dependent)	301
EOS_Ut_DT 	Total Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	301
EOS_Ut_PtT 	Total Specific-Internal-Energy (MJ/kg) (Total Pressure (GPa)- and Temperature (K)-dependent)	301
EOS_V_PtT 	Specific-Volume (m^3/Mg) (Total Pressure (GPa)- and Temperature (K)-dependent)	301

B.4 Category 4: Ion+Cold EOS in SESAME's 303 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Aic_DGic M GPU	Ion Specific-Helmholtz-Free-Energy plus Cold Curve Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Gibbs-Free-Energy plus Cold Curve Specific-Gibbs-Free-Energy (MJ/kg)-dependent)	303
EOS_Aic_DHic M GPU	Ion Specific-Helmholtz-Free-Energy plus Cold Curve Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Enthalpy plus Cold Curve Specific-Enthalpy (MJ/kg)-dependent)	303
EOS_Aic_DPic M GPU	Ion Specific-Helmholtz-Free-Energy plus Cold Curve Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Pressure plus Cold Curve Pressure (GPa)-dependent)	303
EOS_Aic_DSic M GPU	Ion Specific-Helmholtz-Free-Energy plus Cold Curve Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Pressure plus Cold Curve Specific-Entropy ($MJ/kg/K$)-dependent)	303
EOS_Aic_DT GPU	Ion Specific-Helmholtz-Free-Energy plus Cold Curve Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	303
EOS_Aic_DUic M GPU	Ion Specific-Helmholtz-Free-Energy plus Cold Curve Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg)-dependent)	303
EOS_Gic_Daic M GPU	Ion Specific-Gibbs-Free-Energy plus Cold Curve Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Helmholtz-Free-Energy plus Cold Curve Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	303

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Gic_DPic M GPU	Ion Specific-Gibbs-Free-Energy plus Cold Curve Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Pressure plus Cold Curve Pressure (GPa)-dependent)	303
EOS_Gic_DSic M GPU	Ion Specific-Gibbs-Free-Energy plus Cold Curve Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Entropy plus Cold Curve Specific-Entropy ($MJ/kg/K$)-dependent)	303
EOS_Gic_DT GPU	Ion Specific-Gibbs-Free-Energy plus Cold Curve Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	303
EOS_Gic_DUic M GPU	Ion Specific-Gibbs-Free-Energy plus Cold Curve Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg)-dependent)	303
EOS_Hic_DAic M GPU	Ion Specific-Enthalpy plus Cold Curve Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Helmholtz-Free-Energy plus Cold Curve Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	303
EOS_Hic_DPic M GPU	Ion Specific-Enthalpy plus Cold Curve Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Ion Pressure plus Cold Curve Pressure (GPa)-dependent)	303
EOS_Hic_DSic M GPU	Ion Specific-Enthalpy plus Cold Curve Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Entropy plus Cold Curve Specific-Entropy ($MJ/kg/K$)-dependent)	303
EOS_Hic_DT GPU	Ion Specific-Enthalpy plus Cold Curve Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	303

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Hic_DUic M GPU	Ion Specific-Enthalpy plus Cold Curve Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg)-dependent)	303
EOS_Pic_DAic M GPU	Ion Pressure plus Cold Curve Pressure (GPa) (Density (Mg/m^3)- and Ion Specific-Helmholtz-Free-Energy plus Cold Curve Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	303
EOS_Pic_DGic M GPU	Ion Pressure plus Cold Curve Pressure (GPa) (Density (Mg/m^3)- and Ion Specific-Gibbs-Free-Energy plus Cold Curve Specific-Gibbs-Free-Energy (MJ/kg)-dependent)	303
EOS_Pic_DHic M GPU	Ion Pressure plus Cold Curve Pressure (GPa) (Density (Mg/m^3)- and Ion Specific-Enthalpy plus Cold Curve Specific-Enthalpy (MJ/kg)-dependent)	303
EOS_Pic_DSic M GPU	Ion Pressure plus Cold Curve Pressure (GPa) (Density (Mg/m^3)- and Ion Pressure plus Cold Curve Specific-Entropy ($MJ/kg/K$)-dependent)	303
EOS_Pic_DT GPU	Ion Pressure plus Cold Curve Pressure (GPa) (Density (Mg/m^3)- and Temperature (K)-dependent)	303
EOS_Pic_DUic M GPU	Ion Pressure plus Cold Curve Pressure (GPa) (Density (Mg/m^3)- and Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg)-dependent)	303
EOS_Sic_DAic M GPU	Ion Specific-Entropy plus Cold Curve Specific-Entropy ($MJ/kg/K$) (Density (Mg/m^3)- and Ion Specific-Helmholtz-Free-Energy plus Cold Curve Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	303
EOS_Sic_DGic M GPU	Ion Specific-Entropy ($MJ/kg/K$) (Density (Mg/m^3)- and Ion Specific-Gibbs-Free-Energy (MJ/kg)-dependent)	303

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Sic_DHic M GPU	Ion Specific-Entropy ($MJ/kg/K$) (Density (Mg/m^3)- and Ion Specific-Enthalpy (MJ/kg)-dependent)	303
EOS_Sic_DPic M GPU	Ion Specific-Entropy plus Cold Curve Specific-Entropy ($MJ/kg/K$) (Density (Mg/m^3)- and Ion Pressure plus Cold Curve Pressure (GPa)-dependent)	303
EOS_Sic_DT GPU	Ion Specific-Entropy plus Cold Curve Specific-Entropy ($MJ/kg/K$) (Density (Mg/m^3)- and Temperature (K)-dependent)	303
EOS_Sic_DUic M GPU	Ion Specific-Entropy plus Cold Curve Specific-Entropy ($MJ/kg/K$) (Density (Mg/m^3)- and Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg)-dependent)	303
EOS_T_DAic II GPU	Temperature (K) (Density (Mg/m^3)- and Ion Specific-Helmholtz-Free-Energy plus Cold Curve Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	303
EOS_T_DGic II GPU	Temperature (K) (Density (Mg/m^3)- and Ion Specific-Gibbs-Free-Energy plus Cold Curve Specific-Gibbs-Free-Energy (MJ/kg)-dependent)	303
EOS_T_DHic II GPU	Temperature (K) (Density (Mg/m^3)- and Ion Specific-Enthalpy plus Cold Curve Specific-Enthalpy (MJ/kg)-dependent)	303
EOS_T_DPic II GPU	Temperature (K) (Density (Mg/m^3)- and Ion Pressure plus Cold Curve Pressure (GPa)-dependent)	303
EOS_T_DSic II GPU	Temperature (K) (Density (Mg/m^3)- and Ion Pressure plus Cold Curve Specific-Entropy ($MJ/kg/K$)-dependent)	303

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_T_DUic I GPU	Temperature (K) (Density (Mg/m^3)- and Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg)-dependent)	303
EOS_Uic_DAic M GPU	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Helmholtz-Free-Energy plus Cold Curve Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	303
EOS_Uic_DGic M GPU	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Gibbs-Free-Energy plus Cold Curve Specific-Gibbs-Free-Energy (MJ/kg)-dependent)	303
EOS_Uic_DHic M GPU	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Enthalpy plus Cold Curve Specific-Enthalpy (MJ/kg)-dependent)	303
EOS_Uic_DPic M GPU	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Pressure plus Cold Curve Pressure (GPa)-dependent)	303
EOS_Uic_DSic M GPU	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Pressure plus Cold Curve Specific-Entropy ($MJ/kg/K$)-dependent)	303
EOS_Uic_DT GPU	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	303



B.5 Category 5: Electron EOS in SESAME's 304 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Ae.DGe M GPU	Electron Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Specific-Gibbs-Free-Energy (MJ/kg)-dependent)	304
EOS_Ae.DHe M GPU	Electron Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Specific-Enthalpy (MJ/kg)-dependent)	304
EOS_Ae.DPe M GPU	Electron Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Pressure (GPa)-dependent)	304
EOS_Ae.DSe M GPU	Electron Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Specific-Entropy ($MJ/kg/K$)-dependent)	304
EOS_Ae.DT GPU	Electron Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	304
EOS_Ae.DUe M GPU	Electron Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Specific-Internal-Energy (MJ/kg)-dependent)	304
EOS_Ge.DAe M GPU	Electron Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	304
EOS_Ge.DPe M GPU	Electron Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Pressure (GPa)-dependent)	304
EOS_Ge.DSe M GPU	Electron Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Specific-Entropy ($MJ/kg/K$)-dependent)	304
EOS_Ge.DT GPU	Electron Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	304

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Ge_DUe M GPU	Electron Specific-Gibbs-Free-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Specific-Internal-Energy (MJ/kg)-dependent)	304
EOS_He_DAe M GPU	Electron Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Electron Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	304
EOS_He_DPe M GPU	Electron Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Electron Pressure (GPa)-dependent)	304
EOS_He_DSe M GPU	Electron Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Electron Specific-Entropy ($MJ/kg/K$)-dependent)	304
EOS_He_DT GPU	Electron Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	304
EOS_He_DUe M GPU	Electron Specific-Enthalpy (MJ/kg) (Density (Mg/m^3)- and Electron Specific-Internal-Energy (MJ/kg)-dependent)	304
EOS_Pe_DAe M GPU	Electron Pressure (GPa) (Density (Mg/m^3)- and Electron Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	304
EOS_Pe_DGe M GPU	Electron Pressure (GPa) (Density (Mg/m^3)- and Electron Specific-Gibbs-Free-Energy (MJ/kg)-dependent)	304
EOS_Pe_DHe M GPU	Electron Pressure (GPa) (Density (Mg/m^3)- and Electron Specific-Enthalpy (MJ/kg)-dependent)	304
EOS_Pe_DSe M GPU	Electron Pressure (GPa) (Density (Mg/m^3)- and Electron Specific-Entropy ($MJ/kg/K$)-dependent)	304
EOS_Pe_DT GPU	Electron Pressure (GPa) (Density (Mg/m^3)- and Temperature (K)-dependent)	304

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Pe_DUe 	Electron Pressure (<i>GPa</i>) (Density (<i>Mg/m³</i>)- and Electron Specific-Internal-Energy (<i>MJ/kg</i>)-dependent)	304
EOS_Se_DAe 	Electron Specific-Entropy (<i>MJ/kg/K</i>) (Density (<i>Mg/m³</i>)- and Electron Specific-Helmholtz-Free-Energy (<i>MJ/kg</i>)-dependent)	304
EOS_Se_DGe 	Electron Specific-Entropy (<i>MJ/kg/K</i>) (Density (<i>Mg/m³</i>)- and Electron Specific-Gibbs-Free-Energy (<i>MJ/kg</i>)-dependent)	304
EOS_Se_DHe 	Electron Specific-Entropy (<i>MJ/kg/K</i>) (Density (<i>Mg/m³</i>)- and Electron Specific-Enthalpy (<i>MJ/kg</i>)-dependent)	304
EOS_Se_DPe 	Electron Specific-Entropy (<i>MJ/kg/K</i>) (Density (<i>Mg/m³</i>)- and Electron Pressure (<i>GPa</i>)-dependent)	304
EOS_Se_DT 	Electron Specific-Entropy (<i>MJ/kg/K</i>) (Density (<i>Mg/m³</i>)- and Temperature (<i>K</i>)-dependent)	304
EOS_Se_DUe 	Electron Specific-Entropy (<i>MJ/kg/K</i>) (Density (<i>Mg/m³</i>)- and Electron Specific-Internal-Energy (<i>MJ/kg</i>)-dependent)	304
EOS_T_DAe 	Temperature (<i>K</i>) (Density (<i>Mg/m³</i>)- and Electron Specific-Helmholtz-Free-Energy (<i>MJ/kg</i>)-dependent)	304
EOS_T_DGe 	Temperature (<i>K</i>) (Density (<i>Mg/m³</i>)- and Electron Specific-Gibbs-Free-Energy (<i>MJ/kg</i>)-dependent)	304
EOS_T_DHe 	Temperature (<i>K</i>) (Density (<i>Mg/m³</i>)- and Electron Specific-Enthalpy (<i>MJ/kg</i>)-dependent)	304
EOS_T_DPe 	Temperature (<i>K</i>) (Density (<i>Mg/m³</i>)- and Electron Pressure (<i>GPa</i>)-dependent)	304

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_T_DSe II GPU	Temperature (K) (Density (Mg/m^3)- and Electron Specific-Entropy ($MJ/kg/K$)-dependent)	304
EOS_T_DUe II GPU	Temperature (K) (Density (Mg/m^3)- and Electron Specific-Internal-Energy (MJ/kg)-dependent)	304
EOS_Ue_D Ae M GPU	Electron Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	304
EOS_Ue_D Ge M GPU	Electron Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Specific-Gibbs-Free-Energy (MJ/kg)-dependent)	304
EOS_Ue_D He M GPU	Electron Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Specific-Enthalpy (MJ/kg)-dependent)	304
EOS_Ue_D Pe M GPU	Electron Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Pressure (GPa)-dependent)	304
EOS_Ue_D Se M GPU	Electron Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Specific-Entropy ($MJ/kg/K$)-dependent)	304
EOS_Ue_DT GPU	Electron Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	304

B.6 Category 6: Ion EOS in SESAME's 305 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Aiz_DGiz M GPU	Ion Specific-Helmholtz-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Gibbs-Free-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_Aiz_DHiz M GPU	Ion Specific-Helmholtz-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Enthalpy Including Zero Point (MJ/kg)-dependent)	305
EOS_Aiz_DPiz M GPU	Ion Specific-Helmholtz-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Pressure Including Zero Point (GPa)-dependent)	305
EOS_Aiz_DSiz M GPU	Ion Specific-Helmholtz-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Pressure Including Zero Specific-Entropy ($MJ/kg/K$)-dependent)	305
EOS_Aiz_DT GPU	Ion Specific-Helmholtz-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	305
EOS_Aiz_DUiz M GPU	Ion Specific-Helmholtz-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Internal-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_Giz_DAiz M GPU	Ion Specific-Gibbs-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Helmholtz-Free-Energy Including Zero Point (MJ/kg)-dependent)	305

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Giz_DPiz M GPU	Ion Specific-Gibbs-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Pressure Including Zero Point (GPa)-dependent)	305
EOS_Giz_DSiz M GPU	Ion Specific-Gibbs-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Entropy Including Zero Point ($MJ/kg/K$)-dependent)	305
EOS_Giz_DT GPU	Ion Specific-Gibbs-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	305
EOS_Giz_DUiz M GPU	Ion Specific-Gibbs-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Internal-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_Hiz_DAiz M GPU	Ion Specific-Enthalpy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Helmholtz-Free-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_Hiz_DPiz M GPU	Ion Specific-Enthalpy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Pressure Including Zero Point (GPa)-dependent)	305
EOS_Hiz_DSiz M GPU	Ion Specific-Enthalpy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Entropy Including Zero Point ($MJ/kg/K$)-dependent)	305
EOS_Hiz_DT GPU	Ion Specific-Enthalpy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	305
EOS_Hiz_DUiz M GPU	Ion Specific-Enthalpy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Internal-Energy Including Zero Point (MJ/kg)-dependent)	305

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Piz_DAIz M GPU	Ion Pressure Including Zero Point (<i>GPa</i>) (Density (Mg/m^3)- and Ion Specific-Helmholtz-Free-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_Piz_DGIz M GPU	Ion Pressure Including Zero Point (<i>GPa</i>) (Density (Mg/m^3)- and Ion Specific-Gibbs-Free-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_Piz_DHIz M GPU	Ion Pressure Including Zero Point (<i>GPa</i>) (Density (Mg/m^3)- and Ion Specific-Enthalpy Including Zero Point (MJ/kg)-dependent)	305
EOS_Piz_DSiz M GPU	Ion Pressure Including Zero Point (<i>GPa</i>) (Density (Mg/m^3)- and Ion Pressure Including Zero Specific-Entropy ($MJ/kg/K$)-dependent)	305
EOS_Piz_DT GPU	Ion Pressure Including Zero Point (<i>GPa</i>) (Density (Mg/m^3)- and Temperature (<i>K</i>)-dependent)	305
EOS_Piz_DUIz M GPU	Ion Pressure Including Zero Point (<i>GPa</i>) (Density (Mg/m^3)- and Ion Specific-Internal-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_Siz_DAIz M GPU	Ion Specific-Entropy Including Zero Specific-Entropy ($MJ/kg/K$) (Density (Mg/m^3)- and Ion Specific-Helmholtz-Free-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_Siz_DGIz M GPU	Ion Specific-Entropy ($MJ/kg/K$) (Density (Mg/m^3)- and Ion Specific-Gibbs-Free-Energy (MJ/kg)-dependent)	305
EOS_Siz_DHIz M GPU	Ion Specific-Entropy ($MJ/kg/K$) (Density (Mg/m^3)- and Ion Specific-Enthalpy (MJ/kg)-dependent)	305

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Siz_DPiz M GPU	Ion Specific-Entropy Including Zero Specific-Entropy ($MJ/kg/K$) (Density (Mg/m^3)- and Ion Pressure Including Zero Point (GPa)-dependent)	305
EOS_Siz_DT GPU	Ion Specific-Entropy Including Zero Specific-Entropy ($MJ/kg/K$) (Density (Mg/m^3)- and Temperature (K)-dependent)	305
EOS_Siz_DUiz M GPU	Ion Specific-Entropy Including Zero Specific-Entropy ($MJ/kg/K$) (Density (Mg/m^3)- and Ion Specific-Internal-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_T_DAiz II GPU	Temperature (K) (Density (Mg/m^3)- and Ion Specific-Helmholtz-Free-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_T_DGiz II GPU	Temperature (K) (Density (Mg/m^3)- and Ion Specific-Gibbs-Free-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_T_DHiz II GPU	Temperature (K) (Density (Mg/m^3)- and Ion Specific-Enthalpy Including Zero Point (MJ/kg)-dependent)	305
EOS_T_DPiz II GPU	Temperature (K) (Density (Mg/m^3)- and Ion Pressure Including Zero Point (GPa)-dependent)	305
EOS_T_DSiz II GPU	Temperature (K) (Density (Mg/m^3)- and Ion Pressure Including Zero Specific-Entropy ($MJ/kg/K$)-dependent)	305
EOS_T_DUiz II GPU	Temperature (K) (Density (Mg/m^3)- and Ion Specific-Internal-Energy Including Zero Point (MJ/kg)-dependent)	305

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Uiz_DAIz M GPU	Ion Specific-Internal-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Helmholtz-Free-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_Uiz_DGiz M GPU	Ion Specific-Internal-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Gibbs-Free-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_Uiz_DHiz M GPU	Ion Specific-Internal-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Specific-Enthalpy Including Zero Point (MJ/kg)-dependent)	305
EOS_Uiz_DPiz M GPU	Ion Specific-Internal-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Pressure Including Zero Point (GPa)-dependent)	305
EOS_Uiz_DSiz M GPU	Ion Specific-Internal-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Ion Pressure Including Zero Specific-Entropy ($MJ/kg/K$)-dependent)	305
EOS_Uiz_DT GPU	Ion Specific-Internal-Energy Including Zero Point (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	305

B.7 Category 7: Cold curve EOS in SESAME's 306 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Ac.D GPU	Specific-Helmholtz-Free-Energy Cold Curve (MJ/kg) (Density (Mg/m^3)-dependent)	306
EOS_Gc.D GPU	Specific-Gibbs-Free-Energy Cold Curve (MJ/kg) (Density (Mg/m^3)-dependent)	306
EOS_Hc.D GPU	Specific-Enthalpy Cold Curve (MJ/kg) (Density (Mg/m^3)-dependent)	306
EOS_Pc.D GPU	Pressure Cold Curve (GPa) (Density (Mg/m^3)-dependent)	306
EOS_Uc.D GPU	Specific-Internal-Energy Cold Curve (MJ/kg) (Density (Mg/m^3)-dependent)	306

B.8 Category 8: Mass fraction EOS in SESAME's 321 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_M_DT	Mass Fraction (Density(Mg/m^3)- and Temperature (K)-dependent)	321

B.9 Category 9: Vaporization data in SESAME's 401 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Als_Av M	Liquid or Solid Specific-Helmholtz-Free-Energy (MJ/kg) (Vapor Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	401
EOS_Als_Dls M	Liquid or Solid Specific-Helmholtz-Free-Energy (MJ/kg) (Liquid or Solid Density on coexistence line (Mg/m^3)-dependent)	401
EOS_Als_Dv M	Liquid or Solid Specific-Helmholtz-Free-Energy (MJ/kg) (Vapor Density on coexistence line (Mg/m^3)-dependent)	401
EOS_Als_Pv M	Liquid or Solid Specific-Helmholtz-Free-Energy (MJ/kg) (Vapor Pressure (GPa)-dependent)	401
EOS_Als_T	Liquid or Solid Specific-Helmholtz-Free-Energy (MJ/kg) (Temperature (K)-dependent)	401
EOS_Als_Uls M	Liquid or Solid Specific-Helmholtz-Free-Energy (MJ/kg) (Liquid or Solid Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Als_Uv M	Liquid or Solid Specific-Helmholtz-Free-Energy (MJ/kg) (Vapor Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Av_Als M	Vapor Specific-Helmholtz-Free-Energy (MJ/kg) (Liquid or Solid Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	401
EOS_Av_Dls M	Vapor Specific-Helmholtz-Free-Energy (MJ/kg) (Liquid or Solid Density on coexistence line (Mg/m^3)-dependent)	401
EOS_Av_Dv M	Vapor Specific-Helmholtz-Free-Energy (MJ/kg) (Vapor Density on coexistence line (Mg/m^3)-dependent)	401
EOS_Av_Pv M	Vapor Specific-Helmholtz-Free-Energy (MJ/kg) (Vapor Pressure (GPa)-dependent)	401
EOS_Av_T	Vapor Specific-Helmholtz-Free-Energy (MJ/kg) (Temperature (K)-dependent)	401

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Av_Uls M	Vapor Specific-Helmholtz-Free-Energy (MJ/kg) (Liquid or Solid Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Av_Uv M	Vapor Specific-Helmholtz-Free-Energy (MJ/kg) (Vapor Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Dls_Als M	Liquid or Solid Density on coexistence line (Mg/m^3) (Liquid or Solid Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	401
EOS_Dls_Av M	Liquid or Solid Density on coexistence line (Mg/m^3) (Vapor Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	401
EOS_Dls_Dv M	Liquid or Solid Density on coexistence line (Mg/m^3) (Vapor Density on coexistence line (Mg/m^3)-dependent)	401
EOS_Dls_Pv M	Liquid or Solid Density on coexistence line (Mg/m^3) (Vapor Pressure (GPa)-dependent)	401
EOS_Dls_T	Liquid or Solid Density on coexistence line (Mg/m^3) (Temperature (K)-dependent)	401
EOS_Dls_Uls M	Liquid or Solid Density on coexistence line (Mg/m^3) (Liquid or Solid Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Dls_Uv M	Liquid or Solid Density on coexistence line (Mg/m^3) (Vapor Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Dv_Als M	Vapor Density on coexistence line (Mg/m^3) (Liquid or Solid Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	401
EOS_Dv_Av M	Vapor Density on coexistence line (Mg/m^3) (Vapor Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	401
EOS_Dv_Dls M	Vapor Density on coexistence line (Mg/m^3) (Liquid or Solid Density on coexistence line (Mg/m^3)-dependent)	401

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Dv_Pv M	Vapor Density on coexistence line (Mg/m^3) (Vapor Pressure (GPa))-dependent)	401
EOS_Dv_T	Vapor Density on coexistence line (Mg/m^3) (Temperature (K))-dependent)	401
EOS_Dv_Uls M	Vapor Density on coexistence line (Mg/m^3) (Liquid or Solid Specific-Internal-Energy (MJ/kg))-dependent)	401
EOS_Dv_Uv M	Vapor Density on coexistence line (Mg/m^3) (Vapor Specific-Internal-Energy (MJ/kg))-dependent)	401
EOS_Pv_Als M	Vapor Pressure (GPa) (Liquid or Solid Specific-Helmholtz-Free-Energy (MJ/kg))-dependent)	401
EOS_Pv_Av M	Vapor Pressure (GPa) (Vapor Specific-Helmholtz-Free-Energy (MJ/kg))-dependent)	401
EOS_Pv_Dls M	Vapor Pressure (GPa) (Liquid or Solid Density on coexistence line (Mg/m^3))-dependent)	401
EOS_Pv_Dv M	Vapor Pressure (GPa) (Vapor Density on coexistence line (Mg/m^3))-dependent)	401
EOS_Pv_T	Vapor Pressure (GPa) (Temperature (K))-dependent)	401
EOS_Pv_Uls M	Vapor Pressure (GPa) (Liquid or Solid Specific-Internal-Energy (MJ/kg))-dependent)	401
EOS_Pv_Uv M	Vapor Pressure (GPa) (Vapor Specific-Internal-Energy (MJ/kg))-dependent)	401
EOS_T_Als T	Temperature (K) (Liquid or Solid Specific-Helmholtz-Free-Energy (MJ/kg))-dependent)	401

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_T_Av	Temperature (K) I (Vapor Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	401
EOS_T_Dls	Temperature (K) I (Liquid or Solid Density on coexistence line (Mg/m^3)-dependent)	401
EOS_T_Dv	Temperature (K) I (Vapor Density on coexistence line (Mg/m^3)-dependent)	401
EOS_T_Pv	Temperature (K) I (Vapor Pressure (GPa)-dependent)	401
EOS_T_Uls	Temperature (K) I (Liquid or Solid Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_T_Uv	Temperature (K) I (Vapor Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Uls_Als	Liquid or Solid Specific-Internal-Energy (MJ/kg) M (Liquid or Solid Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	401
EOS_Uls_Av	Liquid or Solid Specific-Internal-Energy (MJ/kg) M (Vapor Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	401
EOS_Uls_Dls	Liquid or Solid Specific-Internal-Energy (MJ/kg) M (Liquid or Solid Density on coexistence line (Mg/m^3)-dependent)	401
EOS_Uls_Dv	Liquid or Solid Specific-Internal-Energy (MJ/kg) M (Vapor Density on coexistence line (Mg/m^3)-dependent)	401
EOS_Uls_Pv	Liquid or Solid Specific-Internal-Energy (MJ/kg) M (Vapor Pressure (GPa)-dependent)	401
EOS_Uls_T	Liquid or Solid Specific-Internal-Energy (MJ/kg) (Temperature (K)-dependent)	401
EOS_Uls_Uv	Liquid or Solid Specific-Internal-Energy (MJ/kg) M (Vapor Specific-Internal-Energy (MJ/kg)-dependent)	401

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Uv_Als M	Vapor Specific-Internal-Energy (MJ/kg) (Liquid or Solid Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	401
EOS_Uv_Av M	Vapor Specific-Internal-Energy (MJ/kg) (Vapor Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	401
EOS_Uv_Dls M	Vapor Specific-Internal-Energy (MJ/kg) (Liquid or Solid Density on coexistence line (Mg/m^3)-dependent)	401
EOS_Uv_Dv M	Vapor Specific-Internal-Energy (MJ/kg) (Vapor Density on coexistence line (Mg/m^3)-dependent)	401
EOS_Uv_Pv M	Vapor Specific-Internal-Energy (MJ/kg) (Vapor Pressure (GPa)-dependent)	401
EOS_Uv_T	Vapor Specific-Internal-Energy (MJ/kg) (Temperature (K)-dependent)	401
EOS_Uv_Uls M	Vapor Specific-Internal-Energy (MJ/kg) (Liquid or Solid Specific-Internal-Energy (MJ/kg)-dependent)	401

B.10 Category 10: Melt data in SESAME's 411 and 412 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Af_D GPU	Freeze Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)-dependent)	412
EOS_Af_Pf M	Freeze Specific-Helmholtz-Free-Energy (MJ/kg) (Freeze Pressure (GPa)-dependent)	412
EOS_Af_Tf M	Freeze Specific-Helmholtz-Free-Energy (MJ/kg) (Freeze Temperature (K)-dependent)	412
EOS_Af_Uf M	Freeze Specific-Helmholtz-Free-Energy (MJ/kg) (Freeze Specific-Internal-Energy (MJ/kg)-dependent)	412
EOS_Am_D GPU	Melt Specific-Helmholtz-Free-Energy (MJ/kg) (Density (Mg/m^3)-dependent)	411
EOS_Am_Pm M	Melt Specific-Helmholtz-Free-Energy (MJ/kg) (Melt Pressure (GPa)-dependent)	411
EOS_Am_Tm M	Melt Specific-Helmholtz-Free-Energy (MJ/kg) (Melt Temperature (K)-dependent)	411
EOS_Am_Um M	Melt Specific-Helmholtz-Free-Energy (MJ/kg) (Melt Specific-Internal-Energy (MJ/kg)-dependent)	411
EOS_D_Af I	Density (Mg/m^3) (Freeze Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	412
EOS_D_Am I	Density (Mg/m^3) (Melt Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	411
EOS_D_Pf I	Density (Mg/m^3) (Freeze Pressure (GPa)-dependent)	412
EOS_D_Pm I	Density (Mg/m^3) (Melt Pressure (GPa)-dependent)	411
EOS_D_Tf I	Density (Mg/m^3) (Freeze Temperature (K)-dependent)	412
EOS_D_Tm I	Density (Mg/m^3) (Melt Temperature (K)-dependent)	411

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_D_Uf	Density (Mg/m^3) I (Freeze Specific-Internal-Energy (MJ/kg)-dependent)	412
EOS_D_Um	Density (Mg/m^3) I (Melt Specific-Internal-Energy (MJ/kg)-dependent)	411
EOS_Pf_Af	Freeze Pressure (GPa) M (Freeze Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	412
EOS_Pf_D	Freeze Pressure (GPa) GPU (Density (Mg/m^3)-dependent)	412
EOS_Pf_Tf	Freeze Pressure (GPa) M (Freeze Temperature (K)-dependent)	412
EOS_Pf_Uf	Freeze Pressure (GPa) M (Freeze Specific-Internal-Energy (MJ/kg)-dependent)	412
EOS_Pm_Am	Melt Pressure (GPa) M (Melt Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	411
EOS_Pm_D	Melt Pressure (GPa) GPU (Density (Mg/m^3)-dependent)	411
EOS_Pm_Tm	Melt Pressure (GPa) M (Melt Temperature (K)-dependent)	411
EOS_Pm_Um	Melt Pressure (GPa) M (Melt Specific-Internal-Energy (MJ/kg)-dependent)	411
EOS_Tf_Af	Freeze Temperature (K) M (Freeze Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	412
EOS_Tf_D	Freeze Temperature (K) GPU (Density (Mg/m^3)-dependent)	412
EOS_Tf_Pf	Freeze Temperature (K) M (Freeze Pressure (GPa)-dependent)	412
EOS_Tf_Uf	Freeze Temperature (K) M (Freeze Specific-Internal-Energy (MJ/kg)-dependent)	412

Continued on next page

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Tm_Am M	Melt Temperature (K) (Melt Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	411
EOS_Tm_D GPU	Melt Temperature (K) (Density (Mg/m^3)-dependent)	411
EOS_Tm_Pm M	Melt Temperature (K) (Melt Pressure (GPa)-dependent)	411
EOS_Tm_Um M	Melt Temperature (K) (Melt Specific-Internal-Energy (MJ/kg)-dependent)	411
EOS_Uf_Af M	Freeze Specific-Internal-Energy (MJ/kg) (Freeze Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	412
EOS_Uf_D GPU	Freeze Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)-dependent)	412
EOS_Uf_Pf M	Freeze Specific-Internal-Energy (MJ/kg) (Freeze Pressure (GPa)-dependent)	412
EOS_Uf_Tf M	Freeze Specific-Internal-Energy (MJ/kg) (Freeze Temperature (K)-dependent)	412
EOS_Um_Am M	Melt Specific-Internal-Energy (MJ/kg) (Melt Specific-Helmholtz-Free-Energy (MJ/kg)-dependent)	411
EOS_Um_D GPU	Melt Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)-dependent)	411
EOS_Um_Pm M	Melt Specific-Internal-Energy (MJ/kg) (Melt Pressure (GPa)-dependent)	411
EOS_Um_Tm M	Melt Specific-Internal-Energy (MJ/kg) (Melt Temperature (K)-dependent)	411

B.11 Category 11: Shear Modulus data in SESAME's 431 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_D_Gs I	Density (Mg/m^3) (Shear Modulus (Gpa))-dependent)	431
EOS_Gs_D GPU	Shear Modulus (Gpa) (Density (Mg/m^3))-dependent)	431

B.12 Category 12: Opacity data in SESAME's 500-series tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Keo_DT GPU	Electron Conductive Opacity (<i>OpacityModel</i>) (cm^2/g) (Density (Mg/m^3)- and Temperature (eV)-dependent)	503
EOS_Kp_DT GPU	Planck Mean Opacity (cm^2/g) (Density (Mg/m^3)- and Temperature (eV)-dependent)	505
EOS_Kr_DT GPU	Rosseland Mean Opacity (cm^2/g) (Density (Mg/m^3)- and Temperature (eV)-dependent)	502
EOS_Ogb	Calculated versus Interpolated Opacity Grid Boundary	501
EOS_Zfo_DT GPU	Mean Ion Charge (<i>OpacityModel</i>) (<i>freeelectronsperatom</i>) (Density (Mg/m^3)- and Temperature (eV)-dependent)	504

B.13 Category 13: Conductivity data in SESAME's 600-series tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_B.DT GPU	Thermoelectric Coefficient ($1/cm^2/s$) (Density (Mg/m^3)- and Temperature (eV)-dependent)	604
EOS_Kc.DT GPU	Electron Conductive Opacity (Conductivity Model) (cm^2/g) (Density (Mg/m^3)- and Temperature (eV)-dependent)	605
EOS_Kec.DT GPU	Electrical Conductivity ($1/s$) (Density (Mg/m^3)- and Temperature (eV)-dependent)	602
EOS_Ktc.DT GPU	Thermal Conductivity ($1/cm/s$) (Density (Mg/m^3)- and Temperature (eV)-dependent)	603
EOS_Zfc.DT GPU	Mean Ion Charge (Conductivity Model) (<i>freeelectronsperatom</i>) (Density (Mg/m^3)- and Temperature (eV)-dependent)	601

C TABLE TYPES: EOSPAC VERSION 5 CROSS REFERENCE

Below are tables of defined constants corresponding to all of the data table types available within EOSPAC version 5.

It is important to note that the actual values of these constants may change without notice; therefore, use the constant names – do not hardwire the values into the host code. The EOSPAC 6 Constants are color coded as follows:

- **I** indicates the table is inverted with respect to the first independent variable.
- **II** indicates the table is inverted with respect to the second independent variable.
- **M** indicates the table is a combination of two other tables.
- **MIX** indicates the table is compatible with the [eos.Mix](#) routine.

EOSPAC 6 Constant	EOSPAC 5 Constant	Description	SESAME Table(s)
EOS_B_DT MIX	ES4_THERME	Thermoelectric Coefficient ($1/cm^2/s$) (Density (Mg/m^3)- and Temperature (eV)-dependent)	604
EOS_D_PtT I	ES4_DPTTOT	Density (Mg/m^3) (Total Pressure (GPa)- and Temperature (K)-dependent)	301
EOS_Gs_D	ES4_SHEARM	Shear Modulus (Gpa) (Density (Mg/m^3)-dependent)	431
EOS_Kc_DT MIX	ES4_OPACC3	Electron Conductive Opacity (Conductivity Model) (cm^2/g) (Density (Mg/m^3)- and Temperature (eV)-dependent)	605
EOS_Kec_DT MIX	ES4_ECONDE	Electrical Conductivity ($1/s$) (Density (Mg/m^3)- and Temperature (eV)-dependent)	602

Continued on next page

EOSPAC 6 Constant	EOSPAC 5 Constant	Description	SESAME Table(s)
EOS_Keo_DT MIX	ES4.OPACC2	Electron Conductive Opacity (Opacity Model) (cm^2/g) (Density (Mg/m^3)- and Temperature (eV)-dependent)	503
EOS_Kp_DT MIX	ES4.OPACP	Planck Mean Opacity (cm^2/g) (Density (Mg/m^3)- and Temperature (eV)-dependent)	505
EOS_Kr_DT MIX	ES4.OPACR	Rosseland Mean Opacity (cm^2/g) (Density (Mg/m^3)- and Temperature (eV)-dependent)	502
EOS_Ktc_DT MIX	ES4.TCONDE	Thermal Conductivity ($1/cm/s$) (Density (Mg/m^3)- and Temperature (eV)-dependent)	603
EOS_NullTable	ES4.NULLPTR	null table	n/a
EOS_Pc_D MIX	ES4.PRCLD	Pressure Cold Curve (GPa) (Density (Mg/m^3)-dependent)	306
EOS_Pe_DT MIX	ES4.PRELC	Electron Pressure (GPa) (Density (Mg/m^3)- and Temperature (K)-dependent)	304
EOS_Pe_DUe M MIX	ES4.PNELC	Electron Pressure (GPa) (Density (Mg/m^3)- and Electron Specific-Internal-Energy (MJ/kg)-dependent)	304
EOS_Pf_D	ES4.PFREEZ	Freeze Pressure (GPa) (Density (Mg/m^3)-dependent)	412
EOS_Pic_DT MIX	ES4.PRION	Ion Pressure plus Cold Curve Pressure (GPa) (Density (Mg/m^3)- and Temperature (K)-dependent)	303

Continued on next page

EOSPAC 6 Constant	EOSPAC 5 Constant	Description	SESAME Table(s)
EOS_Pic_DUic M MIX	ES4_PNION	Ion Pressure plus Cold Curve Pressure (<i>GPa</i>) (Density (<i>Mg/m³</i>)- and Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (<i>MJ/kg</i>)-dependent)	303
EOS_Pm_D	ES4_PMELT	Melt Pressure (<i>GPa</i>) (Density (<i>Mg/m³</i>)-dependent)	411
EOS_Pt_DT MIX	ES4_PRTOT	Total Pressure (<i>GPa</i>) (Density (<i>Mg/m³</i>)- and Temperature (<i>K</i>)-dependent)	301
EOS_Pt_DUt M MIX	ES4_PNTOT	Total Pressure (<i>GPa</i>) (Density (<i>Mg/m³</i>)- and Total Specific-Internal-Energy (<i>MJ/kg</i>)-dependent)	301
EOS_T_DPe II MIX	ES4_TPELC	Temperature (<i>K</i>) (Density (<i>Mg/m³</i>)- and Electron Pressure (<i>GPa</i>)-dependent)	304
EOS_T_DPic II MIX	ES4_TPION	Temperature (<i>K</i>) (Density (<i>Mg/m³</i>)- and Ion Pressure plus Cold Curve Pressure (<i>GPa</i>)-dependent)	303
EOS_T_DPt II MIX	ES4_TPTOT	Temperature (<i>K</i>) (Density (<i>Mg/m³</i>)- and Total Pressure (<i>GPa</i>)-dependent)	301
EOS_T_DUe II MIX	ES4_TNELC	Temperature (<i>K</i>) (Density (<i>Mg/m³</i>)- and Electron Specific-Internal-Energy (<i>MJ/kg</i>)-dependent)	304

Continued on next page

EOSPAC 6 Constant	EOSPAC 5 Constant	Description	SESAME Table(s)
EOS_T_DUic II MIX	ES4_TNION	Temperature (K) (Density (Mg/m^3)- and Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg)-dependent)	303
EOS_T_DUt II MIX	ES4_TNTOT	Temperature (K) (Density (Mg/m^3)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_Tf.D II	ES4_TFREEZ	Freeze Temperature (eV) (Density (Mg/m^3)-dependent)	412
EOS_Tm.D II	ES4_TMELT	Melt Temperature (K) (Density (Mg/m^3)-dependent)	411
EOS_Uc.D II MIX	ES4_ENCLD	Specific-Internal-Energy Cold Curve (MJ/kg) (Density (Mg/m^3)-dependent)	306
EOS_Ue.DPe M MIX	ES4_EPELC	Electron Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Electron Pressure (GPa)-dependent)	304
EOS_Ue.DT MIX	ES4_ENELC	Electron Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Temperature (K)-dependent)	304
EOS_Uf.D	ES4_EFREEZ	Freeze Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)-dependent)	412
EOS_Uic.DPic M MIX	ES4_EPION	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg) (Density (Mg/m^3)- and Ion Pressure plus Cold Curve Pressure (GPa)-dependent)	303

Continued on next page

EOSPAC 6 Constant	EOSPAC 5 Constant	Description	SESAME Table(s)
EOS_Uic_DT <i>MIX</i>	ES4_ENION	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (<i>MJ/kg</i>) (Density (<i>Mg/m³</i>)- and Temperature (<i>K</i>)-dependent)	303
EOS_Um_D	ES4_EMELT	Melt Specific-Internal-Energy (<i>MJ/kg</i>) (Density (<i>Mg/m³</i>)-dependent)	411
EOS_Ut_DPt M <i>MIX</i>	ES4_EPTOT	Total Specific-Internal-Energy (<i>MJ/kg</i>) (Density (<i>Mg/m³</i>)- and Total Pressure (<i>GPa</i>)-dependent)	301
EOS_Ut_DT <i>MIX</i>	ES4_ENTOT	Total Specific-Internal-Energy (<i>MJ/kg</i>) (Density (<i>Mg/m³</i>)- and Temperature (<i>K</i>)-dependent)	301
EOS_Zfc_DT <i>MIX</i>	ES4_ZFREE3	Mean Ion Charge (Conductivity Model) (free electrons per atom) (Density (<i>Mg/m³</i>)- and Temperature (<i>eV</i>)-dependent)	601
EOS_Zfo_DT <i>MIX</i>	ES4_ZFREE2	Mean Ion Charge (Opacity Model) (free electrons per atom) (Density (<i>Mg/m³</i>)- and Temperature (<i>eV</i>)-dependent)	504

D OPTIONS: *SETUP PHASE*

Below is a list of defined constants corresponding to the user specified setup phase options available within EOSPAC. This list has been alphabetized according to the defined constant names, which are cross-referenced to the applicable EOSPAC 5[9],[10] defined constants. Unlike EOSPAC 5, these EOSPAC option flags are to be applied to a given table handle using one of two public routines: `eos_ResetOption` and `eos_SetOption` (see chapter 7 sections 1.7 and 1.11 respectively). For each table handle, the `eos_SetOption` routine may be used to enable or disable an optional feature. Alternatively, the `eos_ResetOption` routine may be used to reassert the default option settings if, in fact, such default values are defined in the table below. Take note that some of the options require an associated value passed into the `eos_SetOption` routine parameter named `tableOptionVal`.

It is important to note that the actual values of these constants may change without notice; therefore, use the constant names – do not hardwire the values into the host code.

EOSPAC 6 Constant	Default Option State (<code>tableOptionVal</code>)	Description
EOS_ADJUST_VAP_PRES	Disabled (0)	This provides a mechanism for the host code to pass into EOSPAC 6 adjusted pressure values (corresponding to SAGE's ¹ <code>matdef(2,mat)</code> input variable) for the vapor dome to ensure ambient conditions are reasonable for a specified material. This option is only valid when also using the option named <code>EOS_PT_SMOOTHING</code> . It is important to note that the units of the <code>tableOptionVal</code> must be compatible with Sesame pressure data (GPa). See chapter 9 section 1 for more details.

Continued on next page

¹SAGE is a one-, two-, and three-dimensional, multi-material Eulerian hydrodynamics code (LA-UR-04-2959).

EOSPAC 6 Constant	Default Option State (tableOptionVal)	Description
EOS_APPEND_DATA	Disabled (N/A)	Append the loaded data table and descriptive information to an ASCII file named “TablesLoaded.dat” within the current working directory. The corresponding EOSPAC 5[9],[10] setup option used to enable this feature is $lprnt = TRUE$ passed to ES1TABS.
EOS_CALC_FREE_ENERGY	Disabled (N/A)	Instead of using the corresponding Sesame data, the Helmholtz Free Energy data is calculated using the equations equations (3.1) , (3.9) and (3.10) . If no internal energy data exists for $T = 0$, then the free energy data will not be calculated.
EOS_CHECK_ARGS	Disabled (N/A)	Allow extensive argument checking.
EOS_CREATE_TZERO	Disabled (N/A)	Using linear extrapolation along each isochore, create a $T = 0$ isotherm if it’s unavailable when loading 300-series Sesame data.
EOS_DUMP_DATA	Disabled (N/A)	Write the loaded data table and descriptive information to an ASCII file named “TablesLoaded.dat” within the current working directory. The corresponding EOSPAC 5[9],[10] setup option used to enable this feature is $lprnt = TRUE$ passed to ES1TABS.

Continued on next page

EOSPAC 6 Constant	Default Option State (tableOptionVal)	Description
EOS_INSERT_DATA	Disabled (0)	<p>Insert grid points between each original grid point with respect to all independent variables (i.e., increase grid resolution). The value of the <code>eos_SetOption</code> parameter, <code>tableOptionVal</code>, is to contain the user-defined number of data points to insert between existing data points. The corresponding EOSPAC 5[9],[10] setup option used to enable this feature is $iopt = 10000N$, given $(0 \leq N \leq 9)$ passed to ES1TABS. See chapter 9 section 7.2 about this and EOS_INVERT_AT_SETUP.</p>
EOS_INVERT_AT_SETUP	Disabled (N/A)	<p>Create an inverted table during the Setup Phase (chapter 5) and store it in memory rather than the waiting until the Interpolation Phase (chapter 6) to invert tabulated data. This option is implemented in response to user requests for improved interpolation performance of problems that are heavily-dependent upon inverted data tables.</p> <p>This option is ignored and the EOS_INVALID_OPTION_FLAG error code is returned if the host code attempts to set this option for a non-inverted data table type. See chapter 9 section 7 for a discussion about this option.</p>

Continued on next page

EOSPAC 6 Constant	Default Option State (tableOptionVal)	Description
EOS_MONOTONIC_IN_X	Disabled (N/A)	Enable forced monotonicity with respect to x of $F(x,y)$. The corresponding EOSPAC 5[9],[10] setup option used to enable this feature is $iopt = 100$ passed to ES1TABS.
EOS_MONOTONIC_IN_Y	Disabled (N/A)	Enable forced monotonicity with respect to y of $F(x,y)$. The corresponding EOSPAC 5[9],[10] setup option used to enable this feature is $iopt = 300$ passed to ES1TABS.
EOS_PT_SMOOTHING	Disabled (N/A)	This performs all the necessary data smoothing taken from SAGE. ² See the related setup option named EOS_ADJUST_VAP_PRES and the related interpolation option named EOS_USE_CUSTOM_INTERP. See chapter 9 section 1 for more details.
EOS_SAVE_SPECIES_DATA	Disabled (N/A)	Enable the storage of selected species-specific data that is calculated during the usage of the eos_Mix routine.
EOS_SKIP_EXTRAP_CHECK	Disabled (N/A)	All extrapolation checks are skipped except when using eos_CheckExtrap .

Continued on next page

²SAGE is a one-, two-, and three-dimensional, multi-material Eulerian hydrodynamics code (LA-UR-04-2959).

EOSPAC 6 Constant	Default Option State (tableOptionVal)	Description
EOS_SMOOTH	Disabled (N/A)	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. The corresponding EOSPAC 5[9],[10] setup option used to enable this feature is <i>iopt</i> = 10 passed to ES1TABS.
EOS_SPLIT_COWAN	Disabled (N/A)	Allows splitting for ion data table not found in the database using the cold curve plus Cowan-nuclear model for ions.
EOS_SPLIT_FORCED	Disabled (N/A)	Forces specified splitting option for data table.
EOS_SPLIT_IDEAL_GAS	Disabled (N/A)	Allows splitting for ion data table not found in the database using the cold curve plus ideal gas model for ions.
EOS_SPLIT_NUM_PROP	Disabled (N/A)	Allows splitting for ion data table not found in the database using the cold curve plus number-proportional model for ions.
EOS_USE_MAXWELL_TABLE	Disabled (N/A)	Use the Maxwell data in table 311 instead of the corresponding table 301.

E DATA INFORMATION PARAMETERS

Information about a table can be requested via the `eos_GetTableInfo` routine using the parameters defined in this section. The `eos_GetTableInfo` routine is designed to be general in functionality, so these parameters are grouped according to their prerequisites.

It is important to note that the actual values of these constants may change without notice; therefore, use the constant names – do not hardwire the values into the host code.

Table E-1 lists parameters that require the comment tables (i.e., EOS_Comment) for a material to be loaded and associated with a table handle.

Table E-1: Information parameter(s) related to SESAME’s 100-series tables

Parameter	Description
EOS_Cmnt_Len	Retrieve the length in characters of the comments available for the specified data table

Table E-2 lists parameters that require the general material data table (i.e., EOS_Info) to be loaded and associated with a table handle.

Table E-2: Information parameter(s) related to SESAME’s 201 tables

Parameter	Description
EOS_Exchange_Coeff	Retrieve the exchange coefficient
EOS_Mean_Atomic_Mass	Retrieve the mean atomic mass
EOS_Mean_Atomic_Num	Retrieve the mean atomic number
EOS_Modulus	Retrieve the solid bulk modulus
EOS_Normal_Density	Retrieve the normal density

Table E-3 lists parameters that require data to be loaded and associated with a table handle; however, they don’t apply to SESAME’s 100-series and 201 tables.

Table E-3: Information parameter(s) generally related to SESAME’s tables except for SESAME’s 100-series and 201 tables

Parameter	Description
EOS_F_Convert_Factor	Retrieve the conversion factor corresponding to the dependent variable, $F(x, y)$. This is an alias for EOS_F_CONVERT.
EOS_Log_Val	Retrieve the InfoVal that is non-zero if the data table is in a log10 format.
EOS_X_Convert_Factor	Retrieve the conversion factor corresponding to the primary independent variable, x . This is an alias for EOS_X_CONVERT.
EOS_Y_Convert_Factor	Retrieve the conversion factor corresponding to the secondary independent variable, y . This is an alias for EOS_Y_CONVERT.
EOS_NX	Retrieve the extent of the xVals extrapolation lower/upper bound(s) arrays. This value is dependent upon the table type associated with the table handle, and it can be either 1 or NT. For example, for tables inverted with respect to density this will be the number of temperatures (NT) – for all others this will be 1.
EOS_NY	Retrieve the extent of the yVals extrapolation lower/upper bound(s) arrays. This value is dependent upon the table type associated with the table handle, and it can be either 1 or NR. For example, for tables inverted with respect to temperature this will be the number of densities (NR) – for all others this will be 1.
EOS_X_BOUND_GRID	Retrieve the extrapolation bound(s) grid array ³ for the xVals. If the value returned by EOS_NX is 1, then this will return an arbitrary scalar value of 0.
EOS_X_LOWER_BOUND	Retrieve the extrapolation lower bound(s) array ³ for the $f(x_{min}, y)$, which corresponds to xVals in chapter 6 .

Continued on next page

³The extent of this array is dependent upon the table type associated with the table handle, and it will correspond to the value returned by EOS_NX.

Table E-3: Information parameter(s) generally related to SESAME’s tables except for SESAME’s 100-series and 201 tables

Parameter	Description
EOS_X_UPPER_BOUND	Retrieve the extrapolation upper bound(s) array ³ for the $f(x_{max}, y)$, which corresponds to xVals in chapter 6.
EOS_Y_BOUND_GRID	Retrieve the extrapolation bound(s) grid array ⁴ for the yVals. If the value returned by EOS_NY is 1, then this will return an arbitrary scalar value of 0.
EOS_Y_LOWER_BOUND	Retrieve the extrapolation lower bound(s) array ⁴ for the $f(x, y_{min})$, which corresponds to yVals in chapter 6.
EOS_Y_UPPER_BOUND	Retrieve the extrapolation upper bound(s) array ⁴ for the $f(x, y_{max})$, which corresponds to yVals in chapter 6.

For an arbitrary tabulated surface, $S=f(x,y)$, as shown in the cartoon to the right, the various boundaries, which are defined by the EOS_X_LOWER_BOUND, EOS_X_UPPER_BOUND, EOS_Y_LOWER_BOUND and EOS_Y_UPPER_BOUND options, are shown as $f(x_{min}, y)$, $f(x_{max}, y)$, $f(x, y_{min})$ and $f(x, y_{max})$ respectively. For SESAME data that is not inverted, all four boundary curves are scalar (i.e., minimum and maximum densities and temperatures). For inverted forms, one of the bounding pairs (either $f(x, y_{min})$ and $f(x, y_{max})$ or $f(x_{min}, y)$ and $f(x_{max}, y)$) are scalars and the other are defined as tabulated 1-D curves.

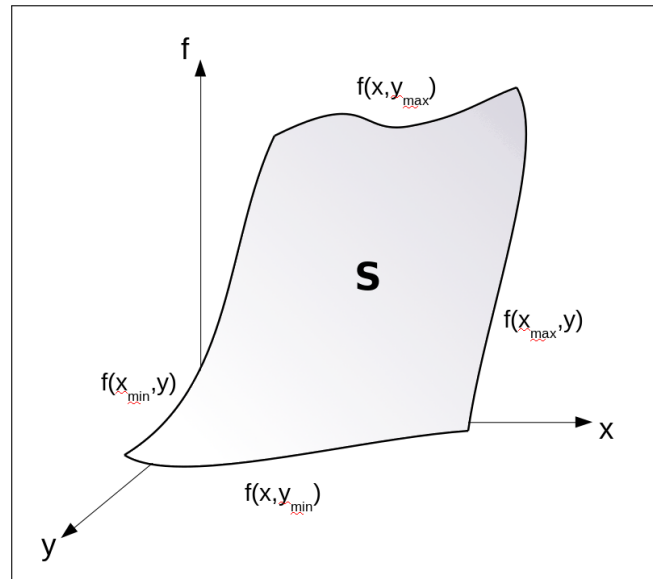


Table E-4 lists parameters that require data to be loaded and associated with a table handle. In other words, all table handles that are associated with data may be queried for the information indicated by these parameters.

⁴The extent of this array is dependent upon the table type associated with the table handle, and it will correspond to the value returned by EOS_NY.

Table E-4: Information parameter(s) generally related to SESAME's tables

Parameter	Description
EOS_Material_ID	Retrieve the SESAME material identification number
EOS_Table_Type	Retrieve the type of data table. Corresponds to the parameters in APPENDICES B and C

[Table E-5](#) lists parameters that require data to be loaded and associated with a table handle; however, they are only valid for non-inverted data tables specifically related to SESAME's 301 and 401 tables.

Table E-5: Information parameter(s) associated with non-inverted data tables

Parameter	Description
EOS_R_Array	Retrieve the density array Note that InfoVals must be allocated to hold NR EOS_REAL values, so querying for the EOS_NR value is first necessary. The conversion factor supplied via the EOS_X_CONVERT option will affect these data.
EOS_T_Array	Retrieve the temperature array Note that InfoVals must be allocated to hold NT EOS_REAL values, so querying for the EOS_NT value is first necessary. The conversion factor supplied via the EOS_Y_CONVERT option will affect these data.
EOS_F_Array	Retrieve the F array This two-dimensional array will be assigned to the one-dimensional array, InfoVals, in a column-major order. Note that InfoVals must be allocated to hold NR*NT EOS_REAL values, so querying for the EOS_NR and EOS_NT values is first necessary. The conversion factor supplied via the EOS_F_CONVERT option will affect these data.
EOS_NR	Retrieve the number of densities

Continued on next page

Table E-5: Information parameter(s) associated with non-inverted data tables

Parameter	Description
EOS_NT	Retrieve the number of temperatures
EOS_Rmin	Retrieve the minimum density. The conversion factor supplied via the EOS_X_CONVERT option will affect these data.
EOS_Rmax	Retrieve the maximum density. The conversion factor supplied via the EOS_X_CONVERT option will affect these data.
EOS_Tmin	Retrieve the minimum temperature. The conversion factor supplied via the EOS_Y_CONVERT option will affect these data.
EOS_Tmax	Retrieve the maximum temperature. The conversion factor supplied via the EOS_Y_CONVERT option will affect these data.
EOS_Fmin	Retrieve the minimum F value. The conversion factor supplied via the EOS_F_CONVERT option will affect these data.
EOS_Fmax	Retrieve the maximum F value. The conversion factor supplied via the EOS_F_CONVERT option will affect these data.
EOS_NUM_PHASES	Retrieve the number of material phases that are tabulated. This is only valid in conjunction with the EOS_M_DT table type.

Table E-6 lists parameters that can be used to fetch internally-interpolated species-specific data associated with a table handle that is used by the `eos_Mix` routine. To gain access to these data, one must enable the `EOS_SAVE_SPECIES_DATA` option (APPENDIX D). Of the information parameters listed in table E-6, the `EOS_X_Species_Data` is probably the most useful, since it corresponds to the species-specific density array that is iteratively-solved for the table type's pressure balance function used during mixing. Of the thirty-three `eos_Mix`-supported table types listed in section 2.3, there exist two important balance functions that are internally-defined for each – these balance functions are correlated to the supported table types (i.e., EOSPAC 6 Constant) in table E-7. Each table type's associated temperature balance function constant is listed for information only, because

all that are not defined as EOS_NullTable correspond to the equilibrium temperature that is either specified or implied by the `eos_Mix` input Y value(s) that are provided by the host code and correspond to the mixed table type (first column of [table E-6](#)). While they have been implemented in the EOSPAC 6 library for many years, these parameters are documented herein, because they are implemented in and actively used by the new `mix_sesame_data` tool that is described in [chapter 8](#).

Table E-6: Information parameter(s) associated with species-specific stored data. For each option listed in this table, the extent of the returned array corresponds to the `nXYPairs` value supplied to the `eos_Mix` routine, and sufficient memory must be made available by the host code to store each requested array. See the `EOS_SAVE_SPECIES_DATA` option in [APPENDIX D](#).

Parameter	Description
<code>EOS_X_Species_Data</code>	Retrieve the density array associated with the current table type's species-specific pressure balance function used during mixing.
<code>EOS_Y_Species_Data</code>	Retrieve the Y array associated with the current table type's species-specific pressure balance function used during mixing.

Table E-7: Internally-defined balance functions used by the `eos_Mix` routine to achieve the desired pressure and temperature equilibrium for the interpolated data corresponding to the defined EOSPAC 6 Constant (i.e., table type).

EOSPAC 6 Constant	SESAME Table(s)	Temperature Balance Function Constant	Pressure Balance Function Constant
<code>EOS_At_DT</code>	301	<code>EOS_NullTable</code>	<code>EOS_Pt_DT</code>
<code>EOS_Pt_DT</code>	301	<code>EOS_NullTable</code>	<code>EOS_Pt_DT</code>
<code>EOS_Ut_DT</code>	301	<code>EOS_NullTable</code>	<code>EOS_Pt_DT</code>

Continued on next page

Table E-7: Internally-defined balance functions used by the `eos_Mix` routine to achieve the desired pressure and temperature equilibrium for the interpolated data corresponding to the defined EOSPAC 6 Constant (i.e., table type).

EOSPAC 6 Constant	SESAME Table(s)	Temperature Balance Function Constant	Pressure Balance Function Constant
EOS_T_DPt	301	EOS_T_DPt	EOS_NullTable
EOS_Ut_DPt	301	EOS_T_DPt	EOS_NullTable
EOS_Pt_DUt	301	EOS_T_DUt	EOS_Pt_DUt
EOS_T_DUt	301	EOS_T_DUt	EOS_Pt_DUt
EOS_Aic_DT	303	EOS_NullTable	EOS_Pic_DT
EOS_Pic_DT	303	EOS_NullTable	EOS_Pic_DT
EOS_Uic_DT	303	EOS_NullTable	EOS_Pic_DT
EOS_T_DPic	303	EOS_T_DPic	EOS_NullTable
EOS_Uic_DPic	303	EOS_T_DPic	EOS_NullTable
EOS_Pic_DUic	303	EOS_T_DUic	EOS_Pic_DUic
EOS_T_DUic	303	EOS_T_DUic	EOS_Pic_DUic
EOS_Ae_DT	304	EOS_NullTable	EOS_Pe_DT
EOS_Pe_DT	304	EOS_NullTable	EOS_Pe_DT

Continued on next page

Table E-7: Internally-defined balance functions used by the `eos_Mix` routine to achieve the desired pressure and temperature equilibrium for the interpolated data corresponding to the defined EOSPAC 6 Constant (i.e., table type).

EOSPAC 6 Constant	SESAME Table(s)	Temperature Balance Function Constant	Pressure Balance Function Constant
EOS_Ue_DT	304	EOS_NullTable	EOS_Pe_DT
EOS_T_DPe	304	EOS_T_DPe	EOS_NullTable
EOS_Ue_DPe	304	EOS_T_DPe	EOS_NullTable
EOS_Pe_DUe	304	EOS_T_DUe	EOS_Pe_DUe
EOS_T_DUe	304	EOS_T_DUe	EOS_Pe_DUe
EOS_Ac_D	306	EOS_NullTable	EOS_Pc_D
EOS_Pc_D	306	EOS_NullTable	EOS_Pc_D
EOS_Uc_D	306	EOS_NullTable	EOS_Pc_D
EOS_Kr_DT	502	EOS_NullTable	EOS_Zfo_DT
EOS_Keo_DT	503	EOS_NullTable	EOS_Zfo_DT
EOS_Zfo_DT	504	EOS_NullTable	EOS_Zfo_DT
EOS_Kp_DT	505	EOS_NullTable	EOS_Zfo_DT
EOS_Zfc_DT	601	EOS_NullTable	EOS_Zfc_DT

Continued on next page

Table E-7: Internally-defined balance functions used by the `eos_Mix` routine to achieve the desired pressure and temperature equilibrium for the interpolated data corresponding to the defined EOSPAC 6 Constant (i.e., table type).

EOSPAC 6 Constant	SESAME Table(s)	Temperature Balance Function Constant	Pressure Balance Function Constant
EOS_Kec_DT	602	EOS_NullTable	EOS_Zfc_DT
EOS_Ktc_DT	603	EOS_NullTable	EOS_Zfc_DT
EOS_B_DT	604	EOS_NullTable	EOS_Zfc_DT
EOS_Kc_DT	605	EOS_NullTable	EOS_Zfc_DT

F META-DATA INFORMATION PARAMETERS

Information about a table can be requested via the `eos_GetMetaData` and `eos_GetTableMetaData` routines using the parameters defined in this section. The `eos_GetMetaData` and `eos_GetTableMetaData` routines are designed to be general in functionality, so these parameters are grouped according to their usage.

It is important to note that the actual values of these constants may change without notice; therefore, use the constant names – do not hardwire the values into the host code.

Table F-1: Information parameter(s) used for the first argument (`infoItem`) of the `eos_GetMetaData` routine

Parameter	Description
All table type constants defined in APPENDICES B and C	Specify the table type of interest

Table F-2: Information parameter(s) used for the second argument (`infoItemCategory`) of the `eos_GetMetaData` routine

Parameter	Description
<code>EOS_Table_Type</code>	Retrieve the specified table type's string representation. Corresponds to the parameters in APPENDICES B and C
<code>EOS_Table_Name</code>	Retrieve the specified table type's descriptive name. Corresponds to the parameters's descriptions in APPENDICES B and C
<code>EOS_Dependent_Var</code>	Retrieve the short string representation of the specified table type's dependent variable as listed in APPENDIX A
<code>EOS_Independent_Var1</code>	Retrieve the short string representation of the specified table type's first independent variable as listed in APPENDIX A

Continued on next page

Table F-2: Information parameter(s) used for the second argument (infoItemCategory) of the `eos_GetMetaData` routine

Parameter	Description
EOS_Independent_Var2	Retrieve the short string representation of the specified table type's second independent variable as listed in APPENDIX A
EOS_Sesame_Table_List	Retrieve the specified table type's associated SESAME table number(s)
EOS_Pressure_Balance_Table_Type	Retrieve the specified table type's associated pressure balance table type as used by the <code>eos_Mix</code> algorithms [11]
EOS_Temperature_Balance_Table_Type	Retrieve the specified table type's associated temperature balance table type as used by the <code>eos_Mix</code> algorithms [11]

Table F-3: Information parameter(s) used for the second argument (infoItem) of the `eos_GetTableMetaData` routine

Parameter	Description
EOS_File_Name	Retrieve the SESAME file name that is associated with the specified table handle
EOS_Material_Name	Retrieve the material name that is associated with the specified table handle
EOS_Material_Source	Retrieve the material source (e.g. author) ⁵ that is associated with the specified table handle
EOS_Material_Date	Retrieve the material creation date ⁵ that is associated with the specified table handle
EOS_Material_Ref	Retrieve the material documentation reference(s) ⁵ that is associated with the specified table handle
EOS_Material_Composition	Retrieve the material composition ⁵ that is associated with the specified table handle

Continued on next page

⁵This information is found the SESAME 101 table, which is loaded using the `EOS_Comments` table type

Table F-3: Information parameter(s) used for the second argument (infoItem) of the `eos_GetTableMetaData` routine

Parameter	Description
EOS_Material_Codes	Retrieve the data generation software name(s) ⁵ that is associated with the specified table handle
EOS_Material_Phases	Retrieve the material phase name(s) ⁵ that is associated with the specified table handle
EOS_Material_Classification	Retrieve the material classification description ⁵ that is associated with the specified table handle. Examples include, but are not limited to, Unknown, Unclassified, Export-Controlled, etc.

G OPTIONS: INTERPOLATION PHASE

Below is a list of defined constants corresponding to the user specified interpolation options available within EOSPAC. This list has been alphabetized according to the defined constant names, which are cross-referenced to the applicable EOSPAC 5[9],[10] defined constants. Unlike EOSPAC 5, these EOSPAC option flags are to be applied to a given table handle using one of two public routines: `eos_ResetOption` and `eos_SetOption` (see chapter 7 sections 1.7 and 1.11 respectively). For each table handle, the `eos_SetOption` routine may be used to enable or disable an optional feature. Alternatively, the `eos_ResetOption` routine may be used to reassert the default option settings if, in fact, such default values are defined in the table below. Take note that some of the options require an associated value passed into the `eos_SetOption` routine parameter named `tableOptionVal`.

It is important to note that the actual values of these constants may change without notice; therefore, use the constant names – do not hardwire the values into the host code.

EOSPAC 6 Constant	Default Option State (tableOptionVal)	Description
EOS_DISCONTINUOUS_DERIVATIVES	Disabled (N/A)	Enable the original linear/bilinear logic, which calculates discontinuous derivatives at the tabulated grid. This option requires the interpolation option, EOS_LINEAR, to be enabled for the specified table handle. See chapter 9 section 6 for a brief discussion of the rationale for this option.

Continued on next page

EOSPAC 6 Constant	Default Option State (tableOptionVal)	Description
EOS_F_CONVERT ⁶	Disabled (1.0)	Set the conversion factor used on the fVals dependent variable value(s). The value of the eos.SetOption parameter, tableOptionVal, is to contain the conversion factor value. See chapter 9 section 7.3 for more details. ⁷
EOS_LINEAR	Disabled (N/A)	Bilinear interpolation. The corresponding EOSPAC 5[9],[10] interpolation option used to enable this feature is idrvs=ES4_BILINE passed to ES1VALS.
EOS_RATIONAL	Enabled (N/A)	Birational interpolation. The corresponding EOSPAC 5[9],[10] interpolation option used to enable this feature is idrvs=ES4_BIRATF passed to ES1VALS.
EOS_SKIP_EXTRAP_CHECK	Disabled (N/A)	All extrapolation checks are skipped unless host calls eos.CheckExtrap .

Continued on next page

⁶The [eos.SetOption](#) parameter, tableOptionVal (see [chapter 7 section 1.11](#)), must be defined to an appropriate number for this option.

⁷The conversion factor value is defined so that it converts values from the SESAME units[1] to the host code units; therefore, it is internally-used as a multiplier to convert the output values to the appropriate host code units.

EOSPAC 6 Constant	Default Option State (tableOptionVal)	Description
EOS_USE_CUSTOM_INTERP	Disabled (N/A)	Use a custom inverse-interpolation algorithm that requires the setup option, EOS_PT_SMOOTHING, to be enabled for the specified table handle. This option is only valid for table types EOS_Ut_PtT and EOS_V_PtT. See chapter 9 section 1 for more details. Note that the partial derivatives, dFx and dFy, are not calculated when this option is set.
EOS_USE_HOST_XY	Disabled (N/A)	This option is not recommended for use. See chapter 9 section 7.4 for additional discussion and details.
EOS_X_CONVERT ⁶	Disabled (1.0)	Set the conversion factor used on the xVals independent variable value(s). The value of the eos_SetOption parameter, tableOptionVal, is to contain the conversion factor value. See chapter 9 section 7.3 for more details. ⁸

Continued on next page

⁸The conversion factor value is defined so that it converts values from the host code units to the SESAME units[1]; therefore, it is internally-used as a divisor to convert the input values to the appropriate SESAME units.

EOSPAC 6 Constant	Default Option State (tableOptionVal)	Description
EOS_Y_CONVERT ⁶	Disabled (1.0)	Set the conversion factor used on the yVals independent variable value(s). The value of eos.SetOption parameter, tableOptionVal, is to contain the conversion factor value. See chapter 9 section 7.3 for more details. ⁸
EOS_XY_MODIFY	Disabled (N/A)	Do not create an internal copy of the xVals and yVals inputs for eos.Interpolate , eos.Mix and eos.CheckExtrap . Modify the xVals and yVals inputs in situ – use host code’s arrays directly. Overrides previously-set EOS_XY_PASSTHRU option. See chapter 9 section 7.4 for additional discussion and details.
EOS_XY_PASSTHRU	Disabled (N/A)	Neither create an internal copy nor modify the xVals and yVals inputs for eos.Interpolate , eos.Mix and eos.CheckExtrap . Use host code’s arrays directly – unmodified. Overrides previously-set EOS_XY_MODIFY option. See chapter 9 section 7.4 for additional discussion and details.

H ERROR CODES

Below is a list of defined constants corresponding to all of the possible error codes returned by EOSPAC. This list has been alphabetized according to the defined constant names, which are cross-referenced to the applicable EOSPAC 5[9],[10] defined constants.

It is important to note that the actual values of these constants may change without notice; therefore, use the constant names – do not hardwire the values into the host code.

NOTE: As of version 6.3, comparison of two error codes now requires the usage of the *eos_ErrorCodesEqual* routine described in chapter 7 section 1.1

EOSPAC 6 Constant (EOSPAC 5 Constant)	Description
EOS_BAD_DATA_TYPE (ES5_BADTABLETYPE)	Data table type is not recognized
EOS_BAD_DERIVATIVE_FLAG (ES5_BADDERIVTYPE)	Derivative is not recognized
EOS_BAD_FREE_ENERGY_DATA	Helmholtz Free Energy data in SESAME is thermodynamically-inconsistent with the Internal Energy data; usage of EOS_CALC_FREE_ENERGY is recommended for this material
EOS_BAD_INTERPOLATION_FLAG (ES5_BADINTRPTYPE)	Interpolation is not recognized
EOS_BAD_MATERIAL_ID (ES5_MATIDZERO)	Material ID is zero
EOS_CANT_INVERT_DATA	Can't invert with respect to the required independent variable
EOS_CANT_MAKE_MONOTONIC	Can't make data monotonic in X
EOS_CONVERGENCE_FAILED (ES5_CONVERGEFAILED)	Iterative algorithm did not converge during inverse interpolation
EOS_DATA_TYPE_NOT_FOUND (ES5_TYPENOTFOUND)	Data table type is not in library
EOS_DATA_TYPE_NO_MATCH	Data types do not match as required for mixing

Continued on next page

EOSPAC 6 Constant (EOSPAC 5 Constant)	Description
EOS_FAILED	Operation failed
EOS_GEN401_AND_NOT_FOUND	401 data was generated and not found
EOS_INDEX_FILE_ERROR	The sesameFilesDir.txt file parser found a syntax error
EOS_INTEGRATION_FAILED	Numerical integration failed or not possible
EOS_INTERP_EXTRAPOLATED	Interpolation caused extrapolation beyond data table boundaries
EOS_INTERP_EXTRAP_PBAL	Pressure balance function extrapolated beyond data table boundaries
EOS_INTERP_EXTRAP_TBAL	Temperature balance function extrapolated beyond data table boundaries
EOS_INVALID_CONC_SUM	The sum of the supplied material concentrations does not equal 1.0
EOS_INVALID_DATA_TYPE	Operation is not defined on this data type
EOS_INVALID_INFO_FLAG	The info flag passed into either eos_GetTableInfo or eos_GetTableMetaData is invalid
EOS_INVALID_INFO_CATEGORY_FLAG	The info category flag passed into eos_GetMetaData is invalid
EOS_INVALID_NXYPAIRS	Invalid nXYPairs value
EOS_INVALID_OPTION_FLAG	The option flag passed into eos_SetOption is invalid
EOS_INVALID_SPLIT_FLAG	The data splitting option is invalid
EOS_INVALID_SUBTABLE_INDEX	Subtable index out of the range
EOS_INVALID_TABLE_HANDLE	Invalid table handle
EOS_MATERIAL_NOT_FOUND (ES5_MATNOTFOUND)	Material ID is not in library
EOS_MEM_ALLOCATION_FAILED (ES5_EXPANDFAILED)	EOS table area cannot be expanded
EOS_MIN_ERROR_CODE_VALUE	Minimum error code value
EOS_NOT_ALLOCATED	Memory not allocated for data
EOS_NOT_INITIALIZED (ES5_NOTINIT)	EOS table area is not initialized

Continued on next page

EOSPAC 6 Constant (EOSPAC 5 Constant)	Description
EOS_NO_COMMENTS	No comments available for this data table
EOS_NO_DATA_TABLE (ES5_NOTABLE)	Data table is not in EOS table area
EOS_NO_SESAMO_FILES (ES5_NOFILESFOUND)	No data library files exist
EOS_OK (ES5_OK)	No errors detected
EOS_OPEN_OUTPUT_FILE_FAILED	Could not open TablesLoaded.dat or related data file
EOS_OPEN_SESAMO_FILE_FAILED (ES5_OPENFAILED)	Could not open data file
EOS_READ_DATA_FAILED (ES5_LDTABLEFAILED)	Could not load data table
EOS_READ_FILE_VERSION_FAILED (ES5_GETVERSNFAILED)	Could not load version from data file
EOS_READ_MASTER_DIR_FAILED (ES5_LDMASTERFAILED)	Could not load master directory
EOS_READ_MATERIAL_DIR_FAILED (ES5_LDMATDIRFAILED)	Could not load material directory
EOS_READ_TOTAL_MATERIALS_FAILED (ES5_GETNMATSFAILED)	Could not read number of materials
EOS_SPLIT_FAILED	The data splitting algorithm failed
EOS_UNDEFINED	The result is undefined
EOS_WARNING	Operation has generated a warning and an associated custom message
EOS_xHi_yHi	Both the x and y arguments were high
EOS_xHi_yLo	The x argument was high, the y argument was low ⁹
EOS_xHi_yOk	The x argument was high, the y argument was OK ¹⁰

Continued on next page

⁹If the y argument corresponds to a temperature value, then a zero temperature was used for interpolation rather than the value supplied by the host code.

¹⁰If the x argument corresponds to a density value, then a zero density was used for interpolation rather than the

EOSPAC 6 Constant (EOSPAC 5 Constant)	Description
EOS_xLo_yHi	The x argument was low, the y argument was OK
EOS_xLo_yLo	Both the x and y arguments were low ^{9, 10}
EOS_xLo_yOk	The x argument was low, the y argument was OK ¹⁰
EOS_xOk_yHi	The x argument is OK and the y argument is high
EOS_xOk_yLo	The x argument is OK and the y argument is low ⁹

value supplied by the host code.
