Numerical implementation and oceanographic application of the thermodynamic potentials of water, vapour, ice, seawater and air – Part 2: The library routines

D. G. Wright¹, R. Feistel², J. H. Reissmann³, K. Miyagawa⁴, D. R. Jackett⁵, W. Wagner⁶, U. Overhoff⁶, C. Guder⁶, A. Feistel⁷, and G. M. Marion⁸

¹Bedford Institute of Oceanography, Dartmouth, NS, B2Y 4A2, Canada
²Leibniz-Institut für Ostseeforschung, Seestraße 15, 18119 Warnemünde, Germany
³Bundesamt für Seeschifffahrt und Hydrographie, Bernhard-Nocht-Straße 78, 20359 Hamburg, Germany
⁴4-12-11-628, Nishiogu, Arakawa-ku, Tokyo 116-0011, Japan
5CSIRO Marine and Atmospheric Research, G.P.O. Box 1538, Hobart, TAS 7001, Australia
6Ruhr-Universität Bochum, Lehrstuhl für Thermodynamik, 44780 Bochum, Germany
7Technische Universität Berlin, Einsteinufer 25, 10587 Berlin, Germany
8Desert Research Institute, Reno, NV, USA

Received: 30 January 2010 – Accepted: 28 February 2010 – Published: 15 March 2010

Correspondence to: D. G. Wright (dan.wright@dfo-mpo.gc.ca)

Published by Copernicus Publications on behalf of the European Geosciences Union.
Abstract

The SCOR/IAPSO\textsuperscript{1} Working Group 127 on Thermodynamics and Equation of State of Seawater has prepared recommendations for new methods and algorithms for numerical estimation of the thermophysical properties of seawater. As an outcome of this work, a new International Thermodynamic Equation of Seawater (TEOS-10) was endorsed by IOC/UNESCO\textsuperscript{2} in June 2009 as the official replacement and extension of the 1980 International Equation of State, EOS-80. As part of this new standard a source code package has been prepared that is now made freely available to users via the World Wide Web. This package includes two libraries referred to as the SIA (Sea-Ice-Air) library and the GSW (Gibbs SeaWater) library. Information on the GSW library may be found on the TEOS-10 web site (http://www.TEOS-10.org). This publication provides an introduction to the SIA library which contains routines to calculate various thermodynamic properties as discussed in the companion paper. The SIA library is very comprehensive, including routines to deal with fluid water, ice, seawater and humid air as well as equilibrium states involving various combinations of these, with equivalent code developed in different languages. The code is hierarchically structured in modules that support (i) almost unlimited extension with respect to additional properties or relations, (ii) an extraction of self-contained sub-libraries, (iii) separate updating of the empirical thermodynamic potentials, and (iv) code verification on different platforms and between different languages. Error trapping is implemented to identify when one or more of the primary routines are accessed significantly beyond their established range of validity. The initial version of the SIA library is available in Visual Basic and FORTRAN as a supplement to this publication and updates will be maintained on the TEOS-10 web site.

\textsuperscript{1}SCOR/IAPSO: Scientific Committee on Oceanic Research/International Association for the Physical Sciences of the Oceans

\textsuperscript{2}IOC/UNESCO: Intergovernmental Oceanographic Commission/United Nations Educational, Scientific and Cultural Organization
1 Introduction

A new International Thermodynamic Equation of Seawater (TEOS-10) was endorsed by IOC/UNESCO in 2009 as the official replacement and extension of the 1980 International Equation of State, EOS-80. General theoretical formulas used in TEOS-10 for a broad range of thermodynamic properties that may be calculated based on the thermodynamic potentials discussed in recently endorsed IAPWS documents (IAPWS, 2008a, 2009a, b, c, 2010) are provided by Feistel et al. (2010b, c). The primary goal of this publication is to introduce a set of library routines developed in order to facilitate the transition to general usage of these new thermodynamic formulations. Two distinct but related libraries have been developed for this purpose. The first deals with pure fluid water (liquid or vapour), ice, seawater and humid air and is referred to as the SIA (Sea-Ice-Air) library since it deals with all three media (including the limiting cases of the salt content of seawater going to zero and either the water vapour content or the air content of moist air going to zero). The SIA library updates and extends the numerical routines published previously by Feistel (2005) and Feistel et al. (2005). The second library is referred to as the Gibbs SeaWater (GSW) library. It is restricted to consideration of seawater properties for the typical terrestrial oceanic (commonly referred to as “Neptunian”) range of conditions. It is a secondary standard in the sense that it is more restricted and based on a mathematical representation of the Gibbs function of pure liquid water (IAPWS, 2009c) in the form of a correlation function computed by regression with respect to data points calculated from elements of the SIA library (Feistel, 2003). It is smaller, focussed on seawater properties and more computationally efficient than the SIA library; it is documented on the TEOS-10 web site (http://www.TEOS-10.org).

Only the SIA library is discussed in detail here. The Primary Standard of this library consists of implementations of IAPWS formulations for fluid water, ice and seawater (IAPWS, 2008a, 2009a, b, c; Feistel et al., 2008), as well as a planned IAPWS document on humid air (IAPWS, 2010). The potential functions described in these

---

3IAPWS: International Association for the Properties of Water and Steam
documents are implemented in terms of Helmholtz functions, $f$, and Gibbs functions, $g$, which contain various empirical coefficients representing the experimental evidence. Superscripts on the potential functions are used to indicate the substance and phase represented; “F” refers to Fluid water in either liquid or vapour form, “W” refers to liquid Water, “V” refers to water Vapour, “Ih” refers to hexagonal pure water Ice, “S” refers to a contribution from sea Salt, “A” refers to (dry) Air, “SW” refers to SeaWater (or Salt dissolved in Water) and “AV” refers to Air with water Vapour in it. From the Primary Standard, the required properties are computed from rigorous thermodynamic relations numerically implemented in various library modules (Fig. 1). These derived properties cover pure substances, their mixtures and composites, expressed in terms of various combinations of input parameters and free of any additional empirical parameters. The modular code permits the extraction of self-contained sub-libraries for specific applications, such as a pure-water or a seawater branch, a sea-ice branch or a humid-air branch.

The library is designed to be easily updated and extended. The potential functions comprising the Primary Standard may be updated simply by substituting newly determined relations with consistent function names, inputs and outputs. The modules in the Primary Standard are rigorously consistent and independent so that any one module can be updated without the need to modify any other module. They are also complete so that an update of the Primary Standard constitutes an update of the full library; improvements of empirical relations in the Primary Standard will be automatically propagated throughout all elements of the library (except level 5, see Sect. 3.6). The library can be easily extended to include new relations by adding them to existing modules or by introducing new modules without the need to modify any existing components of the library. The modular code thus permits virtually unlimited extension with respect to additional required properties or alternative combinations of input parameters.
Initially, through this publication, the SIA library is made available in FORTRAN and Visual Basic to fill the needs of both computationally intensive and more interactive applications. The two versions of the library are constructed and verified to be essentially equivalent, with common function names, inputs and outputs so that a user familiar with one version may easily transition to the other version when this is more convenient for a particular application. Future plans include providing access to the library in MATLAB and C++ with the same implementation philosophy.

The cross-language consistency between the Visual Basic and FORTRAN implementations (and future implementations in other programming languages) requires a careful comparison of the results computed by the two code versions from the same particular function with the same input parameters, covering the valid ranges of those parameters sufficiently densely. From the sheer number of routines and input parameters, it is obvious that this can only be managed by suitable code that calls all library functions at definite grid points and stores the results for comparison between languages (or platforms). Additional code is then required to compare the tables of gridded results to identify inconsistencies while taking account of reasonable numerical roundoff and occasional error returns. The latter poses a specific challenge. In arbitrary parameter spaces, the sufficient conditions for validity form complicated shapes, in several cases not even convex regions. Gridding the cuboid between the parameter extrema includes various points that lie outside the valid range. This often results in either the convergence of iteration procedures to spurious fixed points, or triggering of numerical errors such as “overflow”. It turns out that the results obtained in such cases frequently depend on minor numerical differences that can cause different results to be obtained from different implementations or on different platforms. In the comparison of the outputted check lists, this different behaviour produces virtual inconsistencies that may even come and go with tiny changes made in the code or the grid point location. The only way to solve this problem was to force the code to return a definite error indicator rather than unidentified spurious results within regions that are definitely outside of the known range of validity. This was achieved as detailed in Sect. 5 by
specifying parameter ranges that include all formally valid choices of parameters and do not include parameter values far outside of this formal validity range.

The naming and input/output conventions used in the library are discussed in Sect. 2 and the detailed organisation of routines in the library is discussed in Sect. 3. Practical usage is discussed in Sect. 4. The question of how to determine and implement checks on the ranges of validity for results involving a multi-component system is considered in Sect. 5. Section 6 provides a brief summary. Checks against previous results as well as checks of the “internal” consistency of the routines available in different languages and different libraries are discussed in the Appendix.

Although the code package builds on work published in IAPWS Releases, it has been developed and recommended by the SCOR/IAPSO WG127 and is not officially endorsed by IAPWS. Substantial effort has been dedicated to eliminating errors from the library routines, but the authors cannot guarantee that none remain. They invite users to report errors and related issues to the corresponding authors.

2 Naming and I/O conventions

As already mentioned, the SIA library is currently available in Visual Basic and FORTRAN. The DOS extensions .bas and .F90 are respectively associated with the files in the Visual Basic and FORTRAN versions of the library but they will not be mentioned explicitly unless a language-specific consideration is involved. Names of library elements such as directories, modules, procedures or constants are typeset in the courier font throughout this article for easier identification.

All names in the library are case insensitive; i.e., a function name is never reused with a different case to indicate a different quantity. All functions accept integer and double precision (64 bit) input values compatible with the language standards and all routines return “double” precision floating point outputs. Return values equal to or
greater than the value of the constant ErrorReturn (set to 0.999 999 999 E+99 in
the standard distribution) indicate errors as discussed in Sect. 5. The numerical value
assigned to the constant ErrorReturn used in the library can be modified in the
module Constants.0 if this is desired for reasons such as consistency with external
program parts to be combined with library modules. The only restriction is that its
absolute value must be large compared to realistic values returned by any of the library
routines.

As illustrated in Fig. 1 and discussed in detail in the next section, the library is
organized into levels and modules with the same structure used in the two different lan-
guages. Each module may contain several related routines. Prefixes on routine names
are used to indicate the substance or phase they refer to, or their special purpose (Ta-
ble 1). liq. and vap. refer to routines that deal respectively with the liquid and vapour
states, whereas flu. is used to indicate modules and routines that deal with either
form of fluid water, i.e., either the liquid or vapour state depending on the choice of
input parameters. liq. and sal. refer to the individual components of seawater while
sea. refers to routines dealing with the combined solution. The prefix ice. identifies
routines dealing with pure ice, dry. refers to dry air and air. refers to routines dealing
with dry air, moist air or water vapour. Additional prefixes are used to indicate equi-
librium states between phases. In each case, the two (or three) relevant states are
indicated using the three letter identifiers given in Table 1 separated by an underscore.
For example, the string sea.ice. occurs at the beginning of each routine that returns
equilibrium properties between seawater and ice (freezing point, melting heat etc.).
Routines beginning with set. and get. are used for setting and getting control pa-
rameter values and chk. identifies routines used to produce check values. aux. refers
to private auxiliary routines used primarily in the initialization of iterative routines and
fit. refers to approximate fits that may be of interest to some users. These include the
liquid and vapour components of the industrial formulation IAPWS-IF97, melting tem-
perature as a function of pressure and melting pressure as a function of temperature
for pure water/ice and the Gibbs function for pure water expressed directly in terms of
The prefix on each routine name is followed by a descriptive name that indicates its specific purpose. When necessary, the property name behind the prefix starts with f, g, h or eta to distinguish between different sets of input parameters available for the same property of the same phase. Properties with f are derived from a Helmholtz function and take temperature and density as inputs, those with g are derived from a Gibbs function and take temperature and pressure, those with h are derived from enthalpy and take entropy and pressure, and those with eta are derived from entropy as the potential and take enthalpy and pressure as inputs. Each of those alternative inputs is possibly used in combination with a composition variable, SA for the Absolute Salinity, or A for the dry-air mass fraction.

Suffixes are used on variable and routine names to indicate the units of inputs and outputs. Each of the routines at levels 1–4 of the SIA library has inputs and outputs in basic SI units (kg, m, s, K, newtons (1 N=1 kg m s⁻²), joules (1 J=1 N m) and pascals (1 Pa=1 N m⁻²=1 J m⁻³)). The suffix _si is included to indicate this fact; t_si is the absolute temperature in K, sa_si is the absolute salinity as a dimensionless mass fraction (kg kg⁻¹) between 0 and 1, p_si is the absolute pressure in Pa, and d_si is the density in kg m⁻³. In the case of potential functions that optionally return various partial derivatives, the suffix _si indicates that the value returned is expressed in terms of the combination of those basic SI units appropriate for the particular derivative. Within level 5, routines may use units more familiar to oceanographers (e.g., sea pressure in dbar, temperature in °C, density in g cm⁻³) and indicative suffixes such as _c (celcius), _dbar (decibar) and _c100m (°C per 100 m) are used to indicate this fact. Level 5 also includes routines to convert between various units in common usage. Details are provided in Table S42 of the supplement (http://www.ocean-sci-discuss.net/7/649/2010/osd-7-649-2010-supplement.zip).
3 Library levels

The primary purpose of this section is to describe the rationale used to divide the SIA library into levels.

The library of routines is organized into the same 5 levels discussed in Part 1 (Feistel et al., 2010b) plus an additional level 0 where some physical constants and mathematical algorithms are provided. Throughout the library, there are different thermodynamic potential functions for which the required first and second derivatives with respect to the arguments are available by specifying input parameters. These functions are listed separately in Table 1 of Part 1 and underlined in Table S1 of the supplement (http://www.ocean-sci-discuss.net/7/649/2010/osd-7-649-2010-supplement.zip). Note that although the module names of the library carry the level number as a suffix, the level is not part of any routine name. Hence, to call a particular function such as sea_density_si the user does not need to know at which level or in which module this property is implemented.

The library is structured such that routines from a particular level only need access to other routines contained in modules at the same or lower levels in the library. Mathematically, the relation “A uses B” between any two modules A and B of the library automatically disallows “B uses A” directly or indirectly via chains of used modules, and therefore forms a semi-order.

The factors determining the level occupied by any particular routine are summarized below.

3.1 Level 0: Common constants and algorithms

Each of the modules at level 0 ends in _0. This level is reserved for constants and routines that are used throughout the library but do not directly deal with thermodynamic properties of the geophysical substances. The module Constants_0 specifies the values of physical constants as well as parameter values that determine whether or not certain options will be enabled. The options influence the range of applicability of some
library routines as discussed in Sect. 5. The module Maths_0 includes some simple routines that are used internally. The Visual Basic library includes additional mathematical routines such as Complex_0, implementing some complex-number arithmetic operations which are not originally supported by the language. Routines to determine the molar mass of moist air and to deal with conversions between mole fractions and mass fractions of water vapour in humid air are included in Convert_0. The formulations of Lemmon et al. (2000) and IAPWS-95 (IAPWS, 2009a; Wagner and Pruß, 2002) are used respectively to represent dry air and water vapour contributions to moist air.

The Convert_0 module also includes conversions between Absolute Salinity and Practical Salinity, including the influence of composition anomalies when the geographic location is specified, following McDougall et al. (2009). Details regarding the representation of salinity in the TEOS-10 formulation are discussed in Wright et al. (2010). In some regions, specialized code will be required to provide accurate results. Such specialized code is already included for the Baltic Sea where the anomalies can be particularly strong (Feistel et al., 2010a). When the geographic location is not included, the Absolute Salinity anomaly is specified consistent with the surface waters of the North Atlantic by approximating Absolute Salinity with the Reference-Composition Salinity defined by Millero et al. (2008).

3.2 Level 1: The primary standard: TEOS-10

Each of the modules at level 1 ends in _l. They contain the Primary Standard and may depend on quantities defined at level 0 but are independent of all other modules including other modules at level 1 as well as the contents of the higher levels.

Sixty-nine different algebraic functions are available using the following seven function calls:

\[
\text{flu_f.si(drv_t, drv_d, t.si, d.si)} \quad (1)
\]
\[
\text{ice_g.si(drv_t, drv_p, t.si, p.si)} \quad (2)
\]
In all cases, \( t_{si}, p_{si} \) and \( d_{si} \) take the temperature, pressure and density values in K, Pa and kg m\(^{-3}\), respectively, and \( \text{drv}_t, \text{drv}_p \) and \( \text{drv}_d \) specify the orders of the required derivatives.

\( \text{flu}_f_{si}(\text{drv}_t, \text{drv}_d, t_{si}, d_{si}) \) is the Helmholtz function \( f^F(T,\rho) \) of fluid water discussed in Part 1 and typically referred to as IAPWS-95 (IAPWS, 2009a; Wagner and Prüß, 2002).

\( \text{ice}_g_{si}(\text{drv}_t, \text{drv}_p, t_{si}, p_{si}) \) is the Gibbs function \( g^{\ln}(T,P) \) of ice discussed in Part 1 and typically referred to as IAPWS-06 (IAPWS, 2009b; Feistel and Wagner, 2006).

\( \text{sal}_g_{term}_{si}(\text{term}, \text{drv}_t, \text{drv}_p, t_{si}, p_{si}) \) provides access to the individual temperature- and pressure-dependent coefficients of the seven functions of the series expansion in salinity included in the saline component of the Gibbs potential for seawater, \( g^S(S_A,T,P) \) (Feistel, 2008). \( g^S \) represents the contribution from sea salt to the total Gibbs function \( g^{SW} \) for seawater as discussed in Part 1. The parameter \( \text{term} \) selects one of the seven coefficients of the functions of salinity. This function is introduced in order to deal precisely with the limit \( S_A \to 0 \) for quantities involving the Gibbs function and its derivatives.

\( \text{dry}_f_{si}(\text{drv}_t, \text{drv}_d, t_{si}, d_{si}) \) is the Helmholtz function \( f^A(T,\rho) \) for dry air (Lemmon et al., 2000) discussed in Part 1. In this library, the specific Helmholtz energy is used, which is expressed on a per-mass basis (in J kg\(^{-1}\)) rather than as a molar quantity (in J mol\(^{-1}\)) as in Lemmon et al. (2000).
air.baw.m3mol(drv_t, t.si), air.caaw.m6mol2(drv_t, t.si) and air.caww.m6mol2(drv_t, t.si) are the second and third cross-virial coefficients required to determine the Helmholtz function for moist air. These coefficients describe the deviation from ideal behaviour due to interactions between two or three particles, involving at least one air and one water particle. The coefficients involving only a single component (either air or water) are included in dry_f_si and flu_f_si. The second cross-virial coefficient air.baw.m3mol is taken from Harvey and Huang (2007) and the third coefficients, air.caaw.m6mol2 and air.caww.m6mol2, are taken from Hyland and Wexler (1983). Although contributions to the full virial expansion arising from either dry air or pure water alone are included in dry_f_si and flu_f_si, higher air-water cross-virial coefficients are unknown and are neglected.

Together, the level 1 routines obey the general criteria for axiomatic systems,

1. **Consistency**: it is impossible to derive two different results for the same property by using alternative algebraic combinations of the elements of this set,

2. **Independence**: it is impossible to derive the values of any element within this set from an algebraic combination of other elements of this set,

3. **Completeness**: it is possible to compute all thermodynamic properties of water, vapour, ice, seawater and air from algebraic equations involving the elements of this set.

Adherence to these conditions allows the entire Primary Standard to be upgraded by modifying only routines at this level and for any routine at level 1 to be upgraded independently; the influence is then propagated throughout the library automatically.

### 3.3 Level 2: Explicit properties

The modules at level two each end in _2_. They may depend on quantities defined at levels 0 and 1 as well as other modules at level 2 but they are independent of the contents of the higher levels.
All functions at level 2 and higher ultimately depend on the Primary Standard (level 1) so their accuracies are limited by the uncertainties in the values returned from level 1. Similarly, the ranges of validity cannot exceed those of the functions used from level 1.

Level 2 consists of simple algebraic combinations of the functions (1–5) representing frequently used thermodynamic properties, such as heat capacity, compressibility and sound speed (see Table S1, http://www.ocean-sci-discuss.net/7/649/2010/osd-7-649-2010-supplement.zip, for a full listing). This set of functions is strictly consistent with those of level 1 since no iterative solutions that would require specification of a convergence criterion are permitted at this level. The uncertainties of these quantities are therefore determined entirely by those of level 1 that they depend upon and their ranges of validity are given by the intersection of the ranges of validity for the quantities that they depend on. In contrast to level 1, the quantities at level 2 do not necessarily obey the conditions of independence or completeness.

3.4 Level 3: Implicit properties

The modules from level 3 each end in 3 and are independent of routines found at higher levels in the library.

Function level 3 consists of functions obtained iteratively from algebraic equations involving functions from levels 1 or 2 and information from level 0. Additional numerical uncertainties thus appear as a result of the particular computation algorithms used in these routines. Thus, depending on the (adjustable) accuracy of the iterations performed, rigorous consistency with the functions at levels 1 and 2 is no longer guaranteed. The default settings ensure that this is not a problem for practical applications.

Two groups of level-3 routines are of special interest. First, the Gibbs functions for pure water, seawater and humid air appear at this level. This is because Gibbs functions in general are expressed in terms of $T$ and $P$, and those for pure water and air are here defined in terms of the corresponding Helmholtz functions $f(T,\rho)$. Thus an iterative solution of the equation $P = \rho^2 f_\rho(T,\rho)$ is required to determine $\rho$ from the specified values of $T$ and $P$. The solution procedure is complicated by the fact that it
must recognise mathematical ambiguities; details are discussed in Part 1.

In the second group are quantities that are determined for a specified reference level, for parcels that have been moved to that level while conserving entropy and salinity in the case of seawater or conserving entropy and air fraction in the case of humid air. These quantities require iterative routines to determine values at the specified reference pressure with fixed values of entropy and salinity or air fraction. In many cases, observed pressure excursions of a seawater or air parcel satisfy these constraints to very good approximation. The quantities in this group include potential temperatures, potential enthalpies and potential densities as well as other properties available from enthalpy as a thermodynamic potential, depending on entropy and pressure as the independent variables.

3.5 Level 4: Equilibrium states

The modules from level 4 each end in _4 and are independent of level 5 routines.

The routines at level 4 all deal with composite equilibrium states. These occur when different phases (air, water vapour, liquid water, seawater or ice) coexist in equilibrium at common temperature, pressure and chemical potential values. Important examples are the equilibrium between ice and seawater in sea ice, providing the freezing temperature of seawater, or the equilibrium between liquid water and humid air, providing the saturation vapour pressure which is required for the determination of the relative humidity. Determination of these quantities generally requires determination of one or more input parameters required by level-1, 2 and/or 3 routines through the iterative solution of the equations satisfied at equilibrium (see Part 1). For each routine involving an equilibrium state, the two or more relevant states are indicated by a composite prefix composed of the appropriate three letter identifiers given in Table 1, separated by an underscore. Detailed discussions of these quantities can be found in Part 1 and in IOC (2010).
Rather than implementing various different routines that provide the same property from alternative subsets of input variables, the level-4 modules are organised as “state engines”. The internal state variables of the particular module are first adjusted to the given conditions by calling any of the available `set_equilibrium_at_` routines (see Table S1, blocks S20 through S30; the identification of blocks is discussed in the caption of Table S1). After successful completion, several properties of this equilibrium state can then be retrieved using function calls without input parameters. The state variables corresponding to the determined equilibrium are used as inputs and remain in effect until a different `set_equilibrium_at_` command is executed.

The routines included at levels 2 through 4 provide a useful but somewhat arbitrary selection of derived thermodynamic properties; this selection is not complete, independent or strictly systematic. Additional properties may be added in later versions of the library by the code developers or by users of the library for special applications.

### 3.6 Level 5: Special functions and non-SI or non-basic SI units

The modules containing functions from this level each end in `5`. The main purposes of level 5 are to provide routines to easily convert between various pressure, temperature and salinity units in common use and to provide more computationally efficient routines for oceanographic applications that are called with inputs expressed in units commonly used in the oceanographic literature, which are not basic SI units. It includes the oceanographic routines for seawater that are colloquially referred to as F03 (Feistel, 2003, 2008; IAPWS, 2008a, 2010b) as well as portions of the IAPWS Industrial Formulation for fluid water known as IAPWS-IF97 (IAPWS, 2007), including in particular separate Gibbs functions for liquid and gaseous water.

To deal with conversions between different units, three routines are included in this module. `cnv_pressure` allows conversions between Pa, hPa, kPa, MPa, mbar, bar, dbar, kbar, torr, psi, kg cm\(^{-2}\), mm of Hg, atmospheres and depth below the surface in an isothermal, isohaline and hydrostatic ocean (Saunders, 1981). The conversion between dbar and any other pressure unit includes the conversion between
absolute pressure and sea pressure, i.e., the addition or subtraction of standard atmospheric pressure, 101325 Pa since dbar is expressed as sea pressure whereas results in any other unit default to include the standard atmosphere pressure contribution. \texttt{cnv\_temperature} allows conversions between °F, °C and K with inputs on any of the IPTS-48, IPTS-68 or ITS-90 scales and outputs on either the IPTS-68 or ITS-90 scale. Outputs on the IPTS-48 scale are supported only if the inputs are also on this scale. \texttt{cnv\_salinity} allows conversions between chlorinity (ppt), Knudsen salinity (ppt), conductivity (siemens/m, milli-siemens/cm, mohms/cm) and conductivity ratio, Practical Salinity (on the PSS-78 scale), Reference-Composition Salinity (kg/kg or g/kg; Millero et al., 2008) and Absolute Salinity (kg/kg or g/kg; McDougall et al., 2009).

The level-5 routines with names starting with the string \texttt{gsw\_} mimic the routines available in the GSW library (http://www.TEOS-10.org). These routines have the same names, accept the same input parameters and return the same outputs as the corresponding routines in the GSW library. Use of these routines will be substantially more computationally efficient than calculations based on routines from levels 1 through 4 of the library but they do not always include the full level of optimization used in the GSW library, and they do not support the full range of \( T - P \) conditions of the SIA library. The computational efficiency of these routines derives primarily from the use of \texttt{fit\_liq\_g\_f03\_si}, the Gibbs function formulation for pure liquid water expressed as an explicit function of \( T \) and \( P \); no efficient grouping of common powers or hornering is implemented in these routines. By avoiding additional optimizations we maintain the ability to directly compare the oceanographic routines at level 5 with the associated IAPWS Releases, easy readability of the SIA code, and also provide an independent check on the GSW library routines. Other than increased computational efficiency and the restriction to Neptunian conditions, these routines are effectively equivalent to those based on the Helmholtz function; numerical differences are easily within the range of measurement uncertainties.
4 Practical usage

In this section we provide some basic information to assist new users. We describe where to get the library routines, how to install them and provide some basic information on how to use them. As discussed below, Tables S1 through S42 in the supplement (http://www.ocean-sci-discuss.net/7/649/2010/osd-7-649-2010-supplement.zip) are key to the efficient use of the library routines.

4.1 Getting and installing the library routines on user platforms

The initial version of the code is available as a digital supplement that accompanies this Ocean Sciences contribution. Over time it is expected that the library routines will be modified and/or extended. Updated code and documentation will be available at the TEOS-10 web site (http://www.TEOS-10.org).

The individual libraries can be downloaded as separate packages for each language. Each library package is gzipped so that the size is reduced to order 2 to 3 MB. A typical FORTRAN library package will have a name like SIA_FTNCODE_30-10-09.tar.gz. On a windows system, it can be unpacked using standard routines such as WinZip. On a unix-based system, it can be unpacked using, for example, the following commands.

    gunzip SIA_FTNCODE_30-10-09.tar.gz
    tar xvf SIA_FTNCODE_30-10-09.tar

Executing these commands will create a directory which in this case would be called SIA_FTNCODE_30-10-09.

The basic library code (see Table S1, http://www.ocean-sci-discuss.net/7/649/2010/osd-7-649-2010-supplement.zip) is contained in a single directory called SIA_library. The various check routines listed in Tables A1 and A2 are also included in this directory.
4.2 Finding information on the library routines

New users will be faced with finding what they need amongst the routines included in the library. We have included Table S1 in the supplement (http://www.ocean-sci-discuss.net/7/649/2010/osd-7-649-2010-supplement.zip) primarily to make this step easier.

The overall structure and content of the SIA library are summarized in Table S1. The levels are presented in order and are clearly indicated by suffixes on the module names. At levels 1 through 3, the four columns of Table S1 contain routines associated with pure fluid water, ice, salt or seawater, and air. At level 4 where equilibrium states are considered, the column location is determined by the first element of the name and successive rows of the table hold routines with common second elements.

As already noted, the library is organized such that modules use only information that can be obtained either locally or from lower level modules. The lower level modules required by each module of the library are listed under Uses in Table S1. The routines that are available for external use (either in user routines or in other modules) are listed under Public Routines, also in Table S1.

Additional information on each of the routines listed in Table S1 is provided in the series of Tables S2 through S42. There are separate tables corresponding to Convert_0 at level 0 and for each of the modules at levels 1 through 5.

The following steps will identify relevant routines and determine basic information on how to use them.

1. Consult Table S1 to find a function with a name indicating that it might provide the required information. Routines to calculate Absolute Salinity from other quantities are located at level 0, routines involving a single phase are at levels 1 through 3, multi-component equilibrium properties are at level 4 and functions with non-standard SI input and output units (as often used by oceanographers, for example) are at level 5. Additional routines to convert between different units are available at level 5.
2. Once a function of interest is identified in Table S1 (block Sn), the user can easily obtain additional information on this function from Table Sn in the same supplement. The block identifiers in Table S1 are located in brackets immediately before the module names at the tops of the table cells. Additional information on inputs and outputs for each routine, how to call each routine and what each routine is based on is provided in Tables S2 through S42. References to the appropriate sections of Part 1 where additional information can be found are included in the captions of the supplementary tables.

Upon execution of the library routines, the required arrays of coefficients of the thermodynamic potentials are initialised automatically, and default settings are made for the various iteration routines. Initial values, tolerances, etc. of the iteration algorithms are automatically set to reasonable default values that have been verified to work in the cases considered. However, users may wish to use alternative algorithms, require increased accuracy or need to modify initial conditions for special applications. For these purposes, procedures beginning with set_it_ctrl have been included in the library to allow the user to choose between different available options or reset adjustable parameters. Details are provided in the supplement (http://www.ocean-sci-discuss.net/7/649/2010/osd-7-649-2010-supplement.zip).

4.2.1 FORTRAN-specific information

The FORTRAN version of the SIA library of TEOS-10 routines will normally be accessed through a separate user-written FORTRAN program that calls the library routines to calculate specific quantities of interest. The user’s FORTRAN routine must be written, compiled and then linked to the library routines to create an executable file. The FORTRAN code has been kept simple and modular so that using the library routines should be straightforward. A FORTRAN 90 or 95 compiler is required but otherwise there are no known special requirements.

Two topics warrant special attention, determining the required use associations
for each module and compiling library and user routines. The first of these topics is dealt with through Table S1 (http://www.ocean-sci-discuss.net/7/649/2010/osd-7-649-2010-supplement.zip). All modules required to be included in use associations for each routine are given under the heading Uses for the module where the routine is found. Modules that are used directly by the module containing the routine of interest are given first followed, in brackets, by the modules that are used in turn by these modules. The module where the routine is actually located must, of course, be included as well.

Second, we must compile and link the user’s program to the library routines. Below, we discuss the most automated and portable approach available which makes use of the GNU autoconf/automake tools. Following that we consider a more light-weight approach that is not so automated but still very useful and easy to learn.

If the GNU autoconfig and automake tools are installed on the user’s system, then the basic SIA library can be compiled simply by executing the following command from the directory where autogen.sh is found (TEOS-10-X.Y.Z):

```
./autogen.sh <return>
```

The scripts autogen.sh and configure.in located in the TEOS-10-X.Y.Z directory as well as the Makefile.am files located in the directories TEOS-10-X.Y.Z and SIA.library were written by Malte Thoma, at the Alfred Wegener Institute, Bremerhaven, Germany. The autogen.sh script executes autoconfig and then calls configure to compile each of the identified targets in the SIA.library directory. It also creates makefiles in each directory. All of the module, object and executable files can be deleted by typing “make clean <return>” from the command prompt. Subsequently typing “make <return>” or rerunning autogen.sh will recreate these files. If new modules are created in the SIA.library directory, they will be detected and compiled automatically when autogen.sh is executed.

Use of the GNU autoconf/automake tools is the most automated and portable
method available to manage the FORTRAN library routines. Users familiar with the GNU tools may wish to use these tools to manage their working directories as well. Here, thanks to Malte Thoma, we provide the tools to manage the basic library using the GNU tools and leave it to the user to extend this approach if desired. We also include information on alternative compilation approaches in the README.LIBRARY file within the SIA.library directory. Four options are discussed, starting with the most automated and portable approach proceeding through to a very simple shell script that simply compiles the library modules. Each of these approaches may also be used to manage the user’s working directories. If the user chooses to use the GNU tools to manage the main library routines and to manage user-written routines separately, care must be taken to ensure that a common compiler is used throughout.

The directory EXAMPLES is included with the FORTRAN library package to provide simple examples of how users might manage a working directory that could be located anywhere on the computation platform. The examples include routines to provide easy access to all of the check routines listed in Tables A1 and A2 as well as an interactive routine to determine selected properties for parameter values specified by the user. For compiling and linking these and other user-written routines, we have found a perl script written by Hugh Pumphrey at the University of Edinburgh, UK to be very useful and easy to work with. This script is stored in the file fmkmf.sh which will be found in the SIA.library directory. Perl must be available on the compute platform to make use of this approach but it is free software, licensed under the GNU General Public License.

Information on the use of fmkmf.sh is available at http://www.geos.ed.ac.uk/homes/hcp/fmkmf or http://www.geos.ed.ac.uk/homes/hcp/fmkmf/fmkmf.txt. An example of its use can be found in the script TEOSf90.sh located in the EXAMPLES directory. On most systems, you should only need to edit the first line of this script to provide the address of the directory where the SIA library files are located. Once this is done, simply entering “./TEOSf90.sh Example_Calculations.F90 <return>” from within the EXAMPLES directory will create a makefile for Example_Calculations.F90, and
then execute this makefile to create an executable called Example_Calculations. Simply entering "./Example_Calculations <return>" will execute this file to begin an interactive session to display various thermodynamic properties of fluid water, ice, seawater and moist air. The same procedure can be used to compile and link user routines to the SIA library routines. Other (even more basic) approaches are discussed in the README_EXAMPLES file.

4.2.2 Visual Basic-specific information

The modular structure and the routine names of the core SIA library in VB are the same as in the FORTRAN version. Additional auxiliary code, for tasks such as the computation and comparison of check values, complex calculus or formatted output is organized in a slightly different manner than in FORTRAN. The user does not generally need to be aware of these internal differences. However, one difference that the users should be aware of is the way that check_limits is initialized in the two different codes. In FORTRAN, the value of check_limits is simply initialized as either 0 or 1 in the module Constants_0 to disable or enable range checking, respectively. In the VB6 code, this choice is controlled by the parameter initial_check_limits_value in Constants_0 which is used internally to initialize Property Get check_limits in VB6. Thus, in VB6 the user will manually set the value of initial_check_limits_value to 0 or 1 to disable or enable range checking whereas in FORTRAN the same goal is achieved by setting the value of check_limits.

The Visual Basic (VB6) version of the SIA Library of TEOS-10 can be used in three ways, provided that either MS Excel or MS Visual Basic is already installed on the host computer:

– to call the SIA library functions from within MS Excel spreadsheets,
– to use the SIA library modules as a part of another external VB program,
To use the library functions with MS Excel spreadsheets, open Excel, then press Alt+F11 to open the Visual Basic Editor, import the modules you need with Ctrl+M, and when ready close the Visual Basic Editor with Alt+Q. Now you are able to access SIA function values within your spreadsheet as shown in Fig. 2. Make sure that along with each module you need you also import any other modules it requires, directly or indirectly, as given in the comment lines at the top of each VB module. The main form, i.e. the user interface window of the compiled program (shown in Fig. 4), is not used within MS Excel.

In the open VB Editor of MS Excel, it is possible to inspect, edit, execute or compile the VB code, or call functions and print the return values within the Immediate Window (Ctrl+G), as shown in Fig. 3.

Alternatively, independent of MS Excel, the VB code can be edited, executed or compiled within the proprietary Visual Basic 6 development environment installed on the user’s hardware, or together with other software tools supporting VB or VBA (Visual Basic for Applications).

An example of an executable SIA routine in VB6 is supplied with the library. It is called TEOS10_SIA.exe and can be used “as is” on a personal computer. Copying the executable to your PC and double-clicking on it should produce the screen shown as Fig. 4. Although this executable is of limited practical value, it provides a simple point-and-click demonstration form. The pull-down menu Check offers the re-computation of various check tables published in the articles and documents behind TEOS-10, discussed in the Appendix. The menu Examples displays several lists of selected properties computed from the SIA library functions, such as the seawater properties shown in Fig. 5. The menu File offers to save ASCII files of the check tables, of the examples and of all check values given in the code as comments in the particular library functions. These files are named SIA_CheckTables.txt, SIA_Examples.txt and SIA_CheckValues.txt, respectively, and are written to the folder where the program...
is executed (the user must ensure that writing is permitted in this folder).

The Visual Basic code is kept as simple as possible to support its modification or migration, such as to VB.NET. It intentionally contains only a single Form (the main window), and refrains from any other input, output or error boxes/windows. It does not require any ActiveX components or DLLs except those that are standard parts of VB. All modules are to be copied into one (arbitrary) folder. No file input or output is implemented, with only two exceptions. First, the file gsw.data.dat must be read if composition anomalies relative to Reference Composition seawater (Millero et al., 2008) are to be accounted for in the computation of Absolute Salinity; gsw.data.dat contains a global look-up table used by the functions asal_from_psal and psal_from_asal for this purpose. Second, writing the files available from the menu File is an option available to the user. All variables are explicitly declared. No variables of the type Variant are used. Only one structured variable is declared, Type CplxType in Complex_0. All arrays are declared with the lower index 0.

5 Ranges of validity

The ranges of validity and accuracy of the potential functions for fluid water and ice are described in the IAPWS Releases (2009a, b). For seawater, both the potential functions for pure water and for salt must be valid (IAPWS, 2008a, 2009a). The validity range for the Helmholtz potential for dry air is discussed by Lemmon et al. (2000). For moist air, consisting of dry air plus water vapour, both the potential functions for water vapour (IAPWS, 2009a) and for dry air (Lemmon et al., 2000) must be valid if both components are present. In addition, if both components are present, then both the cross-virial coefficients and the truncated virial expansion must be valid. The validity ranges of the cross-virial coefficients are discussed by Hyland and Wexler (1983) and the results for the two component air-water interaction are updated by Harvey and Huang (2007). Results presented in Part 1 of this publication indicate that when both components are present in significant amounts, the total density of the moist air mixture
must be less than 100 kg m\(^{-3}\) for the truncated virial expansion to be valid. However, if the mass fraction of one of the two components approaches zero, the virial coefficients play no significant role and in this case, the valid range is determined by the conditions required for the validity of the potential function of the remaining component.

The documents mentioned above are adopted to form the foundation for the library routines discussed here and they should be consulted for detailed information on sufficient conditions for applicability. Here, we consider necessary (i.e., less restrictive) conditions for the validity of the library routines and briefly discuss how they relate to the sufficient (i.e., most restrictive) conditions discussed in these seminal publications. Here we use simple necessary conditions for validity as sufficient conditions for invalidity; violation of these conditions is sufficient to formally invalidate results. The necessary conditions for validity of the primary potential functions are listed in Table 2. These are the conditions used to determine whether or not a function call will result in an ErrorReturn value being returned to indicate an invalid result. It is important to note that these simple conditions do not guarantee that the potential functions are evaluated inside their formal ranges of validity since the validity boundaries exhibit complicated geometric shapes in the different parameter diagrams. A discussion of how our range restrictions were specified is included below.

Two modifications of the original potential function formulations have been implemented in the SIA library to extend the range of valid input parameters given in previous publications. First, an extension of the vapour equation has been implemented that permits the computation of sublimation properties down to 50 K (IAPWS, 2008b; Feistel et al., 2010b, c), below which vapour cannot reasonably be expected to exist (Feistel and Wagner, 2007). This extension is included by default in the library; it influences the results only at temperatures below 130 K. Second, the extension introduced by Feistel (2010) to provide more accurate densities at elevated temperatures and salinities of seawater has been introduced as an option. This extension is consistent with the experimental data used for the computation of IAPWS-08, but with this option “on” all results from the Gibbs function of seawater will differ slightly from the
The ranges in which physically meaningful numbers are returned for all of the functions in the library are determined by the maximum ranges specified in Table 2. Although each of the potential functions are smoothly varying and typically return reasonable values within the associated domains, the boundaries of these domains are sometimes outside of the formal ranges of validity. It is therefore important to recognize the relation between these simple cuboid domains and the more detailed and restrictive validity ranges established in the publications referenced above in which the development and validation of the potential functions is reported. To give an example, the density bounds implemented for fluid water use a minimum and a maximum value as necessary conditions. However, densities in the 2-phase region between the liquid and the vapour states, Fig. 6a, include results corresponding to metastable states and to physically unstable situations. The latter results do not obey the sufficient conditions for validity but are well inside the necessary limits.

Below, we illustrate the relation between the range checks implemented in the library and the more precise ranges of validity given in the related background articles. These illustrations deal only with results for the root-level potential functions $f_F(T, \rho)$ for fluid water, $g_{\text{ih}}(T, P)$ for ice Ih, $g^S(S_A, T, P)$ for sea salt dissolved in liquid water, $f_A(T, \rho)$ for dry air, and the air-water cross-virial coefficients $b_{aw}$, $c_{aaw}$ and $c_{aww}$ that represent the effects of molecular interactions between dry air and water vapour in moist air. All range checks on all other quantities are based on those implemented for these quantities. The validity ranges for these four potential functions are discussed in turn below.
5.1 Fluid water: $f^F(T, \rho)$

The validity range of the IAPWS-95 Helmholtz potential $f^F(T, \rho)$ (flu_fsi in the library) for fluid water (IAPWS, 2009a; Wagner and Prüß, 2002) is shown in Fig. 6a in a density-temperature diagram. Valid applications are restricted to pressures below 1 GPa, temperatures between 130 K and 1273 K and exclude the ice and the liquid-vapour 2-phase regions. The 2-phase region separates the stable vapour phase at low density from the stable liquid phase at high density. The points marked “TP” correspond to the triple point which is just a single point on the $T – P$ diagram but is split on a density-temperature diagram due to the co-existence of ice, liquid and vapour with the same temperature and pressure but different densities. The pressure is defined in the two-phase region in the vicinity of the phase transition curve, i.e., in the weakly metastable range. Values computed from the Helmholtz function in the unstable range beyond the metastable band are invalid; properties of unstable water cannot be measured and are unknown.

The heavy dotted rectangular boundary that is open on the left side of Fig. 6a shows the simplified bounds on temperature and density that have been implemented in the library routines (the lower bound on density is at 0 kg m$^{-3}$). Clearly the range checks implemented in the library allow access to regions outside of the ranges that are sufficient to assure validity. Note that the level-1 code will return credible numbers for all areas within the bounding rectangle; no indication of reduced accuracy is given for areas outside of the strict range of validity.

For the special case of oceanographic applications, the range of interest (the “Neptunian” range) occupies only a small portion of the region shown in Fig. 6a. An expanded view of this thin sliver to the far right of Fig. 6a is shown in Fig. 6b. The true upper bounds on temperature and density for IAPWS-95 are such that they will not be exceeded in oceanographic or meteorological applications. The boundary associated with the vapour pressure line is also not a concern since it allows consideration of the full range of temperatures and pressures of interest in oceanography. However, the
lower bound associated with the freezing temperature of pure water requires special consideration since the freezing temperature is lowered in the presence of dissolved sea salt. To allow use of IAPWS-95 to represent the pure water part of the Gibbs function over the full range of oceanographic interest, the library code has been written to return results for the metastable liquid water phase in the region extending down to the thin line marked “freezing-point lowering” in Fig. 6b. The range checks implemented in the library routines permit access to this region.

5.2 Ice: $g_{\text{ih}}(T,P)$

The validity range of the IAPWS-06 Gibbs function $g_{\text{ih}}(T,P)$ of hexagonal ice I (Feistel and Wagner, 2005, 2006; IAPWS, 2009b) is illustrated in Fig. 7a, b by the curved solid bold lines. The valid temperature range extends from 0 K up to the boundary determined by the melting and sublimation curves, and the valid pressure range extends from 200 MPa down to the sublimation line. The Gibbs function is actually valid for lower pressures than those shown here (Feistel and Wagner, 2007) but the sublimation curve is not determined below 130 K due to the restricted validity of the IAPWS-95 equation for vapour, Fig. 6a. As mentioned in a footnote to Table 2, an extension of the vapour equation down to 50 K is implemented in the library that permits the computation of sublimation properties to this limit (IAPWS, 2008b; Feistel et al., 2010b, c); the sublimation curve, and hence the validity boundary, extends smoothly into this region. The high pressure/low temperature region in the upper left part of the valid region returns reasonable values although the lack of experimental data in this region makes their accuracy uncertain. On this temperature-pressure diagram, the triple point is represented by the single $T - P$ location indicated.

The heavy dotted rectangular boundary that is open at low pressures in Fig. 7a indicates the parameter range beyond which the library code will return the ErrorReturn value to indicate that this routine has been accessed outside of the validity range. If the user evaluates $g_{\text{ih}}(T,P)$ (ice_g_si in the library) for the region inside of this rectangle but outside of the formal validity range of IAPWS-06, unreliable values may be returned.
with no warning provided. Clearly this is most likely to occur in the region below and to the right of the sublimation curve. The results returned in this case will be a formal extension of the ice results into this region rather than inaccurate results for vapour.

Figure 7b shows an expanded view of the high-temperature validity boundary. This view shows that the high temperature validity boundary deviates significantly from 273.16 K as the pressure increases and drops below 0 °C for pressures slightly above the standard atmospheric pressure of 101325 Pa. When salt is dissolved in liquid water surrounding the ice, the melting (and equivalently, the freezing) temperature is lowered (Fig. 6b).

5.3 Seawater: \( f^F(T, \rho^W), g^S(S_A, T, P) \)

The range of validity of the Gibbs function for seawater is shown in Fig. 8. Since the valid range for pure fluid water is relatively broad as discussed above, the validity boundaries for seawater are determined by those of the saline part \( g^S(S_A, T, P) \) (sal_g_si in the library). The details of these boundaries are somewhat complicated due to the presence of both phase transitions and data limitations.

The enclosed volume labelled A in the upper left portion of Fig. 8 covers the Neptunian range, including temperatures from the freezing temperature to 40 °C, salinities from 0 to 40 g kg\(^{-1}\) and pressures from less than one standard atmosphere to 100 MPa. In region B, at pressures extending down to the boiling surface, the validity range remains 0 °C to 40 °C for temperature and extends to salinities up to 50 g kg\(^{-1}\), as a result of the work of Poisson and Ghadoumi (1993) on seawater density.

The valid ranges for temperature and salinity are largest at atmospheric pressure due to increased data availability for thermal and colligative properties. This plane is labelled C in Fig. 8; for the Gibbs energy and its temperature-salinity derivatives the valid temperature range extends to 80 °C and the salinity range extends to 120 g kg\(^{-1}\). Region D indicates the extended region of validity at zero salinity, which extends far beyond the temperature and pressure limits of this diagram (Fig. 6). At zero salinity,
valid results corresponding to the metastable liquid water solution are available even below the freezing temperature (region E); as mentioned in Sect. 5.1, this extension was required to permit use the Helmholtz potential function for pure water in the representation of the Gibbs function for seawater at temperatures for which salt water remains fluid but pure liquid water would normally freeze.

Results outside of the regions defined by bold lines in Fig. 8 have few constraints imposed due to limited data availability; they are best considered as extrapolations based on the equations fit to data within the bold outlined regions. In region C, results for the density generally remain physically reasonable, and colligative properties such as heat capacity and the osmotic coefficient at atmospheric pressure are represented within their experimental uncertainty (Feistel, 2008). However, within the region of hot concentrates labelled F in Fig. 8, results for pressure-dependent properties, including density, are not reliable due to limited data availability and possible precipitation of calcium minerals (Marion et al., 2009). For example, attempts to calculate the sound speed within this region may result in a numerical error due to the argument of the square root going negative. New density measurements at high values of temperature and salinity (Millero and Huang, 2009) have already improved the reliability of results in this region (Feistel, 2010) and more recent data hold promise for further improvements in the future (Safarov et al., 2009). The extension to higher temperatures and salinities discussed by Feistel (2010) has been included in the library and can be activated by setting the flag IsExtension2010 = .True. in the library module Constants_0. We note however, that this extension is neither endorsed by IAPWS nor has it been verified independently. This option should therefore be used with caution; by default it is “off”.

To include all valid parameter values in a simple cuboid, the library allows for temperatures from -10 °C to 80 °C, salinities from 0 to 120 g kg\(^{-1}\) and pressures from 100 Pa to 100 MPa. This range is indicated by the heavy dotted lines in Fig. 8 and obviously includes parameter values well outside the true bounds on the validity of the saline potential function, particularly at high values of both temperature and salinity. In some
applications, it may be desirable to flag results at such parameter values as invalid. For example temperature and salinity bounds of −10 °C to 40 °C, and 0 to 40 g kg\(^{-1}\) might be more appropriate for oceanographic applications. These or other modifications of the range boundaries can be easily implemented for special applications by editing the appropriate “min” and “max” limits specified in the library file Constants_0.

5.4 Dry and humid air: \(f^A(T, \rho^A), f^F(T, \rho^V), b_{aw}, c_{aaw}, c_{aww}\)

The range of validity of the Helmholtz function \(f^A(T, \rho^A)\) for dry air (dry_f_si in the library) is determined by the parameter ranges specified by Lemmon et al. (2000). For pressure, it extends from 0 to 70 MPa and for temperature from 60 to 873 K. The maximum air density in this region is 1035.8 kg m\(^{-3}\). If no water vapour is present and range-checking is enabled, the library routines return physically meaningful values for temperatures between 60 and 873 K and for densities from 0 to 1035 kg m\(^{-3}\) as indicated by the heavy dotted lines in Fig. 9. Note that a phase transition to liquid air occurs below the curved boundary indicated in the lower right portion of Fig. 9a. The bounds implemented in the library do not exclude this region and function evaluations for this region will return results obtained for air with liquefied components. Although the formulation of Lemmon et al. (2000) covers liquid air, too, this region is not relevant to oceanographic or meteorological applications and was not checked in the library.

If water vapour is present \((A < 1)\) then the vapour formula of IAPWS-95 and the cross-virial expansion must also be valid. The temperature range where all three cross-virial coefficients are valid is from −80 to +200 °C (Hyland and Wexler, 1983). For the truncated virial expansion to be valid, the leading neglected terms in the expansion of \(f^{AV}\), proportional to \(A^3(1 - A)\rho^3\), \(A^2(1 - A)^2\rho^3\) and \(A(1 - A)^3\rho^3\), must be negligibly small. From a comparison of the results produced by the library routines with experimental data for saturated air, Feistel et al. (2010b, c) conclude that the virial expansion is valid to densities of at least 100 kg m\(^{-3}\) and remains reasonable to at least 200 kg m\(^{-3}\). The range of temperatures and densities permitted by the library routines when water
vapour is present and range checking is “on” uses the more restrictive value as indicated by the inner heavy dotted line in Fig. 9a.

Figure 9b shows the validity range of the cross-virial coefficients overlain on the phase boundary diagram for water vapour, which is modified only slightly by the inclusion of realistic amounts of dry air if the density on the abscissa is understood as the partial density of vapour. (Note that the dotted isochor at 100 kg m\(^{-3}\) always refers to the total density rather than the partial vapour density. If air is added, the limit of 100 kg m\(^{-3}\) for the total density implies a lower limit remaining for the vapour share, i.e. the upper vapour density limit due to the virial coefficients “moves” to the left in the diagram). From Fig. 9b it is apparent that for very moist air, the upper bound on vapour density will be determined by the transitions to the two-phase region rather than by the upper bound on density required for validity of the cross-virial coefficients. As the moisture content of the air decreases, the “leftward movement” of the vapour density limit in Fig. 9 does have the potential to represent a restriction. However, noting that the typical air density at sea level is of order 1.2 kg m\(^{-3}\), i.e. well within the validity range of the virial coefficients, this limitation could only apply (on Earth) under conditions of highly compressed air. A similar argument applies to the lower bound on the temperature if the vapour density is higher than about 10\(^{-6}\) kg m\(^{-3}\), at which in Fig. 9b the bold sublimation line crosses the dotted 193 K isotherm. For temperature-density points in the two-phase region, the homogeneous vapour state is metastable or unstable, causing part of the water mass to condense in the form of liquid or ice until the vapour density of the remaining gas fraction of the composite “Liquid + Vapour” or “Ice + Vapour” is located on the bold curve, representing saturated air or vapour.

If the total density of moist air at given \(T\) is to the left of the condensation line in Fig. 9b (valid for pure water vapour and in good approximation for the partial vapour density of moist air), then, since the partial density of vapour cannot exceed the total density, saturation cannot happen upon increasing the water part of moist air at fixed total density. Hence, the air fraction \(A\) can vary between 0 and 1 in this case without being restricted by the saturation condition. Although the air fraction of physically stable
states of moist air is otherwise additionally limited by the vapour saturation condition, the library routines allow \(0 \leq A \leq 1\) as indicated in Table 2. As for the limits implemented for the other potential functions, the validity range implemented for moist air should be interpreted as necessary and not sufficient conditions for validity.

Note that moist air can stably exist at total densities in the 2-phase region of Fig. 9b if the partial vapour density is outside the 2-phase region. However, the air fraction \(A\) is additionally constrained in this case.

Although the range-checking included in the library routines is intended to include the broadest possible range of physically allowable parameter values using simple cuboid limits, it should be noted that as the density of either the dry-air or vapour component in moist air is decreased, the contribution from the cross-virial coefficients decreases and the validity range in temperature extends beyond the range indicated in Fig. 9a and b. For very small values of dry air content (\(A\) near 0), the upper temperature and density boundaries may extend towards those of the IAPWS-95 values of 1273 K and 1240 kg m\(^{-3}\) while for \(A\) very close to 1, the range may extend to the dry-air limits indicated in Fig. 9a. These cases are not permitted by the bounds currently implemented in the library when range-checking is enabled. This range of values may be considered by disabling range-checking (i.e., by setting `check_limits = 0` in FORTRAN, or `initial_check_limits_value = 0` in VB, in the library file `Constants_0`).

5.5 Range-checks on higher level library functions

For the routines at level 2, any function evaluation that requires access to one of the primary level-1 routines outside of its range of validity is considered invalid. In some cases, the contribution to the function may be essentially negligible (e.g., when a tiny amount of sea salt is dissolved in pure water or a tiny amount of water vapour is added to dry air) but the total evaluation will be seen as invalid as soon as any contribution from an invalid calculation is included. Clearly, this is a conservative approach. If it is too restrictive for a particular application, range checking may be disabled as mentioned above.
For routines at level 3 or higher, the conditions for validity are somewhat more difficult to implement than for levels 1 and 2. This is primarily because iterative solvers are used at levels 3 and 4, and these solvers may access results outside of the validity range during an iteration, but obtain final values that are within the range of validity. Thus one must permit the code to access results beyond the range of validity during the iteration process. The situation is further complicated by the fact that iterative solvers may be nested and they may involve multiple phases. We have allowed for these facts as described below.

Our standard range checks are based on Table 2. If a function evaluation does not involve an iterative solution then these checks are implemented as discussed above. If the function evaluation requires use of an iterative solution technique, then the range checking must be temporarily disabled during this step. To do this, we use a simple integer variable, check_limits, (in VB a Property Get/Let is used to represent this variable in order to permit its automatic initialization on the first call of Property Get check_limits) that is initialized in the routine Constants.0 and is accessible from all relevant routines within the library. When this variable is 1, the checks listed in Table 2 are executed; when it is less than 1, only very basic checks are done to determine whether the inputs and outputs are physically plausible, in particular checking for positive values of absolute salinity, temperature, pressure or density.

Normally, check_limits will be initialized to 1 to enable range checking. On entering each function (or subroutine), the input variables are then checked to determine if they are within the required limits; if they are not, then the function value is set to the ErrorReturn value and the function is immediately exited. If the inputs are valid, then the function is evaluated and subsequently checked to determine if the result produced is within the allowed range; if not then the function value is reset to the ErrorReturn value and execution continues at the line following the function call. It is left to the user to check whether or not an ErrorReturn value has been produced by the code.

When an iterative solver is required, the range checking is temporarily disabled by setting check_limits = check_limits - 1 before calling the iterative routine. On
exit from the solver, the value of check_limits is reset to check_limits + 1. Thus when the first iterative routine is encountered, check_limits is reduced to 0 to disable detailed range checking and if no embedded iterative routines are encountered within the solver then check_limits is reset to 1 on exit so that detailed range checking is re-enabled. Immediately after this is done, the output from the iterative solver is checked to determine if it falls in the allowable range. If it does not, then the results of the iterative solver are set to the ErrorReturn value and execution continues. Note that iterative solvers are implemented as subroutines and all outputs determined by the subroutine are set to the ErrorReturn value when any returned value is found to be out of range.

The above procedure automatically deals with embedded iterative solvers, which sometimes occur in the library. In this case, check_limits is reduced to 0 on entry to the first solver and again reduced by one for each additional embedded solver. On exit from each iterative solver, check_limits is increased by 1 so that on return from the first iterative routine that was called, check_limits will be returned to a value of 1 (provided that 1 was the initial value) and range checking will be re-initiated as required. If an invalid result was returned from any of the embedded solvers, then the results obtained by the entire procedure are set to the ErrorReturn value.

The above implementation of range checking also permits the user to easily turn this feature “off” in order to save computation time or to explore results beyond the permitted range of evaluation. This is achieved simply by initializing check_limits to 0 in FORTRAN, or setting initial_check_limits.value = 0 in VB, in the module Constants_0. When this is done, inputs to function calls are still checked to determine if they are physically meaningful (e.g., absolute salinity, temperature, pressure and density must not be negative), but the more detailed range-checks listed in Table 2 are not executed. Note that the less stringent range checking is normally performed even during the iterative solution procedure, but it has been disabled in a few exceptional cases where temporary violations occur that are irrelevant for the final result.
It may also be required to turn off the range checking only temporarily in some application code, such as testing the effect of the checking. To achieve this in the user’s code, the statement `check_limits = check_limits - 1` before calling the particular code will suspend the range checking until the statement `check_limits = check_limits + 1` returns the checking to its regular mode.

6 Summary

This paper discusses the organization, contents and usage of a library of routines made available to assist users to make use of the new TEOS-10 standard for representing the thermodynamics and equation of state of seawater. Although it is closely associated with the TEOS-10 standard, it is not endorsed by any official body nor can it be guaranteed to be free of errors. Nevertheless, it is hoped that it will be found useful.

The basic structure of the library is discussed in Sects. 2 and 3 and practical applications in different environments are discussed in Sect. 4. The valid ranges over which the library routines may be applied are discussed in Sect. 5.

Two different computation methods are implemented in the library, the usual immediate computation in which the required calculation is done “on demand” based on specified input arguments, and “state-engine” two-step computations. In the case of calculations involving a single phase, there are generally very few possibilities for the specification of input parameters, e.g., temperature and pressure or entropy and pressure. The library provides these properties as functions depending on specified parameter combinations; these functions execute “on demand” the necessary chain of procedures to extract the requested property from the Primary Standard, without storing intermediate results. In contrast, properties of phase equilibria depend on inputs with fewer degrees of freedom but with more possible input combinations. For example, the freezing point of water which occurs at the equilibrium between water and ice, may be determined if just one property is specified, but this property may be the pressure or the temperature or the entropy, etc. Once any one of these properties is specified the
equilibrium state can be completely determined, including each of the other possible input properties. Properties such as the latent heat or the density difference associated with the phase transition are then available for all possible input quantities. Rather than implementing $n_1 \times n_2$ different functions for $n_2$ properties computed from $n_1$ different groups of independent variables, the “state engine” method needs only $n_1 + n_2$ routines, $n_1$ different procedures to set the internal equilibrium state corresponding to the chosen group of specified variables, and $n_2$ procedures to read out the requested property. Here the internal equilibrium state is computed, if necessary, from the Primary Standard in the first step, and stored away to be used in subsequent calculations of equilibrium properties in the second step. Once the equilibrium state is set, it remains in effect for an unlimited number of such calculations until a new equilibrium state is determined. If a new output property is required, the addition of one routine is sufficient to make it available from any of the $n_1$ implemented combination of inputs. If another group of independent variables needs to be used, the addition of one routine provides access to each of the $n_2$ previously implemented properties as functions of the new input parameter group.

A procedure has been implemented to return ErrorReturn (specified as the number 0.999999999E99) when a library routine is called with arguments outside of specified bounds or if it returns values outside of specified bounds. This procedure is straightforward for single-state calculations but is complicated by the need for iterative solutions and by the restriction to intersecting validity ranges when equilibrium calculations are involved. It is important to realize that parameter and function values being within the bounds used in this procedure are necessary conditions for the validity of results rather than sufficient conditions. The relations between the specified necessary conditions and the sufficient conditions identified in the foundational publications are discussed in Sect. 5 and illustrated in Figs. 6–9. In general, the necessary conditions have been specified as close to the sufficient conditions as possible using simple cuboid domains without excluding valid calculations. If required, the user may accept results only in a reduced domain by modifying the specified validity boundaries
in the module Constants_0. Range-checking may also be completely disabled by setting check_limits = 0 in FORTRAN, or initial_check_limits_value = 0 in VB, in the library file Constants_0.

The checks used to detect and correct errors in the library are discussed in the Appendix, including an introduction to auxiliary routines made available to users to easily test the fidelity of local implementations.

Tables S1 through S42 (http://www.ocean-sci-discuss.net/7/649/2010/osd-7-649-2010-supplement.zip) are provided in the supplement to assist the user in finding and using the library routines that perform the desired functions. Table S1 provides a concise listing that the user can easily scan for routines relevant to the problem under consideration. Once a routine of interest is identified in Table S1, the identifier preceding the module name at the top of the table cell indicates the supplementary table to consult for additional information on the routines in this module. This supplementary table gives information on how each routine is called, what it returns and what it is based on. If further information is required, it may be obtained from Part 1 in the sections identified in the captions of the tables in the supplement.

Appendix A

Numerical checks and check values

Libraries such as that presented here are subject to coding errors that must be eliminated. Below, we discuss some of the checks used in an attempt to eliminate errors from the routines included in the library. We encourage users to report any remaining errors they may detect to one of the corresponding authors.

The following checks were applied to verify consistency with previously published information and to verify consistency between the Visual Basic and FORTRAN versions of the library.
1. The numbers produced by the library routines were verified to be consistent with the table values given in Feistel et al. (2008). The module OS2008_5 contains several routines (Table A1) that can be called by users to easily verify consistency of the results produced by local implementations with the results tabulated there.

2. For the level 1 Air routines, results were verified against the original data tabulated in Lemmon et al. (2000) using the routine chk_Lemmon_etal_2000. This check requires the use of the original reference-state coefficients and is discussed in detail in Feistel et al. (2010b, c) where check values that include additional digits are provided for both dry-air and moist-air results. The latter check values can be accessed by calling the routine chk_IAPWS10_table(nn), where nn=13, 14 or 15 to obtain results corresponding to the three routines listed in Table A1. The names of these routines anticipate the appearance of an IAPWS Guideline in 2010 (currently under review) on the representation of moist air in the presence of seawater and ice. The results provided in IAPWS (2010) are preferred over those of Lemmon et al. (2000) as numerical check values due to the increased precision of the tabulated values.

3. The routine chk_IAPWS97_table(nn), nn=5 or 15 can be used to access the two routines associated with IAPWS-IF97 listed in Table A1. They are included to provide checks on our partial implementation of IAPWS-IF97, the industrial formulation of the IAPWS-95 routines (IAPWS, 2007). In particular, they verify the Gibbs functions for liquid water and water vapour. This routine was only used in the SIA library to provide initial values for some iterative solution procedures. Similarly, the routine chk_IAPWS09_table6 (Table A1) provides the ability to easily compare the results computed by library routines with those listed in Table 6 of IAPWS (2009c) dealing with the Gibbs function formulation of Feistel (2003). This formulation is the basis of the routines in Table S39 of the supplement (http://www.ocean-sci-discuss.net/7/649/2010/osd-7-649-2010-supplement.zip) and is used extensively in the GSW_library module (Table S41).
4. In a few cases where straightforward, but tedious analytical calculations were done, simple finite difference checks were used to verify the final results. In particular, each of the relations listed in Tables 1 and 2 of Feistel (2008) were verified for a grid of input values and the analytical derivatives of the virial coefficients were similarly verified using straightforward finite difference checks.

5. In the cases of unit conversions, conversions between different representations of salinity and calculations of quantities that are determined for a specified reference level, moved there with fixed entropy and salinity for seawater or fixed entropy and air fraction for humid air, inverse functions can be used to verify library results. Several such checks are reflected in the recorded check values and numerous others were considered on an ad hoc basis. Many more checks using inverse function are possible as time permits and appropriate inverse functions are developed.

6. Results corresponding to each function contained in the library were created for common input parameter values using both Visual Basic and FORTRAN versions of the library. Results for each function were determined on a grid that included several values for each input parameter, extending over the full cuboid ranges discussed in Sect. 5. Results were stored in separate files for each routine and auxiliary Matlab routines were then used to compare the stored outputs obtained from the different language versions. Any differences between the results obtained from the two different code versions that exceeded those expected due to round-off errors (see Feistel et al., 2008) were identified and the causes of these differences were eliminated. In some cases, checks of values on the validity boundaries had to be avoided since round-off errors could cause values from one language to appear in the valid range while values from the other language appear in the invalid range. This is not a problem in real applications.

To provide users with checks that they are using the library routines correctly or that independent implementations or modifications are consistent, check values are included...
as comment lines in each routine. These check values simply give the correct outputs corresponding to a particular choice (or choices) of input parameter values.

The routines listed in Table A2 are included to allow the user to easily check that a local implementation of the library is providing correct results. When these routines are executed, they produce outputs showing what routine was called with what parameters plus the results produced by the implementation being tested together with the corresponding check values as they appear in the library routines. Note that the check values included in the library are the values produced by one particular version of the code determined on one particular platform. In our experience, these results typically agree with those determined on other platforms or using other programming languages to 12 digits for the results involving a single phase and to 10 or more digits for results involving equilibria between phases. To provide an indication of the expected level of agreement between results obtained on different platforms and using different languages, we have inserted a vertical bar (|) at the first location where values of the results rounded to 12 significant figures obtained from implementations in Visual Basic and FORTRAN differ; the comparisons were done using the default settings for error tolerances. Where no vertical bar appears, results agree to at least 12 significant figures.

Acknowledgements. The authors thank Trevor McDougall for helpful comments and discussions during the course of this work. They thank Malte Thoma and Hugh Pumphrey for assistance with FORTRAN makefiles. This paper is an outcome of the SCOR/IAPSO Working Group 127 on Thermodynamics and Equation of State of Seawater. DW acknowledges partial support from the Department of Fisheries and Oceans’ Center for Ocean Model Development for Applications.

References


Feistel, R.: A Gibbs Function for Seawater Thermodynamics for \(-6\) to \(80\,^\circ\text{C}\) and Salinity up to \(120\,\text{g/kg}\), Deep-Sea Res. I, 55, 1639–1671, 2008.


Harvey, A. H. and Huang, P. H.: First-Principles Calculation of the Air–Water Second Virial


IAPWS: Guideline on an Equation of State for Humid Air in Contact with Seawater and Ice, Consistent with the IAPWS Formulation 2008 for the Thermodynamic Properties of Sea-water. The International Association for the Properties of Water and Steam, Niagara Falls, Canada, 18–23 July 2010, in preparation, 2010.


Lemmon, E. W., Jacobsen, R. T., Penoncello, S. G., and Friend, D. G.: Thermodynamic Prop-


**Table 1.** Summary of prefixes used on procedure names to indicate the substance, phase or special purpose that they are associated with.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>air_</td>
<td>dry air, moist air or water vapour</td>
</tr>
<tr>
<td>aux_</td>
<td>private auxiliary routines used internally within the library</td>
</tr>
<tr>
<td>dry_</td>
<td>air with zero water vapour</td>
</tr>
<tr>
<td>fit_</td>
<td>fits to the results of more precise routines</td>
</tr>
<tr>
<td>flu_</td>
<td>fluid phase of pure water, vapour or liquid</td>
</tr>
<tr>
<td>gsw_</td>
<td>SIA routines that mimic those in the Gibbs SeaWater library</td>
</tr>
<tr>
<td>ice_</td>
<td>pure water, solid phase</td>
</tr>
<tr>
<td>liq_</td>
<td>pure water, liquid phase</td>
</tr>
<tr>
<td>sal_</td>
<td>seawater, liquid phase, saline part</td>
</tr>
<tr>
<td>sea_</td>
<td>seawater, liquid phase</td>
</tr>
<tr>
<td>vap_</td>
<td>pure water, gaseous phase</td>
</tr>
<tr>
<td>–</td>
<td>phase-independent variables</td>
</tr>
</tbody>
</table>
Table 2. Definitions of the primary functions used in the determination of all other functions in the library. Maximum ranges given are necessary conditions for validity. The relations to the sufficient conditions established by the foundational publications are discussed in the main text. The letters $f$ and $g$ are used to indicate Helmholtz and Gibbs functions, respectively. The letters $l$, $m$ and $n$ indicate the order of derivatives with respect to $S$, $T$, $P$ or $\rho$.

<table>
<thead>
<tr>
<th>Function call</th>
<th>Maximum range</th>
<th>Mathematical formula</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>From module Flu.1: This module implements the Helmholtz potential of fluid water and its first and second partial derivatives with respect to temperature and density as defined in IAPWS-95.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$flu_{f ; si}$: Helmholtz potential of liquid water and vapour</td>
<td>$t=T/K$, $d=\rho/(\text{kg m}^{-3})$, $m$, $n \geq 0$, $m+n \leq 2$, $50 \leq t \leq 1273$, $0 &lt; d \leq 1200$</td>
<td>$\frac{\partial^{m+n}}{\partial t^m \partial d^n} F(T, \rho)$</td>
<td>$\text{J m}^{3n} \text{K}^m \text{kg}^{-n+1}$</td>
</tr>
<tr>
<td>From module Ice.1: This module implements the Gibbs potential of hexagonal ice I and its first and second partial derivatives with respect to temperature and pressure as defined in IAPWS-06.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ice_{g ; si}$: Gibbs potential of ice Ih</td>
<td>$t=T/K$, $p=P/\text{Pa}$, $m$, $n \geq 0$, $m+n \leq 2$, $0 \leq t \leq 273.16$, $0 \leq p \leq 2 \times 10^8$</td>
<td>$\frac{\partial^{m+n}}{\partial t^m \partial p^n} g^i(T, P)$</td>
<td>$\text{J}^{-1} \text{m}^{m-n} \text{K}^n \text{kg}^{-m}$</td>
</tr>
<tr>
<td>From module Sal.1: This module implements the coefficients of the salinity expansion of the saline part of the Gibbs potential of seawater and its first and second partial derivatives with respect to temperature and pressure as defined in IAPWS-08.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$sal_{g ; term ; si}$: Salinity expansion term of the Gibbs potential of seawater</td>
<td>$i=1...7$, $t=T/K$, $p=P/\text{Pa}$, $m$, $n \geq 0$, $m+n \leq 2$, $262 \leq t &lt; 353$, $100 \leq p \leq 1 \times 10^8$</td>
<td>$g^s = g_1(T) S_A \ln S_A$, $g^s = \sum_{i=2}^{7} g_i(T, P) S_A^{i/2}$</td>
<td>$\text{J}^{-1} \text{m}^{m-n} \text{K}^n \text{kg}^{-m}$</td>
</tr>
</tbody>
</table>
Table 2. Continued.

<table>
<thead>
<tr>
<th>Function call</th>
<th>Maximum range</th>
<th>Mathematical formula</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>From module Sal₂: This module implements the thermodynamic properties of the saline part of seawater, computed from the saline part of the Gibbs potential of seawater and its first and second partial derivatives with respect to salinity, temperature and pressure as defined in IAPWS-08.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sal₉₆₁₂ (l, m, n, s, t, p)</td>
<td>l, m, n≥0</td>
<td>( \frac{g^{\alpha \beta \gamma}}{S_A \partial \alpha \partial \beta \partial \gamma} g^S(S_A, T, P) )</td>
<td>Jmol⁻¹K³kg⁻¹</td>
</tr>
<tr>
<td>t=T/K</td>
<td>0≤s≤0.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p=P/Pa</td>
<td>262≤t≤353</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100≤p≤1E+8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>From module Air₁: This module implements the Helmholtz potential of dry air and its first and second partial derivatives with respect to temperature and density as defined in Lemmon et al. (2000), plus the second cross-virial coefficient of Harvey and Huang (2007) and the third cross-virial coefficients of Hyland and Wexler (1983), each with their first and second derivatives with respect to temperature.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dry₉₆₁₂ (m, n, t, d)</td>
<td>m, n≥0</td>
<td>( \frac{\partial^2 f^A}{\partial T \partial \rho} f^A(T, \rho) )</td>
<td>Jmol⁻¹K³kg⁻¹</td>
</tr>
<tr>
<td>t=T/K</td>
<td>m+n≤2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d=ρ/(kg m⁻³)</td>
<td>60≤t≤873</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0&lt;d≤1035</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>From module Air₂: This module implements the air-water and water-water cross-virial coefficients as functions of temperature.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>air₀₆₃₂ (n, t)</td>
<td>0≤n≤2</td>
<td>( \frac{\partial^2 C_{aww}}{\partial T \partial \rho} C_{aww}(T) )</td>
<td>m³mol⁻²K⁻¹</td>
</tr>
<tr>
<td>t=T/K</td>
<td>100≤t≤2000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>air₀₆₅₂ (n, t)</td>
<td>0≤n≤2</td>
<td>( \frac{\partial^2 \rho}{\partial T \partial \rho} B_{aww}(T) )</td>
<td>m³mol⁻²K⁻¹</td>
</tr>
<tr>
<td>t=T/K</td>
<td>193≤t≤493</td>
<td></td>
<td></td>
</tr>
<tr>
<td>air₀₆₇₂ (n, t)</td>
<td>0≤n≤2</td>
<td>( \frac{\partial^2 C_{aw}}{\partial T \partial \rho} C_{aww}(T) )</td>
<td>m³mol⁻²K⁻¹</td>
</tr>
<tr>
<td>t=T/K</td>
<td>173≤t≤473</td>
<td></td>
<td></td>
</tr>
<tr>
<td>air₀₆₉₂ (n, t)</td>
<td>0≤n≤2</td>
<td>( \frac{\partial^2 C_{aw}}{\partial T \partial \rho} C_{aww}(T) )</td>
<td>m³mol⁻²K⁻¹</td>
</tr>
<tr>
<td>t=T/K</td>
<td>173≤t≤473</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D. G. Wright et al.

Thermodynamic potentials of water, vapour, ice, seawater and air – Part 2

OSD
7, 649–708, 2010
Table 2. Continued.

<table>
<thead>
<tr>
<th>Function call</th>
<th>Maximum range</th>
<th>Mathematical formula</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>air_f_mix_si(l, m, n, a, t, d)</td>
<td>0≤l, m, n</td>
<td>0≤a≤1</td>
<td>$\frac{\partial^{l+m+n} f_{\text{mix}}}{\partial a \partial t \partial d}$</td>
</tr>
<tr>
<td></td>
<td>l+m+n≤2</td>
<td>193≤t≤473</td>
<td>0&lt;d≤100$^b$</td>
</tr>
<tr>
<td>air_f_si(l, m, n, a, t, d)</td>
<td>0≤l, m, n</td>
<td>0≤a≤1</td>
<td>$\frac{\partial^{l+m+n} f_{\text{si}}}{\partial a \partial t \partial d}$</td>
</tr>
<tr>
<td></td>
<td>l+m+n≤2</td>
<td>60≤t≤873$^c$</td>
<td>0&lt;d≤1036$^c$</td>
</tr>
</tbody>
</table>

$^a$ Note that an extension is implemented that reduces by default the lower bound for validity from the official value of 130 K to 50 K (IAPWS, 2008b; Feistel et al., 2010b). Values above 130 K are not affected.

$^b$ The upper limit of 100 kg m$^{-3}$ is a conservative value established in Feistel et al. (2010c) and in Part 1, and corresponds approximately to the upper pressure limit of 5 MPa given by Hyland and Wexler (1983).

$^c$ Note that the temperature and density ranges specified for humid air are the maximum ranges applicable in the limit $A \rightarrow 1$. When water vapour is present, this range is reduced to the smaller range required for the validity of air_f_mix.
**Table A1.** Routines for comparison of the results from a local implementation with published check values. The functions are implemented in the modules where the function values are most naturally calculated. Several of the tables in module OS2008.5 also appear in IAPWS Releases. In these cases, the same tables may be obtained from the two different library routines (indicated by two different module names grouped together in the first column of this table to emphasize the connection). In Visual Basic, evaluating any of these functions will return formatted ASCII strings that may be printed to a window or saved to a file. In FORTRAN, the results are written to standard output which may be redirected to any desired file. “OS2008” refers to Feistel et al. (2008).

<table>
<thead>
<tr>
<th>Module location</th>
<th>Public function call</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS2008.5</td>
<td>chk_OS2008_Table2</td>
<td>OS2008, Table 2</td>
</tr>
<tr>
<td>OS2008.5</td>
<td>chk_OS2008_Table3</td>
<td>OS2008, Table 3</td>
</tr>
<tr>
<td>OS2008.5</td>
<td>chk_OS2008_TableA1</td>
<td>OS2008, Table A1</td>
</tr>
<tr>
<td>Flu_1</td>
<td>chk_IAPWS95_Table6</td>
<td>IAPWS-95, Table 6</td>
</tr>
<tr>
<td>OS2008.5</td>
<td>chk_OS2008_TableA2</td>
<td>OS2008, Table A2</td>
</tr>
<tr>
<td>Flu_1</td>
<td>chk_IAPWS95_Table7</td>
<td>IAPWS-95, Table 7</td>
</tr>
<tr>
<td>OS2008.5</td>
<td>chk_OS2008_TableA3</td>
<td>OS2008, Table A3</td>
</tr>
<tr>
<td>Liq_Vap_4</td>
<td>chk_IAPWS95_Table8</td>
<td>IAPWS-95, Table 8</td>
</tr>
<tr>
<td>OS2008.5</td>
<td>chk_OS2008_TableA4</td>
<td>OS2008, Table A4</td>
</tr>
<tr>
<td>Ice_1</td>
<td>chk_IAPWS06_Table6</td>
<td>IAPWS-06, Table 6</td>
</tr>
<tr>
<td>OS2008.5</td>
<td>chk_OS2008_TableA5</td>
<td>IAPWS-08, Table 8a</td>
</tr>
<tr>
<td>Sea_3a</td>
<td>chk_IAPWS08_Table8a</td>
<td>OS2008, Table A5</td>
</tr>
<tr>
<td>OS2008.5</td>
<td>chk_OS2008_TableA6</td>
<td>OS2008, Table A6</td>
</tr>
<tr>
<td>Sea_3a</td>
<td>chk_IAPWS08_Table8b</td>
<td>IAPWS-08, Table 8b</td>
</tr>
<tr>
<td>OS2008.5</td>
<td>chk_OS2008_TableA7</td>
<td>OS2008, Table A7</td>
</tr>
<tr>
<td>Sea_3a</td>
<td>chk_IAPWS08_Table8c</td>
<td>IAPWS-08, Table 8c</td>
</tr>
<tr>
<td>OS2008.5</td>
<td>chk_OS2008_TableA8</td>
<td>OS2008, Table A8</td>
</tr>
<tr>
<td>Liq_F03_5</td>
<td>chk_IAPWS09_Table6</td>
<td>IAPWS-09, Table 6</td>
</tr>
<tr>
<td>Air_2</td>
<td>chk_IAPWS10_Table13</td>
<td>IAPWS-10, Table 13</td>
</tr>
<tr>
<td>Air_2</td>
<td>chk_IAPWS10_Table14</td>
<td>IAPWS-10, Table 14</td>
</tr>
<tr>
<td>Air_2</td>
<td>chk_IAPWS10_Table15</td>
<td>IAPWS-10, Table 15</td>
</tr>
<tr>
<td>Air_3b</td>
<td>chk_Lemmon_etal_2000(0)</td>
<td>Lemmon et al. (2000), Table A2 (from the Helmholtz function)</td>
</tr>
<tr>
<td>Air_3b</td>
<td>chk_Lemmon_etal_2000(1)</td>
<td>Lemmon et al. (2000), Table A2 (from the Gibbs function)</td>
</tr>
<tr>
<td>Flu_IF97_5</td>
<td>chk_IAPWS97_table5</td>
<td>IAPWS-IF97, Table 5</td>
</tr>
<tr>
<td>Flu_IF97_5</td>
<td>chk_IAPWS97_table15</td>
<td>IAPWS-IF97, Table 15</td>
</tr>
</tbody>
</table>
Table A2. Routines provided to easily compare results from a local implementation with those produced on the development machines.

<table>
<thead>
<tr>
<th>Values_Flu</th>
<th>Values_Sea</th>
<th>Values_Mix</th>
<th>Values_GSW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses</td>
<td>Uses</td>
<td>Uses</td>
<td>Uses</td>
</tr>
<tr>
<td>constants_0, convert_0, flu_1, flu_2, flu_3a, flu_3b</td>
<td>constants_0, convert_0, sal_1, sal_2, flu_1, flu_2, flu_3a, sea_3a, sea_3b, sea_3c, sea_3d</td>
<td>constants_0, convert_0, maths_0, air_1, air_2, flu_1, flu_2, flu_3a, flu_3b, ice_1, ice_2, sal_1, sal_2, sea_3a, sea_3b, sea_3c, sea_3d, ice_air_4a, ice_air_4b, ice_air_4c, ice_liq_4, ice_vap_4, liq_air_4a, liq_air_4b, liq_air_4c, liq_vap_4, liq_ice_air_4, sea_air_4, sea_ice_4, sea_ice_vap_4, sea_liq_4, sea_vap_4</td>
<td>constants_0, convert_0, maths_0, sal_1, sal_2, liq_f03_5, convert_5, gsw_library_5</td>
</tr>
<tr>
<td>Public Routines</td>
<td>Public Routines</td>
<td>Public Routines</td>
<td>Public Routines</td>
</tr>
<tr>
<td>flu_chk_values</td>
<td>sea_chk_values</td>
<td>mix_chk_values</td>
<td>gsw_chk_values</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Values_Ice</th>
<th>Values_Air</th>
<th>Values_Sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses</td>
<td>Uses</td>
<td>Uses</td>
</tr>
<tr>
<td>constants_0, convert_0, ice_1, ice_2</td>
<td>constants_0, convert_0, maths_0, air_1, air_2, air_3a, air_3b, air_3c</td>
<td>constants_0, convert_0, sal_1, sal_2</td>
</tr>
<tr>
<td>Public Routines</td>
<td>Public Routines</td>
<td>Public Routines</td>
</tr>
<tr>
<td>ice_chk_values</td>
<td>air_chk_values</td>
<td>sal_chk_values</td>
</tr>
</tbody>
</table>
Fig. 1. Modular structure of the SIA library illustrated by a selected subset of library routines. Library levels are indicated to the left of the diagram and boxes represent library modules, each of which contains routines that are conceptually related to each other. The Primary Standard consists of implementations of IAPWS formulations for fluid water, ice, sea salt and dry air. All other thermodynamic properties are then derived from this standard using rigorous thermodynamic relations. The numerical solutions of these relations are systematically implemented in a modular framework that consistently determines all quantities from the Primary Standard without the need for separate correlation equations for individual properties. The current implementation of the library is quite extensive (see Table S1 of the supplement: http://www.ocean-sci-discuss.net/7/649/2010/osd-7-649-2010-supplement.zip) and may be easily extended to provide any desired degree of completeness.
Fig. 2. Calling the SIA Helmholtz function of fluid water, flu_f_si, from within a spreadsheet application.
Fig. 4.2: Running the VB code within the Immediate Window of the Visual Basic Editor available in MS Excel.

Fig. 3. Running the VB code within the Immediate Window of the Visual Basic Editor available in MS Excel.

An example of an executable SIA routine in VB6 is supplied with the library. It is called TEOS10_SIA.exe and can be used “as is” on a personal computer. Copying the executable to your PC and double-clicking on it should produce the screen shown as Fig. 4.3. Although this executable is of limited practical value, it provides a simple point-and-click demonstration form. The pull-down menu Check offers the re-computation of various check tables published in the articles and documents behind TEOS-10, discussed in the Appendix. The menu Examples displays several lists of selected properties computed from the SIA library functions, such as the seawater properties shown in Fig. 4.4. The menu File offers to save ASCII files of the check tables, of the examples and of all check values given in the code as comments in the particular library functions. These files are named SIA_CheckTables.txt, SIA_Examples.txt and SIA_CheckValues.txt, respectively, and are written to the folder where the program is executed (the user must ensure that writing is permitted in this folder).
**Fig. 4.** The user interface window of the executable SIA library program of TEOS-10 included with the Visual Basic download. This executable was compiled in Visual Basic and provides access to check tables, check values and a few basic quantities through the pull-down tabs at the top of the window.
The ranges of validity and accuracy of the potential functions for fluid water and ice are described in the IAPWS Releases (2009a, b). For sea water, both the potential functions for pure water and for salt must be valid (IAPWS, 2008a, 2009a). The validity range for the Helmholtz potential for dry air is discussed by Lemmon et al. (2000). For moist air, consisting of dry air plus water vapour, both the potential functions for water vapour (IAPWS, 2009a) and for dry air (Lemmon et al., 2000) must be valid if both components are present. In addition, if both components are present, then both the cross-virial coefficients and the truncated virial expansion must be valid. The validity ranges of the cross-virial coefficients are discussed by Hyland and Wexler (1983) and the results for the two component air-water interaction are updated by Harvey and Huang (2007). Results presented in Part I of this publication indicate that when both components are present in significant amounts, the total density of the moist air mixture must be less than 100 kg m$^{-3}$ for the truncated virial expansion to be valid. However, if the mass fraction of one of the two components approaches zero, the virial coefficients play no significant role and in this case, the validity range is determined by the conditions required for the validity of the potential function of the remaining component.

The documents mentioned above are adopted to form the foundation for the library routines discussed here and they should be consulted for detailed information on sufficient conditions for applicability. Here, we consider necessary (i.e., less restrictive) conditions for the validity of the library routines and briefly discuss how they relate to the sufficient (i.e., most restrictive) conditions discussed in these seminal publications. Here we use simple necessary conditions for validity as sufficient conditions for the formal invalidity of the formulations. The necessary conditions for validity of the primary potential functions are listed in Table 5.1. These are the conditions used to determine whether or not a function call will result in an ErrorReturn value being returned to indicate an invalid result. It is important to note that these conditions are not exhaustive and other factors may also affect the validity of the results.
Fig. 6. (a) The validity region of the IAPWS-95 Helmholtz potential for fluid water is shown as the bold lines; the temperature bounds are connected by thinner lines indicating a selection of isobars. The heavy dotted lines indicate the range checks implemented in the library code. (b) A magnified view of the small region corresponding to the standard oceanographic (“Neptunian”) range. TP: triple point gas-liquid-solid, CP: critical point. The bold lines show the density and temperature bounds for validity of IAPWS-95 in this region. The deviation of the vapour-pressure line from the 101 325 Pa isobar in the liquid region is below the graphical resolution. Freezing-point lowering occurs with the addition of sea salt. To deal with this effect in the case of seawater, the extension of the pure water properties into the metastable liquid region just above the line marked “Freezing Point Lowering” is required. The heavy dotted line to the right shows the only range check boundary implemented in the library routines that falls in the range shown here (figure adapted from Fig. 1a, b of Part 1).
Fig. 7. (a) Range of validity (solid bold curves) of the Gibbs function of ice Ih according to IAPWS-06 and Feistel and Wagner (2007). The heavy dotted curves indicate the boundaries beyond which the library code returns an ErrorReturn value to indicate that the validity range has been violated. No warning is provided when ice_g_si is accessed outside of the solid bold lines but still within the heavy dotted lines. (b) An expanded view showing the region very near the triple point, TP. Note that the melting temperature of 273.152519 K is actually significantly reduced from 273.16 K at the standard atmospheric pressure of 101 325 Pa (panel a is adapted from Fig. 2 of Part 1).
Fig. 5.3 Range of validity of the IAPWS-08 Gibbs function of seawater. Region A: oceanographic standard range, B: extension to higher salinities, C: extensions to higher temperature and salinity values at atmospheric pressure, D: zero-salinity limit, E: extrapolation into the metastable region, F: hot concentrates; at temperature and salinity values beyond the plane indicated by heavy dotted lines, results that are influenced by the pressure dependence of the Gibbs function are particularly uncertain. (Figure adapted from Fig. 3 of Part I)

The range of validity of the Helmholtz function for dry air (dry_f_si in the library) is determined by the parameter ranges specified by Lemmon et al. (2000). For pressure, it extends from 0 to 70 MPa and for temperature from 60 to 873 K. The maximum air density in this region is 1035.8 kg m\(^{-3}\). If no water vapour is present and range-checking is enabled, the library routines return physically meaningful values for temperatures between 60 and 873 K and for densities from 0 to 1035 kg m\(^{-3}\) as indicated by the heavy dotted lines in Fig. 5.4.

Note that a phase transition to liquid air occurs below the curved boundary indicated in the lower right portion of Fig. 5.4. The bounds implemented in the library do not exclude this region and function evaluations for this region will return results obtained for air with liquefied components. Although the formulation of Lemmon et al. (2000) covers liquid air, too, this...
Fig. 9. (a) Validity ranges of the Helmholtz function for humid air overlain on the phase boundary diagram for dry air. The region indicated by the outer heavy dotted lines gives the range boundaries implemented in the library for dry air (A precisely equal to one). Below the dewpoint curve in the lower right portion of the diagram, dry air condenses. This region below 150 K at high density is irrelevant to oceanographic or meteorological applications but is not excluded by our range checks. To consider humid air, virial coefficients are required. The validity range in temperature of the limiting third virial coefficients is indicated by the bold dotted lines enclosing the words “VIRIAL COEFFICIENTS” in panels (a) and (b). Comparisons with experimental data show good agreement with saturated partial pressures up to a density of 100 kg m⁻³. This density is chosen as the upper bound on the validity range for humid air. (b) The validity range of the Helmholtz function for humid air overlain on the phase boundary diagram for fluid water. Evidently, the upper validity boundary on density is determined by saturation, i.e., the partial transition of vapour to ice or liquid water (the two phase region) rather than the boundary for the validity of the cross-virial coefficients.