Package ‘gsw’

August 9, 2017

Version 1.0-5
Date 2017-08-08
Title Gibbs Sea Water Functions

Copyright Original algorithms and 'Matlab'/C library (c) 2015-2017
WG127 SCOR/IAPSO (Scientific Committee on Oceanic Research /
International Association for the Physical Sciences of the
Oceans, Working Group 127); C wrapper code and R code (c)
2015-2017 Dan Kelley and Clark Richards

Maintainer Dan Kelley <dan.kelley@dal.ca>

Depends R (>= 2.15), testthat

Suggests knitr,

BugReports https://github.com/TEOS-10/GSW-C/issues

Description Provides an interface to the Gibbs 'SeaWater' ('TEOS-10') C
library, version 3.05-4 (commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec', dated 2017-08-
07,
available at <https://github.com/TEOS-10/GSW-C>, which stems from
'Matlab' and other code written by members of Working Group 127 of
'SCOR'/IAPSO' (Scientific Committee on Oceanic Research /
International Association for the Physical Sciences of the Oceans).


License GPL (>= 2) | file LICENSE

LazyLoad yes

LazyData no

Encoding UTF-8

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation yes

Author Dan Kelley [aut, cre, cph] (C wrapper plus R code, tests, and
documentation),
Clark Richards [aut, cph] (C wrapper plus R code, tests, and
documentation),
WG127 SCOR/IAPSO [aut, cph] (Original 'Matlab' and derived code)

Repository CRAN
Date/Publication 2017-08-09 21:05:24 UTC

R topics documented:

argfix ................................................................. 5
gsw ................................................................. 6
gsw_adiabatic_lapse_rate_from_CT ................................. 7
gsw_adiabatic_lapse_rate_ice ...................................... 8
gsw_alpha ............................................................ 9
gsw_alpha_on_beta .................................................. 10
gsw_alpha_wrt_t_exact ............................................. 11
gsw_alpha_wrt_t_ice ................................................ 12
gsw_beta ............................................................. 13
gsw_beta_const_t_exact ............................................ 14
gsw_cabbeling ....................................................... 16
gsw_chem_potential_water_ice ..................................... 17
gsw_chem_potential_water_t_exact ................................. 18
gsw_cp_ice .......................................................... 19
gsw_cp_t_exact ....................................................... 20
gsw_CT_first_derivatives ......................................... 21
gsw_CT_first_derivatives_wrt_t_exact ......................... 22
gsw_CT_freezing .................................................... 23
gsw_CT_freezing_first_derivatives ............................... 24
gsw_CT_freezing_first_derivatives_poly ....................... 25
gsw_CT_freezing_poly ............................................. 26
gsw_CT_from_enthalpy ............................................... 27
gsw_CT_from_entropy ............................................... 28
gsw_CT_from_pt ..................................................... 29
gsw_CT_from_rho ................................................... 30
gsw_CT_from_t ...................................................... 31
gsw_CT_maxdensity ................................................. 32
gsw_CT_second_derivatives ....................................... 33
gsw_C_from_SP ..................................................... 34
gsw_deltaSA_from_SP ............................................... 35
gsw_dilution_coefficient_t_exact ............................... 36
gsw_dynamic_enthalpy .............................................. 37
gsw_enthalpy ....................................................... 38
gsw_enthalpy_CT_exact ............................................ 39
gsw_enthalpy_diff .................................................. 40
gsw_enthalpy_first_derivatives .................................. 42
gsw_enthalpy_first_derivatives_CT_exact ....................... 43
gsw_enthalpy_ice ................................................... 44
gsw_enthalpy_second_derivatives ................................. 45
gsw_enthalpy_second_derivatives_CT_exact ..................... 46
gswe_nthalpy_t_exact .......................................................... 47
gswe_entropy_first_derivatives ....................................... 48
gswe_entropy_from_pt ...................................................... 49
gswe_entropy_from_t ...................................................... 50
gswe_entropy_ice .......................................................... 51
gswe_entropy_second_derivatives .................................. 52
gswe_Fdelta ............................................................... 54
gswe_frazil_properties .................................................. 55
gswe_frazil_properties_potential .................................... 56
gswe_frazil_properties_potential_poly ............................. 57
gswe_frazil_ratios_adiabatic ........................................ 58
gswe_frazil_ratios_adiabatic_poly .................................. 59
gswe_geo_strf_dyn_height ............................................... 60
gswe_geo_strf_dyn_height_pc ......................................... 61
gswe_gibbs ................................................................. 62
gswe_gibbs_ice ............................................................ 64
gswe_grav ................................................................. 65
gswe_Helmholtz_energy_ice ............................................ 66
gswe_ice_fraction_to_freeze_seawater ............................ 67
gswe_internal_energy .................................................... 68
gswe_internal_energy_ice ............................................... 69
gswe_IPV_vs_fNsquared_ratio ......................................... 70
gswe_kappa ................................................................. 71
gswe_kappa_const_t_ice ................................................. 72
gswe_kappa_ice ........................................................... 73
gswe_kappa_t_exact ....................................................... 74
gswe_latentheat_evap_CT ............................................... 75
gswe_latentheat_evap_t ................................................. 76
gswe_latentheat_melting ............................................... 77
gswe_melting_ice_equilibrium_SA_CT_ratio ....................... 78
gswe_melting_ice_equilibrium_SA_CT_ratio_poly ................ 79
gswe_melting_ice_into_seawater ..................................... 80
gswe_melting_ice_SA_CT_ratio ....................................... 81
gswe_melting_ice_SA_CT_ratio_poly ................................ 82
gswe_melting_seaice_into_seawater ................................. 83
Nsquared ................................................................. 84
gswe_pot_enthalpy_from_pt_ice ....................................... 85
gswe_pot_enthalpy_from_pt_ice_poly ................................ 86
gswe_pot_enthalpy_ice_freezing ..................................... 87
gswe_pot_enthalpy_ice_freezing_first_derivatives .............. 88
gswe_pot_enthalpy_ice_freezing_first_derivatives_poly ...... 89
gswe_pot_enthalpy_ice_freezing_poly ............................... 91
gswe_pot_rho_t_exact .................................................... 92
gswe_pressure_coefficient_ice ....................................... 93
gswe_pressure_freezing_CT ............................................. 94
gswe_pt0_from_t .......................................................... 95
gswe_pt0_from_t_ice ...................................................... 96
gswe_pt_first_derivatives ............................................. 97
<table>
<thead>
<tr>
<th>R topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>gsw_pt_from_CT</td>
<td>98</td>
</tr>
<tr>
<td>gsw_pt_from_entropy</td>
<td>99</td>
</tr>
<tr>
<td>gsw_pt_from_pot_enthalpy_ice</td>
<td>100</td>
</tr>
<tr>
<td>gsw_pt_from_pot_enthalpy_ice_poly</td>
<td>101</td>
</tr>
<tr>
<td>gsw_pt_from_t</td>
<td>102</td>
</tr>
<tr>
<td>gsw_pt_from_t_ice</td>
<td>103</td>
</tr>
<tr>
<td>gsw_pt_second_derivatives</td>
<td>104</td>
</tr>
<tr>
<td>gsw_p_from_z</td>
<td>105</td>
</tr>
<tr>
<td>gsw_rho</td>
<td>106</td>
</tr>
<tr>
<td>gsw_rho_alpha_beta</td>
<td>107</td>
</tr>
<tr>
<td>gsw_rho_first_derivatives</td>
<td>108</td>
</tr>
<tr>
<td>gsw_rho_first_derivatives_wrt_enthalpy</td>
<td>109</td>
</tr>
<tr>
<td>gsw_rho_ice</td>
<td>110</td>
</tr>
<tr>
<td>gsw_rho_second_derivatives</td>
<td>111</td>
</tr>
<tr>
<td>gsw_rho_second_derivatives_wrt_enthalpy</td>
<td>113</td>
</tr>
<tr>
<td>gsw_rho_t_exact</td>
<td>114</td>
</tr>
<tr>
<td>gsw_SAAR</td>
<td>115</td>
</tr>
<tr>
<td>gsw_SA_freezing_from_CT</td>
<td>116</td>
</tr>
<tr>
<td>gsw_SA_freezing_from_CT_poly</td>
<td>117</td>
</tr>
<tr>
<td>gsw_SA_freezing_from_t</td>
<td>118</td>
</tr>
<tr>
<td>gsw_SA_freezing_from_t_poly</td>
<td>119</td>
</tr>
<tr>
<td>gsw_SA_from_rho</td>
<td>120</td>
</tr>
<tr>
<td>gsw_SA_from_SP</td>
<td>121</td>
</tr>
<tr>
<td>gsw_SA_from_SP_Baltic</td>
<td>123</td>
</tr>
<tr>
<td>gsw_SA_from_Sstar</td>
<td>124</td>
</tr>
<tr>
<td>gsw_seaice_fraction_to_freeze_seawater</td>
<td>125</td>
</tr>
<tr>
<td>gsw_sigma0</td>
<td>126</td>
</tr>
<tr>
<td>gsw_sigma1</td>
<td>127</td>
</tr>
<tr>
<td>gsw_sigma2</td>
<td>128</td>
</tr>
<tr>
<td>gsw_sigma3</td>
<td>129</td>
</tr>
<tr>
<td>gsw_sigma4</td>
<td>130</td>
</tr>
<tr>
<td>gsw_sound_speed</td>
<td>131</td>
</tr>
<tr>
<td>gsw_sound_speed_ice</td>
<td>132</td>
</tr>
<tr>
<td>gsw_sound_speed_t_exact</td>
<td>133</td>
</tr>
<tr>
<td>gsw_specvol</td>
<td>134</td>
</tr>
<tr>
<td>gsw_specvol_alpha_beta</td>
<td>135</td>
</tr>
<tr>
<td>gsw_specvol_anom_standard</td>
<td>136</td>
</tr>
<tr>
<td>gsw_specvol_first_derivatives</td>
<td>137</td>
</tr>
<tr>
<td>gsw_specvol_first_derivatives_wrt_enthalpy</td>
<td>139</td>
</tr>
<tr>
<td>gsw_specvol_ice</td>
<td>140</td>
</tr>
<tr>
<td>gsw_specvol_second_derivatives</td>
<td>141</td>
</tr>
<tr>
<td>gsw_specvol_second_derivatives_wrt_enthalpy</td>
<td>142</td>
</tr>
<tr>
<td>gsw_specvol_t_exact</td>
<td>144</td>
</tr>
<tr>
<td>gsw_spiciness0</td>
<td>145</td>
</tr>
<tr>
<td>gsw_spiciness1</td>
<td>146</td>
</tr>
<tr>
<td>gsw_spiciness2</td>
<td>147</td>
</tr>
<tr>
<td>gsw_SP_from_C</td>
<td>148</td>
</tr>
<tr>
<td>gsw_SP_from_SA</td>
<td>149</td>
</tr>
</tbody>
</table>
Reshape list elements to match that of the first element

Description

This is mainly used within gsw, to ensure that arguments sent to the C functions are of equal length. This is a convenience, for processing data that often have this condition. For example, a CTD profile is likely to have many values for SP, t, and p, but just a single value for each of longitude and latitude. It is important to call argfix() to handle such cases, because otherwise the underlying C code will be looking past the end of the vectors storing longitude and latitude, which can yield odd results or even segmentation faults.

Usage

argfix(list)

Arguments

list A list of elements, typically arguments that will be used in GSW functions.

Value

A list with all elements of same shape (length or dimension).
Description

Provides an R interface to the TEOS-10 / GSW (Gibbs Sea Water) library, partly for use by the oce package (see http://dankelley.github.io/oce) and partly for general use. It is assumed that users are familiar with the science and methodology of GSW, and that the package vignette (obtained by typing vignette("gsw") in an R window) provides enough orientation to get users started with the gsw functions.

Details

gsw was developed using open-source methodologies, on the GitHub site (https://github.com/TEOS-10/GSW-R), which is part of a set of sites dedicated to GSW formulations in various languages.

The gsw system is to link R functions with the C version of the TEOS-10 library. The R function names are chosen to match those of the Matlab version of GSW, and the function arguments also match with one exception: in gsw, longitude and latitude are indicated with their full names, whereas in Matlab they are indicated with long and lat; since R permits abbreviated function arguments, the shortened names can be used in gsw as well.

The documentation for the gsw functions focuses mainly on the arguments and return values, relying on links to the TEOS-10 webpages for details.


Each function is tested during the building of the package, which means that results are guaranteed to match those of the equivalent Matlab functions to at least 8 digits.

A significant difference from the Matlab case is in the inspection of the dimensions of arguments. The Matlab library has rules for expanding some arguments to match others. For example, if Practical Salinity is a matrix and pressure is a single value, then that single pressure is used throughout a calculation of Absolute Salinity. This convenience is only partly mimicked in the present package. Since the underlying C code works on vectors, the R functions in gsw start by transforming the arguments accordingly. This involves using rep on each argument to get something with length matching the first argument, and, after the computation is complete, converting the return value into a matrix, if the first argument was a matrix. There are some exceptions to this, however. For example, gsw_SA_from_SP and similar functions can handle the case in which the SA argument is a matrix and longitude and latitude are vectors sized to match. This can be handy with gridded datasets. However, the careful analyst will probably prefer to avoid this and other conveniences, supplying properly-matched arguments from the outset.
**Adiabatic Lapse Rate**

**Description**

Note that the unit is K/Pa; multiply by 1e4 to get the more useful K/mb.

**Usage**

```r
 gsw_adiabatic_lapse_rate_from_CT(SA, CT, p)
```

**Arguments**

- `SA` Absolute Salinity [ g/kg ]
- `CT` Conservative Temperature [ degC ]
- `p` sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe4f5'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

adiabatic lapse rate (note unconventional unit) [ K/Pa ]

**References**


**Examples**

```r
 SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
 CT <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4836)
 p <- c(10, 50, 125, 250, 600, 1000)
 lr <- gsw_adiabatic_lapse_rate_from_CT(SA, CT, p)
 expect_equal(lr*1e7, c(0.240199646230069, 0.238457486976761, 0.203635157319712,
 0.1198295668599790, 0.100052760967308, 0.087773070307283))
```
Adiabatic Lapse Rate of Ice

description

Note that the unit is K/Pa; multiply by 1e4 to get the more useful K/mb.

Usage

gsw_adiabatic_lapse_rate_ice(t, p)

Arguments

t in-situ temperature (ITS-90) [degC]
p sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe')., which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

adiabatic lapse rate (note unconventional unit) [K/Pa]

References


Examples

```
t <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
p <- c(10, 50, 125, 250, 600, 1000)
lr <- gsw_adiabatic_lapse_rate_ice(t, p)
expect_equal(lr*1e7, c(0.218777853913651, 0.216559115188599, 0.216867659957613, 0.216988337914416, 0.2171827042780, 0.21810055874804))
```
**gsw_alpha**

| gsw_alpha | Thermal expansion coefficient with respect to Conservative Temperature |

**Description**

Thermal expansion coefficient with respect to Conservative Temperature, using the 75-term equation for specific volume.

**Usage**

`gsw_alpha(SA, CT, p)`

**Arguments**

- `SA`: Absolute Salinity [ g/kg ]
- `CT`: Conservative Temperature [ degC ]
- `p`: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

**Value**

thermal expansion coefficient with respect to Conservative Temperature [ 1/K ]

**References**


**See Also**

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_beta_const_t_exact, gsw_beta, gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_first_derivatives, gsw_rho_ice, gsw_rho_t_exact, gsw_rho, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol
Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
alpha <- gsw_alpha(SA, CT, p)
expect_equal(alpha*1e3, c(0.324464211877393, 0.3226100946880523, 0.281335035247435,
0.173529986885424, 0.146898108553385, 0.130265123640082))
```

gsw_alpha_on_beta  

Thermal expansion coefficient over haline contraction coefficient

Description

Thermal expansion coefficient over haline contraction coefficient, using the 75-term equation for specific volume.

Usage

gsw_alpha_on_beta(SA, CT, p)

Arguments

- **SA**: Absolute Salinity [g/kg]
- **CT**: Conservative Temperature [degC]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe9'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

Value

ratio of thermal expansion coefficient to haline contraction coefficient [(g/kg)/K]

References

### gsw_alpha_wrt_t_exact

**Thermal expansion coefficient with respect to in-situ temperature**

#### Description

Thermal expansion coefficient with respect to in-situ temperature.

#### Usage

```r
gsw_alpha_wrt_t_exact(SA, t, p)
```

#### Arguments

- **SA**: Absolute Salinity [g/kg]
- **t**: in-situ temperature (ITS-90) [degC]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

#### Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bffac'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

#### Value

thermal expansion coefficient with respect to in-situ temperature [1/K]
gsw_alpha_wrt_t_ice

References


See Also

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_first_derivatives, gsw_rho_ice, gsw_rho_t_exact, gsw_rho, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol

Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
t <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
alpha_wrt_t_exact <- gsw_alpha_wrt_t_exact(SA, t, p)
expect_equal(alpha_wrt_t_exact*1e3, c(0.325601747227247, 0.323440083851267, 0.281413883319329, 0.172825692975230, 0.145569941503599, 0.128362986933288))

gsw_alpha_wrt_t_ice

Ice Thermal Expansion Coefficient with Respect to in-situ Temperature

Description

Thermal expansion coefficient of ice, with respect to in-situ temperature.

Usage

gsw_alpha_wrt_t_ice(t, p)

Arguments

t in-situ temperature (ITS-90) [ degC ]
p sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit ’5b4d959e5403119e972f3e863f63e67fa4f5bfec’), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.
**gsw_beta**

**Value**

thermal expansion coefficient with respect to in-situ temperature [ 1/K ]

**References**


**See Also**

Other things related to density: `gsw_CT_from_rho`, `gsw_CT_maxdensity`, `gsw_SA_from_rho`, `gsw_alpha_on_beta`, `gsw_alpha_wrt_t_exact`, `gsw_alpha`, `gsw_beta_const_t_exact`, `gsw_pot_rho_t_exact`, `gsw_rho_alpha_beta`, `gsw_rho_first_derivatives_wrt_enthalpy`, `gsw_rho_first_derivatives`, `gsw_rho_ice`, `gsw_rho_t_exact`, `gsw_rho`, `gsw_sigma0`, `gsw_sigma1`, `gsw_sigma2`, `gsw_sigma3`, `gsw_sigma4`, `gsw_specvol_alpha_beta`, `gsw_specvol_anom_standard`, `gsw_specvol_ice`, `gsw_specvol_t_exact`, `gsw_specvol`

**Examples**

```r
  t <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
  p <- c(  10,   50,   125,   250,   600,   1000)
  alpha <- gsw_alpha_wrt_t_ice(t, p)
  expect_equal(alpha*1e3, c(0.154472408751279, 0.153041866100900, 0.153232698269327, 
                           0.153297634665747, 0.153387461617896, 0.15393895452558))
```

| gsw_beta | Haline contraction coefficient at constant Conservative Temperature |

**Description**

Haline contraction coefficient with respect to Conservative Temperature, using the 75-term equation for specific volume.

**Usage**

`gsw_beta(SA, CT, p)`

**Arguments**

- **SA**: Absolute Salinity [ g/kg ]
- **CT**: Conservative Temperature [ degC ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
gsw_beta_const_t_exact

Details
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e9723f3e863f63e67fa4ff5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value
Haline contraction coefficient at constant Conservative Temperature [ kg/g ]

References

See Also
Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_pot_rho_t_exact, gsw_rh0_alpha_beta, gsw_rh0_first_derivatives_wrt_enthalpy, gsw_rh0_first_derivatives, gsw_rh0_ice, gsw_rh0_t_exact, gsw_rh0, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol

Examples
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4836)
p  <- c( 10,  50, 125, 250, 600, 1000)
beta <- gsw_beta(SA,CT,p)
expect_equal(beta, 1e-3*c(0.717521909550091, 0.717657376442386, 0.726169785748549, 0.750420924314564, 0.754903052075032, 0.756841573481865))

gsw_beta_const_t_exact

Haline contraction coefficient at constant in-situ temperature

Description
Haline contraction coefficient at constant in-situ temperature.

Usage
gsw_beta_const_t_exact(SA, t, p)
Arguments

SA  Absolute Salinity [ g/kg ]

\( t \)  in-situ temperature (ITS-90) [ degC ]

\( p \)  sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Haline contraction coefficient at constant in-situ temperature [ kg/g ]

References


See Also

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta, gsw_pot_rho_t_exact, gsw_rhoe_alpha_beta, gsw_rhoe_first_derivatives_wrt_enthalpy, gsw_rhoe_first_derivatives, gsw_rhoe_ice, gsw_rhoe_t_exact, gsw_rhoe, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol

Examples

```
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
t <- c( 28.7856, 28.4329, 22.8103, 10.2600,  6.8863,  4.4036)
p <- c( 10,  50, 125,  250,  500, 1000)
b <- gsw_beta_const_t_exact(SA, t, p)
expect_equal(b*1e3, c(0.731120837010429, 0.731071779078011, 0.736019128913071, 0.753810501711847, 0.757259405338257, 0.758649268096996))
```
Description

Cabbeling coefficient (75-term equation)

Usage

gsw_cabbeling(SA, CT, p)

Arguments

SA        Absolute Salinity [ g/kg ]
CT        Conservative Temperature [ degC ]
p        sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e5403f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Cabbeling coefficient with respect to Conservative Temperature [ 1/(K^2) ]

References


Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
p <- c(10, 50, 125, 250, 600, 1000)
cabbeling <- gsw_cabbeling(SA, CT, p)
equal(cabbeling[, 1] * 1e4, c(0.086645721047423, 0.086837829466794, 0.092525582052438, 0.108884336975401, 0.112971197222338, 0.115483896148927))
**gsw_chem_potential_water_ice**

*Chemical Potential of Ice*

**Description**

Chemical Potential of Ice

**Usage**

\[
gsw_chem_potential_water_ice(t, p)
\]

**Arguments**

- `t` in-situ temperature (ITS-90) [ degC ]
- `p` sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

chemical potential [ J/kg ]

**References**


**See Also**

Other things related to chemical potential: gsw_chem_potential_water_t_exact

**Examples**

```r
  t <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
  p <- c(10, 50, 125, 250, 600, 1000)
  pot <- gsw_chem_potential_water_ice(t, p)
  expect_equal(pot/10, c(-1.340648365149857, -1.644921413491445, -1.480991678890353, -1.272436055728805, -0.711589477199393, 0.045575390357792))
```
Description

Chemical Potential of Water in Seawater

Usage

gsw_chem_potential_water_t_exact(SA, t, p)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>t</td>
<td>in-situ temperature (ITS-90) [ degC ]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfecc'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

chemical potential [ J/kg ]

References


See Also

Other things related to chemical potential: gsw_chem_potential_water_ice

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
t <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
pot <- gsw_chem_potential_water_t_exact(SA, t, p)
expect_equal(pot, c(-8.545551146284534, -8.008805548342105, -5.103980139874876, -8.634067782745442, 3.335566803473286, 7.555434445971858))
```
**gsw_cp_ice**

Specific heat to ice

---

**Description**

Specific heat of ice

**Usage**

gsw_cp_ice(t, p)

**Arguments**

- **t**
  - in-situ temperature (ITS-90) [degC]
- **p**
  - sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e540319e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

specific heat [J/(K*kg)]

**References**


**Examples**

t <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
p <- c(10, 50, 125, 250, 600, 1000)
cp <- gsw_cp_ice(t, p)
gsw_cp_t_exact

Isobaric heat capacity

Description
Isobaric heat capacity

Usage

gsw_cp_t_exact(SA, t, p)

Arguments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/\text{kg} ]</td>
</tr>
<tr>
<td>t</td>
<td>in-situ temperature (ITS-90) [ \text{degC} ]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [ \text{dbar} ], i.e. absolute pressure [ \text{dbar} ] minus 10.1325 \text{dbar}</td>
</tr>
</tbody>
</table>

Details
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value
heat capacity \[ \text{J/(kg*K)} \]

References


Examples

```
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
t <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
cp_t_exact <- gsw_cp_t_exact(SA, t, p)
equal_equal(cp_t_exact/1e3, c(4.0028880283958537, 4.000900283927373, 3.995546468894633, 3.985076769021378, 3.973593843482723, 3.9601840084786622))
```
**gsw_CT_first_derivatives**

*First Derivatives of Conservative Temperature*

**Description**

First Derivatives of Conservative Temperature

**Usage**

```
gsw_CT_first_derivatives(SA, pt)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>pt</td>
<td>potential temperature (ITS-90) [ degC ]</td>
</tr>
</tbody>
</table>

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfecc'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

A list containing `CT_SA` [ K/(g/kg) ], the derivative of Conservative Temperature with respect to Absolute Salinity, and `CT_pt` [ unitless ], the derivative of Conservative Temperature with respect to potential temperature.

**References**


**Examples**

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
pt <- c(28.7832, 28.4209, 22.7850, 10.2305, 6.8292, 4.3245)
r <- gsw_CT_first_derivatives(SA, pt)
expect_equal(r$CT_SA, c(-0.04198109287806, -0.041558140199508, -0.034739209004865, -0.01871103772892, -0.014075941811725, -0.010571716552295))
expect_equal(r$CT_pt, c(1.002814937296636, 1.002554817053239, 1.001645140295163, 1.000003771100520, 0.999716359504731, 0.999474326580093))
```
**gsw_CT_first_derivatives_wrt_t_exact**

*Derivatives of Conservative Temperature with Respect to or at Constant in-situ Temperature*

**Description**

Derivatives of Conservative Temperature with Respect to or at Constant in-situ Temperature

**Usage**

```r
gsw_CT_first_derivatives_wrt_t_exact(SA, t, p)
```

**Arguments**

- `SA`: Absolute Salinity [ g/kg ]
- `t`: in-situ temperature (ITS-90) [ degC ]
- `p`: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

A list containing:
- `CT_SA_wrt_t` [ K/(g/kg) ], the derivative of Conservative Temperature with respect to Absolute Salinity at constant temperature and pressure,
- `CT_t_wrt_t` [ unitless ], the derivative of Conservative Temperature with respect to temperature at constant Absolute Salinity and pressure,
- `CT_p_wrt_t`, the derivative of Conservative Temperature with respect to pressure at constant Absolute Salinity and temperature.

**References**


**Examples**

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
t <- c( 28.7856, 28.4329, 22.8103, 10.2600,  6.8863,  4.4036)
p <- c( 10,  50, 125,  250,  600, 1000)
r <- gsw_CT_first_derivatives_wrt_t_exact(SA, t, p)
equal_equal(r$CT_SA_wrt_t, c(-0.041988694538987, -0.0415965490888952, -0.034853545749326, -0.019067140454607, -0.015016439826591, -0.012233725491373))
```
### gsw_CT_freezing

**Conservative Temperature of Freezing Seawater**

#### Description

Conservative Temperature of Freezing Seawater

#### Usage

```r
gsw_CT_freezing(SA, p, saturation_fraction = 1)
```

#### Arguments

- **SA**: Absolute Salinity [ g/kg ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **saturation_fraction**: saturation fraction of dissolved air in seawater

#### Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959c54031f9e972f3e863f63e67fa4f5bfe'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

#### Value

Conservative Temperature at freezing of seawater [ degC ].

#### References


#### Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c(10, 50, 125, 250, 600, 1000)
saturation_fraction <- 1
CT <- gsw_CT_freezing(SA, p, saturation_fraction)
expect_equal(CT, c(-1.899683776424096, -1.948791867869104, -2.006240664432488, 
                   -2.092357761318778, -2.359300831770506, -2.677162675412748))
```
**gsw_CT_freezing_first_derivatives**

*First Derivatives of Conservative Temperature for Freezing Water*

**Description**

First Derivatives of Conservative Temperature for Freezing Water

**Usage**

```r
gsw_CT_freezing_first_derivatives(SA, p, saturation_fraction = 1)
```

**Arguments**

- **SA**
  - Absolute Salinity [g/kg]
- **p**
  - sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **saturation_fraction**
  - fraction of air in water [unitless]

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe5'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

A list containing `CTfreezing_SA` [K/(g/kg)], the derivative of Conservative Temperature with respect to Absolute Salinity at constant potential temperature, and `CTfreezing_p` [unitless], the derivative of Conservative Temperature with respect to pressure at constant Absolute Salinity.

**References**


**Examples**

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c(10, 50, 125, 250, 600, 1000)
saturation_fraction <- c(1, 0.8, 0.6, 0.5, 0.4, 0)
r <- gsw_CT_freezing_first_derivatives(SA, p, saturation_fraction)
equal(r$CTfreezing_SA, c(-0.058193253897272, -0.058265158334170, -0.058345661671901, -0.058373842446463, -0.058534544740846, -0.058738846361252))
equal(r$CTfreezing_p, c(-0.7653089432684, -0.766942996466485, -0.76892679988284, -0.774561011527902, -0.787769143040504, -0.802771548245855))
```
First Derivatives of Conservative Temperature for Freezing Water (Polynomial version)

Usage

gsw_CT_freezing_first_derivatives_poly(SA, p, saturation_fraction = 1)

Arguments

SA Absolute Salinity [ g/kg ]
p sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
saturation_fraction fraction of air in water [unitless]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e5403199e972f3e863f63e67fa4f5b6ec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing CT_freezing_SA [ K/(g/kg) ], the derivative of Conservative Temperature with respect to Absolute Salinity at constant potential temperature, and CT_freezing_p [ unitless ], the derivative of Conservative Temperature with respect to pressure at constant Absolute Salinity.

References


Examples

SA <- c( 34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c( 10, 50, 125, 250, 600, 1000)
saturation_fraction <- c( 1, 0.8, 0.6, 0.5, 0.4, 0)
r <- gsw_CT_freezing_first_derivatives_poly(SA, p, saturation_fraction)
expect_equal(r$CT_freezing_SA, c(-0.058191181082769, -0.058263310660779, -0.058343573188907, -0.058370514075271, -0.058528023214462, -0.058722959729433))
expect_equal(r$CT_freezing_p, c(-0.765690732336706, -0.767310677213890, -0.770224214219328, -0.774843488962665, -0.787930403016584, -0.802821704643775))
gsw_CT_freezing_poly  

**Description**

Conservative Temperature Freezing Point (Polynomial version)

**Usage**

```
gsw_CT_freezing_poly(SA, p, saturation_fraction = 1)
```

**Arguments**

- **SA**
  - Absolute Salinity [ g/kg ]
- **p**
  - sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **saturation_fraction**
  - saturation fraction of dissolved air in seawater

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

Conservative Temperature at freezing of seawater [ degC ].

**References**


**Examples**

```R
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c( 10, 50, 125, 250, 600, 1000)
saturation_fraction <- 1
CT_freezing <- gsw_CT_freezing_poly(SA, p, saturation_fraction)
expect_equal(CT_freezing, c(-1.899683776424096, -1.940791867869104, -2.006240664432488, -2.092357761318778, -2.359300831770506, -2.677162675412748))
```
Conservative Temperature from Enthalpy

Usage

gsw_CT_from_enthalpy(sa, h, p)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [g/kg]</td>
</tr>
<tr>
<td>h</td>
<td>Specific enthalpy [J/kg]</td>
</tr>
<tr>
<td>p</td>
<td>Sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Conservative Temperature [degC]

References


See Also

Other things related to enthalpy: gsw_dynamic_enthalpy, gsw_enthalpy_CT_exact, gsw_enthalpy_diff, gsw_enthalpy_first_derivatives_CT_exact, gsw_enthalpy_first_derivatives, gsw_enthalpy_ice, gsw_enthalpy_f_exact, gsw_enthalpy, gsw_frazil_properties_potential_poly, gsw_frazil_properties_potential, gsw_pot_enthalpy_from_pt_ice_poly, gsw_pot_enthalpy_from_pt_ice, gsw_pot_enthalpy_ice_freezing_poly, gsw_pot_enthalpy_ice_freezing, gsw_pt_from_pot_enthalpy_ice_poly, gsw_pt_from_pot_enthalpy_ice, gsw_specvol_first_derivatives_wrt_enthalpy, gsw_specvol_first_derivatives
Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
h <- c(1.15103e5, 1.14014e5, 0.92180e5, 0.43255e5, 0.33087e5, 0.26970e5)
p <- c(10, 50, 125, 250, 600, 1000)
pt <- c(28.7832, 28.4209, 22.7850, 10.2305, 6.8292, 4.3245)
CT <- gsw_CT_from_entropy(SA, h, p)
expect_equal(CT, c(28.809854569021972, 28.439026483379287, 22.786196534098817,
10.26106994920777, 6.827159682675204, 4.323428660306681))

Description

Conservative Temperature from Entropy

Usage

gsw_CT_from_entropy(SA, entropy)

Arguments

SA Absolute Salinity [ g/kg ]
entropy specific entropy [ J/(degC*kg) ]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e86f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Conservative Temperature [ degC ]

References


See Also

Other things related to entropy: gsw_entropy_first_derivatives, gsw_entropy_from_pt, gsw_entropy_from_t, gsw_entropy_ice, gsw_pt_from_entropy
Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
entropy <- c(400.3892, 395.4378, 319.8668, 146.7910, 98.6471, 62.7919)
CT <- gsw_CT_from_entropy(SA, entropy)
expect_equal(CT, c(28.809902787278070, 28.439199226786918, 22.786199266954270, 10.226197672488652, 6.827196739780282, 4.32360294546461))
```

Description

Conservative Temperature from Potential Temperature

Usage

```r
gsw_CT_from_pt(SA, pt)
```

Arguments

- **SA**: Absolute Salinity [g/kg]
- **pt**: potential temperature (ITS-90) [degC]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

Value

Conservative Temperature [degC]

References


Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
pt <- c(28.7832, 28.4209, 22.7850, 10.2305, 6.8292, 4.3245)
CT <- gsw_CT_from_pt(SA, pt)
expect_equal(CT, c(28.809923015982883, 28.439144260767169, 22.786246608464264, 10.226165605435785, 6.8271834176433142, 4.323565182322069))
```
gsw_CT_from_rho

Conservative Temperature from Density, Absolute Salinity and Pressure

Description
Conservative Temperature from Density, Absolute Salinity and Pressure

Usage

gsw_CT_from_rh0(rho, SA, p)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rho</td>
<td>seawater density [ kg/m^3 ]</td>
</tr>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
</tbody>
</table>

Details
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e8b3f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value
A list containing two estimates of Conservative Temperature: CT and CT_multiple, each in [ degC ].

References
http://www.teos-10.org/pubs/gsw/html/gsw_CT_from_rh0.html

See Also
Other things related to density: gsw_CT_maxdensity, gsw_SA_from_rh0, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, gsw_pot_rh0_t_exact, gsw_rh0_alpha_beta, gsw_rh0_first_derivatives_wrt_enthalpy, gsw_rh0_first_derivative, gsw_rh0_ice, gsw_rh0_t_exact, gsw_rh0, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol
Examples

```r
gsw_CT_from_t
```

 rho <- c(1021.8484, 1022.2647, 1024.4207, 1027.7841, 1029.8287, 1031.9916)
 SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
 p <- c(10, 50, 125, 250, 600, 1000)
 r <- gsw_CT_from_rho(rho, SA, p)
 expect_equal(r$CT, c(28.784377302226968, 28.432402127485858, 22.808745445250068,
 10.260169334807866, 6.887336649146716, 4.404594162282834))
```

---

**gsw_CT_from_t** *Convert from temperature to conservative temperature*

**Description**

Convert from temperature to conservative temperature

**Usage**

```r
gsw_CT_from_t(SA, t, p)
```

**Arguments**

- **SA** Absolute Salinity [ g/kg ]
- **t** in-situ temperature (ITS-90) [ degC ]
- **p** sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

Conservative Temperature [ degC ]

**References**

Examples
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
t <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
CT <- gsw_CT_from_t(SA, t, p)
expect_equal(CT, c(28.809919826700281, 28.439227816091140, 22.786175836078498,
                  10.226189266620782, 6.82721363479988, 4.323575748610455))

\[
\text{gsw\_CT\_maxdensity} \quad \text{Conservative Temperature at Maximum Density}
\]

Description
Conservative Temperature at Maximum Density

Usage
\[
gsw\_CT\_maxdensity(SA, p)
\]

Arguments
SA  Absolute Salinity [ g/kg ]
p  sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5b5cfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value
Conservative Temperature [ degC ]

References

See Also
Other things related to density: gsw\_CT\_from\_rho, gsw\_SA\_from\_rho, gsw\_alpha\_on\_beta, gsw\_alpha\_wrt\_t\_exact, gsw\_alpha\_wrt\_t\_ice, gsw\_alpha, gsw\_beta\_const\_t\_exact, gsw\_beta, gsw\_pot\_rho\_t\_exact, gsw\_rho\_alpha\_beta, gsw\_rho\_first\_derivatives\_wrt\_enthalpy, gsw\_rho\_first\_derivatives, gsw\_rho\_ice, gsw\_rho\_t\_exact, gsw\_rho, gsw\_sigma0, gsw\_sigma1, gsw\_sigma2, gsw\_sigma3, gsw\_sigma4, gsw\_specvol\_alpha\_beta, gsw\_specvol\_anom\_standard, gsw\_specvol\_ice, gsw\_specvol\_t\_exact, gsw\_specvol
Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c( 10,  50,  125,  250,  600, 1000)
CT <- gsw_CT_maxdensity(SA, p)
expect_equal(CT, c(-3.731407240089855, -3.861137427731664, -4.060390602245942,
                  -4.306222571955388, -5.089240667106197, -6.028034316992341))

Description

Second Derivatives of Conservative Temperature

Usage

gsw_CT_second_derivatives(SA, pt)

Arguments

SA Absolute Salinity [ g/kg ]
pt potential temperature (ITS-90) [ degC ]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f163e67fa4f5f6e'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing CT_SA_SA [ K/(g/kg)^2 ], the second derivative of Conservative Temperature with respect to Absolute Salinity at constant potential temperature, and CT_SA_pt [ 1/(g/kg) ], the derivative of Conservative Temperature with respect to potential temperature and Absolute Salinity, and CT_pt_pt [ 1/degC ], the second derivative of Conservative Temperature with respect to potential temperature.

References

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
pt <- c(28.7832, 28.4209, 22.7850, 10.2305, 6.8292, 4.3245)
r <- gsw_CT_second_derivatives(SA, pt)
expect_equal(r$CT_SA_SA/1e-3, c(-0.000718502077964, -0.062065324400873, -0.084017055354742,
-0.148446950120131, -0.171270836500246, -0.189920754900116))
expect_equal(r$CT_SA_pt, c(-0.0011974150000869, -0.00119309530139, -0.001226523296082,
-0.00135896286481, -0.001380492698572, -0.001417751669135))
expect_equal(r$CT_pt_pt/1e-3, c(0.123812754427146, 0.1244662808871271, 0.140829458783443,
0.140646803448166, 0.113684095615077, 0.082286843477998))
```

**Description**

Electrical conductivity (in mS/cm) from Practical Salinity. To convert the return value to conductivity ratio, divide by 42.9140 (the value of conductivity at S=35, T=15, and p=0).

**Usage**

```r
gsw_C_from_SP(SP, t, p)
```

**Arguments**

- `SP`: Practical Salinity (PSS-78) [unitless]
- `t`: in-situ temperature (ITS-90) [degC]
- `p`: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e5403119e972f3e863f6367fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

Electrical conductivity [mS/cm]

**References**

See Also

Other things related to salinity: `gsw_SA_from_SP_Baltic`, `gsw_SA_from_SP`, `gsw_SA_from_Sstar`, `gsw_SP_from_C`, `gsw_SP_from_SA`, `gsw_SP_from_SK`, `gsw_SP_from_SR`, `gsw_SP_from_Sstar`, `gsw_SR_from_SP`, `gsw_Sstar_from_SA`, `gsw_Sstar_from_SP`, `gsw_deltaSA_from_SP`

Other things related to conductivity: `gsw_SP_from_C`

Examples

```r
SP <- c(34.5487, 34.7275, 34.8605, 34.6810, 34.5680, 34.5600)
t <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
C <- gsw_C_from_SP(SP, t, p)
expect_equal(C, c(56.412599581571186, 56.31618560269953, 50.670369333973944, 38.134518936104350, 35.056577637635257, 32.986550607990118))
```

deltaSA Absolute Salinity Anomaly from Practical Salinity

Description

Absolute Salinity Anomaly from Practical Salinity

Usage

```r
gsw_deltaSA_from_SP(SP, p, longitude, latitude)
```

Arguments

- **SP**: Practical Salinity (PSS-78) [ unitless ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **longitude**: longitude in decimal degrees, positive to the east of Greenwich. (This is called `long` in the TEOS-10 Matlab code.)
- **latitude**: latitude in decimal degrees, positive to the north of the equator. (This is called `lat` in the TEOS-10 Matlab code.)

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

Value

`deltaSA` Absolute Salinity Anomaly [ g/kg ]
References


See Also

Other things related to salinity: gsw_C_from_SP, gsw_SA_from_SP_Baltic, gsw_SA_from_SP, gsw_SA_from_Sstar, gsw_SP_from_C, gsw_SP_from_SA, gsw_SP_from_SK, gsw_SP_from_SR, gsw_SP_from_Sstar, gsw_SR_from_SP, gsw_Sstar_from_SA, gsw_Sstar_from_SP

Examples

```r
SP = c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p = c( 10, 50, 125, 250, 600, 1000)
lat = c( 4, 4, 4, 4, 4, 4)
long = c(188, 188, 188, 188, 188, 188)
deltaSA = gsw_deltaSA_from_SP(SP, p, long, lat)
expect_equal(deltaSA, c(0.00067203365230, 0.00026836122231, 0.000665803155705,
                       0.002706154619403, 0.005652977406832, 0.009444734661606))
```

gsw_dilution_coefficient_t_exact

_Dilution coefficient_

Description

Dilution coefficient

Usage

```r
gsw_dilution_coefficient_t_exact(SA, t, p)
```

Arguments

- **SA** Absolute Salinity [ g/kg ]
- **t** in-situ temperature (ITS-90) [ degC ]
- **p** sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e5403191e972f3e863f63e67fa4f5f6c'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.
Value

dilution coefficient [ (J/kg)(kg/g) ]

References


Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
t <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
dc <- gsw_dilution_coefficient_t_exact(SA, t, p)
expect_equal(dc, c(79.140342151532040, 79.104983526833820, 77.503312816847389,
73.53562653715272, 72.483378545466564, 71.760667498673087))
```

Description

Dynamic enthalpy of seawater (75-term equation)

Usage

gsw_dynamic_enthalpy(SA, CT, p)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>CT</td>
<td>Conservative Temperature [ degC ]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

dynamic enthalpy [ J/kg ]
**gsw_enthalpy**

Specific enthalpy of seawater (75-term equation)

**Description**

Specific enthalpy of seawater (75-term equation)

**Usage**

```r
gsw_enthalpy(SA, CT, p)
```

**Arguments**

- **SA**: Absolute Salinity [g/kg]
- **CT**: Conservative Temperature [degC]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e5403119e972f3e863f63e67fa4f5bfc'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.
gsw_enthalpy_CT_exact

Value

specific enthalpy [ J/kg ]

References


See Also

Other things related to enthalpy: gsw_CT_from_enthalpy, gsw_dynamic_enthalpy, gsw_enthalpy_CT_exact, gsw_enthalpy_diff, gsw_enthalpy_first_derivatives_CT_exact, gsw_enthalpy_first_derivatives, gsw_enthalpy_ice, gsw_enthalpy_t_exact, gsw_frazil_properties_potential_poly, gsw_frazil_properties_potential, gsw_pot_enthalpy_from_pt_ice_poly, gsw_pot_enthalpy_from_pt_ice, gsw_pot_enthalpy_ice_freezing_poly, gsw_pot_enthalpy_ice_freezing, gsw_pt_from_pot_enthalpy_ice_poly, gsw_pt_from_pot_enthalpy_ice, gsw_specvol_first_derivatives_wrt_enthalpy, gsw_specvol_first_derivatives

Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
p <- c(10, 50, 125, 250, 600, 1000)

e <- gsw_enthalpy(SA, CT, p)
expect_equal(e/1e5, c(1.151031813559086, 1.140146926828028, 0.921800138366058,
0.432553713026279, 0.330871609742468, 0.269706841603465))

gsw_enthalpy_CT_exact  Seawater Specific Enthalpy in terms of Conservative Temperature

Description

Seawater Specific Enthalpy in terms of Conservative Temperature

Usage

gsw_enthalpy_CT_exact(SA, CT, p)

Arguments

SA       Absolute Salinity [ g/kg ]
CT       Conservative Temperature [ degC ]
p       sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959ec54031f9e972f3e863f63e67fa4f5bfe'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

specific enthalpy [ J/kg ]

References


See Also

Other things related to enthalpy: gsw_C_from_enthalpy, gsw_dynamic_enthalpy, gsw_enthalpy_diff, gsw_enthalpy_first_derivatives_CT_exact, gsw_enthalpy_first_derivatives, gsw_enthalpy_ice, gsw_enthalpy_t_exact, gsw_enthalpy, gsw_frazil_properties_potential_poly, gsw_frazil_properties_potential, gsw_pot_enthalpy_from_pt_ice_poly, gsw_pot_enthalpy_from_pt_ice, gsw_pot_enthalpy_ice_freezing_poly, gsw_pot_enthalpy_ice_freezing, gsw_pt_from_pot_enthalpy_ice_poly, gsw_pt_from_pot_enthalpy_ice, gsw_specvol_first_derivatives_wrt_enthalpy, gsw_specvol_first_derivatives

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8999, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
p <- c(10, 50, 125, 250, 600, 1000)
e <- gsw_enthalpy_CT_exact(SA, CT, p)
equal(e, c(1.151831813321767, 1.140146925585614, 0.921800131787836,
0.43255371231579, 0.330871615358722, 0.269706848807403))
```

```r

---

**gsw_enthalpy_diff**  
Specific Enthalpy Difference with Pressure

Description

Specific enthalpy difference [ J/kg ].

Usage

```r
gsw_enthalpy_diff(SA, CT, p_shallow, p_deep)
```
Arguments

SA  Absolute Salinity [ g/kg ]
CT  Conservative Temperature [ degC ]
p_shallow  pressure at a shallower depth [ dbar ]
p_deep  pressure at a deeper depth [ dbar ]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit ’5b4d959e54031f9e972f3e863f63e67fa4f5bfec’), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

specific enthalpy difference [ J/kg ]

References


See Also

Other things related to enthalpy: gsw_CT_from_enthalpy, gsw_dynamic_enthalpy, gsw_enthalpy_CT_exact, gsw_enthalpy_first_derivatives_CT_exact, gsw_enthalpy_first_derivatives, gsw_enthalpy_ice, gsw_enthalpy_t_exact, gsw_enthalpy, gsw_frazil_properties_potential_poly, gsw_frazil_properties_potential, gsw_pot_enthalpy_from_pt_ice_poly, gsw_pot_enthalpy_from_pt_ice, gsw_pot_enthalpy_ice_freezing_poly, gsw_pot_enthalpy_ice_freezing, gsw_pt_from_pot_enthalpy_ice_poly, gsw_pt_from_pot_enthalpy_ice, gsw_specvol_first_derivatives_wrt_enthalpy, gsw_specvol_first_derivatives

Examples

SA <- c( 34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c( 28.7856, 28.4329, 22.8103, 10.2600,  6.8863,  4.4036)
p_shallow <- c(10,  50,  125,  250,  600, 1000)
p_deep <- c( 110, 150,  225,  350,  700, 1100)
ed <- gsw_enthalpy_diff(SA, CT, p_shallow, p_deep)
gsw_enthalpy_first_derivatives

First Derivatives of Enthalpy

Description

First Derivatives of Enthalpy

Usage

gsw_enthalpy_first_derivatives(SA, CT, p)

Arguments

 SA  Absolute Salinity [ g/kg ]
 CT  Conservative Temperature [ degC ]
  p   sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

a list containing h_SA [ (J/kg)/(g/kg) ], the derivative of enthalpy wrt Absolute Salinity, and h_CT [ (J/kg)/degC ], the derivative of enthalpy wrt Conservative Temperature.

References


See Also

Other things related to enthalpy: gsw_CT_from_enthalpy, gsw_dynamic_enthalpy, gsw_enthalpy_CT_exact, gsw_enthalpy_diff, gsw_enthalpy_first_derivatives_CT_exact, gsw_enthalpy_ice, gsw_enthalpy_t_exact, gsw_enthalpy, gsw_frazil_properties_potential_poly, gsw_frazil_properties_potential, gsw_pot_enthalpy_from_pt_ice_poly, gsw_pot_enthalpy_from_pt_ice, gsw_pot_enthalpy_ice_freezing_poly, gsw_pot_enthalpy_ice_freezing, gsw_pt_from_pot_enthalpy_ice_poly, gsw_pt_from_pot_enthalpy_ice, gsw_specvol_first_derivatives_wrt_enthalpy, gsw_specvol_first_derivatives
Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
d <- gsw_enthalpy_first_derivatives_CT_exact(SA, CT, p)
equal(d$h_SA, c(-0.070223912348929, -0.351159768365102, -0.887025065692568,
-1.829602387915694, -4.423463748270238, -7.405100077558673))
equal(d$h_CT, c(3.991899705530481, 3.992025640520101, 3.992210365030743,
3.992284150250490, 3.992685389122658, 3.993014168534175))

Description

First Derivatives of Enthalpy wrt CT

Usage

gsw_enthalpy_first_derivatives_CT_exact(SA, CT, p)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>CT</td>
<td>Conservative Temperature [ degC ]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e540319e972f3e86fi63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

a list containing h_SA [ (J/kg)/(g/kg) ], the derivative of enthalpy wrt Absolute Salinity, and h_CT [ (J/kg)/degC ], the derivative of enthalpy wrt Conservative Temperature.

Bugs

The HTML documentation suggests that this function returns 3 values, but there are only 2 returned values in the C code used here (and the matlab code on which that is based). Also, the d/dSA check values given the HTML are not reproduced by the present function. This was reported on Mar 18, 2017 as https://github.com/TEOS-10/GSW-Matlab/issues/7. See https://github.com/TEOS-10/GSW-R/issues/34
**gsw_enthalpy_ice**

### Description

Specific enthalpy of ice [ J/kg ]. Note that this is a negative quantity.

### Usage

`gsw_enthalpy_ice(t, p)`

### Arguments

- **t**
  - in-situ temperature (ITS-90) [ degC ]

- **p**
  - sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

### Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfcc'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.
Value

specific enthalpy [ J/kg ]

References


See Also

Other things related to enthalpy: gsw_CT_from_enthalpy, gsw_dynamic_enthalpy, gsw_enthalpy_CT_exact, gsw_enthalpy_diff, gsw_enthalpy_first_derivatives_CT_exact, gsw_enthalpy_first_derivatives, gsw_enthalpy_t_exact, gsw_enthalpy, gsw_frazil_properties_potential_poly, gsw_frazil_properties_potential, gsw_pot_enthalpy_from_pt_ice_poly, gsw_pot_enthalpy_from_pt_ice, gsw_pot_enthalpy_ice_freezing_poly, gsw_pot_enthalpy_ice_freezing, gsw_pt_from_pot_enthalpy_ice_poly, gsw_pt_from_pot_enthalpy_ice, gsw_specvol_first_derivatives_wrt_enthalpy, gsw_specvol_first_derivatives

Examples

t <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
p <- c( 10,  50, 125,  250,  600, 1000)
se <- gsw_enthalpy_ice(t, p)
expect_equal(se/1e5, c(-3.554414597446597, -3.60338857687490, -3.58309884253586, -3.558998379233944, -3.494811024956881, -3.402784319238127))

Description

Second Derivatives of Enthalpy

Usage

gsw_enthalpy_second_derivatives(SA, CT, p)

Arguments

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>CT</td>
<td>Conservative Temperature [ degC ]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
</tbody>
</table>
Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f363e67fa4f5bf6ec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing $h_{\text{SA-SA}}$ [ (J/kg)/(g/kg)$^2$ ], the second derivative of enthalpy with respect to Absolute Salinity, $h_{\text{SA-CT}}$ [ (J/kg)/(K*g/kg) ], the derivative of enthalpy with respect to Absolute Salinity and Conservative Temperature, and $h_{\text{CT-CT}}$ [ (J/kg)/degC$^2$ ], the second derivative of enthalpy with respect to Conservative Temperature.

References


Examples

```r
SA <- c(34.7118, 34.8915, 35.8256, 34.8472, 34.7366, 34.7324)
CT <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)

r <- gsw_enthalpy_second_derivatives(SA, CT, p)
expect_equal(r$h_SA_SA, c(0.0000000922482023, 0.000404963500641, 0.00105980046742, 0.00243188963823, 0.006019611828423, 0.010225411250217))
expect_equal(r$h_SA_CT, c(0.000130004715129, 0.000653614489248, 0.001877220817849, 0.005470392103793, 0.014314756132297, 0.025195603327700))
expect_equal(r$h_CT_CT, c(0.000714303909834, 0.003584401249266, 0.009718730753139, 0.024064471995224, 0.061547884081343, 0.107493969308119))
```
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e540319e972f3e863f63e67fa4f5bfe5'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing \( h_{\text{SA-SA}} \) [ \((\text{J/kg})/(\text{g/kg})^2\) ], the second derivative of enthalpy with respect to Absolute Salinity, \( h_{\text{SA-CT}} \) [ \((\text{J/kg})/(\text{K} \cdot \text{g/kg})\) ], the derivative of enthalpy with respect to Absolute Salinity and Conservative Temperature, and \( h_{\text{CT-CT}} \) [ \((\text{J/kg})/\text{degC}^2\) ], the second derivative of enthalpy with respect to Conservative Temperature.

References


Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
r <- gsw_enthalpy_second_derivatives_CT_exact(SA, CT, p)
expect_equal(r$h_SA_SA, c(0.00082767011576, 0.000414469343141, 0.00108958017293, 0.002472193425998, 0.006103171596328, 0.018377465312463))
expect_equal(r$h_SA_CT, c(0.000130320164426, 0.000655016236924, 0.001879127443985, 0.005468695168037, 0.014315709900526, 0.025192691262061))
expect_equal(r$h_CT_CT, c(0.000714397830954, 0.000584965664329, 0.0009703337655703, 0.024044402143825, 0.061449390733344, 0.107333638394904))
```

Description

Seawater Specific Enthalpy in terms of in-situ Temperature

Usage

```r
gsw_enthalpy_t_exact(SA, t, p)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [g/kg]</td>
</tr>
<tr>
<td>t</td>
<td>in-situ temperature (ITS-90) [degC]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
</tbody>
</table>
Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959ec540319e972f3e863f53e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

specific enthalpy [ J/kg ]

References


See Also

Other things related to enthalpy: gsw_CT_from_enthalpy, gsw_dynamic_enthalpy, gsw_enthalpy_CT_exact, gsw_enthalpy_diff, gsw_enthalpy_first_derivatives_CT_exact, gsw_enthalpy_first_derivatives, gsw_enthalpy_ice, gsw_enthalpy, gsw_frazil_properties_potential_poly, gsw_frazil_properties_potential, gsw_pot_enthalpy_from_pt_ice_poly, gsw_pot_enthalpy_from_pt_ice, gsw_pot_enthalpy_freezing_poly, gsw_pot_enthalpy_freezing, gsw_pt_from_pot_enthalpy_ice_poly, gsw_pt_from_pot_enthalpy_ice, gsw_specvol_first_derivatives_wrt_enthalpy, gsw_specvol_first_derivatives

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
t <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4836)
p <- c(10, 50, 125, 250, 600, 1000)
e <- gsw_enthalpy_t_exact(SA, t, p)
expect_equal(e/1e5, c(1.151032604783763, 1.140148036012021, 0.921799209310966, 0.432553283808897, 0.330872159700175, 0.269705880448018))
```

---

 gsentropy_first_derivatives

**First Derivatives of Entropy**

Description

First Derivatives of Entropy

Usage

gsw_entropy_first_derivatives(SA, CT)
Arguments

SA  Absolute Salinity [ g/kg ]
CT  Conservative Temperature [ degC ]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

a list containing eta_SA [ (J/(kg*degC) / (g/kg)], the derivative of entropy wrt Absolute Salinity, and eta_CT [ (J/(kg*degC^2)], the derivative of entropy wrt Conservative Temperature.

References


See Also

Other things related to entropy: gsw_CT_from_entropy, gsw_entropy_from_pt, gsw_entropy_from_t, gsw_entropy_ice, gsw_pt_from_entropy

Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262,  6.8272,  4.3236)
d <- gsw_entropy_first_derivatives(SA, CT)
expect_equal(d$eta_SA, c(-0.263286800711655, -0.263977276574528, -0.255367497912925, -0.238066586439561, -0.234438268606436, -0.23282684341694))
expect_equal(d$eta_CT, c(13.221031210083824, 13.236911191313675, 13.48904628681361, 14.086599016583795, 14.257729576432077, 14.38642994564941))

gsw_entropy_from_pt

Specific Entropy into Absolute Salinity and Potential Temperature

Description

Calculates specific entropy in terms of Absolute Salinity and Potential Temperature.

Usage

gsw_entropy_from_pt(SA, pt)
**Arguments**

- **SA**: Absolute Salinity [ g/kg ]
- **pt**: potential temperature (ITS-90) [ degC ]

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

specific entropy [ J/(kg*degC) ]

**References**


**See Also**

Other things related to entropy: [gsw_CT_from_entropy](https://github.com/TEOS-10/GSW-C), [gsw_entropy_first_derivatives](https://github.com/TEOS-10/GSW-C), [gsw_entropy_from_t](https://github.com/TEOS-10/GSW-C), [gsw_entropy_ice](https://github.com/TEOS-10/GSW-C), [gsw_pt_from_entropy](https://github.com/TEOS-10/GSW-C)

**Examples**

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
pt <- c(28.7832, 28.4210, 22.7858, 10.2305, 6.8292, 4.3245)
e <- gsw_entropy_from_pt(SA, pt)
expect_equal(e/1e2, c(4.003894674443156, 3.954383994925507, 3.198674385897981,
                   1.467905482842553, 0.98649100565646, 0.627913567234252))
```

**Description**

Calculates specific entropy in terms of Absolute Salinity, in-situ temperature and pressure.

**Usage**

```r
gsw_entropy_from_t(SA, t, p)
```
Arguments

SA  Absolute Salinity [ g/kg ]
t  in-situ temperature (ITS-90) [ degC ]
p  sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

specific entropy [ J/(kg*K) ]

References


See Also

Other things related to entropy: gsw_CT_from_entropy, gsw_entropy_first_derivatives, gsw_entropy_from_pt, gsw_entropy_ice, gsw_pt_from_entropy

Examples

```
SA <- c(34.7118, 34.8915, 35.8256, 34.8472, 34.7366, 34.7324)
t <- c( 28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c( 10, 50, 125, 250, 600, 1000)
e <- gsw_entropy_ice(SA, t, p)
equal(e1e2, c(4.003894252787245, 3.954381784340642, 3.198664981986740, 1.4679088815899072, 0.986473408657975, 0.627915087346090))
```

```
gsw_entropy_ice  Entropy of ice

Description

Entropy of ice

Usage

gsw_entropy_ice(t, p)
```
Arguments

- `t`: in-situ temperature (ITS-90) [degC]
- `p`: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5f6ec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

entropy [J/(kg*degC)]

References


See Also

Other things related to entropy: gsw_CT_from_entropy, gsw_entropy_first_derivatives, gsw_entropy_from_pt, gsw_entropy_from_t, gsw_pt_from_entropy

Examples

t <- c(-10.7856, -13.4329, -12.8183, -12.2600, -10.8863, -8.4036)
p <- c(10, 50, 125, 250, 600, 1000)
e <- gsw_entropy_ice(t, p)
expect_equal(e/1e3, c(-1.303663820598987, -1.324090218294577, -1.319426394193644, -1.315402956671801, -1.305426590579231, -1.287021035328113))

---

gsw_entropy_second_derivatives

*Second Derivatives of Entropy*

Description

Second Derivatives of Entropy

Usage

gsw_entropy_second_derivatives(SA, CT)
Arguments

SA  Absolute Salinity [ g/kg ]
CT  Conservative Temperature [ degC ]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5f6c'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing eta_SA_SA [ (J/(K*kg))/(g/kg)^2 ], the second derivative of entropy with respect to Absolute Salinity, eta_SA_CT [ (J/(K*kg))/(K*g/kg) ], the derivative of entropy with respect to Absolute Salinity and Conservative Temperature, and eta_CT_CT [ (J/(K*kg))/K^2 ], the second derivative of entropy with respect to Conservative Temperature.

Bugs

As of March 27, 2017, the test values listed in “Examples” do not match values provided at the TEOS-10 website listed in “References”, but they match with values given by the Matlab code that is provided on the TEOS-10 website. It is expected that the TEOS-10 website will be updated by May 2017. As those updates to the TEOS-10 website become available, the present comment will be revised or deleted.

References


See Also

Other functions with suspicious test values on the TEOS-10 website: gsw_rho_second_derivatives_wrt_enthalpy, gsw_specvol_second_derivatives_wrt_enthalpy, gsw_t_freezing_first_derivatives_poly

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
r <- gsw_entropy_second_derivatives(SA, CT)
expect_equal(r$eta_SA_SA, c(-0.007627718929669, -0.007591969960708, -0.007528186784540, -0.007455177590576, -0.007441108287466, -0.007414368396280))
expect_equal(r$eta_SA_CT, c(-0.001833104216751, -0.001819473824306, -0.001580843823414, -0.000930111408561, -0.000717011215195, -0.000548410546838))
expect_equal(r$eta_CT_CT, c(-0.043665623731189, -0.043781336189326, -0.04550611440888, -0.049708939454018, -0.050938690879443, -0.051875017843472))
```
gsw_Fdelta  

**Ratio of Absolute to Preformed Salinity, minus 1**

**Description**

Ratio of Absolute to Preformed Salinity, minus 1

**Usage**

gsw_Fdelta(p, longitude, latitude)

**Arguments**

- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **longitude**: longitude in decimal degrees, positive to the east of Greenwich. (This is called `long` in the TEOS-10 Matlab code.)
- **latitude**: latitude in decimal degrees, positive to the north of the equator. (This is called `lat` in the TEOS-10 Matlab code.)

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

**Value**

(S/SStar)-1 [ unitless ]

**References**


**Examples**

```r
p <- c(10, 50, 125, 250, 600, 1000)
latitude <- c(4, 4, 4, 4, 4, 4)
longitude <- c(188, 188, 188, 188, 188, 188)
r <- gsw_Fdelta(p, longitude, latitude)
expect_equal(r/1e-3, c(0.006472389923452, 0.010352848168433, 0.025541937543450, 0.104348729347986, 0.218678084205081, 0.365415365571266))
```
**gsw_frazil_properties**  
*Properties of Frazil ice*

**Description**

Calculation of Absolute Salinity, Conservative Temperature, and ice mass fraction based on bulk Absolute Salinity, bulk enthalpy, and pressure

**Usage**

```r
gsw_frazil_properties(SA_bulk, h_bulk, p)
```

**Arguments**

- `SA_bulk`: Absolute Salinity of a combination of seawater and ice [g/kg]
- `h_bulk`: enthalpy of a mixture of seawater and ice [J/kg]
- `p`: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f6367fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

a list containing `SA_final`, `h_final` and `w_Ih_final`.

**References**


**Examples**

```r
SA_bulk <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)  
h_bulk <- c(-4.5544e4, -4.6033e4, -4.5830e4, -4.5589e4, -4.4948e4, -4.4027e4)  
p <- c(10, 50, 125, 250, 600, 1000)  
r <- gsw_frazil_properties(SA_bulk, h_bulk, p)  
expect_equal(r$SA_final, c(39.111030663000442, 39.407625769681573, 39.595789974885108, 39.481230845372889, 39.591177095552503, 39.82647709177123))  
expect_equal(r$CT_final, c(-2.156311126114311, -2.204672298963783, -2.273689262333450, -2.363714136353600, -2.644541006680772, -2.977651291726651))  
expect_equal(r$w_Ih_final, c(0.112480568014322, 0.114600300867556, 0.115421108602301, 0.117372990660305, 0.122617649983886, 0.127906590822347))
```
Description

Calculation of Absolute Salinity, Conservative Temperature, and ice mass fraction based on bulk Absolute Salinity, bulk potential enthalpy, and pressure

Usage

gsw_frazil_properties_potential(SA_bulk, h_pot_bulk, p)

Arguments

- **SA_bulk**: Absolute Salinity of a combination of seawater and ice [g/kg]
- **h_pot_bulk**: potential enthalpy of a mixture of seawater and ice [J/kg]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

a list containing SA_final, h_final and w_Ih_final.

References


See Also

Other things related to enthalpy: gsw_CT_from_enthalpy, gsw_dynamic_enthalpy, gsw_enthalpy_CT_exact, gsw_enthalpy_diff, gsw_enthalpy_first_derivatives_CT_exact, gsw_enthalpy_first_derivatives, gsw_enthalpy_ice, gsw_enthalpy_t_exact, gsw_enthalpy, gsw_frazil_properties_potential_poly, gsw_pot_enthalpy_from_pt_ice_poly, gsw_pot_enthalpy_from_pt_ice, gsw_pot_enthalpy_ice_freezing_poly, gsw_pot_enthalpy_ice_freezing, gsw_pt_from_pot_enthalpy_ice_poly, gsw_pt_from_pot_enthalpy_ice, gsw_specvol_first_derivatives_wrt_enthalpy, gsw_specvol_first_derivatives
gsw_frazil_properties_potential_poly

Properties of Frazil ice i.t.o. potential enthalpy (polynomial version)

Description
Calculation of Absolute Salinity, Conservative Temperature, and ice mass fraction based on bulk Absolute Salinity, bulk potential enthalpy, and pressure

Usage

gsw_frazil_properties_potential_poly(SA_bulk, h_pot_bulk, p)

Arguments

- SA_bulk: Absolute Salinity of a combination of seawater and ice [g/kg]
- h_pot_bulk: potential enthalpy of a mixture of seawater and ice [J/kg]
- p: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfc'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing SA_final, h_final and w_Ih_final.

References

See Also

Other things related to enthalpy: gsw_CT_from_enthalpy, gsw_dynamic_enthalpy, gsw_enthalpy_CT_exact, gsw_enthalpy_diff, gsw_enthalpy_first_derivatives_CT_exact, gsw_enthalpy_first_derivatives, gsw_enthalpy_ice, gsw_enthalpy_t_exact, gsw_enthalpy_adiabatic, gsw_enthalpy_properties_potential, gsw_pot_enthalpy_from_pt_ice_poly, gsw_pot_enthalpy_from_pt_ice, gsw_pot_enthalpy_ice_freezing_poly, gsw_pot_enthalpy_ice_freezing, gsw_pt_from_pot_enthalpy_ice_poly, gsw_pt_from_pot_enthalpy_ice, gsw_specvol_first_derivatives_wrt_enthalpy, gsw_specvol_first_derivatives

Examples

```r
gsw_frazil_ratios_adiabatic

# Ratios of SA, CT and p changes when Frazil Ice Forms

description

Ratios of changes in sa, ct and p that occur when frazil ice forms due to changes in pressure upon the mixture of seawater and ice.

Usage

gsw_frazil_ratios_adiabatic(SA, p, w_Ih)

Arguments

- **SA**: Absolute Salinity [g/kg]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **w_Ih**: initial mass fraction (ice) / (water + ice)

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e5403199e972f3e86f963e67fa4f5bfecc'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.
**Value**

a list containing dSA_dCT_frazil, dSA_dP_frazil and dCT_dP_frazil.

**References**


**Examples**

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c(10, 50, 125, 250, 600, 1000)
w_Ih <- c(0.9, 0.84, 0.4, 0.25, 0.05, 0.01)
r <- gsw_frazil_ratios_adiaabatic_poly(SA, p, w_Ih)
expect_equal(r$dSA_dCT_frazil, c(0.035152378000401, 1.932548405396193, 0.613212115609003, 0.516193092738565, 0.436656742034200, 0.425827266533876))
expect_equal(r$dSA_dP_frazil/1e-6, c(-0.197406834470366, -0.132213926580032, -0.0455580136143659, -0.038806356507548, -0.03335012722953744, -0.032350141940822))
expect_equal(r$dCT_dP_frazil/1e-7, c(-0.65401727338347, -0.689317412221414, -0.743301297684333, -0.751910946738026, -0.768138213038669, -0.781384728059898))
```

**gsw_frazil_ratios_adiaabatic_poly**

Ratios of SA, CT and p changes when Frazil Ice Forms (polynomial form)

**Description**

Ratios of changes in SA, CT and p that occur when frazil ice forms due to changes in pressure upon the mixture of seawater and ice.

**Usage**

`gsw_frazil_ratios_adiaabatic_poly(SA, p, w_Ih)`

**Arguments**

- **SA** Absolute Salinity [g/kg]
- **p** sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **w_Ih** initial mass fraction (ice) / (water + ice)

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031919e972f3e863f63e67fa4f5bfc'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.


**Value**

a list containing `dSA_dCT_frazil`, `dSA_dP_frazil` and `dCT_dP_frazil`.

**References**


**Examples**

```r
SA <- c( 34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c(10, 50, 125, 250, 600, 1000)
w_Ih <- c( 0.9, 0.84, 0.4, 0.25, 0.05, 0.01)
r <- gsw_frazil_ratios_adia_poly(SA, p, w_Ih)
expect_equal(r$dSA_dCT_frazil, c(3.035308957896530, 1.932631198810934, 0.613220785586734,
                               0.516106221687200, 0.436657158542033, 0.425827675768018))
expect_equal(r$dSA_dP_frazil/1e-6, c(-0.197512213108610, -0.133280971893621, -0.045599951957139,
                                   -0.038820466574251, -0.033548047632788, -0.033352365425407))
expect_equal(r$dCT_dP_frazil/1e-7, c(-0.658715350062703, -0.689634794137768, -0.743613932027895,
                                    -0.752179782823459, -0.768292629045686, -0.783236208526200))
```

**Description**

This calculates a geopotential anomaly, called either the dynamic height anomaly or the geostrophic
streamfunction in the TEOS-10 document listed as [1] below; users should read that and the refer-
ences therein for more details on the definition and its calculation here.

To get the column-integrated value in meters, take the first value of the returned vector and divide
by 9.7963 m/s². Note that this yields an integral with the top measured pressure (not zero) as an
upper limit.

**Usage**

```r
gsw_geo_strf_dyn_height(SA, CT, p, p_ref = 0)
```

**Arguments**

- **SA**: Absolute Salinity [ g/kg ]
- **CT**: Conservative Temperature [ degC ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **p_ref**: reference pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
Details
Because of the scheme used in the underlying C code, the pressures must be in order, and must
not have any repeats. Also, there must be at least 4 pressure values. Violating any of these three
restrictions yields an error.

If \( p_{\text{ref}} \) exceeds the largest \( p \) value, a vector of zeros is returned, in accordance with the underlying
C code.

The present R function works with a wrapper to a C function contained within the GSW-C system
(Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git com-
mitt '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system
(https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab sys-
more about these software systems, their authorships, and the science behind it all.

Value
A vector containing geopotential anomaly in \( m^2/s^2 \) for each level. For more on the units, see [2].

References

Examples
\[
\begin{align*}
\text{SA} & \leftarrow \text{c}(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324) \\
\text{CT} & \leftarrow \text{c}(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236) \\
p & \leftarrow \text{c}(10, 50, 125, 250, 600, 1000) \\
p_{\text{ref}} & \leftarrow 1000 \\
dh & \leftarrow \text{gsw_geo_strf_dyn_height}(\text{SA, CT, p, p}_{\text{ref}}) \\
\text{expect_equal}(dh, \text{c}(17.039204557769487, 14.665853784722286, 10.912861136923812, \\
7.567928838774945, 3.393524055565328, 0))
\end{align*}
\]
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>CT</td>
<td>Conservative Temperature [ degC ]</td>
</tr>
<tr>
<td>delta_p</td>
<td>difference in sea pressure between the deep and shallow limits of layers within which SA and CT are assumed to be constant. Note that delta_p must be positive.</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e540319e972f3e863f63e67fa4f5bfc3'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing dyn_height, the dynamic height anomaly [ m^2/s^2 ], and p_mid [ dbar ], the pressures at the layer centres. Note that the dynamic height anomaly unit, also known as a "dynamic meter", corresponds to approximately 1.02 metres of sealevel height (see e.g. Talley et al., 2011. Descriptive Physical Oceanography, 6th edition. Elsevier).

References


Examples

```r
sa <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
ct <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
delta_p <- c(10, 40, 75, 125, 350, 400)
r <- gsw_geo_strf_dyn_height_pc(sa, ct, delta_p)
equal(r$dyn_height, c(-0.300346215853487, -1.755165998114308, -4.423531083131365, -6.81659136254657, -9.453175257818430, -12.72109624991439))
equal(r$p_mid/1e2, c(0.050000000000000, 0.300000000000000, 0.875000000000000, 1.875000000000000, 4.250000000000000, 8.000000000000000))
```

Description

Gibbs Energy of Seawater, and its Derivatives

Usage

```r
gsw_gibbs(ns, nt, np, SA, t, p = 0)
```
Arguments

ns  An integer, the order of the SA derivative. Must be 0, 1, or 2.
nt  An integer, the order of the t derivative. Must be 0, 1, or 2.
np  An integer, the order of the p derivative. Must be 0, 1, or 2.
SA  Absolute Salinity [ g/kg ]
t  in-situ temperature (ITS-90) [ degC ]
p  sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e86363e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Gibbs energy [ J/kg ] if ns=nt=np=0. Derivative of energy with respect to SA [ J/kg/(g/kg)^ns ] if ns is nonzero and nt=np=0, etc. Note that derivatives with respect to pressure are in units with Pa, not dbar.

Caution

The TEOS-10 webpage for gsw_gibbs does not provide test values, so the present R version should be considered untested.

References


Examples

```R
library(gsw)
p <- seq(0, 100, 1)
SA <- rep(35, length(p))
t <- rep(-5, length(p))
## Check the derivative wrt pressure. Note the unit change
E <- gsw_gibbs(0, 0, 0, SA, t, p)
# Estimate derivative from linear fit (try plotting: it is very linear)
m <- lm(E ~ p)
print(summary(m))
plot(p, E)
abline(m)

dEdp1 <- coef(m)[2]
# Calculate derivative ... note we multiply by 1e4 to get from 1/Pa to 1/dbar
dEdp2 <- 1e4 * gsw_gibbs(0, 0, 1, SA[1], t[1], p[1])
## Ratio
dEdp1 / dEdp2
```
gsw_gibbs_ice  
Gibbs Energy of Ice, and its Derivatives

Description

Gibbs Energy of Ice, and its Derivatives

Usage

gsw_gibbs_ice(nt, np, t, p = 0)

Arguments

nt An integer, the order of the t derivative. Must be 0, 1, or 2.
np An integer, the order of the p derivative. Must be 0, 1, or 2.
t in-situ temperature (ITS-90) [degC]
p sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfeb'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Gibbs energy [J/kg] if ns=nt=np=0. Derivative of energy with respect to t [J/kg/(degC)^nt ] if nt is nonzero, etc. Note that derivatives with respect to pressure are in units with Pa, not dbar.

Caution

The TEOS-10 webpage for gsw_gibbs_ice does not provide test values, so the present R version should be considered untested.

References

Examples

```r
library(gsw)
p <- seq(0, 100, 1)
t <- rep(-5, length(p))
## Check the derivative wrt pressure. Note the unit change
E <- gsw_gibbs_ice(0, 0, t, p)
# Estimate derivative from linear fit (try plotting: it is very linear)
m <- lm(E ~ p)
print(summary(m))
plot(p, E)
abline(m)
dEdp1 <- coef(m)[2]
# Calculate derivative ... note we multiply by 1e4 to get from 1/Pa to 1/ubar

dEdp2 <- 1e4 * gsw_gibbs_ice(0, 1, t[1], p[1])
## Ratio
dEdp1 / dEdp2
```

---

gsw_grav  Gravitational Acceleration

Description

Gravitational Acceleration

Usage

`gsw_grav(latitude, p = 0)`

Arguments

- `latitude` latitude in decimal degrees, positive to the north of the equator. (This is called `lat` in the TEOS-10 Matlab code.)
- `p` sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e5403f19e972f3e863f63e67fa4f5bfe3'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

Value

gravitational acceleration [ m/s^2 ]
References


Examples

```r
lat <- c(-90, -60, -30, 0)
grav <- gsw_grav(lat)
expect_equal(grav, c(9.832186205884799, 9.819178859991149,
                     9.793249257048750, 9.780327000000000))
```

gsw_Helmholtz_energy_ice

**Helmholtz Energy of Ice**

Description

Helmholtz Energy of Ice

Usage

```r
gsw_Helmholtz_energy_ice(t, p)
```

Arguments

- `t` in-situ temperature (ITS-90) [ degC ]
- `p` sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f6367fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Helmholtz energy if ice [ J/kg ]

References

**gsw_ice_fraction_to_freeze_seawater**

---

**Ice Fraction to Cool Seawater to Freezing**

**Description**

Ice Fraction to Cool Seawater to Freezing

**Usage**

```r
gsw_ice_fraction_to_freeze_seawater(SA, CT, p, t_Ih)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>CT</td>
<td>Conservative Temperature [ degC ]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
<tr>
<td>t_Ih</td>
<td>initial temperature of ice [ degC ]</td>
</tr>
</tbody>
</table>

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

**Value**

a list containing SA_freeze, CT_freeze and w_Ih.

**References**

**Examples**

```r
SA <- c( 34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c( 28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c( 10, 50, 125, 250, 600, 1000)
t_ih <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
r <- gsw_ic ice_fraction_to_freeze_seawater(SA, CT, p, t_ih)
expect_equal(r$SA_freeze, c(25.823952352620722, 26.120495895535438, 27.460572941868072, 30.629978769577168, 31.458222332943784, 32.121170316796444))
expect_equal(r$CT_freeze, c(-1.389936216242376, -1.437013334134283, -1.569815847128818, -1.84641965657020, -2.16676673735941, -2.52273087978756))
expect_equal(r$w_ih, c(0.256046867272203, 0.251379393388925, 0.215985652155336, 0.121020375537284, 0.094378196687535, 0.075181377710828))
```

**Description**

Specific Internal Energy of Seawater (75-term equation)

**Usage**

```r
gsw_internal_energy(SA, CT, p)
```

**Arguments**

- **SA** Absolute Salinity [ g/kg ]
- **CT** Conservative Temperature [ degC ]
- **p** sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

specific internal energy [ J/kg ]

**References**

Examples

```
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
e <- gsw_internal_energy_ice(SA, CT, p)
expect_equal(e/1e5, c(1.148091576956162, 1.134013145527675, 0.909571141498779,
                   0.408593072177020, 0.273985276460357, 0.175019409258405))
```

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5f6c'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

specific internal energy [ J/kg ]

References


Examples

```
t_1h <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
p <- c(10, 50, 125, 250, 600, 1000)
e <- gsw_internal_energy_ice(t_1h, p)
expect_equal(e/1e5, c(-3.55660992432442, -3.6099926216929878, -3.597799043634774,
                    -3.587312078410920, -3.561207060376329, -3.512700418975375))
```
gsw_IPV_vs_fNsquared_ratio

Ratio of vert. gradient of pot. density to vert grad of locally-referenced pot density

Description

Note that the C library had to be patched to get this working: a new version of the library will address the bug directly.

Usage

gsw_IPV_vs_fNsquared_ratio(SA, CT, p, p_ref = 0)

Arguments

SA Absolute Salinity [ g/kg ]
CT Conservative Temperature [ degC ]
p sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
p_ref reference pressure [ dbar ]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

list containing IPV_vs_fNsquared_ratio [ unitless ] and mid-point pressure p_mid [ dbar ]

References


Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
p <- c( 10,  50,  125,  250,  600, 1000)
p_ref <- 0
r <- gsw_IPV_vs_fNsquared_ratio(SA, CT, p, p_ref)
expect_equal(r$IPV_vs_fNsquared_ratio, c(0.999742244888022, 0.996939883468178, 0.986141997098021,
0.931595598713477, 0.861224354872028))
expect_equal(r$p_mid, c(30, 87.5, 187.5, 425, 800))
gsw_kappa  

Isentropic Compressibility of Seawater (75-term equation)

Description

Isentropic Compressibility of Seawater (75-term equation)

Usage

gsw_kappa(SA, CT, p)

Arguments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>CT</td>
<td>Conservative Temperature [ degC ]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe6'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

isentropic compressibility [ 1/Pa ] (not 1/bar)

References


See Also

Other things related to compressibility: gsw_kappa_const_t_ice, gsw_kappa_ice, gsw_kappa_t_exact

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c( 28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c( 10, 50, 125, 250, 600, 1000)
kappa <- gsw_kappa(SA, CT, p)
extpect_equal(kappa*1e9, c(0.411343648791300, 0.411105416128894, 0.416566236026610, 0.435588650838751, 0.438782500588955, 0.439842289994702))
```
gsw_kappa_const_t_ice  Isothermal Compressibility of Ice

Description
Calculate isothermal compressibility of ice, in 1/Pa.

Usage
```
gsw_kappa_const_t_ice(t, p)
```

Arguments
- `t` in-situ temperature (ITS-90) [ degC ]
- `p` sea pressure [dbar], i.e. absolute pressure [bar] minus 10.1325 dbar

Details
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value
isothermal compressibility of ice [ 1/Pa ] (not 1/ubar)

References

See Also
Other things related to compressibility: gsw_kappa_ice, gsw_kappa_t_exact, gsw_kappa

Examples
```
t <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
p <- c( 10,  50, 125,  250,  600, 1000)
kappa <- gsw_kappa_const_t_ice(t, p)
expect_equal(kappa*1e9, c(0.115874753261484, 0.115384948953145, 0.115442212717850,
0.115452884634531, 0.115454824232421, 0.115619994536961))
```
isentropic compressibility of ice, in 1/Pa.

Arguments

\( t \)  
in-situ temperature (ITS-90) [ degC ]

\( p \)  
sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bafc'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

isentropic compressibility of ice [ 1/Pa ] (not 1/dbar)

References


See Also

Other things related to compressibility: gsw_kappa_const_t_ice, gsw_kappa_t_exact, gsw_kappa

Examples

```r
  t <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
  p <- c( 10,  50, 125, 250, 600, 1000)
  kappa <- gsw_kappa_ice(t, p)
  expect_equal(kappa*1e9, c(0.112495239053936, 0.112070687842183, 0.112119091047584, 0.112126584739297, 0.112123513812840, 0.112262589530974))
```
gsw_kappa_t_exact  
*Isentropic compressibility of seawater (exact)*

**Description**

Isentropic compressibility of seawater (exact)

**Usage**

```r
gsw_kappa_t_exact(SA, t, p)
```

**Arguments**

- **SA**: Absolute Salinity [g/kg]
- **t**: in-situ temperature (ITS-90) [degC]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e5403199e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

isentropic compressibility [1/Pa] (not 1/ubar)

**References**


**See Also**

Other things related to compressibility: `gsw_kappa_const_t_ice`, `gsw_kappa_ice`, `gsw_kappa`

**Examples**

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
kappa <- gsw_kappa(SA, CT, p)
expect_equal(kappa*1e9, c(0.411343648791300, 0.41105416128094, 0.416566236026610, 0.43558850838751, 0.438782500588955, 0.439842289994702))
```
**gsw_latentheat_evap_CT**

*Latent heat of evaporation*

---

**Description**

Latent heat of evaporation

**Usage**

```
gsw_latentheat_evap_CT(SA, CT)
```

**Arguments**

- **SA**: Absolute Salinity [ g/kg ]
- **CT**: Conservative Temperature [ degC ]

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e5403f9e972f3e863f63e67fa4f5bfe5'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

latent heat of evaporation [ J/kg ]

**References**


**See Also**

Other things related to latent heat: `gsw_latentheat_evap_t`, `gsw_latentheat_melting`

**Examples**

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
lh <- gsw_latentheat_evap_CT(SA, CT)
expect_equal(lh/1e6, c(2.429947187462561, 2.430774073049213, 2.444220372158452, 2.474127109232524, 2.482151446148560, 2.488052297193594))
```
gsw_latentheat_evap_t  Latent heat of evaporation

Description

Latent heat of evaporation

Usage

gsw_latentheat_evap_t(SA, t)

Arguments

SA  Absolute Salinity [ g/kg ]
   t  in-situ temperature (ITS-90) [ degC ]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

latent heat of evaporation [ J/kg ]

References


See Also

Other things related to latent heat: gsw_latentheat_evap_CT, gsw_latentheat_melting

Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
   t <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
   lh = gsw_latentheat_evap_t(SA, t)
   expect_equal(lh/1e6, c(2.429882982734836, 2.430730236218543, 2.444217294049004, 2.474137411322517, 2.482156276375029, 2.488054617630297))
Latent Heat of Melting

usage

gsw_latentheat_melting(SA, p)

Arguments

SA | Absolute Salinity [ g/kg ]
---|-----------------------------
p | sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

latent heat of freezing [ J/kg ]

References


See Also

Other things related to latent heat: gsw_latentheat_evap_CT, gsw_latentheat_evap_t

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c(10, 50, 125, 250, 600, 1000)
lh <- gsw_latentheat_melting(SA, p)
expect_equal(lh/1e5, c(3.299496680271213, 3.298613352397986, 3.297125622834541, 3.294973895330757, 3.288480445559747, 3.280715862416388))
```
Calculate \( \frac{d(SA)}{d(CT)} \) for Ice Melting in near-freezing Seawater

**Usage**

\[
gsw\_melting\_ice\_equilibrium\_SA\_CT\_ratio(SA, p)
\]

**Arguments**

- **SA**: Absolute Salinity \([\text{g/kg}]\)
- **p**: Sea pressure \([\text{dbar}]\), i.e. absolute pressure \([\text{dbar}]\) minus 10.1325 \text{dbar}

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

Ratio of change in \(SA\) to change in \(CT\) \([\text{g/kg/degC}]\).

**References**


**Examples**

```r
SA <- c( 34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324 )
p <- c( 10, 50, 125, 250, 600, 1000 )
r <- gsw_melting_ice_equilibrium_SA_CT_ratio(SA, p)
expect_equal(r, c(0.420209509196985, 0.422511693121631, 0.424345003216433, 0.422475836091426, 0.422023427778221, 0.423037622331042))
```
`gsw_melting_ice_equilibrium_SA_CT_ratio_poly`

*Calculate d(SA)/d(CT) for Ice Melting in near-freezing Seawater (Polynomial version)*

**Description**

Calculate d(SA)/d(CT) for Ice Melting in near-freezing Seawater (Polynomial version)

**Usage**

`gsw_melting_ice_equilibrium_SA_CT_ratio_poly(SA, p)`

**Arguments**

- `SA` Absolute Salinity [ g/kg ]
- `p` sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

ratio of change in SA to change in CT [ g/kg/degC ].

**References**


**Examples**

```r
SA <- c( 34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c( 10, 50, 125, 250, 600, 1000)
r <- gsw_melting_ice_equilibrium_SA_CT_ratio_poly(SA, p)
equal_equal(r, c(0.420209444587263, 0.4225116664682796, 0.424345538275708,
0.422475965003649, 0.422023755182266, 0.423038080717229))
```
**gsw_melting_ice_into_seawater**

*Calculate properties related to ice melting in seawater*

**Description**

Calculate properties related to ice melting in seawater

**Usage**

```
gsw_melting_ice_into_seawater(SA, CT, p, w_ih, t_ih)
```

**Arguments**

- **SA**  
  Absolute Salinity [ g/kg ]
- **CT**  
  Conservative Temperature [ degC ]
- **p**  
  sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **w_ih**  
  initial mass fraction (ice) / (water + ice)
- **t_ih**  
  initial temperature of ice [ degC ]

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e5403119e972f3e86f63e67fa4f5bfc'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

A list containing **SA_final**, **CT_final** and **w_ih_final**.

**References**


**Examples**

```
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(4.7856, 2.4329, 1.8103, 1.2600, 0.6886, 0.4403)
p <- c(10, 50, 125, 250, 600, 1000)
w_ih <- c(0.0560, 0.02513, 0.02159, 0.01210, 0.00943, 0.00751)
t_ih <- c(-4.7856, -4.4329, -3.8103, -2.6600, -3.8863, -3.4836)
r <- gsw_melting_ice_into_seawater(SA, CT, p, w_ih, t_ih)
expect_equal(r$SA_final, c(32.767939199999994, 34.01476604999998, 34.269397295999994,
                       34.425548880000001, 34.409033862000001, 34.471559675999998))
```
**Description**

Calculate $d(SA)/d(CT)$ for Ice Melting in Seawater

**Usage**

```r
gsw_melting_ice_SA_CT_ratio(SA, CT, p, t_ih)
```

**Arguments**

- **SA**: Absolute Salinity [ g/kg ]
- **CT**: Conservative Temperature [ degC ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **t_ih**: initial temperature of ice [ degC ]

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GRW-C](https://github.com/TEOS-10/GRW-C), as git commit ’5b4d959e54031f9e972f3e863f6363e67fa4f5bfe`)’, which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GRW-Fortran](https://github.com/TEOS-10/GRW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GRW-Matlab](https://github.com/TEOS-10/GRW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

ratio of change in SA to change in CT [ g/kg/degC ].

**References**


**Examples**

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(3.7856, 3.4329, 2.8103, 1.2600, 0.6886, 0.4403)
p <- c(10, 50, 125, 250, 600, 1000)
t_ih <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
r <- gsw_melting_ice_SA_CT_ratio(SA, CT, p, t_ih)
expect_equal(r, c(0.373840909022490, 0.371878514972099, 0.377104664622191, 0.382777696796156, 0.387133845152000, 0.393947316026914))
```
### gsw_melting_ice_SA_CT_ratio_poly

*Calculate d(SA)/d(CT) for Ice Melting in Seawater (Polynomial version)*

### Description

Calculate d(SA)/d(CT) for Ice Melting in Seawater (Polynomial version)

### Usage

```r
gsw_melting_ice_SA_CT_ratio_poly(SA, CT, p, t_ih)
```

### Arguments

- **SA**: Absolute Salinity [g/kg]
- **CT**: Conservative Temperature [degC]
- **p**: Sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **t_ih**: Initial temperature of ice [degC]

### Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e540311f9e972f3e863f63e67fa455b6c'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

### Value

Ratio of change in SA to change in CT [g/kg/degC].

### References


### Examples

```r
SA <- c( 34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c( 3.7856, 3.4329, 2.8103, 1.2600, 0.6886, 0.4403)
p <- c( 10, 50, 125, 250, 600, 1000)
t_ih <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
r <- gsw_melting_ice_SA_CT_ratio_poly(SA, CT, p, t_ih)
equal(r, c(0.373840908629278, 0.371878512745054, 0.3771046558031030,
0.382777681212224, 0.387133812279563, 0.393947267481204))
```
**gsw_melting_seaice_into_seawater**

*Calculate properties related to seaice melting in seawater*

**Description**

Calculate properties related to seaice melting in seawater

**Usage**

```r
gsw_melting_seaice_into_seawater(SA, CT, p, w_seaice, SA_seaice, t_seaice)
```

**Arguments**

- `SA`: Absolute Salinity [ g/kg ]
- `CT`: Conservative Temperature [ degC ]
- `p`: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- `w_seaice`: mass fraction (seaice) / (water + seaice)
- `SA_seaice`: Absolute Salinity of seaice
- `t_seaice`: temperature of seaice

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959c54031f9e972f3e863f63e67fa4f5bfe'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

A list containing `SA_final` and `CT_final`.

**References**


**Examples**

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(4.7856, 2.4329, 1.8103, 1.2600, 0.6886, 0.4403)
p <- c(10, 50, 125, 250, 600, 1000)
w_seaice <- c(0.0560, 0.02513, 0.02159, 0.01210, 0.00943, 0.00751)
SA_seaice <- c(5, 4.8, 3.5, 2.5, 1, 0.4)
t_seaice <- c(-4.7856, -4.4329, -3.8103, -4.2600, -3.8863, -3.4036)
r <- gsw_melting_seaice_into_seawater(SA, CT, p, w_seaice, SA_seaice, t_seaice)
```
Description

The result is computed based on first-differencing a computed density with respect pressure, and this can yield noisy results with CTD data that have not been smoothed and decimated. It also yields infinite values, for repeated adjacent pressure (e.g. this occurs twice with the ctd dataset provided in the oce package).

Usage

```r
gsw_Nsquared(SA, CT, p, latitude = 0)
```

Arguments

- **SA**: Absolute Salinity [ g/kg ]
- **CT**: Conservative Temperature [ degC ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **latitude**: latitude in decimal degrees, positive to the north of the equator. (This is called `lat` in the TEOS-10 Matlab code.)

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfc'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

- list containing N2 [ 1/s^2 ] and mid-point pressure p_mid [ dbar ]

References

**gsw_pot_enthalpy_from_pt_ice**

**Potential Enthalpy of Ice**

---

**Description**

Potential Enthalpy of Ice

**Usage**

`gsw_pot_enthalpy_from_pt_ice(pt_ice)`

**Arguments**

- `pt_ice` : potential temperature of ice (ITS-90) [ degC ]

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

**Value**

- potential enthalpy [ J/kg ]

**References**

**gsw_pot_enthalpy_from_pt_ice_poly**

**See Also**
Other things related to enthalpy: `gsw_CT_from_enthalpy`, `gsw_dynamic_enthalpy`, `gsw_enthalpy_CT_exact`, `gsw_enthalpy_diff`, `gsw_enthalpy_first_derivatives_CT_exact`, `gsw_enthalpy_first_derivatives`, `gsw_enthalpy_ice`, `gsw_enthalpy_t_exact`, `gsw_enthalpy`, `gsw_frazil_properties_potential_poly`, `gsw_frazil_properties_potential`, `gsw_pot_enthalpy_from_pt_ice_poly`, `gsw_pot_enthalpy_ice_freezing_poly`, `gsw_pot_enthalpy_ice_freezing`, `gsw_pt_from_pot_enthalpy_ice_poly`, `gsw_pt_from_pot_enthalpy_ice`, `gsw_specvol_first_derivatives_wrt_enthalpy`, `gsw_specvol_first_derivatives`

**Examples**
```r
pt0 ice <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
e <- gsw_pot_enthalpy_from_pt_ice_poly(pt0 ice)
equal(e/1e5, c(-3.55549449611868, -3.608607069998877, -3.5961538980859193,
-3.58512317806596, -3.557490528226009, -3.507198313847837))
```

---

**gsw_pot_enthalpy_from_pt_ice_poly**

**Potential Enthalpy of Ice (Polynomial version)**

**Description**
Potential Enthalpy of Ice (Polynomial version)

**Usage**
```r
gsw_pot_enthalpy_from_pt_ice_poly(pt0 ice)
```

**Arguments**
- `pt0 ice`: potential temperature of ice (ITS-90) [ degC ]

**Details**
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e540319e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

**Value**
- potential enthalpy [ J/kg ]

**References**
See Also
Other things related to enthalpy: gsw_CT_from_enthalpy, gsw_dynamic_enthalpy, gsw_enthalpy_CT_exact, gsw_enthalpy_diff, gsw_enthalpy_first_derivatives_CT_exact, gsw_enthalpy_first_derivatives, gsw_enthalpy_ice, gsw_enthalpy_t_exact, gsw_enthalpy, gsw_frazil_properties_potential_poly, gsw_frazil_properties_potential, gsw_pot_enthalpy_from_pt_ice, gsw_pot_enthalpy_ice_freezing_poly, gsw_pot_enthalpy_ice_freezing, gsw_pt_from_pot_enthalpy_ice_poly, gsw_pt_from_pot_enthalpy_ice, gsw_specvol_first_derivatives_wrt_enthalpy, gsw_specvol_first_derivatives

Examples
pt0_ice <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
e <- gsw_pot_enthalpy_ice_freezing(pt0_ice)
expect_equal(e/1e5, c(-3.555499482216265, -3.60868710959428, -3.596153924697033, -3.585123214831169, -3.557490561327994, -3.507198320793373))

Description
Potential Enthalpy of Ice at Freezing Point

Usage
gsw_pot_enthalpy_ice_freezing(SA, p, saturation_fraction = 1)

Arguments
SAAbsolute Salinity [ g/kg ]
psea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
saturation_fractionfraction of air in water [unitless]

Details
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value
potential enthalpy [ J/kg ]
Bugs

1. The C source underlying this function lacks an argument, saturation_fraction, which is present in the Matlab source, and so that argument is ignored here.


References


See Also

Other things related to enthalpy: gsw_CT_from_enthalpy, gsw_dynamic_enthalpy, gsw_enthalpy_CT_exact, gsw_enthalpy_diff, gsw_enthalpy_first_derivatives_CT_exact, gsw_enthalpy_first_derivatives, gsw_enthalpy_ice, gsw_enthalpy_t_exact, gsw_enthalpy, gsw_frazil_properties_potential_poly, gsw_frazil_properties_potential, gsw_pot_enthalpy_from_pt_ice_poly, gsw_pot_enthalpy_from_pt_ice, gsw_pot_enthalpy_ice_freezing_poly, gsw_pt_from_pot_enthalpy_ice_poly, gsw_pt_from_pot_enthalpy_ice, gsw_specvol_first_derivatives_wrt_enthalpy, gsw_specvol_first_derivatives

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c(10, 50, 125, 250, 600, 1000)
saturation_fraction = 1
e <- gsw_pot_enthalpy_ice_freezing(SA, p, saturation_fraction)
## Not run:
expect_equal(e/1e5, c(-3.373409558967978, -3.374434164002012, -3.376117536928847, -3.378453698871986, -3.385497832886802, -3.393768587631489))

## End(Not run)
```

---

gsw_pot_enthalpy_ice_freezing_first_derivatives

*First Derivatives of Potential Enthalpy*

Description

First Derivatives of Potential Enthalpy

Usage

```r
gsw_pot_enthalpy_ice_freezing_first_derivatives(SA, p)
```
Arguments

SA  Absolute Salinity [ g/kg ]

p  sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing pot_enthalpy_ice_freezing_SA [ (J/kg)/(g/kg) ], the derivative of potential enthalpy with respect to Absolute Salinity, and pot_enthalpy_ice_freezing_p [ unitless ], the derivative of Conservative Temperature with respect to potential temperature. (Note that the second quantity is denoted pot_enthalpy_ice_freezing_P in the documentation for the Matlab function.)

References


Examples

```
sa <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c(10, 50, 125, 250, 600, 1000)
r <- gsw_pot_enthalpy_ice_freezing_first_derivatives_poly(sa, p)
expect_equal(r$pot_enthalpy_ice_freezing_SA/1e2,
c(-1.183484968590718, -1.184125268891200, -1.184619267864844,
  -1.184026131143674, -1.183727706650925, -1.183814873741961))
expect_equal(r$pot_enthalpy_ice_freezing_p/1e-3,
c(-0.202809399383260, -0.20387335312542, -0.203473018454630,
  -0.204112435106666, -0.205889571619502, -0.207895691215823))
```

---

gsw_pot_enthalpy_ice_freezing_first_derivatives_poly

First Derivatives of Potential Enthalpy (Polynomial version)

Description

First Derivatives of Potential Enthalpy (Polynomial version)
Usage

gsw_pot_enthalpy_ice_freezing_first_derivatives_poly(SA, p)

Arguments

SA               Absolute Salinity [ g/kg ]

p                sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.0.5-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e540319e972f3e863f63e67fa4f5bfe5c'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing pot_enthalpy_ice_freezing_SA [ (J/kg)/(g/kg) ], the derivative of potential enthalpy with respect to Absolute Salinity, and pot_enthalpy_ice_freezing_p [ unitless ], the derivative of Conservative Temperature with respect to potential temperature. (Note that the second quantity is denoted pot_enthalpy_ice_freezing_p in the documentation for the Matlab function.)

References


Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c( 10,  50, 125,  250,  600, 1000)
r <- gsw_pot_enthalpy_ice_freezing_first_derivatives_poly(SA, p)
expect_equal(r$pot_enthalpy_ice_freezing_SA/1e2,
          c(-1.183498006918154, -1.184135169530662, -1.184626138334419,
            -1.18403266542549, -1.18372737143588, -1.183805326863513)))
expect_equal(r$pot_enthalpy_ice_freezing_p/1e-3,
          c(-0.202934280214689, -0.20313695011241, -0.203515960539503,
             -0.204145112153220, -0.205898365024147, -0.207885289186464))
Description

Potential Enthalpy of Ice at Freezing Point (Polynomial version)

Usage

gsw_pot_enthalpy_ice_freezing_poly(SA, p, saturation_fraction = 1)

Arguments

SA      Absolute Salinity [ g/kg ]

p      sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

saturation_fraction

fraction of air in water [unitless]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

potential enthalpy [ J/kg ]

References


See Also

Other things related to enthalpy: gsw_CT_from_enthalpy, gsw_dynamic_enthalpy, gsw_enthalpy_CT_exact, gsw_enthalpy_diff, gsw_enthalpy_first_derivatives_CT_exact, gsw_enthalpy_first_derivatives, gsw_enthalpy_ice, gsw_enthalpy_t_exact, gsw_enthalpy, gsw_frazil_properties_potential_poly, gsw_frazil_properties_potential, gsw_pot_enthalpy_from_pt_ice_poly, gsw_pot_enthalpy_from_pt_ice, gsw_pot_enthalpy_ice_freezing, gsw_pt_from_pot_enthalpy_ice_poly, gsw_pt_from_pot_enthalpy_ice, gsw_specvol_first_derivatives_wrt_enthalpy, gsw_specvol_first_derivatives
gsw_pot_rho_t_exact

Potential density

Description

Potential density

Usage

gsw_pot_rho_t_exact(SA, t, p, p_ref)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>t</td>
<td>in-situ temperature (ITS-90) [ degC ]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
<tr>
<td>p_ref</td>
<td>reference pressure [ dbar ]</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe6'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

Value

potential density [ kg/m³ ]

References


See Also

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_first_derivatives, gsw_rho_ice, gsw_rho_t_exact, gsw_rho, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
t <- c( 28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(  10,   50,   125,   250,   600,  1000)
p_ref <- 0
prho <- gsw_pot_rho_t_exact(SA, t, p, p_ref)
sa <- c(1.021798145811089, 1.022052484416980, 1.02389358361958, 1.026667621124443, 1.02710730868492, 1.027409631264134)
```

\[ gsw\_pressure\_coefficient\_ice \]

\[ \text{Pressure Coefficient for Ice} \]

**Description**

Pressure Coefficient for Ice

**Usage**

```r
gsw\_pressure\_coefficient\_ice(t, p)
```

**Arguments**

- `t` in-situ temperature (ITS-90) [degC]
- `p` sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

**Value**

specific internal energy [Pa/degC]
References


Examples

```r
t <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
p <- c(10, 50, 125, 250, 600, 1000)
pc <- gsw_pressure_coefficient_ice(t, p)
expect_equal(pc/1e6, c(1.333098059787838, 1.326359005133730, 1.327354133828322,
                          1.327793888831923, 1.328549609231685, 1.331416733490227))
```

---

*gsw_pressure_freezing_CT*

*Pressure at which Seawater Freezes*

Description

Pressure at which Seawater Freezes

Usage

```
gsw_pressure_freezing_CT(SA, CT, saturation_fraction = 1)
```

Arguments

- **SA**: Absolute Salinity [ g/kg ]
- **CT**: Conservative Temperature [ degC ]
- **saturation_fraction**: fraction of air in water [unitless]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e5403f19e972f3e863f63e67fa4f5bfc'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

pressure at which freezing will occur [ dbar ]

References

**gsw_pt0_from_t**

**Examples**

```r
gsw_pt0_from_t(SA = c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324), 
t = c(-1.8996, -1.9407, -2.0062, -2.0923, -2.3593, -2.6771), 
psat = c(1, 0.8, 0.6, 0.5, 0.4, 0), 
p = gsw_pressure_freezing_CT(SA, t, psat))
expect_equal(p/1e3, c(0.009890538270710, 0.050376826585933, 0.125933117050624, 
0.251150973076077, 0.60141775836021, 1.002273338145043))
```

---

**gsw_pt0_from_t**  
*Potential temperature referenced to the surface*

**Description**

Potential temperature referenced to the surface

**Usage**

```r
gsw_pt0_from_t(SA, t, p)
```

**Arguments**

- **SA**: Absolute Salinity [ g/kg ]
- **t**: in-situ temperature (ITS-90) [ degC ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

potential temperature [ degC ]

**References**

**Examples**

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
t <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
pt0 <- gsw_pt0_from_t_ice(SA, t, p)
expect_equal(pt0, c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324))
```

**gsw_pt0_from_t_ice**

**Potential Temperature of Ice Referenced to the Surface**

**Description**

Potential Temperature of Ice Referenced to the Surface

**Usage**

```r
gsw_pt0_from_t_ice(t, p)
```

**Arguments**

- `t` in-situ temperature (ITS-90) [degC]
- `p` sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

**Value**

potential temperature [degC]

**References**

http://www.teos-10.org/pubs/gsw/html/gsw_pt0_from_t_ice.html

**Examples**

```r
t <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
p <- c(10, 50, 125, 250, 600, 1000)
pt0 <- gsw_pt0_from_t_ice(t, p)
expect_equal(pt0, c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036))
```
First Derivatives of Potential Temperature

Usage

gsw_pt_first_derivatives(SA, CT)

Arguments

SA Absolute Salinity [ g/kg ]
CT Conservative Temperature [ degC ]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing pt_SA [ K/(g/kg) ], the derivative of potential temperature with respect to Absolute Salinity, and pt_CT [ unitless ], the derivative of potential temperature with respect to Conservative Temperature.

References


Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
r <- gsw_pt_first_derivatives(SA, CT)
expect_equal(r$pt_SA, c(0.041863223165431, 0.041452303483011, 0.034682095247246, 0.018711079068408, 0.014079958329844, 0.010577326129948))
expect_equal(r$pt_CT, c(0.997192967140242, 0.997451686508335, 0.998357568277750, 0.999996224076267, 1.000283719083268, 1.000525947028218))
gsw_pt_from_CT

Potential temperature from Conservative Temperature

Description

Potential temperature from Conservative Temperature

Usage

gsw_pt_from_CT(SA, CT)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>CT</td>
<td>Conservative Temperature [ degC ]</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e540319e972f3e863f63e67fa4f5bfee'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

potential temperature [ degC ]

References


Examples

```r
SA <- c(34.7118, 34.8915, 35.6256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
pt <- gsw_pt_from_CT(SA, CT)
expect_equal(pt, c(28.731777048624573, 28.420955597191984, 22.784953468087107, 10.230534394434429, 6.829216587061605, 4.324534835990236))
```
**gsw_pt_from_entropy**  
*Potential Temperature from Entropy*

**Description**  
Potential Temperature from Entropy

**Usage**  
gsw_pt_from_entropy(SA, entropy)

**Arguments**  
SA **Absolute Salinity** [ g/kg ]  
entropy **specific entropy** [ J/(degC*kg) ]

**Details**  
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**  
potential temperature [ degC ]

**References**  

**See Also**  
Other things related to entropy: *gsw_CT_from_entropy*, *gsw_entropy_first_derivatives*, *gsw_entropy_from_pt*, *gsw_entropy_from_t*, *gsw_entropy_ice*

**Examples**  
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)  
entropy <- c(400.3892, 395.4378, 319.8668, 146.7910, 98.6471, 62.7919)  
pt <- gsw_pt_from_entropy(SA, entropy)  
expect_equal(pt, c(28.783179828078666, 28.420954825949291, 22.784952736245351, 10.230532066931868, 6.829213325916900, 4.32457782985845))
**gsw_pt_from_pot_enthalpy_ice**

*Potential Temperature from Potential Enthalpy of Ice*

**Description**

Potential Temperature from Potential Enthalpy of Ice

**Usage**

```
gsw_pt_from_pot_enthalpy_ice(pot_enthalpy_ice)
```

**Arguments**

- `pot_enthalpy_ice`
  - potential enthalpy of ice [ J/kg ]

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

**Value**

potential temperature [ degC ]

**References**


**See Also**

Other things related to enthalpy: gsw_CT_from_enthalpy, gsw_dynamic_enthalpy, gsw_enthalpy_CT_exact, gsw_enthalpy_diff, gsw_enthalpy_first_derivatives_CT_exact, gsw_enthalpy_first_derivatives, gsw_enthalpy_ice, gsw_enthalpy_t_exact, gsw_enthalpy, gsw_frazil_properties_potential_poly, gsw_frazil_properties_potential, gsw_pot_enthalpy_from_pt_ice_poly, gsw_pot_enthalpy_from_pt_ice, gsw_pot_enthalpy_ice_freezing_poly, gsw_pot_enthalpy_ice_freezing, gsw_pt_from_pot_enthalpy_ice_poly, gsw_specvol_first_derivatives_wrt_enthalpy, gsw_specvol_first_derivatives

**Examples**

```
pot_enthalpy_ice <- c(-3.5544e5, -3.6033e5, -3.5830e5, -3.5589e5, -3.4948e5, -3.4027e5)
pot_enthalpy_ice <- gsw_pt_from_pot_enthalpy_ice(pot_enthalpy_ice)
expect_equal(pt, c(-10.733087588125384, -13.167397822300588, -12.154205899172704, -10.956202704066083, -7.794963188206421, -3.314905214262531))
```
Description

Potential Temperature from Potential Enthalpy of Ice (Polynomial version)

Usage

```r
gsw_pt_from_pot_enthalpy_ice_poly(pot_enthalpy_ice)
```

Arguments

- `pot_enthalpy_ice`
  - potential enthalpy of ice [ J/kg ]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

Value

- potential temperature [ degC ]

References


See Also

Other things related to enthalpy: `gsw_CT_from_enthalpy`, `gsw_dynamic_enthalpy`, `gsw_enthalpy_CT_exact`, `gsw_enthalpy_diff`, `gsw_enthalpy_first_derivatives_CT_exact`, `gsw_enthalpy_first_derivatives`, `gsw_enthalpy_ice`, `gsw_enthalpy_t_exact`, `gsw_enthalpy`, `gsw_frazil_properties_potential_poly`, `gsw_frazil_properties_potential`, `gsw_pot_enthalpy_from_pt_ice_poly`, `gsw_pot_enthalpy_from_pt_ice`, `gsw_pot_enthalpy_ice_freezing_poly`, `gsw_pot_enthalpy_ice_freezing`, `gsw_pt_from_pot_enthalpy_ice`, `gsw_specvol_first_derivatives_wrt_enthalpy`, `gsw_specvol_first_derivatives`
Examples

```r
pot_enthalpy_ice <- c(-3.5544e5, -3.6033e5, -3.5830e5, -3.5589e5, -3.4948e5, -3.4027e5)
pt <- gsw_pt_from_pot_enthalpy_ice_poly(pot_enthalpy_ice)
expect_equal(pt, c(-10.733085986035007, -13.167396204945987, -12.154204137867396, -10.956201046447006, -7.794963341294590, -3.314907552013722))
```

---

**gsw_pt_from_t**  
*Potential Temperature from in-situ Temperature*

### Description

Potential Temperature from in-situ Temperature

### Usage

```r
gsw_pt_from_t(SA, t, p, p_ref = 0)
```

### Arguments

- **SA**: Absolute Salinity [ g/kg ]
- **t**: in-situ temperature (ITS-90) [ degC ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **p_ref**: reference pressure [ dbar ]

### Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959c54031f9e972f3e863f63e67fa4f5bfe'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

### Value

- **potential temperature**: [ degC ]

### References

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
t <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
p_ref <- 0
pt <- gsw_pt_from_t(SA, t, p, p_ref)
expect_equal(pt, c(28.783196819670632, 28.420983342398962, 22.784930399117108,
                  10.230523661095731, 6.829230224409661, 4.324510571845719))
```

---

gsw_pt_from_t_ice  
Potential Temperature of Ice from in-situ Temperature

Description

Potential Temperature of Ice from in-situ Temperature

Usage

`gsw_pt_from_t_ice(t, p, p_ref = 0)`

Arguments

t  in-situ temperature (ITS-90) [ degC ]

p  sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

p_ref  reference pressure [ dbar ]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e5403199e972f3e86f63e67fa4f5b6ec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

potential temperature [ degC ]

References

Examples

t <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
p <- c(10, 50, 125, 250, 600, 1000)
p_ref <- 0 # not actually needed, since 0 is the default
pt <- gsw_pt_from_t_ice(t, p, p_ref)
expect_equal(pt, c(-10.787778988205272, -13.443730926050661, -12.83742705699676,
                   -12.314321615760921, -11.017040858094234, -8.622907355083147))

Description

Second Derivatives of Potential Temperature

Usage

gsw_pt_second_derivatives(SA, CT)

Arguments

SA          Absolute Salinity [ g/kg ]
CT          Conservative Temperature [ degC ]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing pt_SA_SA [ K/(g/kg)^2 ], the second derivative of potential temperature with respect to Absolute Salinity at constant potential temperature, and pt_SA_pt [ 1/(g/kg) ], the derivative of potential temperature with respect to Conservative Temperature and Absolute Salinity, and pt_pt_pt [ 1/degC ], the second derivative of potential temperature with respect to Conservative Temperature.

References

Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
r <- gsw.pt_second_derivatives(SA, CT)
expect_equal(r$pt_SA_SA/1e-3, c(0.160307058371208, 0.160785497957769, 0.168647220588324,
                    0.198377949876584, 0.21018199321236, 0.220018966513329))
expect_equal(r$pt_SA_CT, c(0.001185581323691, 0.001187068518686, 0.001217629686266,
                   0.001333254154015, 0.001379674342678, 0.001418371539325))
expect_equal(r$pt_CT_CT/1e-3, c(-0.121979811279463, -0.123711264754503, -0.140136818504977,
                   -0.140645384127949, -0.113781055410824, -0.082417269009484))

Description

Pressure from height (75-term equation)

Usage

gsw_p_from_z(z, latitude)

Arguments

z
  height, zero at surface (but note last 2 args) and positive upwards \[ \text{m} \]
latitude
  latitude in decimal degrees, positive to the north of the equator. (This is called \text{lat} in the TEOS-10 Matlab code.)

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at \url{https://github.com/TEOS-10/GSW-C}, as git commit '5b4d959e5403199e972f3e863f63e67fa4f5bfc'), which stems from the GSW-Fortran system (\url{https://github.com/TEOS-10/GSW-Fortran}) which in turn stems from the GSW-Matlab system (\url{https://github.com/TEOS-10/GSW-Matlab}). Consult \url{http://www.teos-10.org} to learn more about these software systems, their authorships, and the science behind it all.

Value

sea pressure \[ \text{d-bar} \]

References

\url{http://www.teos-10.org/pubs/gsw/html/gsw_p_from_z.html}

See Also

Other things related to depth: \textit{gsw_z_from_p}
Examples

```r
z <- c(10, 50, 125, 250, 600, 1000)
latitude <- 4
p <- gsw_p_from_z(z, latitude)
expect_equal(p/1e3, c(0.010055726724518, 0.050283543374874, 0.125731858435610, 
0.251540299593468, 0.604210012340727, 1.007990337692001))
```

**gsw_rho**  
*In-situ density*

**Description**

In-situ density, using the 75-term equation for specific volume.

**Usage**

```r
gsw_rho(SA, CT, p)
```

**Arguments**

- **SA**  
  Absolute Salinity [ g/kg ]

- **CT**  
  Conservative Temperature [ degC ]

- **p**  
  sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

in-situ density [ kg/m³ ]

**References**


**See Also**

Other things related to density: `gsw_CT_from_rho`, `gsw_CT_maxdensity`, `gsw_SA_from_rho`, `gsw_alpha_on_beta`, `gsw_alpha_wrt_t_exact`, `gsw_alpha_wrt_t_ice`, `gsw_alpha`, `gsw_beta_const_t_exact`, `gsw_beta`, `gsw_pot_rho_t_exact`, `gsw_rho_alpha_beta`, `gsw_rho_first_derivatives_wrt_enthalpy`, `gsw_rho_first_derivatives`, `gsw_rho_ice`, `gsw_rhos_exact`, `gsw_sigma0`, `gsw_sigma1`, `gsw_sigma2`, `gsw_sigma3`, `gsw_sigma4`, `gsw_specvol_alpha_beta`, `gsw_specvol_anom_standard`, `gsw_specvol_t_exact`, `gsw_specvol`
Examples

```
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
p <- c(10, 50, 125, 250, 600, 1000)
rho <- gsw_rho(SA, CT, p)
expect_equal(rho/1e3, c(1.021839935738108, 1.022262457966867, 1.02427195413316,
1.027790152759127, 1.029837779000189, 1.032002453224572))
```

```
gsw_rho_alpha_beta In-situ density, thermal expansion coefficient and haline contraction coefficient (75-term equation)
```

Description

Calculate the in-situ density, the expansion coefficient (with respect to Conservative Temperature) and the haline contraction coefficient (with respect to Absolute Salinity), using the 75-term equation.

Usage

```
gsw_rho_alpha_beta(SA, CT, p)
```

Arguments

- **SA** Absolute Salinity [ g/kg ]
- **CT** Conservative Temperature [ degC ]
- **p** sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e85d63e67fa4f5f6ec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing in-situ density rho [ kg/m^3 ], thermal expansion coefficient alpha [ 1/degC ], and haline contraction coefficient beta [ kg/g ].

References

See Also

Other things related to density: `gsw_CT_from_rho`, `gsw_CT_maxdensity`, `gsw_SA_from_rho`, `gsw_alpha_on_beta`, `gsw_alpha_wrt_t_exact`, `gsw_alpha_wrt_ice`, `gsw_alpha`, `gsw_beta_const_t_exact`, `gsw_beta`, `gsw_pot_rho_t_exact`, `gsw_rho_first_derivatives_wrt_enthalpy`, `gsw_rho_first_derivatives`, `gsw_rho_ice`, `gsw_rho_t_exact`, `gsw_rho`, `gsw_sigma0`, `gsw_sigma1`, `gsw_sigma2`, `gsw_sigma3`, `gsw_sigma4`, `gsw_specvol_alpha_beta`, `gsw_specvol_anom_standard`, `gsw_specvol_ice`, `gsw_specvol_t_exact`, `gsw_specvol`

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
p <- c(10, 50, 125, 250, 600, 1000)
r <- gsw_rho_alpha_beta(SA, CT, p)
```
Value

list containing drho_dSA [ kg^2/(g m^3) ], drho_dCT [ kg/(K m^3) ] and drho_dp [ kg/(Pa m^3) ]

References


See Also

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_ice, gsw_rho_t_exact, gsw_rho, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol

Examples

```r
gsw_rho_first_derivatives_wrt_enthalpy(SA, CT, p)
```

\[ r = gsw\_rho\_first\_derivatives\_wrt\_enthalpy(SA, CT, p) \]

```r
expect_equal(r$drho_dSA, c(0.733153791778356, 0.733624109867480, 0.743950957375504, 0.771357282286743, 0.777581141431288, 0.781278296628328))
expect_equal(r$drho_dCT, c(-0.331729027977015, -0.329838643311336, -0.288013324730644, -0.178012962919839, -0.150654632545556, -0.13355647386894))
expect_equal(r$drho_dp, 1e-6*c(0.420382360738476, 0.420251070273888, 0.426773054953941, 0.447763615252861, 0.452011501791479, 0.454118117103094))
```

```r
r
```

\[ gsw\_rho\_first\_derivatives\_wrt\_enthalpy \]

\textit{Density First Derivatives wrt enthalpy (75-term equation)}

Description

Density First Derivatives wrt enthalpy (75-term equation)

Usage

\[ gsw\_rho\_first\_derivatives\_wrt\_enthalpy(SA, CT, p) \]

Arguments

\begin{itemize}
  \item \textbf{SA} \hspace{1cm} \text{Absolute Salinity [ g/kg ]}
  \item \textbf{CT} \hspace{1cm} \text{Conservative Temperature [ degC ]}
  \item \textbf{p} \hspace{1cm} \text{sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar}
\end{itemize}
Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit 'b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing \( \rho_{SA \text{ wrt } h} \) \([\text{kg/m}^3]/(\text{g/kg})/(\text{J/kg})\] and \( \rho_h \) \([\text{kg/m}^3]/(\text{J/kg})\].

References


See Also

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives, gsw_rho_ice, gsw_rho_t_exact, gsw_rhog, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
p <- c(10, 50, 125, 250, 600, 1000)
r <- gsw_rho_first_derivatives_wrt_enthalpy(SA, CT, p)
expect_equal(r$rho_SA_wrt_h, c(0.733147960400929, 0.733595114830609, 0.74388977148835, 0.771275693831993, 0.777414200397148, 0.781030546537425))
expect_equal(r$rho_h*1e4, c(-0.831005413475887, -0.826243794873652, -0.721438289309903, -0.445892608094272, -0.377326924646647, -0.334475962698187))
```

```
gsw_rho_ice In-situ density of ice
```

Description

In-situ density of ice \([\text{kg/m}^3]\)

Usage

gsw_rho_ice(t, p)
Arguments

\[ t \text{ in-situ temperature (ITS-90) [ degC ]} \]

\[ p \text{ sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar} \]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

in-situ density [ kg/m^3 ]

References


See Also

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_first_derivatives, gsw_rho_t_exact, gsw_rho, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol

Examples

```r
    t <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
    p <- c(10, 50, 125, 250, 600, 1000)
    rho <- gsw_rho_ice(t, p)
    expect_equal(rho, c(918.2879969148962, 918.7843487325120, 918.6962796312600, 918.7513732275766, 918.9291139833307, 919.0032237449378))
```

---

### gsw_rho_second_derivatives

**Second Derivatives of Density**

**Description**

Second Derivatives of Density
Usage

gsw_rho_second_derivatives(SA, CT, p)

Arguments

SA  Absolute Salinity [ g/kg ]
CT  Conservative Temperature [ degC ]
p  sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec' ), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing 
rho_SA_SA [ (kg/m^3)/(g/kg)^2 ], the second derivative of density with respect to Absolute Salinity, rho_SA_CT [ (g/kg)/(g/kg)/degC ], the derivative of density with respect to Absolute Salinity and Conservative Temperature, and rho_CT_CT [ (kg/m^3)/degC^2 ], the second derivative of density with respect to Conservative Temperature.

References


Examples

SA <- c(34.7118, 34.8915, 35.8256, 34.8472, 34.7366, 34.7324)
CT <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c( 10,  50,  125,  250,  600,  1000)
r <- gsw_rho_second_derivatives(SA, CT, p)
equal(r$rho_SA_SA/1e-3, c(0.207364734477357, 0.207415414547223, 0.192903197286004, 0.135809142211237, 0.122627562106876, 0.114042431905783))
equal(r$rho_SA_CT, c(-0.001832856561477, -0.001837354806146, -0.001988065808878, -0.0025668181494807, -0.002708939464658, -0.002798484050141))
equal(r$rho_CT_CT, c(-0.007241243282334, -0.007267807914635, -0.007964270843331, -0.010080164822017, -0.010572009761984, -0.010939294762280))
equal(r$rho_SA_p/1e-8, c(-0.007202931942412, -0.007755861200845, -0.009254966987409, -0.186661486272630, -0.110022261844248, -0.112287954167177))
equal(r$rho_CT_p/1e-8, c(-0.11659792537549, -0.117744271236102, -0.141712549466964, -0.214441626736539, -0.237704139801551, -0.255296606304874))
**gsw_rho_second_derivatives_wrt_enthalpy**  
*Second Derivatives of Density wrt Enthalpy*

**Description**  
Second Derivatives of Density wrt Enthalpy

**Usage**  
gsw_rho_second_derivatives_wrt_enthalpy(SA, CT, p)

**Arguments**  
- **SA**: Absolute Salinity [ g/kg ]
- **CT**: Conservative Temperature [ degC ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**  
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**  
A list containing rho_SA_SA [ (kg/m^3)/(g/kg)^2 ], the second derivative of density with respect to Absolute Salinity, rho_SA_h [ (g/kg)/(g/kg)/(J/kg)] , the derivative of density with respect to Absolute Salinity and enthalpy, and rho_h_h [ (kg/m^3)/(J/kg)^2 ], the second derivative of density with respect to enthalpy.

**Bugs**  
As of March 27, 2017, the test values listed in “Examples” do not match values provided at the TEOS-10 website listed in “References”, but they match with values given by the Matlab code that is provided on the TEOS-10 website. It is expected that the TEOS-10 website will be updated by May 2017. As those updates to the TEOS-10 website become available, the present comment will be revised or deleted.

**References**  
See Also

Other functions with suspicious test values on the TEOS-10 website: gsw_entropy_second_derivatives, gsw_specvol_second_derivatives_wrt_enthalpy, gsw_t_freezing_first_derivatives_poly

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
t <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
p <- c(10, 50, 125, 250, 600, 1000)
r <- gsw_rho_second_derivatives_wrt_enthalpy(SA, t, p)
expect_equal(r$rho_SA_SA/1e-3, c(0.207312267114544, 0.207065033523473, 0.191848346945039, 0.133182862881598, 0.116049034622904, 0.102745309429078))
expect_equal(r$rho_SA_h/1e-6, c(-0.45905300088382, -0.468370569872258, -0.496606515416296, -0.642833108558133, -0.682059162941161, -0.706793055445909))
expect_equal(r$rho_h_h/1e-9, c(-0.454213854637790, -0.455984900233909, -0.49980703989387, -0.62837767293403, -0.664821595759388, -0.687367088752173))
```

### gsw_rho_t_exact

**In-situ Density of Seawater**

**Description**

In-situ Density of Seawater

**Usage**

```r
gsw_rho_t_exact(SA, t, p)
```

**Arguments**

- **SA**: Absolute Salinity [ g/kg ]
- **t**: in-situ temperature (ITS-90) [ degC ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5f6c'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

in-situ density [ kg/m^3 ]
**gsw_SAAR**

**Description**
Absolute Salinity Anomaly Ratio

**Usage**
gsw_SAAR(p, longitude, latitude)

**Arguments**
- **p**: sea pressure [dbar], i.e., absolute pressure [dbar] minus 10.1325 dbar
- **longitude**: longitude in decimal degrees, positive to the east of Greenwich. (This is called `long` in the TEOS-10 Matlab code.)
- **latitude**: latitude in decimal degrees, positive to the north of the equator. (This is called `lat` in the TEOS-10 Matlab code.)

**Details**
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e5403119e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.
**Value**

a list containing SAAR, which is the (unitless) Absolute Salinity Anomaly Ratio, and in_ocean is set to 1 if SAAR is nonzero, or to 0 otherwise.

**Bugs**

The definition of in_ocean is incorrect, because the C function named gsw_saar, which is called by the present R function, does not calculate in_ocean, as the base Matlab function named gsw_SAAR does. However, examination of the Matlab code shows that in_ocean is set to 0 along with SAAR, whenever the original estimate of the latter is nonfinite. Thus, points that would be signalled as being on the land by the Matlab code are indicated in the same way with the present R function. However, other points may also be indicated as being on land, if SAAR is simply zero in the first calculation. Whether this poses a problem in practice is an open question, since it seems likely that this function would only be called with oceanic locations, anyway. If problems arise for users, a patch can be written to improve things.

**References**


**Examples**

```r
p <- c(10, 50, 125, 250, 600, 1000)
longitude <- c(188, 188, 188, 188, 188, 188)
latitude <- c(4, 4, 4, 4, 4, 4)
SAAR <- gsw_SAAR(p, longitude, latitude)
expect_equal(1e3*SAAR$SAAR, c(0.004794295602143, 0.007668755837570, 0.18919828449091, 0.077293264028981, 0.161974583039298, 0.270652408428964))
expect_equal(SAAR$in_ocean, rep(1, 6))
```

---

**gsw_SA_freezing_from_CT**

*Compute Absolute Salinity at Freezing Conservative Temperature*

**Description**

Compute Absolute Salinity at Freezing Conservative Temperature

**Usage**

gsw_SA_freezing_from_CT(CT, p, saturation_fraction = 1)

**Arguments**

- **CT**: Conservative Temperature [degC]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **saturation_fraction**: fraction of air in water [unitless]
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

**Value**

Absolute Salinity [ g/kg ]

**References**


**Examples**

```r
CT <- c(-0.11901, -0.15608, -0.72138, -1.97738, -2.31728, -2.56764)
p <- c(10, 50, 125, 250, 600, 1000)
saturation_fraction <- 1
SA <- gsw_SA_freezing_from_CT_poly(CT, p, saturation_fraction)
expect_equal(SA, c(2.280500648179144, 2.41686765108550, 11.973503162175106, 32.868973869711390, 34.017513292374431, 32.859871943514150))
```

**gsw_SA_freezing_from_CT_poly**

Compute Absolute Salinity at Freezing Point (Polynomial version)

**Description**

Compute Absolute Salinity at Freezing Point (Polynomial version)

**Usage**

```r
gsw_SA_freezing_from_CT_poly(CT, p, saturation_fraction = 1)
```

**Arguments**

- `CT` Conservative Temperature [ degC ]
- `p` sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- `saturation_fraction` fraction of air in water [unitless]
Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959c54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Absolute Salinity [ g/kg ]

References


Examples

CT <- c(-0.11901, -0.15608, -0.72138, -1.97738, -2.31728, -2.56764)
p <- c(10, 50, 125, 250, 600, 1000)
saturation_fraction <- 1
SA <- gsw_SA_freezing_from_CT_poly(CT, p, saturation_fraction)
expect_equal(SA, c(2.28181026779295, 2.4181343292641376, 11.971996354752958, 32.867931280363138, 34.015087798162732, 32.856434894818825))

---

gsw_SA_freezing_from_t

Compute Absolute Salinity at Freezing in-situ Temperature

Description

Compute Absolute Salinity at Freezing in-situ Temperature

Usage

gsw_SA_freezing_from_t(t, p, saturation_fraction = 1)

Arguments

t in-situ temperature (ITS-90) [ degC ]
p sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
saturation_fraction fraction of air in water [unitless]
Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Absolute Salinity [ g/kg ]

References


Examples

t <- c(-0.11901, -0.15608, -0.72138, -1.97738, -2.31728, -2.56764)
p <- c(-50, 50, 125, 250, 600, 1000)
saturation_fraction <- 1
SA <- gsw_SA_freezing_from_t_poly(t, p, saturation_fraction)
expect_equal(SA, c(2.01579844008186, 2.156742019102164, 11.679080083422074, 32.844195564019278, 34.138949682974413, 33.100945437175568))
gsw_SA_from_rho

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f66e67fa4f5bfe5'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Absolute Salinity [ g/kg ]

References


Examples

t <- c(-0.11901, -0.15608, -0.72138, -1.97738, -2.31728, -2.56764)
p <- c(10, 50, 125, 250, 600, 1000)
saturation_fraction <- 1
SA <- gsw_SA_freezing_from_t_poly(t, p, saturation_fraction)
expect_equal(SA, c(2.017072489768256, 2.151989342038462, 11.677649626115608, 32.84312811499026, 34.136459306273451, 33.0974275226251822))

Description

Compute Absolute Salinity from Density, etc

Usage

gsw_SA_from_rh(rho, CT, p)

Arguments

rho  
seawater density [ kg/m^3 ]

CT  
Conservative Temperature [ degC ]

p  
sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
gsw_SA_from_SP

Details
The present R function works with a wrapper to a C function contained within the GSW-C system
(Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git com-
mit '5b4d959e54031f9e972f3e863f63e67fa4f5bfc'), which stems from the GSW-Fortran system
(https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab sys-
more about these software systems, their authorships, and the science behind it all.

Value
Absolute Salinity [ g/kg ]

References

See Also
Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_alpha_on_beta,
gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta,
gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_first_derivatives,
gsw_rho_ice, gsw_rho_t_exact, gsw_rho, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3,
gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact,
gsw_specvol

Examples
rho <- c(1021.8482, 1022.2647, 1024.4207, 1027.7841, 1029.8287, 1031.9916)
CT <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
SA <- gsw_SA_from_rho(rho, CT, p)
expect_equal(SA, c(34.71200120418108, 34.89172388488869, 35.02622257609505,
34.84716042234572, 34.736398269039945, 34.732228881079742))

Description
Calculate Absolute Salinity from Practical Salinity, pressure, longitude, and latitude.

Usage
gsw_SA_from_SP(SP, p, longitude, latitude)
Arguments

`SP`  
Practical Salinity (PSS-78) [ unitless ]

`p`  
sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

`longitude`  
longitude in decimal degrees, positive to the east of Greenwich. (This is called long in the TEOS-10 Matlab code.)

`latitude`  
latitude in decimal degrees, positive to the north of the equator. (This is called lat in the TEOS-10 Matlab code.)

Details

If `SP` is a matrix and if its dimensions correspond to the lengths of longitude and latitude, then the latter are converted to analogous matrices with `expand.grid`.

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e86f3f63e6f7fa4f5b6ec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

Value

Absolute Salinity [ g/kg ]

References


See Also

Other things related to salinity: `gsw_C_from_SP`, `gsw_SA_from_SP_Baltic`, `gsw_SA_from_Sstar`, `gsw_SP_from_C`, `gsw_SP_from_SA`, `gsw_SP_from_SK`, `gsw_SP_from_SR`, `gsw_SP_from_Sstar`, `gsw_SR_from_SP`, `gsw_Sstar_from_SA`, `gsw_Sstar_from_SP`, `gsw_deltaSA_from_SP`

Examples

```r
SP <- c(34.5487, 34.7275, 34.8605, 34.6810, 34.5680, 34.5600)
p <- c( 10,  50,  125,  250,  600, 1000)
lat <- c(  4,   4,   4,   4,   4,   4)
long <- c( 188, 188, 188, 188, 188, 188)
SA <- gsw_SA_from_SP(SP, p, long, lat)
expect_equal(SA, c(34.711778344814114, 34.891522618230098, 35.025544862476920,
                  34.847229026189588, 34.736628474576851, 34.732363065590846))
```
gsw_SA_from_SP_Baltic  Convert from Practical Salinity to Absolute Salinity (Baltic)

Description

Calculate Absolute Salinity from Practical Salinity, pressure, longitude, and latitude.

Usage

gsw_SA_from_SP_Baltic(SP, longitude, latitude)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>Practical Salinity (PSS-78) [ unitless ]</td>
</tr>
<tr>
<td>longitude</td>
<td>longitude in decimal degrees, positive to the east of Greenwich. (This is called long in the TEOS-10 Matlab code.)</td>
</tr>
<tr>
<td>latitude</td>
<td>latitude in decimal degrees, positive to the north of the equator. (This is called lat in the TEOS-10 Matlab code.)</td>
</tr>
</tbody>
</table>

Details

If SP is a matrix and if its dimensions correspond to the lengths of longitude and latitude, then the latter are converted to analogous matrices with expand.grid.

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e86363e67e4f53bc'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Absolute Salinity [ g/kg ]

References


See Also

Other things related to salinity: gsw_C_from_SP, gsw_SA_from_SP, gsw_SA_from_Sstar, gsw_SP_from_C, gsw_SP_from_SA, gsw_SP_from_SK, gsw_SP_from_SR, gsw_SP_from_Sstar, gsw_SR_from_SP, gsw_Sstar_from_SA, gsw_Sstar_from_SP, gsw_deltaSA_from_SP
Examples

```r
SP <- c( 6.5683, 6.6719, 6.8108, 7.2629, 7.4825, 10.2796)
lon <- c( 20, 20, 20, 20, 20, 20)
lat <- c( 59, 59, 59, 59, 59, 59)
SA <- gsw_SA_from_SP_Baltic(SP, lon, lat)
expect_equal(SA, c(6.669945432342856, 6.773776430742856, 6.912986138057142,
5.366094191885713, 7.586183837142856, 10.389520570971428))
```

---

**gsw_SA_from_Sstar**  
Absolute Salinity from Preformed Salinity

**Description**

Calculate Absolute Salinity from Preformed Salinity, pressure, longitude, and latitude.

**Usage**

```r
gsw_SA_from_Sstar(Sstar, p, longitude, latitude)
```

**Arguments**

- **Sstar**  
  Preformed Salinity [ g/kg ]
- **p**  
  sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **longitude**  
  longitude in decimal degrees, positive to the east of Greenwich. (This is called `long` in the TEOS-10 Matlab code.)
- **latitude**  
  latitude in decimal degrees, positive to the north of the equator. (This is called `lat` in the TEOS-10 Matlab code.)

**Details**

If Sstar is a matrix and if its dimensions correspond to the lengths of longitude and latitude, then the latter are converted to analogous matrices with `expand.grid`.

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e7fa4f5bfc'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

Absolute Salinity [ g/kg ]

**References**

See Also

Other things related to salinity: `gsw_C_from_SP`, `gsw_SA_from_SP`, `gsw_SP_from_C`, `gsw_SP_from_SA`, `gsw_SP_from_SK`, `gsw_SP_from_SR`, `gsw_SP_from_Sstar`, `gsw_SR_from_SP`, `gsw_Sstar_from_SA`, `gsw_Sstar_from_SP`, `gsw_deltasA_from_SP`

Examples

```r
SP <- c(34.7115, 34.8912, 35.0247, 34.8436, 34.7291, 34.7197)
p <- c(10, 50, 125, 250, 600, 1000)
lat <- c(4, 4, 4, 4, 4, 4)
long <- c(188, 188, 188, 188, 188, 188)
SA <- gsw_SA_from_Sstar(SP, p, long, lat)
expect_equal(SA, c(34.71724663585905, 34.891561223296009, 35.025594598699882, 34.847235885385913, 34.736694493054166, 34.73238711902753))
```

---

gsw_seaice_fraction_to_freeze_seawater

*Sea ice Fraction to Cool Seawater to Freezing*

Description

Sea ice Fraction to Cool Seawater to Freezing

Usage

```r
gsw_seaice_fraction_to_freeze_seawater(SA, CT, p, SA_seaice, t_seaice)
```

Arguments

- **SA**: Absolute Salinity [ g/kg ]
- **CT**: Conservative Temperature [ degC ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **SA_seaice**: Absolute Salinity of sea ice [ g/kg ]
- **t_seaice**: initial temperature of sea ice [ degC ]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfc8'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing `SA_freeze`, `CT_freeze` and `w_Ih`. 

---
**gsw_sigma0**

**Potential density anomaly referenced to 0 dbar**

### Description

This uses the 75-term density equation, and returns potential density referenced to a pressure of 0 dbar, minus 1000 kg/m^3.

### Usage

```r
gsw_sigma0(SA, CT)
```

### Arguments

- **SA**  
  Absolute Salinity [ g/kg ]
- **CT**  
  Conservative Temperature [ degC ]

### Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

### Value

Potential density anomaly [ kg/m^3 ]


gsw_sigma1

References

http://www.teos-10.org/pubs/gsw/html/gsw_sigma0.html

See Also

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_first_derivatives, gsw_rho_ice, gsw_rho_t_exact, gsw_rho, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol

Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.0899, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
sigma0 <- gsw_sigma0(SA, CT)
expect_equal(sigma0, c(21.797900819337656, 22.052215404397316, 23.892985387893923, 26.667608665972011, 27.107380455119710, 27.405748977090885))

---

**gsw_sigma1**

*Potential density anomaly referenced to 1000 dbar*

Description

This uses the 75-term density equation, and returns potential density referenced to a pressure of 1000 dbar, minus 1000 kg/m^3.

Usage

gsw_sigma1(SA, CT)

Arguments

- **SA**  
  Absolute Salinity [ g/kg ]

- **CT**  
  Conservative Temperature [ degC ]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit `5b4d959e54031b9e972f3e863f63e67fa4f5bfec`), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.
**Value**

potential density anomaly [ kg/m^3 ]

**References**


**See Also**

Other things related to density: 
gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_first_derivatives, gsw_rho_ice, gsw_rho_t_exact, gsw_rho, gsw_sigma0, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol

**Examples**

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8999, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
sigma1 <- gsw_sigma1(SA, CT)
expect_equal(sigma1, c(25.95561850310202, 26.213131422420247, 28.125423775188438, 31.120360038882382, 31.63772422273368, 32.002453224572037))
```

**Description**

This uses the 75-term density equation, and returns potential density referenced to a pressure of 2000 dbar, minus 1000 kg/m^3.

**Usage**

`gsw_sigma2(SA, CT)`

**Arguments**

- **SA** Absolute Salinity [ g/kg ]
- **CT** Conservative Temperature [ degC ]

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e5403119e972f3e863f63e67fa4f5f6c'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.
Value

potential density anomaly [ kg/m^3 ]

References


See Also

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_first_derivatives, gsw_rho_ice, gsw_rho_t_exact, gsw_rho, gsw_rho0, gsw_sigma0, gsw_sigma1, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol

Examples

```
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
sigma2 <- gsw_sigma2(SA, CT)
expect_equal(sigma2, c(30.023152223799116, 30.28378336283477, 32.26555684029719, 35.474550881051073, 36.067289438047737, 36.492606494879510))
```

Description

This uses the 75-term density equation, and returns potential density referenced to a pressure of 3000 dbar, minus 1000 kg/m^3.

Usage

gsw_sigma3(SA, CT)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>CT</td>
<td>Conservative Temperature [ degC ]</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e540319e972f3e863f6367fa4f5bfc'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.
Value
potential density anomaly with reference pressure 3000 dbar [ kg/m^3 ]

References

See Also
Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, 
gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, 
gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_first_derivatives 
gsw_rho_ice, gsw_rho_t_exact, gsw_rho, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma4, 
gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, 
gsw_specvol

Examples
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8999, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
sigma3 <- gsw_sigma3(SA, CT)
expect_equal(sigma3, c(34.003747849093675, 34.267409891564057, 36.316415829697917, 
39.73236769377039, 40.397934186745033, 40.881795690566832))

Description
This uses the 75-term density equation, and returns potential density referenced to a pressure of 
4000 dbar, minus 1000 kg/m^3.

Usage
gsw_sigma4(SA, CT)

Arguments
SA Absolute Salinity [ g/kg ]
CT Conservative Temperature [ degC ]

Details
The present R function works with a wrapper to a C function contained within the GSW-C system 
(Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e5403129e972f3e863f63e67fa4f58bfc'), which stems from the GSW-Fortran system 
(https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system 
more about these software systems, their authorships, and the science behind it all.
gsw_sound_speed

Value

potential density anomaly with reference pressure 4000 dbar [ kg/m^3 ]

References


See Also

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, gsw_pot_rho_t_exact, gsw_rhalpha beta, gsw_rhalpha_first_derivatives_wrt_enthalpy, gsw_rhalpha_first_derivatives_wrt_hydrostatic_pressure, gsw_rhalpha_ice, gsw_rhalpha_t_exact, gsw_rhalpha, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol

Examples

SA <- c(34.7118, 34.8915, 35.8256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
sigma4 <- gsw_sigma4(SA, CT)
expect_equal(sigma4, c(37.900374609834898, 38.166979617032439, 40.280876075282549, 43.89609103421953, 44.631677245327637, 45.17181731202039))

gsw_sound_speed  Sound speed

Description

Speed of sound in seawater, using the 75-term equation for specific volume.

Usage

gsw_sound_speed(SA, CT, p)

Arguments

SA  Absolute Salinity [ g/kg ]
CT  Conservative Temperature [ degC ]
p  sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e540319e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.
Value

sound speed [ m/s ]

References


See Also

Other things related to sound: gsw_sound_speed_ice, gsw_sound_speed_t_exact

Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
speed <- gsw_sound_speed_ice(SA, CT, p)
expect_equal(speed/1e3, c(1.542426412426373, 1.542558891663385, 1.530001535436184,
1.494551099295314, 1.487622786765276, 1.484271672296205))

---

gsw_sound_speed_ice      Sound speed in ice

Description

Speed of sound in ice.

Usage

gsw_sound_speed_ice(t, p)

Arguments

- `t`  
in-situ temperature (ITS-90) [ degC ]
- `p`  
sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

sound speed [ m/s ]
gsw_sound_speed_t_exact

References


See Also

Other things related to sound: gsw_sound_speed_t_exact, gsw_sound_speed

Examples

t <- c(-10.7856, 13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
p <- c(10, 50, 125, 250, 600, 1000)
speed <- gsw_sound_speed_t_exact(t, p)
expect_equal(speed/1e3, c(3.111311360346254, 3.116492565497544, 3.115833462003452,
3.115637032488204, 3.115377253092692, 3.113321384499191))

Description

Sound Speed in Seawater

Usage

gsw_sound_speed_t_exact(SA, t, p)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>t</td>
<td>in-situ temperature (ITS-90) [ degC ]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5b4c'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

sound speed [ m/s ]
Specific Volume of Seawater

Description

Specific Volume of Seawater

Usage

gsw_specvol(SA, CT, p)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>CT</td>
<td>Conservative Temperature [ degC ]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Specific volume (1/density)
gsw_specvol_alpha_beta

References


See Also

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta,
gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta,
gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_first_derivatives,
gsw_rho_ice, gsw_rho_t_exact, gsw_rho, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3,
gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact

Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8999, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
p <- c(10, 50, 125, 250, 600, 1000)
specvol <- gsw_specvol(SA, CT, p)
expect_equal(specvol*1e3, c(0.978626852431313, 0.97822365781325, 0.976155264597929,
0.97296125801157, 0.971026719344908, 0.968989944622149))

---

gsw_specvol_alpha_beta

Specific Volume, alpha, and beta

Description

Specific Volume, alpha, and beta

Usage

gsw_specvol_alpha_beta(SA, CT, p)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [g/kg]</td>
</tr>
<tr>
<td>CT</td>
<td>Conservative Temperature [degC]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e5403119e972f3e863f63e67fa4f5bfe'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.
Value

a list holding specvol, the specific volume [ m$^3$/kg ], alpha, the thermal expansion coefficient [ 1/degC ], and beta, the haline contraction coefficient [ kg/g ].

References


See Also

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_first_derivatives, gsw_rho_ice, gsw_rho_t_exact, gsw_rho, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol

Examples

```r
SA <- c(34.7118, 34.8915, 35.8256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
p <- c(10, 50, 125, 250, 600, 1000)
r <- gsw_specvol_anom_standard(SA, CT, p)

equal(r$specvol/1e-3, c(0.978626852431313, 0.978222365701325, 0.976155264597929,
0.972961258011157, 0.971026719344908, 0.96898994622149))
equal(r$alpha/1e-3, c(0.324638934509245, 0.322655537959731, 0.2811457232101171,
0.173199716344780, 0.146289673594824, 0.129414845334599))
equal(r$beta/1e-3, c(0.717483987596135, 0.717647512290095, 0.726211643644768,
0.750508751749777, 0.755052064788492, 0.757050813384370))
```

---

gsw_specvol_anom_standard

Specific volume anomaly [standard] (75-term equation)

Description

Note that the TEOS function named specific_volume_anomaly is not provided in the C library, so it is not provided in R, either.

Usage

gsw_specvol_anom_standard(SA, CT, p)

Arguments

- **SA**: Absolute Salinity [ g/kg ]
- **CT**: Conservative Temperature [ degC ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e543f19e972f3e863f6367fa4f5bfe'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Specific volume anomaly [m^3/kg]

References


See Also

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_first_derivatives, gsw_rho_ice, gsw_rho_t_exact, gsw_rho, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_ice, gsw_specvol_t_exact, gsw_specvol,

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 20.2262, 6.8272, 4.3236)
p  <- c(  10,  50, 125, 250,  600, 1000)
a  <- gsw_specvol_anom_standard(SA, CT, p)
expect_equal(a*1e5, c(0.601051894897400, 0.578609769250563, 0.405600538950092,
                      0.142190453761838, 0.104335535578967, 0.076383389577725))
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>CT</td>
<td>Conservative Temperature [ degC ]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f6e367fa4f5f6ec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing $v_{SA}$ [(m^3/kg)/(g/kg)], the derivative of specific volume with respect to Absolute Salinity, $v_{CT}$ [(m^3/kg)/degC], the derivative of specific volume with respect to Conservative Temperature, and $v_{p}$ [(m^3/kg)/dbar], the derivative of specific volume with respect to pressure. (Note that the last quantity is denoted $v_{p}$ in the documentation for the Matlab function.)

References


See Also

Other things related to enthalpy: `gsw_CT_from_enthalpy`, `gsw_dynamic_enthalpy`, `gsw_enthalpy_CT_exact`, `gsw_enthalpy_diff`, `gsw_enthalpy_first_derivatives_CT_exact`, `gsw_enthalpy_first_derivatives`, `gsw_enthalpy_ice`, `gsw_enthalpy_t_exact`, `gsw_enthalpy`, `gsw_frazil_properties_potential_poly`, `gsw_frazil_properties_potential`, `gsw_pot_enthalpy_from_pt_ice_poly`, `gsw_pot_enthalpy_from_pt_ice`, `gsw_pot_enthalpy_ice_freezing_poly`, `gsw_pot_enthalpy_ice_freezing`, `gsw_pt_from_pot_enthalpy_ice_poly`, `gsw_pt_from_pot_enthalpy_ice`, `gsw_specvol_first_derivatives_wrt_enthalpy`

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.6272, 4.3236)
p <- c(10, 50, 125, 250, 600, 1000)
r <- gsw_specvol_first_derivatives(SA, CT, p)
expect_equal(r$v_SA/1e-6, c(-0.702149096451073, -0.702018847212088, -0.708895319156155, -0.730208155560782, -0.733175729486169, -0.733574625737474))
expect_equal(r$v_CT/1e-6, c(0.317700378655437, 0.315628536349601, 0.274441778380000, 0.168516613901993, 0.142051181824820, 0.125401683814057))
expect_equal(r$v_p/1e-12, c(-0.4802527990904794, -0.402146232553089, -0.406663124765787, -0.423877042622481, -0.426198431093548, -0.426390351853655))
```
First Derivatives of Specific Volume wrt Enthalpy

Usage

gsw_specvol_first_derivatives_wrt_enthalpy(SA, CT, p)

Arguments

SA  Absolute Salinity [ g/kg ]
CT  Conservative Temperature [ degC ]
p   sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e86363e67fa4f5beec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing v_SA_wrt_h and v_h.

References


See Also

Other things related to enthalpy: gsw_CT_from_enthalpy, gsw_dynamic_enthalpy, gsw_enthalpy_CT_exact, gsw_enthalpy_diff, gsw_enthalpy_first_derivatives_CT_exact, gsw_enthalpy_first_derivatives, gsw_enthalpy_ice, gsw_enthalpy_t_exact, gsw_enthalpy, gsw_frazil_properties_potential_poly, gsw_frazil_properties_potential, gsw_pot_enthalpy_from_pt_ice_poly, gsw_pot_enthalpy_from_pt_ice, gsw_pot_enthalpy_ice_freezing_poly, gsw_pot_enthalpy_ice_freezing, gsw_pt_from_pot_enthalpy_ice_poly, gsw_pt_from_pot_enthalpy_ice, gsw_specvol_first_derivatives
gsw_specvol_ice

Specific Volume of Ice

Description

Specific Volume of Ice

Usage

```r
  gsw_specvol_ice(t, p)
```

Arguments

- `t` in-situ temperature (ITS-90) [degC]
- `p` sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5b6fc'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Specific volume [m³/kg]

References

See Also

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_first_derivatives, gsw_rho_ice, gsw_rho_t_exact, gsw_rho, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_t_exact, gsw_specvol

Examples

t <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
p <- c(10, 50, 125, 250, 600, 1000)
v <- gsw_specvol_ice(t, p)
expect_equal(v, c(0.0010885982980677, 0.00108848945509, 0.001088499019939, 0.001088433747301, 0.001088223220685, 0.001088135464776))

---

**gsw_specvol_second_derivatives**

*Second Derivatives of Specific Volume*

Description

Second Derivatives of Specific Volume

Usage

gsw_specvol_second_derivatives(SA, CT, p)

Arguments

SA Absolute Salinity [g/kg]
CT Conservative Temperature [degC]
p sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.
Value
A list containing \( \text{specvol}_{\text{SA}}_{\text{SA}} \) \( \begin{bmatrix} (\text{m}^3/\text{kg})/\begin{bmatrix} \text{g}/\text{kg} \end{bmatrix}^2 \end{bmatrix} \), the second derivative of specific volume with respect to Absolute Salinity, \( \text{specvol}_{\text{SA}}_{\text{CT}} \) \( \begin{bmatrix} (\text{m}^3/\text{kg})/\begin{bmatrix} \text{g}/\text{kg} \end{bmatrix}/\text{degC} \end{bmatrix} \), the derivative of specific volume with respect to Absolute Salinity and Conservative Temperature, \( \text{specvol}_{\text{CT}}_{\text{CT}} \) \( \begin{bmatrix} (\text{m}^3/\text{kg})/\text{degC}^2 \end{bmatrix} \), the second derivative of specific volume with respect to Conservative Temperature, \( \text{specvol}_{\text{SA}}_{\text{P}} \) \( \begin{bmatrix} (\text{m}^3/\text{kg})/\begin{bmatrix} \text{g}/\text{kg} \end{bmatrix}/\text{dbar} \end{bmatrix} \), the derivative of specific volume with respect to Absolute Salinity and pressure, and \( \text{specvol}_{\text{CT}}_{\text{P}} \) \( \begin{bmatrix} (\text{m}^3/\text{kg})/\begin{bmatrix} \text{K} \end{bmatrix}/\text{dbar} \end{bmatrix} \), the derivative of specific volume with respect to Conservative Temperature and pressure.

References

Examples
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
r <- gsw_specvol_second_derivatives(wrt_enthalpy(SA, CT, p))
expect_equal(r$\text{specvol}_{\text{SA}}_{\text{SA}}/1e-8, c(0.000906777599140, 0.000915086639384, 0.004568844270812, 0.096725108896007, 0.09911765836648, 0.10030227946072))
expect_equal(r$\text{specvol}_{\text{SA}}_{\text{CT}}/1e-8, c(0.129965321178084, 0.130523953162130, 0.149555815430615, 0.217023290441810, 0.233892039070486, 0.243659989480325))
expect_equal(r$\text{specvol}_{\text{CT}}_{\text{CT}}/1e-7, c(0.071409582006642, 0.071582962051991, 0.077436153664104, 0.09532936274850, 0.100105336953738, 0.103044572835472))
expect_equal(r$\text{specvol}_{\text{SA}}_{\text{P}}/1e-14, c(0.141281359467752, 0.141507584673426, 0.147247234588907, 0.164580347761218, 0.168069801298412, 0.169948275518754))
expect_equal(r$\text{specvol}_{\text{CT}}_{\text{P}}/1e-14, c(0.085542828707964, 0.086723632576213, 0.112156562396990, 0.188269835959500, 0.21161556759369, 0.22860957504991))
Details
The present R function works with a wrapper to a C function contained within the GSW-C system
(Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git com-
mmit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system
(https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab sys-
more about these software systems, their authorships, and the science behind it all.

Value
A list containing specvol_SA_SA [ (m^3/kg)/(g/kg)^2 ], the second derivative of specific volume
with respect to Absolute Salinity, specvol_SA_h [ (m^3/kg)/(g/kg)/(J/kg) ], the derivative of spe-
cific volume with respect to Absolute Salinity and enthalpy, and specvol_h_h [ (m^3/kg)/(J/kg)^2
], the second derivative of specific volume with respect to enthalpy.

Bugs
As of March 27, 2017, the test values listed in “Examples” do not match values provided at the
TEOS-10 website listed in “References”, but they match with values given by the Matlab code that is
provided on the TEOS-10 website. It is expected that the TEOS-10 website will be updated by
May 2017. As those updates to the TEOS-10 website become available, the present comment will
be revised or deleted.

References
html

See Also
Other functions with suspicious test values on the TEOS-10 website: gsw_entropy_second_derivatives,
gsw_rho_second_derivatives_wrt_enthalpy, gsw_t_freezing_first_derivatives_poly

Examples
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
p <- c(10, 50, 125, 250, 600, 1000)
r <- gsw_specvol_second_derivatives_wrt_enthalpy(SA, CT, p)
expect_equal(r$specvol_SA_SA/1e-8, c(0.080900028996264, 0.080937999675000, 0.084663065647101,
0.09697364985384, 0.09972745342293, 0.10135303979356))
expect_equal(r$specvol_SA_h/1e-12, c(0.325437133570796, 0.327060462851431, 0.375273569184178,
0.54518833073084, 0.58942488189351, 0.616101548209175))
expect_equal(r$specvol_h_h/1e-15, c(0.447949998681476, 0.449121446914278, 0.485998151346315,
0.595980711669896, 0.628708349875318, 0.647433212216398))
Specific Volume of Seawater

Usage

gsw_specvol_t_exact(SA, t, p)

Arguments

- **SA**: Absolute Salinity [ g/kg ]
- **t**: in-situ temperature (ITS-90) [ degC ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

Value

Specific volume [ m³/kg ]

References


See Also

Other things related to density: gsw_CT_from_rho, gsw_CT_maxdensity, gsw_SA_from_rho, gsw_alpha_on_beta, gsw_alpha_wrt_t_exact, gsw_alpha_wrt_t_ice, gsw_alpha, gsw_beta_const_t_exact, gsw_beta, gsw_pot_rho_t_exact, gsw_rho_alpha_beta, gsw_rho_first_derivatives_wrt_enthalpy, gsw_rho_first_derivatives, gsw_rho_ice, gsw_rho_t_exact, gsw_rho, gsw_sigma0, gsw_sigma1, gsw_sigma2, gsw_sigma3, gsw_sigma4, gsw_specvol_alpha_beta, gsw_specvol_anom_standard, gsw_specvol_ice, gsw_specvol
**Examples**

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
t <- c(28.7856, 28.4329, 22.8103, 10.2600, 6.8863, 4.4036)
p <- c(10, 50, 125, 250, 600, 1000)
v <- gsw_spiciness0(SA, t, p)
expect_equal(v*1e3, c(0.978626625025472, 0.978222143734527, 0.976154768597586,
0.972961211575438, 0.971826779948624, 0.96898990731808))
```

**Description**

Calculate seawater spiciness referenced to 0 dbar (i.e. the surface).

**Usage**

```r
gsw_spiciness0(SA, CT)
```

**Arguments**

- **SA** Absolute Salinity [g/kg]
- **CT** Conservative Temperature [degC]

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit ’5b4d959c54031f9e972f3e863f63e67fa4f5bfec’), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

**Value**

spiciness [kg/m^3]

**References**

http://www.teos-10.org/pubs/gsw/html/gsw_spiciness0.html

**Examples**

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
spiciness <- gsw_spiciness0(SA, CT)
expect_equal(spiciness, c(5.728998558542941, 5.749940496782486, 4.163547112671111,
1.069362556641764, 0.426428274444305, 0.089725188494086))
```
Description

Calculate seawater spiciness referenced to 1000 dbar.

Usage

gsw_spiciness1(sa, ct)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>CT</td>
<td>Conservative Temperature [ degC ]</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

spiciness [ kg/m^3 ]

References


See Also

Other things related to spiciness: gsw_spiciness2

Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
spiciness <- gsw_spiciness1(SA, CT)
expect_equal(spiciness, c(6.31103832212324, 6.326411175472160, 4.667218659743284, 1.351722468726905, 0.628494082166029, 0.224779784908478))
Description

Calculate seawater spiciness referenced to 2000 dbar.

Usage

gsw_spiciness2(SA, CT)

Arguments

SA  Absolute Salinity [ g/kg ]
CT  Conservative Temperature [ degC ]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

spiciness [ kg/m^3 ]

References


See Also

Other things related to spiciness: gsw_spiciness1

Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
spiciness <- gsw_spiciness2(SA, CT)
expect_equal(spiciness, c(6.874671751873180, 6.884616399155135, 5.154458892387083, 1.624327800598636, 0.823490797424952, 0.355069307641827))
gsw_SP_from_C  
Convert from Electrical Conductivity to Practical Salinity

**Description**

Convert from Electrical Conductivity to Practical Salinity

**Usage**

gsw_SP_from_C(C, t, p)

**Arguments**

- **C**: conductivity [mS/cm]
- **t**: in-situ temperature (ITS-90) [degC]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

Practical Salinity (PSS-78) [unitless]

**References**


**See Also**

Other things related to salinity: `gsw_C_from_SP`, `gsw_SA_from_SP_Baltic`, `gsw_SA_from_SP`, `gsw_SA_from_Sstar`, `gsw_SP_from_SA`, `gsw_SP_from_SK`, `gsw_SP_from_SR`, `gsw_SP_from_Sstar`, `gsw_SR_from_SP`, `gsw_Sstar_from_SA`, `gsw_Sstar_from_SP`, `gsw_deltaSA_from_SP`

Other things related to conductivity: `gsw_C_from_SP`
Examples

\[
\begin{align*}
C & \leftarrow c(34.5487, 34.7275, 34.8605, 34.6810, 34.5680, 34.5600) \\
t & \leftarrow c(28.7856, 28.4329, 22.8130, 10.2600, 6.8863, 4.4036) \\
p & \leftarrow c(10, 50, 125, 250, 600, 1000) \\
SP & \leftarrow gsw\_SP\_from\_SA(C, t, p) \\
expect\_equal(SP, c(20.000869599086951, 20.265511864874270, 22.981513062527689, 31.204503263727982, 34.032315787432829, 36.400308494388170))
\end{align*}
\]

Description

Calculate Practical Salinity from Absolute Salinity, pressure, longitude, and latitude.

Usage

\[
gsw\_SP\_from\_SA(SA, p, longitude, latitude)
\]

Arguments

- **SA**: Absolute Salinity [g/kg]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **longitude**: longitude in decimal degrees, positive to the east of Greenwich. (This is called long in the TEOS-10 Matlab code.)
- **latitude**: latitude in decimal degrees, positive to the north of the equator. (This is called lat in the TEOS-10 Matlab code.)

Details

If SP is a matrix and if its dimensions correspond to the lengths of longitude and latitude, then the latter are converted to analogous matrices with `expand.grid`.

Note: unlike the corresponding Matlab function, this does not return a flag indicating whether the location is in the ocean.

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git committ '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

Practical Salinity (PSS-78) [unitless]
Calculate Practical Salinity from Knudsen Salinity

**Description**

Calculate Practical Salinity from Knudsen Salinity

**Usage**

```r
gsw_SP_from_SK(SK)
```

**Arguments**

- **SK**
  - Knudsen Salinity [ parts per thousand, ppt ]

**Value**

Practical Salinity (PSS-78) [ unitless ]

**References**


**See Also**

Other things related to salinity: `gsw_C_from_SP`, `gsw_SA_from_SP_Baltic`, `gsw_SA_from_SP`, `gsw_SA_from_Sstar`, `gsw_SP_from_C`, `gsw_SP_from_SA`, `gsw_SP_from_SR`, `gsw_SP_from_Sstar`, `gsw_SR_from_SP`, `gsw_Sstar_from_SA`, `gsw_Sstar_from_SP`, `gsw_deltaSA_from_SP`
Examples

```r
SK <- c(34.5487, 34.7275, 34.8605, 34.6810, 34.5600, 34.5600)
SP <- gsw_SP_from_SK(SK)
expect_equal(SP, c(34.548342096952908, 34.72795637119113, 34.860409847645435,
                  34.680755706371187, 34.567658670360110, 34.559651800554022))
```

Description

Calculate Practical Salinity from Reference Salinity

Usage

```r
gsw_SP_from_SR(SR)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR</td>
<td>Reference Salinity [ g/kg ]</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959c54031f9e972f3e863f63e67fa4f5f6fc'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

Value

Practical Salinity (PSS-78) [ unitless ]

References


See Also

Other things related to salinity: `gsw_C_from_SP`, `gsw_SA_from_SP_Baltic`, `gsw_SA_from_SP`, `gsw_SA_from_Sstar`, `gsw_SP_from_C`, `gsw_SP_from_SA`, `gsw_SP_from_SK`, `gsw_SP_from_Sstar`, `gsw_SR_from_SP`, `gsw_Sstar_from_SA`, `gsw_Sstar_from_SP`, `gsw_deltaSA_from_SP`
Examples
SR <- c(34.5487, 34.7275, 34.8605, 34.6810, 34.5600, 34.5600)
SP <- gsw_SP_from_sr(SR)
expect_equal(SP, c(34.38552667080714, 34.564513505458834, 34.696889296869848,
34.518231743800094, 34.405762086435850, 34.39779632817147))

Description
Practical Salinity from Preformed Salinity

Usage
gsw_SP_from_Sstar(Sstar, p, longitude, latitude)

Arguments
Sstar Preformed Salinity \[ \text{g/kg} \]
p sea pressure \[ \text{dbar} \], i.e. absolute pressure \[ \text{dbar} \] minus 10.1325 dbar
longitude longitude in decimal degrees, positive to the east of Greenwich. (This is called long in the TEOS-10 Matlab code.)
latitude latitude in decimal degrees, positive to the north of the equator. (This is called lat in the TEOS-10 Matlab code.)

Details
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfe'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value
Practical Salinity (PSS-78) \[ \text{unitless} \]

References

See Also
Other things related to salinity: gsw_C_from_SP, gsw_SA_from_SP_Baltic, gsw_SA_from_SP, gsw_SA_from_Sstar, gsw_SP_from_C, gsw_SP_from_SA, gsw_SPI_from_SK, gsw_SP_from_SR, gsw_SR_from_SP, gsw_Sstar_from_SA, gsw_Sstar_from_SP, gsw_deltaSA_from_SP
**Examples**

```r
gsw_SR_from_SP
```

**Description**

Calculate Reference Salinity from Practical Salinity

**Usage**

```r
gsw_SR_from_SP(SP)
```

**Arguments**

- `SP` Practical Salinity (PSS-78) [unitless]

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e86f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

Reference Salinity [g/kg]

**References**


**See Also**

Other things related to salinity: `gsw_C_from_SP`, `gsw_SA_from_SP_Baltic`, `gsw_SA_from_SP`, `gsw_SA_from_Sstar`, `gsw_SP_from_C`, `gsw_SP_from_SA`, `gsw_SP_from_SK`, `gsw_SP_from_SR`, `gsw_SP_from_Sstar`, `gsw_Sstar_from_SA`, `gsw_Sstar_from_SP`, `gsw_deltaSA_from_SP`
Examples

```r
SP <- c(34.5487, 34.7275, 34.8605, 34.6810, 34.5680, 34.5600)
SR <- gsw_sstar_from_sa(SP)
print(SR)
```

```
[ 34.71611927085727  34.891255045714303  35.024882197714305 
  34.844535778285724  34.731002934857159  34.722965211428587]
```

---

### gsw_Sstar_from_SA

**Convert from Absolute Salinity to Preformed Salinity**

#### Description

Calculate Preformed Salinity from Absolute Salinity, pressure, longitude, and latitude.

#### Usage

```r
gsw_Sstar_from_SA(sa, p, longitude, latitude)
```

#### Arguments

- `sa`: Absolute Salinity [g/kg]
- `p`: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- `longitude`: longitude in decimal degrees, positive to the east of Greenwich. (This is called `long` in the TEOS-10 Matlab code.)
- `latitude`: latitude in decimal degrees, positive to the north of the equator. (This is called `lat` in the TEOS-10 Matlab code.)

#### Details

If `SA` is a matrix and if its dimensions correspond to the lengths of longitude and latitude, then the latter are converted to analogous matrices with `expand.grid`.

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

#### Value

Preformed Salinity [g/kg]

#### References

gsw_Sstar_from_SP

See Also

Other things related to salinity: gsw_C_from_SP, gsw_SA_from_SP_Baltic, gsw_SA_from_SP, gsw_SA_from_Sstar, gsw_SP_from_C, gsw_SP_from_SA, gsw_SP_from_SK, gsw_SP_from_SR, gsw_SP_from_Sstar, gsw_SR_from_SP, gsw_Sstar_from_SP, gsw_Sstar_from_SP, gsw_deltasa_from_SP

Examples

```
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c( 10,  50, 125,  250,  600,  1000)
lat <- c(  4,  4,  4,   4,   4,   4)
long <- c(188, 188, 188, 188, 188, 188)
Sstar <- gsw_Sstar_from_SA(SA, p, long, lat)
equal(Sstar, c(34.711575335926490, 34.891138777337822, 35.024705401162166, 34.84356418358302, 34.729005527604883, 34.71971883389462))
```

---

gsw_Sstar_from_SP  Convert from Practical Salinity to Preformed Salinity

Description

Calculate Preformed Salinity from Practical Salinity, pressure, longitude, and latitude.

Usage

```
gsw_Sstar_from_SP(SP, p, longitude, latitude)
```

Arguments

- `SP` Practical Salinity (PSS-78) [unitless]
- `p` sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- `longitude` longitude in decimal degrees, positive to the east of Greenwich. (This is called long in the TEOS-10 Matlab code.)
- `latitude` latitude in decimal degrees, positive to the north of the equator. (This is called lat in the TEOS-10 Matlab code.)

Details

If SP is a matrix and if its dimensions correspond to the lengths of longitude and latitude, then the latter are converted to analogous matrices with `expand.grid`.

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit `5b4d959e54031f9e972f3e863f63e67fa4f5bfc`), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.
Thermobaric coefficient (75-term equation)

Description
Thermobaric coefficient (75-term equation)

Usage
```r
gsw_thermobaric(SA, CT, p)
```

Arguments
- **SA**: Absolute Salinity [ g/kg ]
- **CT**: Conservative Temperature [ degC ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e540319e972f3e863f63e67fa4f5bfc'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.
**gsw_Turner_Rsubrho**

**Value**

thermobaric coefficient wrt Conservative Temperature \[ 1/(K \text{ Pa}) \]

**References**


**Examples**

```r
SA <- c(34.7118, 34.8915, 35.0265, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
p <- c(10, 50, 125, 250, 600, 1000)
tb <- gsw_thermobaric(SA, CT, p)
expect_equal(tb*1e11, c(0.152618598186650, 0.153662896162852, 0.17349325875738,
                      0.23281016808414, 0.251984724005424, 0.26666303289558))
```

**gsw_Turner_Rsubrho**

*Turner Angle and Density Ratio*

**Description**

This uses the 75-term density equation. The values of Turner Angle Tu and density ratio Rrho are calculated at mid-point pressures, \( p_{\text{mid}} \).

**Usage**

```r
gsw_Turner_Rsubrho(SA, CT, p)
```

**Arguments**

- **SA**: Absolute Salinity [g/kg]
- **CT**: Conservative Temperature [degC]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e5403f19e972f3e863f63e67fa4f5b1e'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

**Value**

List containing Tu [degrees], Rsubrho [unitless], and \( p_{\text{mid}} \) [dbar]
References


Examples

SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
p <- c( 10, 50, 125, 250, 600, 1000)
r <- gsw_Turner_Rsbrho(SA, CT, p)
expect_equal(r$Tu, c(-2.063858905281147, 41.758435216784427, 47.606966981687535, 53.710351151706369, 45.52706385821527))
expect_equal(r$Rsbrho, 100*c(-0.009304335069039, -0.170564834348709, 0.219627771740757, 0.065271424662002, 1.087044054679743))
expect_equal(r$p_mid, 100*c(0.300, 0.875, 1.875, 4.250, 8.000))

### gsw_t_deriv_chem_potential_water_t_exact

Derivative of Chemical Potential of Water in Seawater wrt Temperature

#### Description

Derivative of Chemical Potential of Water in Seawater wrt Temperature

#### Usage

```r
gsw_t_deriv_chem_potential_water_t_exact(SA, t, p)
```

#### Arguments

- **SA**: Absolute Salinity [ g/kg ]
- **t**: in-situ temperature (ITS-90) [ degC ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

#### Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959ec54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

#### Value

- **derivative**: [ J/(g*degC) ]
gsw_t_freezing

Freezing Temperature of Seawater

Description

This uses the C function named gsw_t_freezing_exact, because the C function named gsw_t_freezing does not produce check values that match the Matlab function called gsw_t_freezing (see references for those test values).

Usage

gsw_t_freezing(SA, p, saturation_fraction = 1)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
<tr>
<td>saturation_fraction</td>
<td>fraction of air in water [unitless]</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.0.5-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5fbfc'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

in-situ freezing temperature (ITS-90) [ degC ]

References

Examples

```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c(  10,  50, 125,  250, 1000)
saturation_fraction <- 1
tf <- gsw_t_freezing(SA, p, saturation_fraction)
expect_equal(tf, c(-1.902730710149803, -1.942908619287183, -2.006861069199743,
                    -2.090985086875259, -2.351293130342102, -2.660498762776720))
```

Description

Derivatives of Freezing Water Properties

Usage

```r
gsw_t_freezing_first_derivatives(SA, p, saturation_fraction = 1)
```

Arguments

- **SA**: Absolute Salinity [ g/kg ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- **saturation_fraction**: fraction of air in water [unitless]

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.0.5-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

Value

A list containing
- **t_freezing_SA** [ K/(g/kg) ], the derivative of freezing temperature with Absolute Salinity
- **t_freezing_p** [ K/dbar ], the derivative with respect to pressure.

References

Derivatives of Freezing Water Properties (Polynomial version)

Usage

gsw_t_freezing_first_derivatives_poly(SA, p, saturation_fraction = 1)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Absolute Salinity [ g/kg ]</td>
</tr>
<tr>
<td>p</td>
<td>sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar</td>
</tr>
<tr>
<td>saturation_fraction</td>
<td>fraction of air in water [unitless]</td>
</tr>
</tbody>
</table>

Details

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value

a list containing tfreezing_SA [ K/(g/kg) ], the derivative of freezing temperature with Absolute Salinity and tfreezing_p [ K/dbar ], the derivative with respect to pressure.
Bugs

As of March 27, 2017, the test values listed in “Examples” do not match values provided at the TEOS-10 website listed in “References”, but they match with values given by the Matlab code that is provided on the TEOS-10 website. It is expected that the TEOS-10 website will be updated by May 2017. As those updates to the TEOS-10 website become available, the present comment will be revised or deleted.

References


See Also

Other functions with suspicious test values on the TEOS-10 website: gsw_entropy_second_derivatives, gsw_rhoe_second_derivatives_wrt_enthalpy, gsw_specvol_second_derivatives_wrt_enthalpy

Examples

```r
SA <- c( 34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
p <- c( 10, 50, 125, 250, 600, 1000)
saturation_fraction <- c( 1, 0.8, 0.6, 0.5, 0.4, 0)
derivs <- gsw_t_freezing_first_derivatives_poly(SA, p, saturation_fraction)
expect_equal(derivs$tfreezing_SA, c(-0.056810211094078, -0.056855567524973, -0.056901968693345, -0.056903498206432, -0.056975157476629, -0.057083526206200))
expect_equal(derivs$tfreezing_p/1e-7, c(-0.748987354878138, -0.75028853857513, -0.752676389629787, -0.756549680608529, -0.767482625710990, -0.779985619685683))
```

---

gsw_t_from_CT  

**In situ temperature from Conservative Temperature**

Description

In situ temperature from Conservative Temperature

Usage

```r
gsw_t_from_CT(SA, CT, p)
```

Arguments

- **SA**  
  Absolute Salinity [ g/kg ]
- **CT**  
  Conservative Temperature [ degC ]
- **p**  
  sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
Details
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.

Value
in-situ temperature (ITS-90) [ degC ]

References

Examples
```r
SA <- c(34.7118, 34.8915, 35.0256, 34.8472, 34.7366, 34.7324)
CT <- c(28.8099, 28.4392, 22.7862, 10.2262, 6.8272, 4.3236)
p <- c(10, 50, 125, 250, 600, 1000)
t <- gsw_t_from_CT(SA, CT, p)
expect_equal(t, c(28.785580227725703, 28.432872246163946, 22.810323087627076, 10.260010752788906, 6.886286301829376, 4.403624452383043))
```

---

**gsw_t_from_pt0_ice**

*In situ Temperature from Potential Temperature at 0dbar*

Description
In situ Temperature from Potential Temperature at 0bar

Usage
```r
gsw_t_from_pt0_ice(pt0_ice, p)
```

Arguments
- **pt0_ice**: potential temperature of ice (ITS-90) [ degC ]
- **p**: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar

Details
The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at https://github.com/TEOS-10/GSW-C, as git commit '5b4d959e54031f9e972f3e863f63e67fa4f5bfec'), which stems from the GSW-Fortran system (https://github.com/TEOS-10/GSW-Fortran) which in turn stems from the GSW-Matlab system (https://github.com/TEOS-10/GSW-Matlab). Consult http://www.teos-10.org to learn more about these software systems, their authorships, and the science behind it all.
**gsw_z_from_p**

**Value**

in-situ temperature (ITS-90) [degC]

**References**

http://www.teos-10.org/pubs/gsw/html/gsw_t_from_pt0_ice.html

**Examples**

```r
pt0_ice <- c(-10.7856, -13.4329, -12.8103, -12.2600, -10.8863, -8.4036)
p <- c( 10,  50, 125, 250,  600, 1000)
t <- gsw_t_from_pt0_ice(pt0_ice, p)
expect_equal(t, c(-10.78312084414074, -13.42206863839141, -12.78317022330448,
                -12.205667526492839, -10.755496924674144, -8.184121042593350))
```

**Description**

Computation of height (above sea level) from pressure, using the 75-term equation for specific volume.

**Usage**

gsw_z_from_p(p, latitude)

**Arguments**

- `p`: sea pressure [dbar], i.e. absolute pressure [dbar] minus 10.1325 dbar
- `latitude`: latitude in decimal degrees, positive to the north of the equator. (This is called lat in the TEOS-10 Matlab code.)

**Details**

The present R function works with a wrapper to a C function contained within the GSW-C system (Version 3.05-4 dated 2017-08-07, available at [https://github.com/TEOS-10/GSW-C](https://github.com/TEOS-10/GSW-C), as git commit '5b4d959e5403f19e972f3e863f63e67fa4f5bfc'), which stems from the GSW-Fortran system ([https://github.com/TEOS-10/GSW-Fortran](https://github.com/TEOS-10/GSW-Fortran)) which in turn stems from the GSW-Matlab system ([https://github.com/TEOS-10/GSW-Matlab](https://github.com/TEOS-10/GSW-Matlab)). Consult [http://www.teos-10.org](http://www.teos-10.org) to learn more about these software systems, their authorships, and the science behind it all.

**Value**

height [m]
saar

References


See Also

Other things related to depth: gsw_p_from_z

Examples

z <- gsw_z_from_p(c(10, 50, 125, 250, 600, 1000), 4)
expect_equal(z/1e2, c(-0.09944583469453, -0.497180897012550, -1.242726219409978,
-2.484700576540589, -5.958253480356214, -9.920919060719987))

Description

This dataset is not intended for users, but rather for internal use within the gsw package. The dataset stores the 1.4M lookup table defined in the 8.3M file src/gsw_saar_data.c in the C library. (The .c file exceeds CRAN limitations on size.)

Details

The data are designed to replace C elements defined as below in src/gsw_saar_data.c:

```c
static int gsw_nx=91, gsw_ny=45, gsw_nz=45;
static double longs_ref[91];
static double lats_ref[45];
static double p_ref[45];
static double ndepth_ref[4095];
static double saar_ref[184275];
static double delta_sa_ref[184275];
```

R storage is in a list named saar, with elements named as in the C code, i.e. gsw_nx etc.

C storage for these variables is allocated as needed, and the data are inserted, when gsw is launched. Thus, the existing C library code "knows" about the data as local storage, which keeps alterations to the C library to a minimum.

The saar dataset was created by the following R code. The netcdf file used in this code comes from the GSW-Fortran repository (at commit baa0c09fffc7ed1f74972a1a2902d8754caab4cb) and its md5 value is dacb3f981e8e710ac2e83477701b3905.

```r
library(ncdf4)
nc <- nc_open("~/git/GSW-Fortran/test/gsw_data_v3.0.nc")
## Use as.vector() since these will all get handed into C, which does not understand matrices.
p_ref <- as.vector(ncvar_get(nc, "p_ref"))
lats_ref <- as.vector(ncvar_get(nc, "lats_ref"))
```
longs_ref <- as.vector(ncvar_get(nc, "longs_ref"))
ndepth_ref <- as.vector(ncvar_get(nc, "ndepth_ref"))
ndepth_ref[!is.finite(ndepth_ref)] <- -9e99
saar_ref <- as.vector(ncvar_get(nc, "SAAR_ref"))
saar_ref[!is.finite(saar_ref)] <- -9e99
delta_sa_ref <- as.vector(ncvar_get(nc, "deltaSA_ref"))
delta_sa_ref[!is.finite(delta_sa_ref)] <- -9e99
saar <- list(gsw_nx=gsw_nx, gsw_ny=gsw_ny, gsw_nz=gsw_nz,
              longs_ref=longs_ref, lats_ref=lats_ref, p_ref=p_ref, ndepth_ref=ndepth_ref,
              saar_ref=saar_ref, delta_sa_ref=delta_sa_ref)
save(saar, file="saar.rda")
tools::resaveRdaFiles("saar.rda")
nc_close(nc)
Index

argfix, 5
expand.grid, 122–124, 149, 154, 155
gsw, 6
gsw-package (gsw), 6
gsw_adiabatic_lapse_rate_from_CT, 7
gsw_adiabatic_lapse_rate_ice, 8
gsw_alpha, 9, 11–15, 30, 32, 93, 106, 108–111, 115, 121, 127–131, 135–137, 141, 144
gsw_alpha_on_beta, 9, 10, 12–15, 30, 32, 39, 106, 108–111, 115, 121, 127–131, 135–137, 141, 144
gsw_alpha_wrt_t_exact, 9, 11, 11, 13–15, 30, 32, 93, 106, 108–111, 115, 121, 127–131, 135–137, 141, 144
gsw_alpha_wrt_t_ice, 9, 11, 12, 12, 14, 15, 30, 32, 93, 106, 108–111, 115, 121, 127–131, 135–137, 141, 144
gsw_beta, 9, 11–13, 13, 15, 30, 32, 93, 106, 108–111, 115, 121, 127–131, 135–137, 141, 144
gsw_beta_const_t_exact, 9, 11–14, 14, 30, 32, 93, 106, 108–111, 115, 121, 127–131, 135–137, 141, 144
gsw_C_from_SP, 34, 36, 122, 123, 125, 148, 150–153, 155, 156
gsw_cabbeling, 16
gsw_chem_potential_water_ice, 17, 18
gsw_chem_potential_water_t_exact, 17, 18
gsw_cp_ice, 19
gsw_cp_t_exact, 20
gsw_CT_first_derivatives, 21
gsw_CT_first_derivatives_wrt_t_exact, 22
gsw_CT_freezing, 23
gsw_CT_freezing_first_derivatives, 24
gsw_CT_freezing_first_derivatives_poly, 25
gsw_CT_freezing_poly, 26
gsw_CT_from_enthalpy, 27, 38–42, 44, 45, 48, 56, 58, 86–88, 91, 100, 101, 138, 139
gsw_CT_from_entropy, 28, 49–52, 99
gsw_CT_from_pt, 29
gsw_CT_from_rho, 9, 11–15, 30, 32, 93, 106, 108–111, 115, 121, 127–131, 135–137, 141, 144
gsw_CT_from_t, 31
gsw_CT_maxdensity, 9, 11–15, 30, 32, 93, 106, 108–111, 115, 121, 127–131, 135–137, 141, 144
gsw_CT_second_derivatives, 33
gsw_deltaSA_from_SP, 35, 35, 122, 123, 125, 148, 150–153, 155, 156
gsw_dilution_coefficient_t_exact, 36
gsw_dynamic_enthalpy, 27, 37, 39–42, 44, 45, 48, 56, 58, 86–88, 91, 100, 101, 138, 139
gsw_enthalpy, 27, 38, 38, 40–42, 44, 45, 48, 56, 58, 86–88, 91, 100, 101, 138, 139
gsw_enthalpy_CT_exact, 27, 38, 39, 39, 41, 42, 44, 45, 48, 56, 58, 86–88, 91, 100, 101, 138, 139
gsw_enthalpy_diff, 27, 38–40, 40, 42, 44, 45, 48, 56, 58, 86–88, 91, 100, 101, 138, 139
gsw_enthalpy_first_derivatives, 27, 38–41, 42, 44, 45, 48, 56, 58, 86–88, 91, 100, 101, 138, 139
gsw_enthalpy_first_derivatives_CT_exact, 27, 38–42, 43, 45, 48, 56, 58, 86–88, 91, 100, 101, 138, 139
gsw_enthalpy_ice, 27, 38–42, 44, 44, 48, 56, 58, 86–88, 91, 100, 101, 138, 139
gsw_enthalpy_second_derivatives, 45
gsw_thermobaric, 156

157

gsw_Turner_Rsubrho, 157

gsw_z_from_p, 105, 164

rep, 6

saar, 165