pythermalcomfort

Release 1.3.0

Oct 22, 2020

Contents

CHAPTER 1

Overview

docs	
tests	
package	

Package to calculate several thermal comfort indices (e.g. PMV, PPD, SET, adaptive) and convert physical variables.

Please cite us if you use this package: Tartarini, F., Schiavon, S., 2020. pythermalcomfort: A Python package for thermal comfort research. SoftwareX 12, 100578. https://doi.org/10.1016/j.softx.2020.100578

- Free software: MIT license

1.1 Installation

```
pip install pythermalcomfort
```

You can also install the in-development version with:

```
pip install https://github.com/CenterForTheBuiltEnvironment/pythermalcomfort/archive/
↪master.zip
```

1.2 Documentation

https://pythermalcomfort.readthedocs.io/

1.3 Examples and Tutorials

Examples files on how to use some of the functions

YouTube tutorials playlist

1.4 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. Click here to learn more on how to contribute to the project.

CHAPTER 2

Installation

At the command line:

```
pip install pythermalcomfort
```

CHAPTER 3

Examples and Tutorials

Examples files on how to use some of the functions

YouTube tutorials playlist

CHAPTER 4

Functions documentation

4.1 Comfort models

4.1.1 PMV PPD

pythermalcomfort.models.**pmv_ppd**(*tdb*, *tr*, *vr*, *rh*, *met*, *clo*, *wme=0*, *standard='ISO'*, *units='SI'*)

Returns Predicted Mean Vote (PMV) and Predicted Percentage of Dissatisfied (PPD) calculated in accordance to main thermal comfort Standards. The PMV is an index that predicts the mean value of the thermal sensation votes (self-reported perceptions) of a large group of people on a sensation scale expressed from –3 to +3 corresponding to the categories "cold," "cool," "slightly cool," "neutral," "slightly warm, " "warm," and "hot."[1]. The PPD is an index that establishes a quantitative prediction of the percentage of thermally dissatisfied people determined from PMV[1].

Parameters

- **tdb** (*float*) – dry bulb air temperature, default in [°C] in [°F] if *units* = 'IP'
- **tr** (*float*) – mean radiant temperature, default in [°C] in [°F] if *units* = 'IP'
- **vr** (*float*) – relative air velocity, default in [m/s] in [fps] if *units* = 'IP'

  Note: vr is the relative air velocity caused by body movement and not the air speed measured by the air velocity sensor. The relative air velocity can be calculate using the function *pythermalcomfort.psychrometrics.v_relative()*.
- **rh** (*float*) – relative humidity, [%]
- **met** (*float*) – metabolic rate, [met]
- **clo** (*float*) – clothing insulation, [clo]

  Note: The ASHRAE 55 Standard suggests that the dynamic clothing insulation is used as input in the PMV model. The dynamic clothing insulation can be calculated using the function *pythermalcomfort.psychrometrics.clo_dynamic()*.

[1] ANSI, & ASHRAE. (2017). Thermal Environmental Conditions for Human Occupancy. Atlanta.

- **wme** (*float*) – external work, [met] default 0
- **standard** (*str (default="ISO")*) – comfort standard used for calculation
  - If "ISO", then the ISO Equation is used
  - If "ASHRAE", then the ASHRAE Equation is used

  Note: While the PMV equation is the same for both the ISO and ASHRAE standards, the ASHRAE Standard Use of the PMV model is limited to air speeds below 0.20 m/s (40 fpm). When air speeds exceed 0.20 m/s (40 fpm), the comfort zone boundaries are adjusted based on the SET model.
- **units** (*str default="SI"*) – select the SI (International System of Units) or the IP (Imperial Units) system.

Returns

- *PMV* – Predicted Mean Vote
- *PPD* – Predicted Percentage of Dissatisfied occupants, [%]

Notes

You can use this function to calculate the PMV and PPD in accordance with either the ASHRAE 55 2017 Standard[1] or the ISO 7730 Standard[2].

Examples

```
>>> from pythermalcomfort.models import pmv_ppd
>>> # calculate relative air velocity
>>> vr = v_relative(v=0.1, met=1.2)
>>> # as you can see the relative air velocity is 0.16 m/s which is
significantly higher than v
>>> results = pmv_ppd(tdb=25, tr=25, vr=vr, rh=50, met=1.2, clo=0.5, wme=0,
standard="ISO")
>>> print(results)
{'pmv': -0.09, 'ppd': 5.2}

>>> print(results['pmv'])
-0.09

>>> # for users who wants to use the IP system
>>> results_ip = pmv_ppd(tdb=77, tr=77, vr=0.4, rh=50, met=1.2, clo=0.5,
units="IP")
>>> print(results_ip)
{'pmv': 0.01, 'ppd': 5.0}
```

Raises

- `StopIteration` – Raised if the number of iterations exceeds the threshold
- `ValueError` – The 'standard' function input parameter can only be 'ISO' or 'ASHRAE'

[2] ISO. (2005). ISO 7730 - Ergonomics of the thermal environment — Analytical determination and interpretation of thermal comfort using calculation of the PMV and PPD indices and local thermal comfort criteria.

4.1.2 PMV

pythermalcomfort.models.**pmv**(*tdb*, *tr*, *vr*, *rh*, *met*, *clo*, *wme=0*, *standard='ISO'*, *units='SI'*)

Returns Predicted Mean Vote (PMV) calculated in accordance to main thermal comfort Standards. The PMV is an index that predicts the mean value of the thermal sensation votes (self-reported perceptions) of a large group of people on a sensation scale expressed from –3 to +3 corresponding to the categories "cold," "cool, " "slightly cool," "neutral," "slightly warm," "warm," and "hot."[1]

Parameters

- **tdb** (*float*) – dry bulb air temperature, default in [°C] in [°F] if *units* = 'IP'
- **tr** (*float*) – mean radiant temperature, default in [°C] in [°F] if *units* = 'IP'
- **vr** (*float*) – relative air velocity, default in [m/s] in [fps] if *units* = 'IP'

  Note: vr is the relative air velocity caused by body movement and not the air speed measured by the air velocity sensor. It can be calculate using the function *pythermalcomfort.psychrometrics.v_relative()*.
- **rh** (*float*) – relative humidity, [%]
- **met** (*float*) – metabolic rate, [met]
- **clo** (*float*) – clothing insulation, [clo]

  Note: The ASHRAE 55 Standard suggests that the dynamic clothing insulation is used as input in the PMV model. The dynamic clothing insulation can be calculated using the function *pythermalcomfort.psychrometrics.clo_dynamic()*.
- **wme** (*float*) – external work, [met] default 0
- **standard** (*str (default="ISO")*) – comfort standard used for calculation
  - If "ISO", then the ISO Equation is used
  - If "ASHRAE", then the ASHRAE Equation is used

  Note: While the PMV equation is the same for both the ISO and ASHRAE standards, the ASHRAE Standard Use of the PMV model is limited to air speeds below 0.20 m/s (40 fpm). When air speeds exceed 0.20 m/s (40 fpm), the comfort zone boundaries are adjusted based on the SET model. See ASHRAE 55 2017 Appendix H for more information[1].
- **units** (*str default="SI"*) – select the SI (International System of Units) or the IP (Imperial Units) system.

Returns **PMV** (*float*) – Predicted Mean Vote

Notes

You can use this function to calculate the PMV[1][2].

Examples

```
>>> from pythermalcomfort.models import pmv
>>> # calculate relative air velocity
>>> vr = v_relative(v=0.1, met=1.2)
>>> # as you can see the relative air velocity is 0.16 m/s which is
significantly higher than v
>>> results = pmv(tdb=25, tr=25, vr=vr, rh=50, met=1.2, clo=0.5)
```

(continues on next page)

(continued from previous page)

```
>>> print(results)
-0.09
```

4.1.3 Standard Effective Temperature (SET)

pythermalcomfort.models.**set_tmp**(*tdb*, *tr*, *v*, *rh*, *met*, *clo*, *wme=0*, *body_surface_area=1.8258*, *patm=101325*, *units='SI'*)

Calculates the Standard Effective Temperature (SET). The SET is the temperature of an imaginary environment at 50% (rh), <0.1 m/s (20 fpm) average air speed (v), and tr = tdb , in which the total heat loss from the skin of an imaginary occupant with an activity level of 1.0 met and a clothing level of 0.6 clo is the same as that from a person in the actual environment with actual clothing and activity level.

Parameters

- **tdb** (*float*) – dry bulb air temperature, default in [°C] in [°F] if *units* = 'IP'
- **tr** (*float*) – mean radiant temperature, default in [°C] in [°F] if *units* = 'IP'
- **v** (*float*) – air velocity, default in [m/s] in [fps] if *units* = 'IP'
- **rh** (*float*) – relative humidity, [%]
- **met** (*float*) – metabolic rate, [met]
- **clo** (*float*) – clothing insulation, [clo]
- **wme** (*float*) – external work, [met] default 0
- **body_surface_area** (*float*) – body surface area, default value 1.8258 [m2] in [ft2] if *units* = 'IP'
- **patm** (*float*) – atmospheric pressure, default value 101325 [Pa] in [atm] if *units* = 'IP'
- **units** (*str default="SI"*) – select the SI (International System of Units) or the IP (Imperial Units) system.

Returns **SET** (*float*) – Standard effective temperature, [°C]

Notes

You can use this function to calculate the SET temperature in accordance with the ASHRAE 55 2017 Standard[1].

Examples

```
>>> from pythermalcomfort.models import set_tmp
>>> set_tmp(tdb=25, tr=25, v=0.1, rh=50, met=1.2, clo=.5)
25.3

>>> # for users who wants to use the IP system
>>> set_tmp(tdb=77, tr=77, v=0.328, rh=50, met=1.2, clo=.5, units='IP')
77.6
```

4.1.4 Cooling Effect

pythermalcomfort.models.**cooling_effect**(*tdb*, *tr*, *vr*, *rh*, *met*, *clo*, *wme=0*, *units='SI'*)

Returns the value of the Cooling Effect (CE) calculated in compliance with the ASHRAE 55 2017 Standard[1]. The CE of the elevated air speed is the value that, when subtracted equally from both the average air temperature and the mean radiant temperature, yields the same SET under still air as in the first SET calculation under elevated air speed. The cooling effect is calculated only for air speed higher than 0.1 m/s.

Parameters

- **tdb** (*float*) – dry bulb air temperature, default in [°C] in [°F] if *units* = 'IP'
- **tr** (*float*) – mean radiant temperature, default in [°C] in [°F] if *units* = 'IP'
- **vr** (*float*) – relative air velocity, default in [m/s] in [fps] if *units* = 'IP'

  Note: vr is the relative air velocity caused by body movement and not the air speed measured by the air velocity sensor. It can be calculate using the function *pythermalcomfort.psychrometrics.v_relative()*.
- **rh** (*float*) – relative humidity, [%]
- **met** (*float*) – metabolic rate, [met]
- **clo** (*float*) – clothing insulation, [clo]
- **wme** (*float*) – external work, [met] default 0
- **units** (*str default="SI"*) – select the SI (International System of Units) or the IP (Imperial Units) system.

Returns *ce* – Cooling Effect, default in [°C] in [°F] if *units* = 'IP'

Examples

```
>>> from pythermalcomfort.models import cooling_effect
>>> ce = cooling_effect(tdb=25, tr=25, vr=0.3, rh=50, met=1.2, clo=0.5)
>>> print(ce)
1.64

>>> # for users who wants to use the IP system
>>> ce = cooling_effect(tdb=77, tr=77, vr=1.64, rh=50, met=1, clo=0.6, units="IP")
>>> print(ce)
3.74
```

Raises `ValueError` – If the cooling effect could not be calculated

4.1.5 Adaptive ASHRAE

pythermalcomfort.models.**adaptive_ashrae**(*tdb*, *tr*, *t_running_mean*, *v*, *units='SI'*)

Determines the adaptive thermal comfort based on ASHRAE 55. The adaptive model relates indoor design temperatures or acceptable temperature ranges to outdoor meteorological or climatological parameters.

Parameters

- **tdb** (*float*) – dry bulb air temperature, default in [°C] in [°F] if *units* = 'IP'
- **tr** (*float*) – mean radiant temperature, default in [°C] in [°F] if *units* = 'IP'

- **t_running_mean** (*float*) – running mean temperature, default in [°C] in [°C] in [°F] if *units* = 'IP'
- **v** (*float*) – air velocity, default in [m/s] in [fps] if *units* = 'IP'
- **units** (*str default="SI"*) – select the SI (International System of Units) or the IP (Imperial Units) system.

Returns

- **tmp_cmf** (*float*) – Comfort temperature a that specific running mean temperature, default in [°C] or in [°F]
- **tmp_cmf_80_low** (*float*) – Lower acceptable comfort temperature for 80% occupants, default in [°C] or in [°F]
- **tmp_cmf_80_up** (*float*) – Upper acceptable comfort temperature for 80% occupants, default in [°C] or in [°F]
- **tmp_cmf_90_low** (*float*) – Lower acceptable comfort temperature for 90% occupants, default in [°C] or in [°F]
- **tmp_cmf_90_up** (*float*) – Upper acceptable comfort temperature for 90% occupants, default in [°C] or in [°F]
- **acceptability_80** (*bol*) – Acceptability for 80% occupants
- **acceptability_90** (*bol*) – Acceptability for 90% occupants

Notes

You can use this function to calculate if your conditions are within the *adaptive thermal comfort region*. Calculations with comply with the ASHRAE 55 2017 Standard[1].

Examples

```
>>> from pythermalcomfort.models import adaptive_ashrae
>>> results = adaptive_ashrae(tdb=25, tr=25, t_running_mean=20, v=0.1)
>>> print(results)
{'tmp_cmf': 24.0, 'tmp_cmf_80_low': 20.5, 'tmp_cmf_80_up': 27.5,
'tmp_cmf_90_low': 21.5, 'tmp_cmf_90_up': 26.5, 'acceptability_80': True,
'acceptability_90': False}

>>> print(results['acceptability_80'])
True
# The conditions you entered are considered to be comfortable for by 80% of the
occupants

>>> # for users who wants to use the IP system
>>> results = adaptive_ashrae(tdb=77, tr=77, t_running_mean=68, v=0.3, units='ip')
>>> print(results)
{'tmp_cmf': 75.2, 'tmp_cmf_80_low': 68.9, 'tmp_cmf_80_up': 81.5,
'tmp_cmf_90_low': 70.7, 'tmp_cmf_90_up': 79.7, 'acceptability_80': True,
'acceptability_90': False}

>>> results = adaptive_ashrae(tdb=25, tr=25, t_running_mean=9, v=0.1)
ValueError: The running mean is outside the standards applicability limits
# The adaptive thermal comfort model can only be used
# if the running mean temperature is higher than 10°C
```

Raises `ValueError` – Raised if the input are outside the Standard's applicability limits

4.1.6 Adaptive EN

`pythermalcomfort.models.`**`adaptive_en`**(*tdb*, *tr*, *t_running_mean*, *v*, *units='SI'*)

Determines the adaptive thermal comfort based on EN 16798-1 2019[3]

Parameters

- **tdb** (*float*) – dry bulb air temperature, default in [°C] in [°F] if *units* = 'IP'
- **tr** (*float*) – mean radiant temperature, default in [°C] in [°F] if *units* = 'IP'
- **t_running_mean** (*float*) – running mean temperature, default in [°C] in [°C] in [°F] if *units* = 'IP'
- **v** (*float*) – air velocity, default in [m/s] in [fps] if *units* = 'IP'

  Note: Indoor operative temperature correction is applicable for buildings equipped with fans or personal systems providing building occupants with personal control over air speed at occupant level. For operative temperatures above 25°C the comfort zone upper limit can be increased by 1.2 °C (0.6 < v < 0.9 m/s), 1.8 °C (0.9 < v < 1.2 m/s), 2.2 °C (v > 1.2 m/s)

- **units** (*str default="SI"*) – select the SI (International System of Units) or the IP (Imperial Units) system.

Returns

- **tmp_cmf** (*float*) – Comfort temperature at that specific running mean temperature, default in [°C] or in [°F]
- **acceptability_cat_i** (*bol*) – If the indoor conditions comply with comfort category I
- **acceptability_cat_ii** (*bol*) – If the indoor conditions comply with comfort category II
- **acceptability_cat_iii** (*bol*) – If the indoor conditions comply with comfort category III
- **tmp_cmf_cat_i_up** (*float*) – Upper acceptable comfort temperature for category I, default in [°C] or in [°F]
- **tmp_cmf_cat_ii_up** (*float*) – Upper acceptable comfort temperature for category II, default in [°C] or in [°F]
- **tmp_cmf_cat_iii_up** (*float*) – Upper acceptable comfort temperature for category III, default in [°C] or in [°F]
- **tmp_cmf_cat_i_low** (*float*) – Lower acceptable comfort temperature for category I, default in [°C] or in [°F]
- **tmp_cmf_cat_ii_low** (*float*) – Lower acceptable comfort temperature for category II, default in [°C] or in [°F]
- **tmp_cmf_cat_iii_low** (*float*) – Lower acceptable comfort temperature for category III, default in [°C] or in [°F]

Notes

You can use this function to calculate if your conditions are within the EN adaptive thermal comfort region. Calculations with comply with the EN 16798-1 2019[1].

[3] EN, & BSI. (2019). Energy performance of buildings - Ventilation for buildings. BSI Standards Limited 2019.

Examples

```
>>> from pythermalcomfort.models import adaptive_en
>>> results = adaptive_en(tdb=25, tr=25, t_running_mean=20, v=0.1)
>>> print(results)
{'tmp_cmf': 25.4, 'acceptability_cat_i': True, 'acceptability_cat_ii': True,
'acceptability_cat_iii': True, ... }

>>> print(results['acceptability_cat_i'])
True
# The conditions you entered are considered to comply with Category I

>>> # for users who wants to use the IP system
>>> results = adaptive_en(tdb=77, tr=77, t_running_mean=68, v=0.3, units='ip')
>>> print(results)
{'tmp_cmf': 77.7, 'acceptability_cat_i': True, 'acceptability_cat_ii': True,
'acceptability_cat_iii': True, ... }

>>> results = adaptive_en(tdb=25, tr=25, t_running_mean=9, v=0.1)
ValueError: The running mean is outside the standards applicability limits
# The adaptive thermal comfort model can only be used
# if the running mean temperature is between 10 °C and 30 °C
```

Raises `ValueError` – Raised if the input are outside the Standard's applicability limits

4.1.7 Solar gain on people

pythermalcomfort.models.**solar_gain**(*sol_altitude*, *sol_azimuth*, *sol_radiation_dir*, *sol_transmittance*, *f_svv*, *f_bes*, *asw=0.7*, *posture='seated'*, *floor_reflectance=0.6*)

Calculates the solar gain to the human body using the Effective Radiant Field (ERF)[1]. The ERF is a measure of the net energy flux to or from the human body. ERF is expressed in W over human body surface area [w/m2]. In addition, it calculates the delta mean radiant temperature. Which is the amount by which the mean radiant temperature of the space should be increased if no solar radiation is present.

Parameters

- **sol_altitude** (*float*) – Solar altitude, degrees from horizontal [deg]. Ranges between 0 and 90.
- **sol_azimuth** (*float*) – Solar azimuth, degrees clockwise from North [deg]. Ranges between 0 and 180.
- **posture** (*str*) – Default 'seated' list of available options 'standing', 'supine' or 'seated'
- **sol_radiation_dir** (*float*) – Direct-beam solar radiation, [W/m2]. Ranges between 200 and 1000. See Table C2-3 of ASHRAE 55 2017[1].
- **sol_transmittance** (*float*) – Total solar transmittance, ranges from 0 to 1. The total solar transmittance of window systems, including glazing unit, blinds, and other façade treatments, shall be determined using one of the following methods: i) Provided by manufacturer or from the National Fenestration Rating Council approved Lawrence Berkeley National Lab International Glazing Database. ii) Glazing unit plus venetian blinds or other complex or unique shades shall be calculated using National Fenestration Rating Council approved software or Lawrence Berkeley National Lab Complex Glazing Database.
- **f_svv** (*float*) – Fraction of sky vault exposed to body, ranges from 0 to 1.

- **f_bes** (*float*) – Fraction of the possible body surface exposed to sun, ranges from 0 to 1. See Table C2-2 and equation C-7 ASHRAE 55 2017[1].
- **asw** (*float*) – The average short-wave absorptivity of the occupant. It will range widely, depending on the color of the occupant's skin as well as the color and amount of clothing covering the body. A value of 0.7 shall be used unless more specific information about the clothing or skin color of the occupants is available. Note: Short-wave absorptivity typically ranges from 0.57 to 0.84, depending on skin and clothing color. More information is available in Blum (1945).
- **floor_reflectance** (*float*) – Floor refectance. It is assumed to be constant and equal to 0.6.

Notes

More information on the calculation procedure can be found in Appendix C of[1].

Returns

- **erf** (*float*) – Solar gain to the human body using the Effective Radiant Field [W/m2]
- **delta_mrt** (*float*) – Delta mean radiant temperature. The amount by which the mean radiant temperature of the space should be increased if no solar radiation is present.

Examples

```
>>> from pythermalcomfort.models import solar_gain
>>> results = solar_gain(sol_altitude=0, sol_azimuth=120,
sol_radiation_dir=800, sol_transmittance=0.5, f_svv=0.5, f_bes=0.5,
asw=0.7, posture='seated')
>>> print(results)
{'erf': 42.9, 'delta_mrt': 10.3}
```

4.1.8 Universal Thermal Climate Index (UTCI)

pythermalcomfort.models.**utci**(*tdb*, *tr*, *v*, *rh*, *units='SI'*)

Determines the Universal Thermal Climate Index (UTCI). The UTCI is the equivalent temperature for the environment derived from a reference environment. It is defined as the air temperature of the reference environment which produces the same strain index value in comparison with the reference individual's response to the real environment. It is regarded as one of the most comprehensive indices for calculating heat stress in outdoor spaces. The parameters that are taken into account for calculating UTCI involve dry-bulb temperature, mean radiation temperature, the pressure of water vapor or relative humidity, and wind speed (at the elevation of 10m)[7].

Parameters

- **tdb** (*float*) – dry bulb air temperature, default in [°C] in [°F] if *units* = 'IP'
- **tr** (*float*) – mean radiant temperature, default in [°C] in [°F] if *units* = 'IP'
- **v** (*float*) – relative air velocity, default in [m/s] in [fps] if *units* = 'IP'
- **rh** (*float*) – relative humidity, [%]
- **units** (*str default="SI"*) – select the SI (International System of Units) or the IP (Imperial Units) system.

[7] Zare, S., Hasheminejad, N., Shirvan, H.E., Hemmatjo, R., Sarebanzadeh, K., Ahmadi, S., 2018. Comparing Universal Thermal Climate Index (UTCI) with selected thermal indices/environmental parameters during 12 months of the year. Weather Clim. Extrem. 19, 49–57. https://doi.org/10.1016/j.wace.2018.01.004

Returns **utci** (*float*) – Universal Thermal Climate Index, [°C] or in [°F]

Notes

You can use this function to calculate the Universal Thermal Climate Index (*UTCI*) The applicability wind speed value must be between 0.5 and 17 m/s.

Examples

```
>>> from pythermalcomfort.models import utci
>>> utci(tdb=25, tr=25, v=1.0, rh=50)
24.6

>>> # for users who wants to use the IP system
>>> utci(tdb=77, tr=77, v=3.28, rh=50, units='ip')
76.4
```

Raises `ValueError` – Raised if the input are outside the Standard's applicability limits

4.1.9 Clothing prediction

`pythermalcomfort.models.clo_tout` (*tout*, *units='SI'*)

Representative clothing insulation Icl as a function of outdoor air temperature at 06:00 a.m[4].

Parameters

- **tout** (*float*) – outdoor air temperature at 06:00 a.m., default in [°C] in [°F] if *units* = 'IP'
- **units** (*str default="SI"*) – select the SI (International System of Units) or the IP (Imperial Units) system.

Returns **clo** (*float*) – Representative clothing insulation Icl, [clo]

Notes

The ASHRAE 55 2017 states that it is acceptable to determine the clothing insulation Icl using this equation in mechanically conditioned buildings[1].

Examples

```
>>> from pythermalcomfort.models import clo_tout
>>> clo_tout(tout=27)
0.46
```

[4] Schiavon, S., & Lee, K. H. (2013). Dynamic predictive clothing insulation models based on outdoor air and indoor operative temperatures. Building and Environment, 59, 250–260. doi.org/10.1016/j.buildenv.2012.08.024

4.1.10 Vertical air temperature gradient

pythermalcomfort.models.**vertical_tmp_grad_ppd**(*tdb*, *tr*, *vr*, *rh*, *met*, *clo*, *vertical_tmp_grad*, *units='SI'*)

Calculates the percentage of thermally dissatisfied people with a vertical temperature gradient between feet and head[1]. This equation is only applicable for vr < 0.2 m/s (40 fps).

Parameters

- **tdb** (*float*) – dry bulb air temperature, default in [°C] in [°F] if *units* = 'IP'

  Note: The air temperature is the average value over two heights: 0.6 m (24 in.) and 1.1 m (43 in.) for seated occupants and 1.1 m (43 in.) and 1.7 m (67 in.) for standing occupants.

- **tr** (*float*) – mean radiant temperature, default in [°C] in [°F] if *units* = 'IP'

- **vr** (*float*) – relative air velocity, default in [m/s] in [fps] if *units* = 'IP'

  Note: vr is the relative air velocity caused by body movement and not the air speed measured by the air velocity sensor. It can be calculate using the function *pythermalcomfort.psychrometrics.v_relative()*.

- **rh** (*float*) – relative humidity, [%]

- **met** (*float*) – metabolic rate, [met]

- **clo** (*float*) – clothing insulation, [clo]

- **vertical_tmp_grad** (*float*) – vertical temperature gradient between the feet and the head, default in [°C/m] in [°F/ft] if *units* = 'IP'

- **units** (*str default="SI"*) – select the SI (International System of Units) or the IP (Imperial Units) system.

Returns

- **PPD_vg** (*float*) – Predicted Percentage of Dissatisfied occupants with vertical temperature gradient, [%]

- **Acceptability** (*bol*) – The ASHRAE 55 2017 standard defines that the value of air speed at the ankle level is acceptable if PPD_ad is lower or equal than 5 %

Examples

```
>>> from pythermalcomfort.models import vertical_tmp_grad_ppd
>>> results = vertical_tmp_grad_ppd(25, 25, 0.1, 50, 1.2, 0.5, 7)
>>> print(results)
{'PPD_vg': 12.6, 'Acceptability': False}
```

4.1.11 Ankle draft

pythermalcomfort.models.**ankle_draft**(*tdb*, *tr*, *vr*, *rh*, *met*, *clo*, *v_ankle*, *units='SI'*)

Calculates the percentage of thermally dissatisfied people with the ankle draft (0.1 m) above floor level[1]. This equation is only applicable for vr < 0.2 m/s (40 fps).

Parameters

- **tdb** (*float*) – dry bulb air temperature, default in [°C] in [°F] if *units* = 'IP'

  Note: The air temperature is the average value over two heights: 0.6 m (24 in.) and 1.1 m (43 in.) for seated occupants and 1.1 m (43 in.) and 1.7 m (67 in.) for standing occupants.

- **tr** (*float*) – mean radiant temperature, default in [°C] in [°F] if *units* = 'IP'
- **vr** (*float*) – relative air velocity, default in [m/s] in [fps] if *units* = 'IP'

  Note: vr is the relative air velocity caused by body movement and not the air speed measured by the air velocity sensor. It can be calculate using the function *pythermalcomfort.psychrometrics.v_relative()*.
- **rh** (*float*) – relative humidity, [%]
- **met** (*float*) – metabolic rate, [met]
- **clo** (*float*) – clothing insulation, [clo]
- **v_ankle** (*float*) – air speed at the 0.1 m (4 in.) above the floor, default in [m/s] in [fps] if *units* = 'IP'
- **units** (*str default="SI"*) – select the SI (International System of Units) or the IP (Imperial Units) system.

Returns

- **PPD_ad** (*float*) – Predicted Percentage of Dissatisfied occupants with ankle draft, [%]
- **Acceptability** (*bol*) – The ASHRAE 55 2017 standard defines that the value of air speed at the ankle level is acceptable if PPD_ad is lower or equal than 20 %

Examples

```
>>> from pythermalcomfort.models import ankle_draft
>>> results = ankle_draft(25, 25, 0.2, 50, 1.2, 0.5, 0.3, units="SI")
>>> print(results)
{'PPD_ad': 18.6, 'Acceptability': True}
```

References

4.2 Psychrometrics functions

pythermalcomfort.psychrometrics.**clo_dynamic**(*clo*, *met*, *standard='ASHRAE'*)

Estimates the dynamic clothing insulation of a moving occupant. The activity as well as the air speed modify the insulation characteristics of the clothing and the adjacent air layer. Consequently the ISO 7730 states that the clothing insulation shall be corrected[2]. The ASHRAE 55 Standard, instead, only corrects for the effect of the body movement, and states that the correction is permitted but not required.

Parameters

- **clo** (*float*) – clothing insulation, [clo]
- **met** (*float*) – metabolic rate, [met]
- **standard** (*str (default="ASHRAE")*) –
  - If "ASHRAE", uses Equation provided in Section 5.2.2.2 of ASHRAE 55 2017

Returns **clo** (*float*) – dynamic clothing insulation, [clo]

pythermalcomfort.psychrometrics.**enthalpy**(*tdb*, *hr*)

Calculates air enthalpy

Parameters

- **tdb** (*float*) – air temperature, [°C]
- **hr** (*float*) – humidity ratio, [kg water/kg dry air]

Returns **enthalpy** (*float*) – enthalpy [J/kg dry air]

pythermalcomfort.psychrometrics.**p_sat**(*tdb*)

Calculates vapour pressure of water at different temperatures

Parameters **tdb** (*float*) – air temperature, [°C]

Returns **p_sat** (*float*) – operative temperature, [Pa]

pythermalcomfort.psychrometrics.**p_sat_torr**(*tdb*)

Estimates the saturation vapor pressure in [torr]

Parameters **tdb** (*float*) – dry bulb air temperature, [C]

Returns **p_sat** (*float*) – saturation vapor pressure [torr]

pythermalcomfort.psychrometrics.**psy_ta_rh**(*tdb*, *rh*, *patm=101325*)

Calculates psychrometric values of air based on dry bulb air temperature and relative humidity. For more accurate results we recommend the use of the the Python package psychrolib.

Parameters

- **tdb** (*float*) – air temperature, [°C]
- **rh** (*float*) – relative humidity, [%]
- **patm** (*float*) – atmospheric pressure, [Pa]

Returns

- **p_vap** (*float*) – partial pressure of water vapor in moist air, [Pa]
- **hr** (*float*) – humidity ratio, [kg water/kg dry air]
- **t_wb** (*float*) – wet bulb temperature, [°C]
- **t_dp** (*float*) – dew point temperature, [°C]
- **h** (*float*) – enthalpy [J/kg dry air]

pythermalcomfort.psychrometrics.**running_mean_outdoor_temperature**(*temp_array*, *alpha=0.8*, *units='SI'*)

Estimates the running mean temperature

Parameters

- **temp_array** (*list*) – array containing the mean daily temperature in descending order (i.e. from newest/yesterday to oldest) $[\Theta_{day-1}, \Theta_{day-2}, \ldots, \Theta_{day-n}]$. Where Θ_{day-1} is yesterday's daily mean temperature. The EN 16798-1 2019[3] states that n should be equal to 7
- **alpha** (*float*) – constant between 0 and 1. The EN 16798-1 2019[3] recommends a value of 0.8, while the ASHRAE 55 2017 recommends to choose values between 0.9 and 0.6, corresponding to a slow- and fast- response running mean, respectively. Adaptive comfort theory suggests that a slow-response running mean (alpha = 0.9) could be more appropriate for climates in which synoptic-scale (day-to- day) temperature dynamics are relatively minor, such as the humid tropics.
- **units** (*str default="SI"*) – select the SI (International System of Units) or the IP (Imperial Units) system.

Returns **t_rm** (*float*) – running mean outdoor temperature

pythermalcomfort.psychrometrics.**t_dp**(*tdb*, *rh*)
Calculates the dew point temperature.

Parameters

- **tdb** (*float*) – dry-bulb air temperature, [°C]
- **rh** (*float*) – relative humidity, [%]

Returns **t_dp** (*float*) – dew point temperature, [°C]

pythermalcomfort.psychrometrics.**t_mrt**(*tg*, *tdb*, *v*, *d=0.15*, *emissivity=0.9*)
Converts globe temperature reading into mean radiant temperature in accordance with ISO 7726:1998[5]

Parameters

- **tg** (*float*) – globe temperature, [°C]
- **tdb** (*float*) – air temperature, [°C]
- **v** (*float*) – air velocity, [m/s]
- **d** (*float*) – diameter of the globe, [m] default 0.15 m
- **emissivity** (*float*) – emissivity of the globe temperature sensor, default 0.9

Returns **tr** (*float*) – mean radiant temperature, [°C]

pythermalcomfort.psychrometrics.**t_o**(*tdb*, *tr*, *v*)
Calculates operative temperature in accordance with ISO 7726:1998[5]

Parameters

- **tdb** (*float*) – air temperature, [°C]
- **tr** (*float*) – mean radiant temperature temperature, [°C]
- **v** (*float*) – air velocity, [m/s]

Returns **to** (*float*) – operative temperature, [°C]

pythermalcomfort.psychrometrics.**t_wb**(*tdb*, *rh*)
Calculates the wet-bulb temperature using the Stull equation[6]

Parameters

- **tdb** (*float*) – air temperature, [°C]
- **rh** (*float*) – relative humidity, [%]

Returns **tdb** (*float*) – wet-bulb temperature, [°C]

pythermalcomfort.psychrometrics.**units_converter**(*from_units='ip'*, ***kwargs*)
Converts IP values to SI units

Parameters

- **from_units** (*str*) – specify system to convert from
- ****kwargs** (*[t, v]*)

Returns *converted values in SI units*

[5] ISO. (1998). ISO 7726 - Ergonomics of the thermal environment instruments for measuring physical quantities.
[6] Stull, R., 2011. Wet-Bulb Temperature from Relative Humidity and Air Temperature. J. Appl. Meteorol. Climatol. 50, 2267–2269. doi.org/10.1175/JAMC-D-11-0143.1

pythermalcomfort.psychrometrics.**v_relative**(*v*, *met*)
Estimates the relative air velocity which combines the average air velocity of the space plus the relative air velocity caused by the body movement.

Parameters

- **v** (*float*) – air velocity measured by the sensor, [m/s]
- **met** (*float*) – metabolic rate, [met]

Returns **vr** (*float*) – relative air velocity, [m/s]

4.3 Reference values clo and met

4.3.1 Met typical tasks, [met]

pythermalcomfort.utilities.**met_typical_tasks = {'Basketball': 6.3, 'Calisthenics': 3.5,**
This dictionary contains the met values of typical tasks.

Example

```
>>> from pythermalcomfort.utilities import met_typical_tasks
>>> print(met_typical_tasks['Filing, standing'])
1.4
```

4.3.2 Clothing insulation of typical ensembles, [clo]

pythermalcomfort.utilities.**clo_typical_ensembles = {'Jacket, Trousers, long-sleeve shirt':**
This dictionary contains the total clothing insulation of typical typical ensembles.

Example

```
>>> from pythermalcomfort.utilities import clo_typical_ensembles
>>> print(clo_typical_ensembles['Typical summer indoor clothing'])
0.5
```

4.3.3 Insulation of individual garments, [clo]

pythermalcomfort.utilities.**clo_individual_garments = {'Ankle socks': 0.02, 'Boots': 0.1,**
This dictionary contains the clo values of individual clothing elements. To calculate the total clothing insulation you need to add these values together.

Example

```
>>> from pythermalcomfort.utilities import clo_individual_garments
>>> print(clo_individual_garments['T-shirt'])
0.08

>>> # calculate total clothing insulation
>>> i_cl = clo_individual_garments['T-shirt'] + clo_individual_garments["Men's␣
↪underwear"] +
>>>         clo_individual_garments['Thin trousers'] + clo_individual_garments['Shoes␣
↪or sandals']
```

(continues on next page)

(continued from previous page)

```
>>> print(i_cl)
0.29
```

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

5.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.2 Documentation improvements

pythermalcomfort could always use more documentation, whether as part of the official pythermalcomfort docs, in docstrings, or even on the web in blog posts, articles, and such.

5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/CenterForTheBuiltEnvironment/pythermalcomfort/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

5.4 Development

To set up *pythermalcomfort* for local development:

1. Fork pythermalcomfort (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:CenterForTheBuiltEnvironment/pythermalcomfort.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes run all the checks and docs builder with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

To add a function

1. Add the function to the python file *src/pythermalcomfort/models.py* and document it.
2. Add any related functions that are used by your function either in *src/pythermalcomfort/utilities.py* or *src/pythermalcomfort/psychrometrics.py*. See existing code as example.
3. Test your function by writing a test in *tests/test_XXXX.py*. Test it by running tox -e pyXX where XX is the Python version you want to use, e.g. 37
4. Add *autofunction* to *doc.reference.pythermalcomfort.py*.

CHAPTER 6

Authors

- Federico Tartarini
- Stefano Schiavon
- Tyler Hoyt
- Chris Mackey

Derivative work

pythermalcomfort is a derivative work of the following software projects:

- CBE Comfort Tool for indoor thermal comfort calculations. Available under GPL.
- ladybug-comfort. Available under GPL.
- UTCI Fortran Code for outdoor thermal comfort calculations. Available under MIT.

CHAPTER 7

Changelog

7.1 1.3.0 (2020-10-19)

- Changed PMV elevated air speed limit from 0.2 to 0.1 m/s

7.2 1.2.3 (2020-09-09)

- Fixed error in the calculation of erf
- Updated validation table erf

7.3 1.2.2 (2020-08-21)

- Changed default diameter in t_mrt
- Improved documentation

7.4 1.2.0 (2020-07-29)

- Significantly improved calculation speed using numba. Wrapped set and pmv functions

7.5 1.0.6 (2020-07-24)

- Minor speed improvement changed math.pow with **
- Added validation PMV validation table from ISO 7730

7.6 1.0.4 (2020-07-20)

- Improved speed calculation of the Cooling Effect
- Bisection has been replaced with Brentq function from scipy

7.7 1.0.3 (2020-07-01)

- Annotated variables in the SET code.

7.8 1.0.2 (2020-06-11)

- Fixed an error in the bisection equation used to calculated Cooling Effect.

7.9 1.0.0 (2020-06-09)

- Major stable release.

7.10 0.7.0 (2020-06-09)

- Added equation to calculate the dynamic clothing insulation

7.11 0.6.3 (2020-04-11)

- Fixed error in calculation adaptive ASHRAE
- Added some examples

7.12 0.6.3 (2020-03-17)

- Renamed function to_calc to t_o
- Fixed error calculation of relative air velocity
- renamed input parameter ta to tdb
- Added function to calculate mean radiant temperature from black globe temperature
- Added function to calculate solar gain on people
- Added functions to calculate vapour pressure, wet-bulb temperature, dew point temperature, and psychrometric data from dry bulb temperature and RH
- Added authors
- Added dictionaries with reference clo and met values
- Added function to calculate enthalpy

7.13 0.5.2 (2020-03-11)

- Added function to calculate the running mean outdoor temperature

7.14 0.5.1 (2020-03-06)

- There was an error in version 0.4.2 in the calculation of PMV and PPD with elevated air speed, i.e. vr > 0.2 which has been fixed in this version
- Added function to calculate the cooling effect in accordance with ASHRAE

7.15 0.4.1 (2020-02-17)

- Removed compatibility with python 2.7 and 3.5

7.16 0.4.0 (2020-02-17)

- Created adaptive_EN, v_relative, t_clo, vertical_tmp_gradient, ankle_draft functions and wrote tests.
- Added possibility to decide with measuring system to use SI or IP.

7.17 0.3.0 (2020-02-13)

- Created set_tmp, adaptive_ashrae, UTCI functions and wrote tests.
- Added warning to let the user know if inputs entered do not comply with Standards applicability limits.

7.18 0.1.0 (2020-02-11)

- Created pmv, pmv_ppd functions and wrote tests.
- Documented code.

7.19 0.0.0 (2020-02-11)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index