

# Package ‘meteoEVT’

May 8, 2026

**Type** Package

**Title** Computation and Visualization of Energetic and Vortical Atmospheric Quantities

**Version** 0.1.0

**Date** 2022-09-02

**Maintainer** Laura Mack <laum52@zedat.fu-berlin.de>

**Description** Energy-Vorticity theory (EVT) is the fundamental theory to describe processes in the atmosphere by combining conserved quantities from hydrodynamics and thermodynamics. The package 'meteoEVT' provides functions to calculate many energetic and vortical quantities, like potential vorticity, Bernoulli function and dynamic state index (DSI) [e.g. Weber and Nevir, 2008, <doi:10.1111/j.1600-0870.2007.00272.x>], for given gridded data, like ERA5 reanalyses. These quantities can be studied directly or can be used for many applications in meteorology, e.g., the objective identification of atmospheric fronts. For this purpose, separate function are provided that allow the detection of fronts based on the thermic front parameter [Hewson, 1998, <doi:10.1017/S1350482798000553>], the F diagnostic [Parfitt et al., 2017, <doi:10.1002/2017GL073662>] and the DSI [Mack et al., 2022, <doi:10.48550/arXiv.2208.11438>].

**Author** Laura Mack [aut, cre]

**Imports** grDevices, graphics, purrr, ncd4

**License** GPL (>= 2)

**URL** <https://github.com/noctiluc3nt/meteoEVT>

**Encoding** UTF-8

**RoxygenNote** 7.2.0

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-09-05 08:30:02 UTC

## Contents

meteoEVT-package . . . . .	2
calc_bernoulli . . . . .	3

calc_density . . . . .	4
calc_dsi . . . . .	4
calc_ensrophy . . . . .	6
calc_fdiag . . . . .	7
calc_frontogenesis . . . . .	8
calc_helicity . . . . .	9
calc_lamb . . . . .	10
calc_pv . . . . .	11
calc_tfp . . . . .	13
calc_theta . . . . .	13
calc_vorticity . . . . .	14
crossprod . . . . .	15
df_dp . . . . .	16
df_dx . . . . .	16
df_dy . . . . .	17
df_dz . . . . .	18
div . . . . .	18
fill_horiz . . . . .	19
frontid . . . . .	20
grad . . . . .	22
jacobian . . . . .	23
readin_dim . . . . .	24
readin_era5 . . . . .	24
rot . . . . .	25
scalarprod . . . . .	26

<b>Index</b>	<b>27</b>
--------------	-----------

---

meteoEVT-package	<i>Introduction</i>
------------------	---------------------

---

## Description

Energy-Vorticity theory (EVT) is the fundamental theory to describe processes in the atmosphere by combining conserved quantities from hydrodynamics and thermodynamics. The package 'meteoEVT' provides functions to calculate many energetic and vortical quantities, like potential vorticity, Bernoulli function and dynamic state index (DSI) (Weber and Nevir, 2008), for given gridded data, like ERA5 reanalyses. These quantities can be studied directly or can be used for many applications in meteorology, e.g., the objective identification of atmospheric fronts. For this purpose, separate function are provided that allow the detection of fronts based on the thermic front parameter (Hewson, 1998), the F diagnostic (Parfitt et al., 2017) and the DSI (Mack et al., 2022).

## Details

Phenomenons in the Earth's atmosphere, like tropical hurricanes or extratropical cyclones, can adequately be characterized by a combination of energetic and vortical quantities. These quantities can also be used for a consistent theoretical description of these phenomenons. This package provides functions to calculate Bernoulli function, vorticity, enstrophy, helicity, Lamb vector and potential

vorticity based on given gridded data sets. Additionally, by using energy-vortex theory an adiabatic, stationary and inviscid basic state of the Earth's atmosphere can be derived, which is itself a solution of the primitive equations. The derivation from this basic state is given by the dynamic state index (DSI), which can be used for the study of, e.g., cyclones and fronts. Recently, the DSI was used to identify atmospheric fronts objectively from reanalysis data and thereby provides an alternative way for front detection. For this purpose, this package provides functions to calculate the DSI and use it to identify atmospheric fronts. This method can be compared with state-of-the-art front identification methods based on the thermic front parameter or the F diagnostic.

## References

- Weber, T. and N vir, P. (2008). Storm tracks and cyclone development using the theoretical concept of the Dynamic State Index (DSI). *Tellus A*, 60(1):1–10, doi:10.1111/j.1600-0870.2007.00272.x.
- Parfitt, R., Czaja, A., and Seo, H. (2017). A simple diagnostic for the detection of atmospheric fronts. *Geophys. Res. Lett.*, 44:4351–4358, doi:10.1002/2017GL073662.
- Hewson, T. D. (1998). Objective fronts. *Meteorol. Appl.*, 5:37–65, doi:10.1017/S1350482798000553.
- Mack, L., Rudolph, A. and N vir, P. (2022). Identifying atmospheric fronts based on diabatic processes using the dynamic state index (DSI), arXiv:2208.11438.

---

calc_bernoulli	<i>Bernoulli function</i>
----------------	---------------------------

---

## Description

Calculates the Bernoulli function, i.e. total energy density, as sum of potential, kinetic and thermal energy density

## Usage

```
calc_bernoulli(t_fld, u_fld, v_fld, w_fld, phi_fld)
```

## Arguments

t_fld	temperature field [K]
u_fld	zonal velocity field [m/s]
v_fld	meridional velocity field [m/s]
w_fld	vertical velocity field [m/s]
phi_fld	geopotential height [gpm]

## Value

Bernoulli function field [m<sup>2</sup>/s<sup>2</sup>]

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
bernoulli=calc_bernoulli(data$temp,data$u,data$v,data$w,data$z)
```

---

calc_density	<i>Density</i>
--------------	----------------

---

**Description**

Calculates the density of an ideal fluid

**Usage**

```
calc_density(t_fld, lev_p)
```

**Arguments**

t_fld	temperature field [K]
lev_p	vector containing pressure levels [Pa]

**Value**

density [kg/m<sup>3</sup>]

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
density=calc_density(data$temp,data$lev)
```

---

calc_dsi	<i>Dynamic State Index (DSI)</i>
----------	----------------------------------

---

**Description**

Calculates the dynamic state index DSI

**Usage**

```

calc_dsi(
  t_fld,
  u_fld,
  v_fld,
  w_fld,
  phi_fld,
  lev_p,
  lat = NULL,
  dx = 0.25,
  dy = 0.25,
  zvort_only = FALSE,
  relative = FALSE,
  pv_fld = NULL,
  mode = "lonlat"
)

```

**Arguments**

t_fld	temperature field [K]
u_fld	zonal velocity field [m/s]
v_fld	meridional velocity field [m/s]
w_fld	vertical velocity field [m/s]
phi_fld	geopotential height [gpm]
lev_p	vector containing pressure levels [Pa]
lat	vector containing latitude
dx	x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
zvort_only	logical, TRUE: if only the vertical vorticity (zvort) should be calculated, FALSE: for the whole vorticity vector, default: FALSE
relative	logical, TRUE: only relative vorticity, FALSE: whole (absolute) vorticity, default: FALSE
pv_fld	optional pv field (if e.g., PV is directly taken from ERA5 and not calculated separately)
mode	use 'lonlat' if the data is given on a lon-lat-grid or 'cartesian' if the data is given on an equidistant cartesian grid

**Value**

dynamic state index [ $K^2 \cdot m^4 / (kg^2 \cdot s^3)$ ]

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
dsi=calc_dsi(data$temp,data$u,data$v,data$w,data$z,lev_p=data$lev,lat=data$lat)
```

---

calc_enstrophy	<i>Enstrophy density</i>
----------------	--------------------------

---

**Description**

Calculates the enstrophy density (vorticity squared) either in 2d or 3d

**Usage**

```
calc_enstrophy(
  u_fld,
  v_fld,
  w_fld = NULL,
  lev_p,
  lat = NULL,
  dx = 0.25,
  dy = 0.25,
  zvort_only = TRUE,
  relative = TRUE,
  zvort_fld = NULL,
  mode = "lonlat"
)
```

**Arguments**

u_fld	zonal velocity field [m/s]
v_fld	meridional velocity field [m/s]
w_fld	vertical velocity field [m/s]
lev_p	vector containing pressure levels [Pa]
lat	vector containing latitude
dx	x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
zvort_only	logical, TRUE: if only 2d enstrophy (based on z-vorticity) should be calculated, FALSE: for 3d enstrophy (based on 3d vorticity), default: TRUE
relative	logical, TRUE: only relative vorticity, FALSE: whole (absolute) vorticity should be used for calculation of enstrophy, default: TRUE

zvort_fld	optional zvort field (if e.g., zvort is directly taken from ERA5 and not calculated separately)
mode	use 'lonlat' if the data is given on a lon-lat-grid or 'cartesian' if the data is given on an equidistant cartesian grid

**Value**

enstrophy density field [1/s<sup>2</sup>]

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
#3d enstrophy
ens3d=calc_enstrophy(data$u,data$v,data$w,data$lev,lat=data$lat)
#2d enstrophy as scalar
ens2d=calc_enstrophy(data$u,data$v,lev_p=data$lev,lat=data$lat,zvort_only=TRUE)
```

---

calc_fdiag	<i>F diagnostic</i>
------------	---------------------

---

**Description**

Calculates the F diagnostic

**Usage**

```
calc_fdiag(
  t_fld,
  u_fld,
  v_fld,
  w_fld,
  lev_p,
  lat = NULL,
  dx = 0.25,
  dy = 0.25,
  mode = "lonlat"
)
```

**Arguments**

t_fld	temperature field [K]
u_fld	zonal velocity field [m/s]
v_fld	meridional velocity field [m/s]
w_fld	vertical velocity field [m/s]
lev_p	vector containing pressure levels [Pa]

lat	only for lonlat mode: vector containing latitude
dx	x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
mode	the horizontal coordinate system, options are lonlat for a longitude-latitude-grid (default), or cartesian for an equidistant cartesian grid

**Value**

F diagnostic (dimensionless)

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
fdiag=calc_fdiag(data$temp,data$u,data$v,data$w,data$lev,data$lat)
```

---

calc\_frontogenesis      *Petterssen Frontogenesis Function*

---

**Description**

Calculates the Petterssen frontogenesis function based on the potential temperature

**Usage**

```
calc_frontogenesis(
  t_fld,
  u_fld,
  v_fld,
  w_fld,
  lev_p,
  mode = "lonlat",
  lat = NULL,
  dx = 0.25,
  dy = 0.25
)
```

**Arguments**

t_fld	temperature field [K]
u_fld	zonal velocity field [m/s]
v_fld	meridional velocity field [m/s]
w_fld	vertical velocity field [m/s]
lev_p	vector containing pressure levels [Pa]

mode	the coordinate system, options are lonlat for a longitude-latitude-grid (default), or cartesian for an equidistant cartesian grid
lat	only for lonlat mode: vector containing latitude
dx	x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')

**Value**

Petterssen Frontogenesis Function

---

calc_helicity	<i>Helicity density</i>
---------------	-------------------------

---

**Description**

Calculates the helicity density (scalar product of wind vector and vorticity vector) either for the whole vector (3d) or only for the vertical component (updraft helicity)

**Usage**

```
calc_helicity(
  u_fld,
  v_fld,
  w_fld,
  lev_p,
  lat = NULL,
  dx = 0.25,
  dy = 0.25,
  vert_only = FALSE,
  relative = TRUE,
  zvort_fld = NULL,
  mode = "lonlat"
)
```

**Arguments**

u_fld	zonal velocity field [m/s]
v_fld	meridional velocity field [m/s]
w_fld	vertical velocity field [m/s]
lev_p	vector containing pressure levels [Pa]
lat	vector containing latitude
dx	x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')

dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
vert_only	logical, TRUE: if only the updraft helicity $w*\zeta$ (based on z-vorticity) should be calculated, FALSE: for 3d helicity (based on 3d vorticity), default: FALSE
relative	logical, TRUE: only relative vorticity, FALSE: whole (absolute) vorticity should be used for calculation of enstrophy, default: TRUE
zvort_fld	optional zvort field (if e.g., zvort is directly taken from ERA5 and not calculated separately)
mode	use 'lonlat' if the data is given on a lon-lat-grid or 'cartesian' if the data is given on an equidistant cartesian grid

**Value**

helicity density field [m/s<sup>2</sup>]

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
#3d helicity
hel=calc_helicity(data$u,data$v,data$w,data$lev,lat=data$lat)
#updraft helicity
up_hel=calc_helicity(data$u,data$v,data$w,data$lev,lat=data$lat,vert_only=TRUE)
```

---

calc\_lamb

*Lamb vector (sometimes called vortex energy)*

---

**Description**

Calculates the Lamb vector (cross product of wind vector and vorticity vector)

**Usage**

```
calc_lamb(
  u_fld,
  v_fld,
  w_fld,
  lev_p,
  lat = NULL,
  dx = 0.25,
  dy = 0.25,
  relative = TRUE,
  zvort_fld = NULL,
  mode = "lonlat"
)
```

**Arguments**

u_fld	zonal velocity field [m/s]
v_fld	meridional velocity field [m/s]
w_fld	vertical velocity field [m/s]
lev_p	vector containing pressure levels [Pa]
lat	vector containing latitude
dx	x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
relative	logical, TRUE: only relative vorticity, FALSE: whole (absolute) vorticity should be used for calculation of enstrophy, default: TRUE
zvort_fld	optional zvort field (if e.g., zvort is directly taken from ERA5 and not calculated separately)
mode	use 'lonlat' if the data is given on a lon-lat-grid or 'cartesian' if the data is given on an equidistant cartesian grid

**Value**

lamb vector [m/s<sup>2</sup>]

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
lamb=calc_lamb(data$u,data$v,data$w,data$lev,lat=data$lat)
```

---

calc\_pv

*Potential Vorticity (PV)*

---

**Description**

Calculates the potential vorticity

**Usage**

```
calc_pv(
  t_fld,
  u_fld,
  v_fld,
  w_fld,
  lev_p,
  lat = NULL,
  dx = 0.25,
```

```

dy = 0.25,
zvort_only = FALSE,
relative = FALSE,
zvort_fld = NULL,
mode = "lonlat"
)

```

### Arguments

t_fld	temperature field [K]
u_fld	zonal velocity field [m/s]
v_fld	meridional velocity field [m/s]
w_fld	vertical velocity field [m/s]
lev_p	vector containing pressure levels [Pa]
lat	vector containing latitude
dx	x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
zvort_only	logical, TRUE: if only the vertical vorticity (zvort) should be calculated, FALSE: for the whole vorticity vector, default: FALSE
relative	logical, TRUE: only relative vorticity, FALSE: whole (absolute) vorticity, default: FALSE
zvort_fld	optional zvort field (if e.g., zvort is directly taken from ERA5 and not calculated separately)
mode	use 'lonlat' if the data is given on a lon-lat-grid or 'cartesian' if the data is given on an equidistant cartesian grid

### Value

potential vorticity field [ $K \cdot m^2 / (kg \cdot s)$ ]

### Examples

```

myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
#PV based on all three components
pv=calc_pv(data$temp,data$u,data$v,data$w,data$lev,lat=data$lat)
#PV only based on vertical component
pv_vert=calc_pv(data$temp,data$u,data$v,data$w,lev_p=data$lev,lat=data$lat,zvort_only=TRUE)

```

---

calc_tfp	<i>Thermic Front Parameter (TFP)</i>
----------	--------------------------------------

---

**Description**

Calculates the thermic front parameter based on the potential temperature

**Usage**

```
calc_tfp(t_fld, lev_p, lat = NULL, dx = 0.25, dy = 0.25, mode = "lonlat")
```

**Arguments**

t_fld	temperature field [K]
lev_p	vector containing pressure levels [Pa]
lat	only for lonlat mode: vector containing latitude
dx	x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
mode	the horizontal coordinate system, options are 'lonlat' for a longitude-latitude-grid (default), or 'cartesian' for an equidistant cartesian grid

**Value**

thermic front parameter [K/m<sup>2</sup>]

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
tfp=calc_tfp(data$temp,data$lev,data$lat)
```

---

calc_theta	<i>Potential temperature</i>
------------	------------------------------

---

**Description**

Calculates the potential temperature

**Usage**

```
calc_theta(t_fld, lev_p)
```

**Arguments**

t\_fld            temperature field [K]  
lev\_p            vector containing pressure levels [Pa]

**Value**

density [kg/m<sup>3</sup>]

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
theta=calc_theta(data$temp,data$lev)
```

---

calc_vorticity	<i>Vorticity</i>
----------------	------------------

---

**Description**

Calculates the vorticity

**Usage**

```
calc_vorticity(
  u_fld,
  v_fld,
  w_fld,
  lev_p,
  lat = NULL,
  dx = 0.25,
  dy = 0.25,
  zvort_only = FALSE,
  relative = FALSE,
  zvort_fld = NULL,
  mode = "lonlat"
)
```

**Arguments**

u\_fld            zonal velocity field [m/s]  
v\_fld            meridional velocity field [m/s]  
w\_fld            vertical velocity field [m/s]  
lev\_p            vector containing pressure levels [Pa]  
lat              vector containing latitude  
dx              x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat'  
or e.g. 1000 m in cartesian coordinates with mode='cartesian')

dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
zvort_only	logical, TRUE: if only the vertical vorticity (zvort) should be calculated, FALSE: for the whole vorticity vector, default: FALSE
relative	logical, TRUE: only relative vorticity, FALSE: whole (absolute) vorticity, default: FALSE
zvort_fld	optional zvort field (if e.g., zvort is directly taken from ERA5 and not calculated separately)
mode	use 'lonlat' if the data is given on a lon-lat-grid or 'cartesian' if the data is given on an equidistant cartesian grid

**Value**

vorticity field [1/s]

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
#3d vorticity
xi=calc_vorticity(data$u,data$v,data$w,data$lev,lat=data$lat)
#z-vorticity as scalar
zeta=calc_vorticity(data$u,data$v,data$w,data$lev,lat=data$lat,zvort_only=TRUE)
```

---

crossprod

*cross product*

---

**Description**

Calculates the cross product of two given 3d vector fields

**Usage**

```
crossprod(fld1, fld2)
```

**Arguments**

fld1	field 1 with dimensions (lon,lat,p,3)
fld2	field 2 with dimensions (lon,lat,p,3)

**Value**

field containing the cross product

df\_dp

*df\_dp***Description**

Calculates the p derivative (pressure system) using central differences

**Usage**

```
df_dp(fld, plev = 5000)
```

**Arguments**

fld	field with dimensions (lon,lat,p)
plev	a scalar containing the p resolution (if equidistant) or a vector containing pressure levels in Pa (for non-equidistant)

**Value**

field containing the partial derivative w.r.t. p

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
theta=calc_theta(data$temp,data$lev)
dtheta_dp=df_dp(theta)
```

df\_dx

*df\_dx***Description**

Calculates the x derivative using central differences (for lonlat-grid or cartesian grid)

**Usage**

```
df_dx(fld, lat = NULL, dx = 0.25, mode = "lonlat")
```

**Arguments**

fld	field with dimensions (lon,lat,p)
lat	only for lonlat mode: vector containing latitude
dx	x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
mode	the coordinate system, options are lonlat for a longitude-latitude-grid (default), or cartesian for an equidistant cartesian grid

**Value**

field containing the partial derivative w.r.t. x

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
theta=calc_theta(data$temp,data$lev)
dtheta_dx=df_dx(theta,data$lat)
```

df\_dy

*df\_dy***Description**

Calculates the y derivative using central differences

**Usage**

```
df_dy(fld, dy = 0.25, mode = "lonlat")
```

**Arguments**

fld	with dimensions (lon,lat,p)
dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
mode	the coordinate system, options are lonlat for a longitude-latitude-grid (default), or cartesian for an equidistant cartesian grid

**Value**

field containing the partial derivative w.r.t. y

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
theta=calc_theta(data$temp,data$lev)
dtheta_dy=df_dy(theta,dy=0.25)
```

---

 df\_dz

 $df_{dz}$ 


---

**Description**

Calculates the z derivative

**Usage**

```
df_dz(fld, rho, plev = 5000)
```

**Arguments**

fld	field with dimensions (lon,lat,p)
rho	field with dimensions (lon,lat,p) for density or a scalar rho (for constant density)
plev	a scalar containing the p resolution (if equidistant) or a vector containing pressure levels in Pa (for non-equidistant)

**Value**

field containing the partial derivative w.r.t. z

---

 div

 $divergence$ 


---

**Description**

Calculates the divergence of a vector field

**Usage**

```
div(
  fld,
  lat = NULL,
  d = 3,
  system = "p",
  rho = NULL,
  dx = 0.25,
  dy = 0.25,
  plev = 5000,
  mode = "lonlat"
)
```

**Arguments**

fld	field with dimensions (lon,lat,p,d)
lat	vector containing latitude (only for mode='lonlat')
d	scalar for dimension (use d=2 for horizontal gradient and d=3 for 3d-gradient)
system	for type of coordinate system (use 'p' for pressure system and 'z' for height system)
rho	field with dimensions (lon,lat,p) for density or a scalar rho (for constant density)
dx	x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
plev	a scalar containing the p resolution (if equidistant) or a vector containing pressure levels in Pa (for non-equidistant)
mode	the coordinate system, options are lonlat for a longitude-latitude-grid (default), or cartesian for an equidistant cartesian grid

**Value**

field containing the divergence of fld

---

fill_horiz	<i>Plotting a xy domain with custom boundaries, colour paletts and optional world map</i>
------------	---

---

**Description**

Plotting a xy domain with custom boundaries, colour paletts and optional world map

**Usage**

```
fill_horiz(
  x,
  y,
  fld,
  levels = 1:100,
  main = "",
  worldmap = TRUE,
  legend_loc = "topright",
  legend_title = "",
  legend_only = FALSE,
  Lab = NULL,
  ...
)
```

**Arguments**

x	array containing x-axis values (e.g. longitude)
y	array containing y-axis values (e.g. latitude)
fld	field (which should be plotted) with dimensions (x,y)
levels	levels for colour bar
main	character containing main title of the plot
worldmap	should the world map contours be plotted (default TRUE)
legend_loc	location of legend
legend_title	character containing legend title
legend_only	logical TRUE only legend should be plotted, or FALSE everything should be plotted (default)
Lab	lab palette from type colorRampPalette
...	additional graphic parameters

**Value**

no return

---

frontid

*Front Identification und Statistics*

---

**Description**

Calculates frontal zones based on a chosen method (TFP, F diagnostic, DSI) and provides statistics of the distribution of meteorological quantities inside the determined frontak zones.

**Usage**

```
frontid(
  t_fld,
  u_fld = NULL,
  v_fld = NULL,
  w_fld = NULL,
  phi_fld = NULL,
  lev_p,
  lat = NULL,
  method = "tfp",
  threshold = 2 * 10^-10,
  dx = 0.25,
  dy = 0.25,
  fronts_only = FALSE,
  mode = "lonlat"
)
```

**Arguments**

t_fld	temperature field [K]
u_fld	zonal velocity field [m/s]
v_fld	meridional velocity field [m/s]
w_fld	vertical velocity field [m/s]
phi_fld	geopotential height [gpm]
lev_p	vector containing pressure levels [Pa]
lat	only for lonlat mode: vector containing latitude
method	character containing the method, use 'tfp' for TFP method, 'f' for F diagnostic and 'dsi' for DSI method
threshold	scalar containing a suitable threshold (e.g., $2 \times 10^{-10}$ for TFP method, 1 or for F diagnostic, $10^{-16}$ for DSI method)
dx	x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
fronts_only	if you only want to calculate the frontal regions and not their properties (default FALSE)
mode	the horizontal coordinate system, options are lonlat for a longitude-latitude-grid (default), or cartesian for an equidistant cartesian grid

**Value**

list containing the used method and used threshold, field with logicals containing the detected frontal zones and numerics of temperature, u-wind, v-wind, w-wind, geopotential, vorticity, PV and DSI inside the determined frontal zones

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)

#front identification using the thermic front parameter (example without front statistic)
tfp_fronts=frontid(data$temp,lev_p=data$lev,lat=data$lat,fronts_only=TRUE)

#front identification using F diagnostic (example with front statistic)
f_fronts=frontid(data$temp,data$u,data$v,data$w,data$z,lev_p=data$lev,lat=data$lat,
method='f',threshold=2,fronts_only=FALSE)

#front identification using the dynamic state index (example with statistic)
dsi_fronts=frontid(data$temp,data$u,data$v,data$w,data$z,lev_p=data$lev,lat=data$lat,
method='dsi',threshold=4*10^-16,fronts_only=FALSE)
```

---

grad                      *gradient of a scalar field*

---

### Description

Calculates the gradient

### Usage

```
grad(
  fld,
  lat = NULL,
  d = 3,
  system = "p",
  rho = NULL,
  dx = 0.25,
  dy = 0.25,
  plev = 5000,
  mode = "lonlat"
)
```

### Arguments

fld	field with dimensions (lon,lat,p)
lat	vector containing latitude
d	scalar for dimension (use d=2 for horizontal gradient and d=3 for 3d-gradient)
system	for type of coordinate system (use 'p' for pressure system and 'z' for height system)
rho	field with dimensions (lon,lat,p) for density or a scalar rho (for constant density)
dx	x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
plev	a scalar containing the p resolution (if equidistant) or a vector containing pressure levels in Pa (for non-equidistant)
mode	the coordinate system, options are lonlat for a longitude-latitude-grid (default), or cartesian for an equidistant cartesian grid

### Value

field containing the gradient with dimension (lon,lat,p,d)

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
theta=calc_theta(data$temp,data$lev)
theta_grad=grad(theta,data$lat)
```

jacobian

*Jacobian matrix and determinant***Description**

Calculates the Jacobian matrix and Jacobian determinant for 2 or 3 given scalar fields

**Usage**

```
jacobian(
  fld1,
  fld2,
  fld3 = NULL,
  lat = NULL,
  d = 3,
  system = "p",
  rho = NULL,
  dx = 0.25,
  dy = 0.25,
  plev = 5000,
  mode = "lonlat"
)
```

**Arguments**

fld1	field 1 with dimensions (lon,lat,p)
fld2	field 2 with dimensions (lon,lat,p)
fld3	field 3 with dimensions (lon,lat,p)
lat	vector containing latitude
d	scalar for dimension (use d=2 for 2 input fields and d=3 for 3 inpt fields)
system	for type of coordinate system (use 'p' for pressure system and 'z' for height system)
rho	field with dimensions (lon,lat,p) for density or a scalar rho (for constant density)
dx	x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
plev	a scalar containing the p resolution (if equidistant) or a vector containing pressure levels in Pa (for non-equidistant)
mode	the coordinate system, options are lonlat for a longitude-latitude-grid (default), or cartesian for an equidistant cartesian grid

**Value**

list containing Jacobian matrix and determinant

---

readin_dim	<i>read in dimensions</i>
------------	---------------------------

---

**Description**

: reads dimensions of ERA5 data

**Usage**

```
readin_dim(filename)
```

**Arguments**

filename            name of file to read in

**Value**

no return

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data_dims = readin_dim(myfile)
```

---

readin_era5	<i>read in ERA5 data</i>
-------------	--------------------------

---

**Description**

: reads ERA5 data

**Usage**

```
readin_era5(filename)
```

**Arguments**

filename            name of file to read in

**Value**

no return

**Examples**

```
myfile=system.file("extdata", "era5_storm-zeynep.nc", package = "meteoEVT")
data = readin_era5(myfile)
```

---

rot	<i>rotation</i>
-----	-----------------

---

**Description**

Calculates the rotation of a vector field

**Usage**

```
rot(
  fld,
  lat = NULL,
  d = 3,
  system = "p",
  rho = NULL,
  dx = 0.25,
  dy = 0.25,
  plev = 5000,
  mode = "lonlat"
)
```

**Arguments**

fld	with dimensions (lon,lat,p,d)
lat	vector containing latitude
d	scalar for dimension (use d=2 for horizontal gradient and d=3 for 3d-gradient)
system	for type of coordinate system (use 'p' for pressure system and 'z' for height system)
rho	field with dimensions (lon,lat,p) for density or a scalar rho (for constant density)
dx	x resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
dy	y resolution in the corresponding unit (e.g. 0.25 degree for ERA5 with mode='lonlat' or e.g. 1000 m in cartesian coordinates with mode='cartesian')
plev	a scalar containing the p resolution (if equidistant) or a vector containing pressure levels in Pa (for non-equidistant)
mode	the coordinate system, options are lonlat for a longitude-latitude-grid (default), or cartesian for an equidistant cartesian grid

**Value**

field containing the divergence of fld

---

scalarprod

*scalar product*

---

**Description**

Calculates the scalar product of two given fields

**Usage**

```
scalarprod(fld1, fld2)
```

**Arguments**

fld1            field 1 with dimensions (lon,lat,p,d)

fld2            field 2 with dimensions (lon,lat,p,d)

**Value**

field of the scalar product with dimensions (lon,lat,p)

# Index

[calc\\_bernoulli](#), 3  
[calc\\_density](#), 4  
[calc\\_dsi](#), 4  
[calc\\_enstrophy](#), 6  
[calc\\_fdiag](#), 7  
[calc\\_frontogenesis](#), 8  
[calc\\_helicity](#), 9  
[calc\\_lamb](#), 10  
[calc\\_pv](#), 11  
[calc\\_tfp](#), 13  
[calc\\_theta](#), 13  
[calc\\_vorticity](#), 14  
[crossprod](#), 15

[df\\_dp](#), 16  
[df\\_dx](#), 16  
[df\\_dy](#), 17  
[df\\_dz](#), 18  
[div](#), 18

[fill\\_horiz](#), 19  
[frontid](#), 20

[grad](#), 22

[jacobian](#), 23

[meteoEVT-package](#), 2

[readin\\_dim](#), 24  
[readin\\_era5](#), 24  
[rot](#), 25

[scalarprod](#), 26