

# Interpola

## v2.0

Generated by Doxygen 1.8.18



---

<b>1 Interpola</b>	<b>1</b>
1.1 Mass conservative interpolation in overlapping grids	3
<b>2 Modules Index</b>	<b>7</b>
2.1 Modules List	7
<b>3 File Index</b>	<b>9</b>
3.1 File List	9
<b>4 Module Documentation</b>	<b>11</b>
4.1 vars_dat Module Reference	11
4.1.1 Detailed Description	14
4.1.2 Variable Documentation	14
4.1.2.1 cday	14
4.1.2.2 cenlat	14
4.1.2.3 cenlon	15
4.1.2.4 cname	15
4.1.2.5 cunits	15
4.1.2.6 current_date	15
4.1.2.7 dix	15
4.1.2.8 djx	15
4.1.2.9 dlat	16
4.1.2.10 dlon	16
4.1.2.11 dpob	16
4.1.2.12 dx	16
4.1.2.13 dxe	16
4.1.2.14 dy	16
4.1.2.15 dye	17
4.1.2.16 ed	17
4.1.2.17 ei	17
4.1.2.18 eix	17
4.1.2.19 ejx	17
4.1.2.20 elat	17
4.1.2.21 elon	18
4.1.2.22 ename	18
4.1.2.23 epob	18
4.1.2.24 gmt	18
4.1.2.25 grid_id	18
4.1.2.26 isice	18
4.1.2.27 islake	19

4.1.2.28 isoilwater . . . . .	19
4.1.2.29 isurban . . . . .	19
4.1.2.30 iswater . . . . .	19
4.1.2.31 itime . . . . .	19
4.1.2.32 julday . . . . .	19
4.1.2.33 julyr . . . . .	20
4.1.2.34 map_proj_char . . . . .	20
4.1.2.35 mapproj . . . . .	20
4.1.2.36 mecha . . . . .	20
4.1.2.37 mminlu . . . . .	20
4.1.2.38 moadcenlat . . . . .	20
4.1.2.39 ndims . . . . .	21
4.1.2.40 nh . . . . .	21
4.1.2.41 num_land_cat . . . . .	21
4.1.2.42 pollat . . . . .	21
4.1.2.43 pollon . . . . .	21
4.1.2.44 radm . . . . .	21
4.1.2.45 sdim . . . . .	22
4.1.2.46 stdlon . . . . .	22
4.1.2.47 times . . . . .	22
4.1.2.48 title . . . . .	22
4.1.2.49 tpob . . . . .	22
4.1.2.50 trulat1 . . . . .	22
4.1.2.51 trulat2 . . . . .	23
4.1.2.52 tvar . . . . .	23
4.1.2.53 unlimdimid . . . . .	23
4.1.2.54 xlat . . . . .	23
4.1.2.55 xlon . . . . .	23
4.1.2.56 zlev . . . . .	23
<b>5 File Documentation</b>	<b>25</b>
5.1 calculos.F90 File Reference . . . . .	25
5.1.1 Function/Subroutine Documentation . . . . .	25
5.1.1.1 conversion() . . . . .	25
5.2 Interpola.F90 File Reference . . . . .	26
5.2.1 Function/Subroutine Documentation . . . . .	26
5.2.1.1 interpola() . . . . .	26
5.3 lee_files.F90 File Reference . . . . .	27
5.3.1 Function/Subroutine Documentation . . . . .	27

---

5.3.1.1 check()	27
5.3.1.2 file_reading()	27
5.4 README.md File Reference	28
5.5 salidas.F90 File Reference	28
5.5.1 Function/Subroutine Documentation	28
5.5.1.1 crea_attr()	28
5.5.1.2 file_out()	29
5.6 vars_dat_mod.F90 File Reference	29
<b>Index</b>	<b>33</b>



# Chapter 1

## Interpola

Emission Interpolation to a new mesh by using a conservative flux method

The new mesh is provided by the wrfinput file and the original grid with emissions are provided from wrfchem file  
input files:

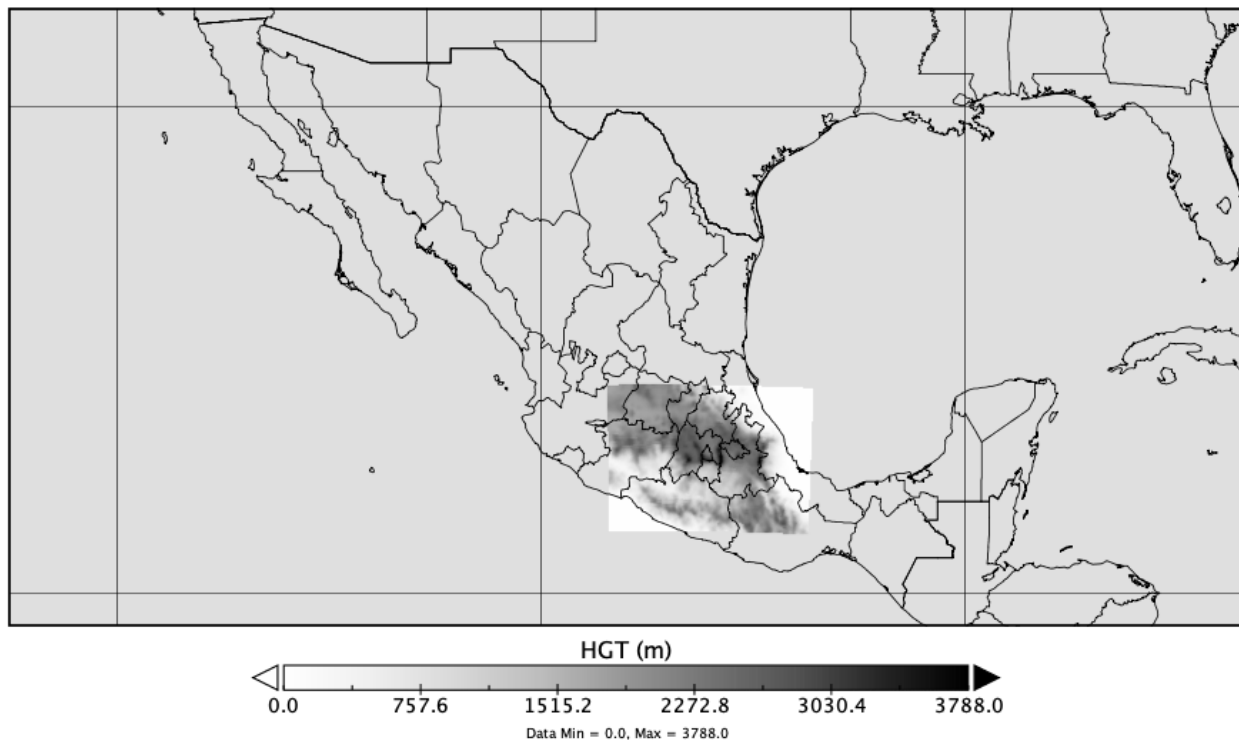
```
wrfchemin.nc ! A 12 hours emission file to be interpolated (0 to 11 hour or 12 to 23 hour)
wrfinput      ! Domain where emissions will be interpolated
```

output file:

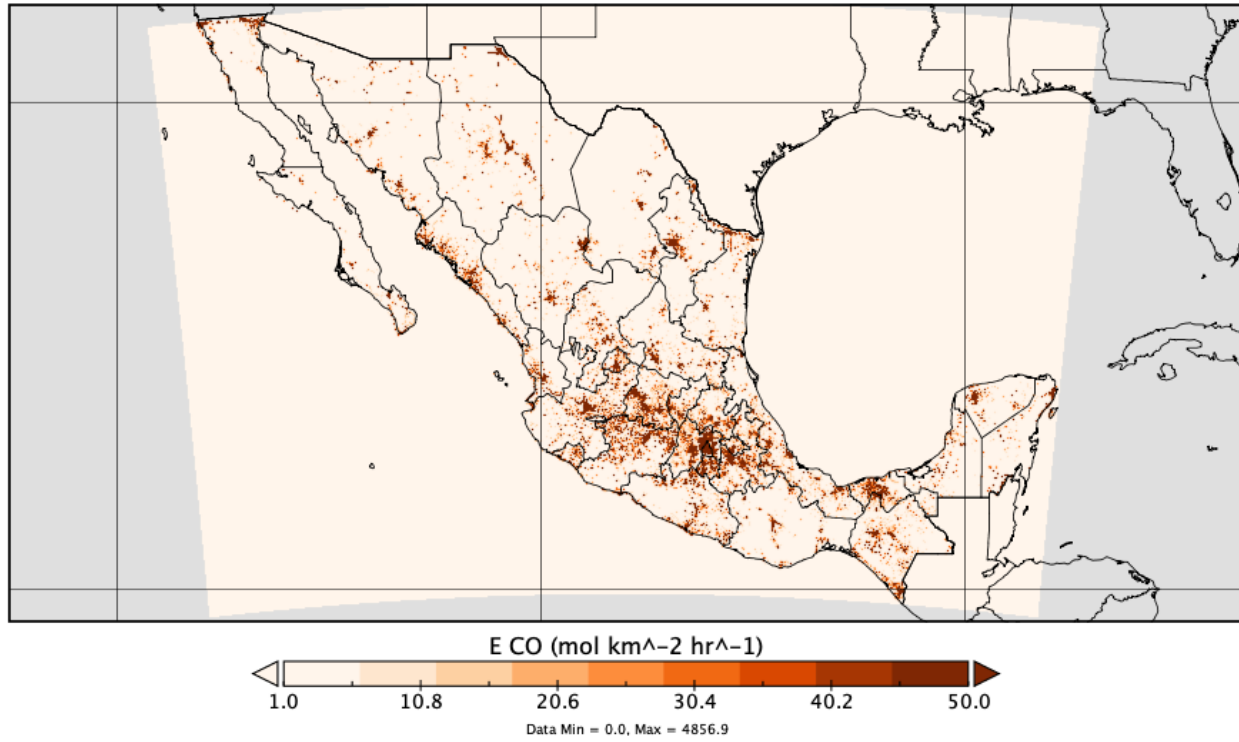
```
wrfchemi_00z_d01 or wrfchemi_12z_d01 ! 00z or 12z based on emissions file.
                                         ! d01, d02,... based on wrfinput
```

wrfchemin.nc - file contain emissions starting with "E\_"

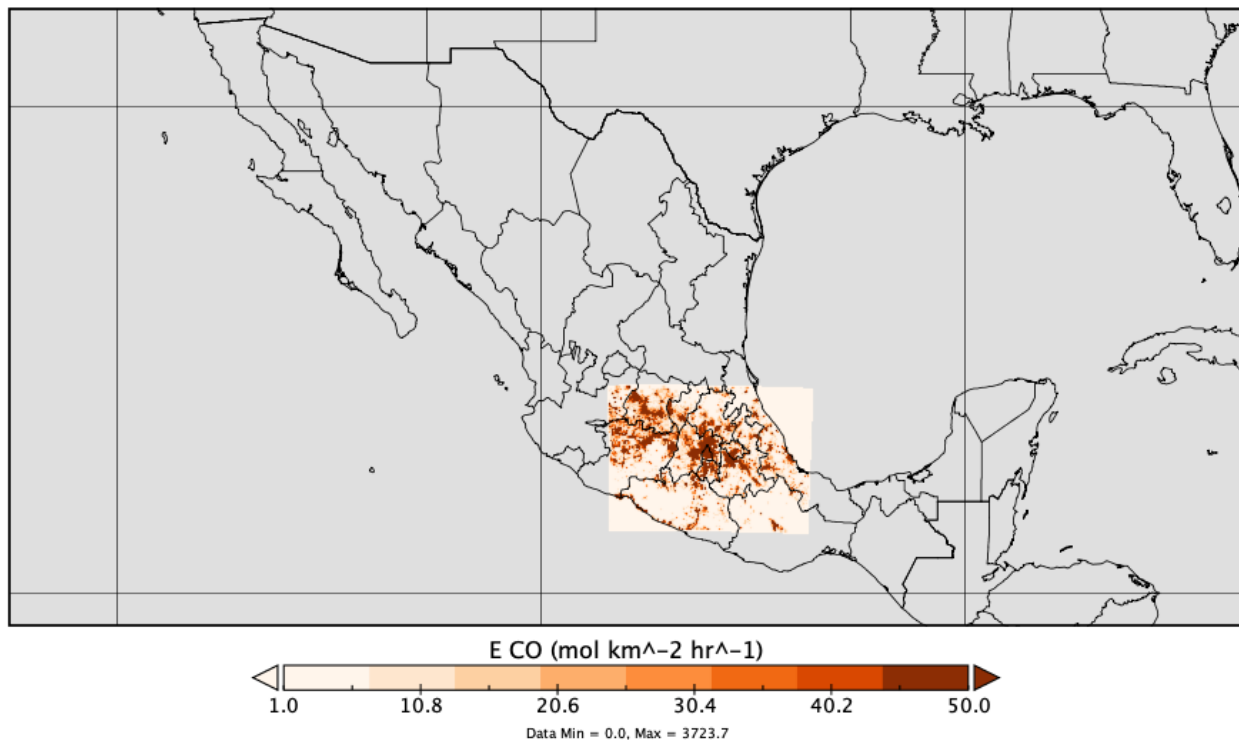
Domain to interpolate



## Emission Domain



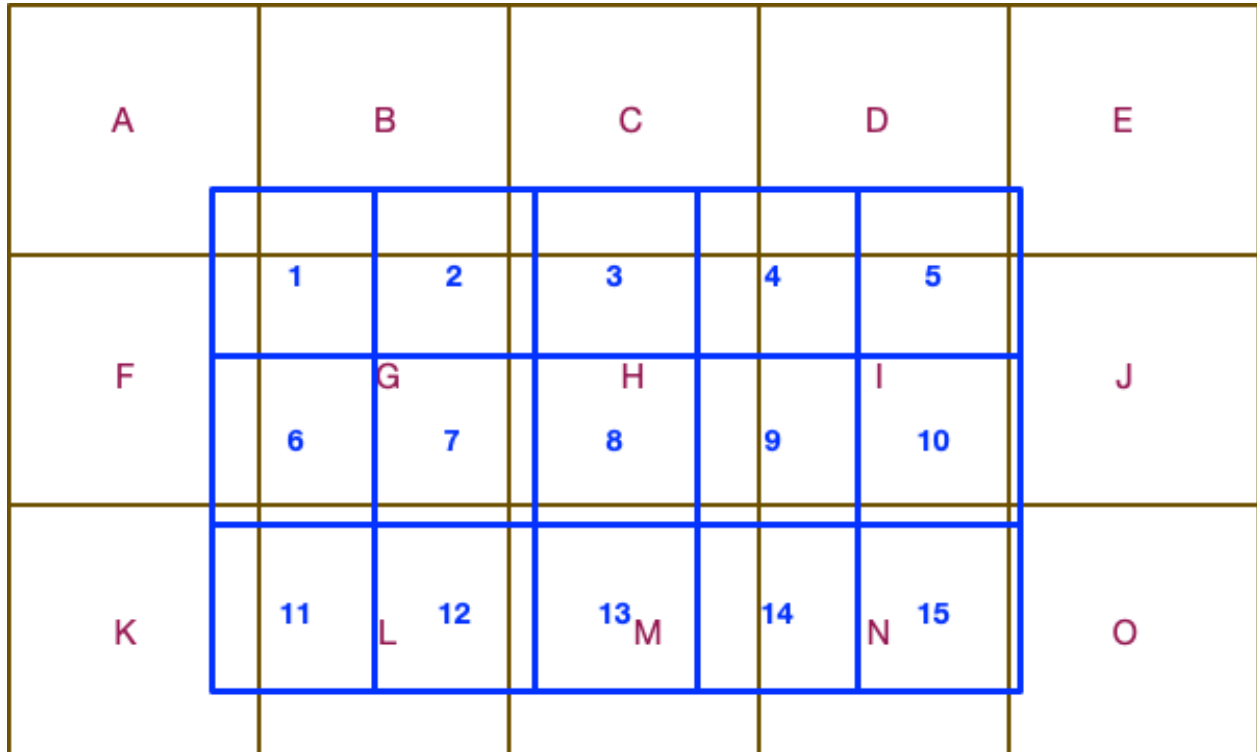
Output file: wrfchemi\_00z\_d02  
00Z based on emissions and d02 based on wrfinput





## 1.1 Mass conservative interpolation in overlapping grids

Emissions Inventories are generated for represent global, regional or local emissions, air quality models domains are displayed in a different grid (dimensions and location) than the emissions. On other hand, emissions are a flux and it is necessary to use a mass conservative interpolation in order to avoid inconsistencies in the emissions. An example of overlapping grids is presented in the followgin figure.

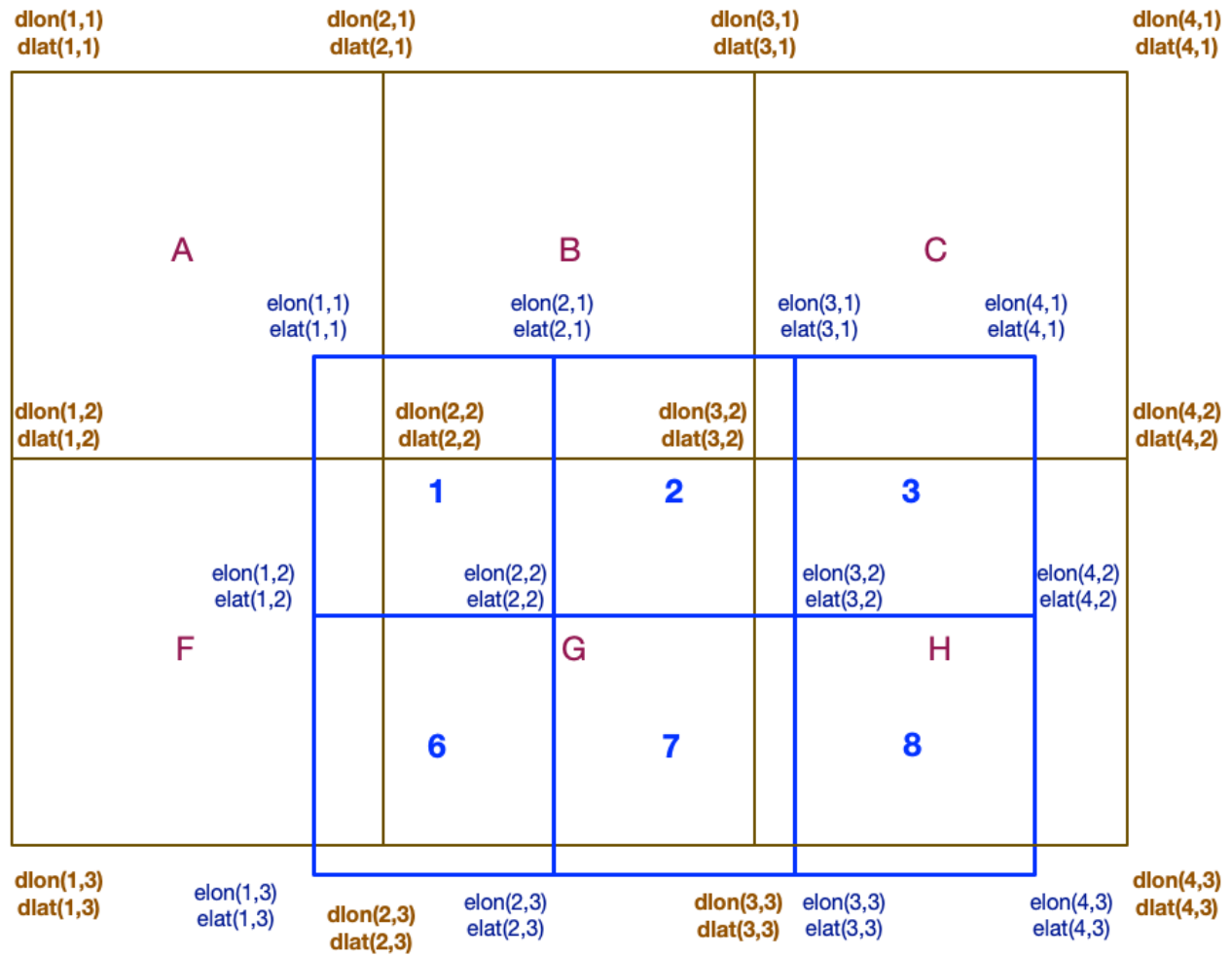


Grid with letters represents the new domain (it has coordinates dlat,dlon) and grid with numbers is the emissions (it has coordinates elat,elon). For N emission there are N+1 coordinates in each axis.

For the new domain to interpolate the emissions the following variables are set:

New domain	Emissions domain
$y_{lat1} = .5 * (dlat(i, j-1) + dlat(i, j))$	$elat1 = 0.5 * (elat(ii, jj) + elat(ii, jj-1))$
$y_{lat2} = .5 * (dlat(i, j+1) + dlat(i, j))$	$elat2 = 0.5 * (elat(ii, jj) + elat(ii, jj+1))$
$x_{lon1} = .5 * (dlon(i-1, j) + dlon(i, j))$	$elon1 = 0.5 * (elon(ii, jj) + elon(ii-1, jj))$
$x_{lon2} = .5 * (dlon(i+1, j) + dlon(i, j))$	$elon2 = 0.5 * (elon(ii, jj) + elon(ii+1, jj))$

The following figure presents the locations of the coordinates in each mesh.

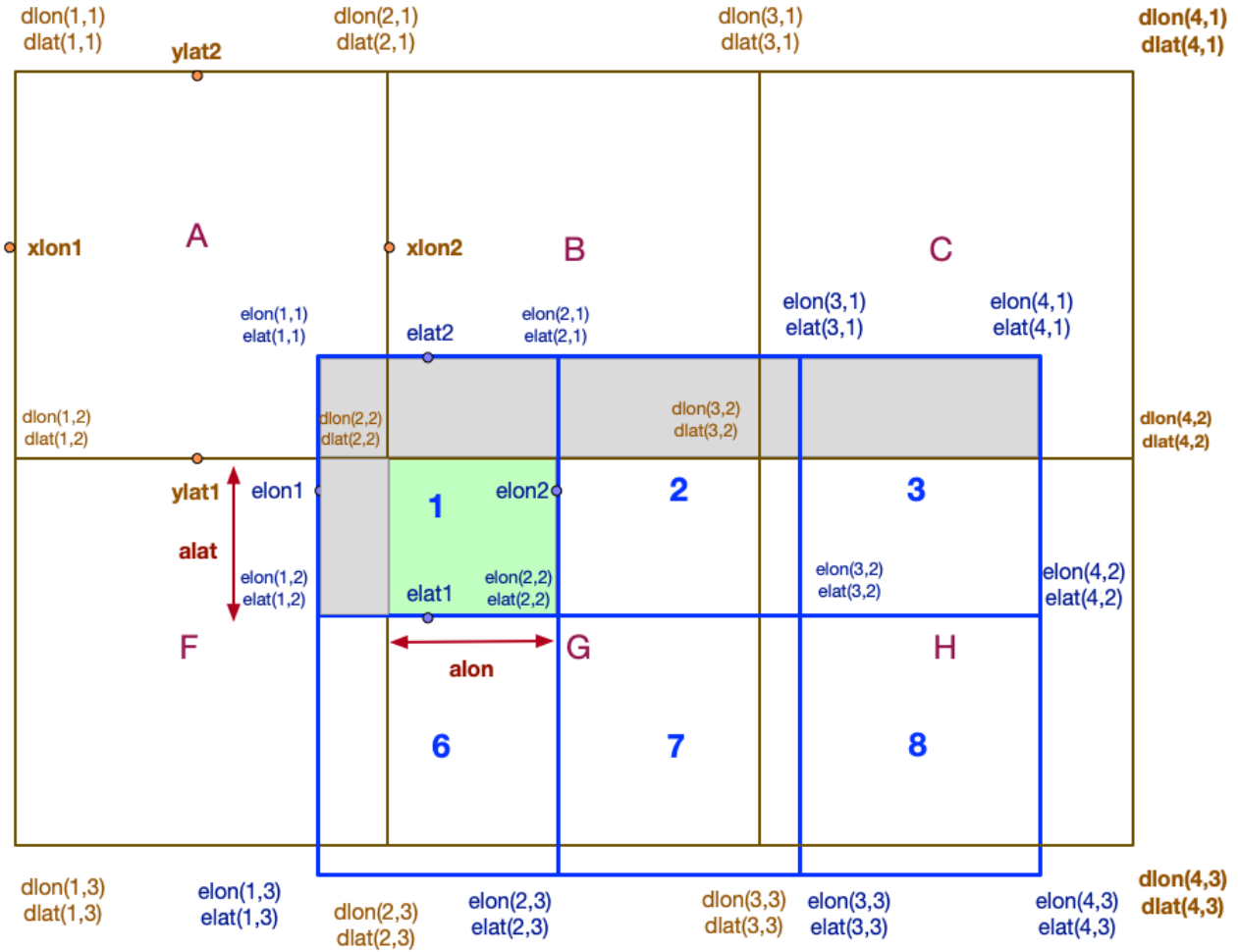


The following procedure is followed for obtaining the new emissions in the new grid:

```

alat=(min(ylat2,elat2)-max(ylat1,elat1))
alon=(min(xlon2,elon2)-max(xlon1,elon1))
area=max(0.,alat*alon)

```



Emissions in cell **A** are a flux emission fraction from 1, for **B** a flux fractions from 1 and 2, and for **C** fluxes from 2 and 3, **F** fluxes from 1 and 6, **G** from 1, 2, 6 and 7 and **H** from 2, 3, 4, 7, 8 and 9, and so on.



## Chapter 2

# Modules Index

### 2.1 Modules List

Here is a list of all modules with brief descriptions:

<a href="#">vars_dat</a>	Set up variables used during the process . . . . .	<a href="#">11</a>
--------------------------	--	--------------------



# Chapter 3

## File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">calculos.F90</a>	25
<a href="#">Interpola.F90</a>	26
<a href="#">lee_files.F90</a>	27
<a href="#">salidas.F90</a>	28
<a href="#">vars_dat_mod.F90</a>	29





## Chapter 4

# Module Documentation

### 4.1 vars\_dat Module Reference

Set up variables used during the process.

#### Variables

- integer, parameter `nh` =24
- integer `ndims`  
*Number of dimension in wrfinput file.*
- integer `zlev`  
*Number of emissions layers (1 to 8)*
- integer `radm` =0  
*number of emissions classes*
- real, dimension(:,:,:), allocatable `ei`  
*emissions input file dimensions `nx, ny, level, nh, radm`*
- real, dimension(:,:,:), allocatable `ed`  
*emissions in new DOMAIN file dimensions `nx, ny, level, nh, radm`*
- real, dimension(:,:), allocatable `elat`  
*Latitudes from input file emissions.*
- real, dimension(:,:), allocatable `elon`  
*Longitudes from input file emissions.*
- real, dimension(:,:), allocatable `epob`  
*Density population from input file emissions.*
- real, dimension(:,:), allocatable `dlat`  
*Latitudes in new domain `nx, ny` from new domain `ed`.*
- real, dimension(:,:), allocatable `dlon`  
*Longitudes in new domain `nx, ny` from new domain `ed`.*
- real, dimension(:,:), allocatable `dpob`  
*Density population in new domain `nx, ny` from new domain `ed`.*
- real, dimension(:,:), allocatable `xlon`

- Longitudes in emissions domain  $nx, ny$ .*

  - real, dimension(:, :, :), allocatable [xlat](#)
- Latitudes in emissions domain  $nx, ny$ .*

  - integer [dix](#)
- Number of values in longitude in new file.*

  - integer [dix](#)
- Number of values in latitude in new file.*

  - integer [eix](#)
- Number of values in longitude in emissions file.*

  - integer [ejx](#)
- Number of values in latitude in emissions file.*

  - integer [grid\\_id](#)
- Domain number (d01, d02, etc.) from wrfinput.*

  - integer [julyr](#)
- Julian year in emissions file.*

  - integer [julday](#)
- Julian day in emissions file.*

  - integer [mapproj](#)
- Map projection type.*

  - integer [iswater](#)
- Value for land use water.*

  - integer [islake](#)
- Value for land use lake.*

  - integer [isice](#)
- Value for land use ice.*

  - integer [isurban](#)
- Value for land use urban.*

  - integer [isoilwater](#)
- Value for land use ice.*

  - integer [unlimdimid](#)
- ID unlimit variable (time)*

  - real [cenlat](#)
- Central latitude.*

  - real [cenlon](#)
- Central longitude.*

  - real [dx](#)
- Grid dimension in m output file x.*

  - real [dy](#)
- Grid dimension in m output file y.*

  - real [dx](#)
- Grid dimension in m emissions file x.*

  - real [dye](#)
- Grid dimension in m emissions file y.*

  - real [trulat1](#)
- True latitud lower.*

  - real [trulat2](#)
- True latitud higer.*

- real [moadcenlat](#)  
*Mother of all domains center latitude.*
- real [stdlon](#)  
*Standard longitude.*
- real [pollat](#)  
*The pole latitude.*
- real [pollon](#)  
*The pole longitude.*
- real [gmt](#)  
*GMT time.*
- real [num\\_land\\_cat](#)  
*Number of land categories.*
- character(len=3) [cday](#)  
*Day type (lun, mar, mie, jue, vie, sab, dom)*
- character(len=19) [mminlu](#)  
*Land use input description.*
- character(len=19) [map\\_proj\\_char](#)  
*Map projection description.*
- character(len=19) [itime](#)  
*Counter for time in file.*
- character(len=38) [title](#)  
*Title description input/output files for V4 should have V4.0.*
- character(len=19), dimension(1, 1) [times](#)  
*Start date in input emissions file.*
- character(len=19) [current\\_date](#)  
*Current date in input emissions file.*
- character(len=19) [mecha](#)  
*Chemical mechanism name.*
- character(len=19), dimension(:), allocatable [sdim](#)  
*Vector of dimensions descriptions.*
- character(len=11), dimension(:), allocatable [ename](#)  
*Emissions description long.*
- character(len=50), dimension(:), allocatable [cname](#)  
*Emissions name variable short.*
- character(len=50), dimension(:), allocatable [cunits](#)  
*Units in emissions vars.*
- logical, dimension(:), allocatable [tvar](#)  
*true if input var is an emissions variable*
- logical [tpob](#)  
*true if input emissions files contains density population*

### 4.1.1 Detailed Description

Set up variables used during the process.

#### Author

Jose Agustin Garcia Reynoso

#### Date

28-08-2012

#### Emissions Inventories Variables

#### Parameters

<i>nh</i>	Number of hours during the day
-----------	--------------------------------

### 4.1.2 Variable Documentation

#### 4.1.2.1 cday

```
character(len=3) vars_dat::cday
```

Day type (lun, mar, mie, jue, vie, sab, dom)

#### 4.1.2.2 cenlat

```
real vars_dat::cenlat
```

Central latitude.

#### 4.1.2.3 cenlon

```
real vars_dat::cenlon
```

Central longitude.

#### 4.1.2.4 cname

```
character(len=50), dimension(:), allocatable vars_dat::cname
```

Emissions name variable short.

#### 4.1.2.5 cunits

```
character(len=50), dimension(:), allocatable vars_dat::cunits
```

Units in emissions vars.

#### 4.1.2.6 current\_date

```
character (len=19) vars_dat::current_date
```

Current date in input emissions file.

#### 4.1.2.7 dix

```
integer vars_dat::dix
```

Number of values in longitude in new file.

#### 4.1.2.8 djx

```
integer vars_dat::djx
```

Number of values in latitude in new file.

#### 4.1.2.9 dlat

```
real, dimension(:,:), allocatable vars_dat::dlat
```

Latitudes in new domain  $nx, ny$  from new domain  $ed$ .

#### 4.1.2.10 dlon

```
real, dimension(:,:), allocatable vars_dat::dlon
```

Longitudes in new domain  $nx, ny$  from new domain  $ed$ .

#### 4.1.2.11 dpob

```
real, dimension(:,:), allocatable vars_dat::dpob
```

Density population in new domain  $nx, ny$  from new domain  $ed$ .

#### 4.1.2.12 dx

```
real vars_dat::dx
```

Grid dimension in m output file  $x$ .

#### 4.1.2.13 dxe

```
real vars_dat::dxe
```

Grid dimension in m emissions file  $x$ .

#### 4.1.2.14 dy

```
real vars_dat::dy
```

Grid dimension in m output file  $y$ .

#### 4.1.2.15 dye

```
real vars_dat::dye
```

Grid dimension in m emissions file y.

#### 4.1.2.16 ed

```
real, dimension(:,:,:,:), allocatable vars_dat::ed
```

emissions in new DOMAIN file dimensions nx, ny, level, nh, radm

#### 4.1.2.17 ei

```
real, dimension(:,:,:,:), allocatable vars_dat::ei
```

emissions input file dimensions nx, ny, level, nh, radm

#### 4.1.2.18 eix

```
integer vars_dat::eix
```

Number of values in longitude in emissions file.

#### 4.1.2.19 ejx

```
integer vars_dat::ejx
```

Number of values in latitude in emissions file.

#### 4.1.2.20 elat

```
real, dimension(:,:), allocatable vars_dat::elat
```

Latitudes from input file emissions.

**4.1.2.21 elon**

```
real, dimension(:,:), allocatable vars_dat::elon
```

Longitudes from input file emissions.

**4.1.2.22 ename**

```
character(len=11), dimension(:), allocatable vars_dat::ename
```

Emissions description long.

**4.1.2.23 epob**

```
real, dimension(:,:), allocatable vars_dat::epob
```

Density population from input file emissions.

**4.1.2.24 gmt**

```
real vars_dat::gmt
```

GMT time.

**4.1.2.25 grid\_id**

```
integer vars_dat::grid_id
```

Domain number (d01, d02, etc.) from wrfinput.

**4.1.2.26 isice**

```
integer vars_dat::isice
```

Value for land use ice.



**4.1.2.27 islake**

```
integer vars_dat::islake
```

Value for land use lake.

**4.1.2.28 isoilwater**

```
integer vars_dat::isoilwater
```

Value for land use ice.

**4.1.2.29 isurban**

```
integer vars_dat::isurban
```

Value for land use urban.

**4.1.2.30 iswater**

```
integer vars_dat::iswater
```

Value for land use water.

**4.1.2.31 itime**

```
character(len=19) vars_dat::itime
```

Counter for time in file.

**4.1.2.32 julday**

```
integer vars_dat::julday
```

Julian day in emissions file.

#### 4.1.2.33 julyr

```
integer vars_dat::julyr
```

Julian year in emissions file.

#### 4.1.2.34 map\_proj\_char

```
character(len=19) vars_dat::map_proj_char
```

Map projection description.

#### 4.1.2.35 mapproj

```
integer vars_dat::mapproj
```

Map projection type.

#### 4.1.2.36 mecha

```
character (len=19) vars_dat::mecha
```

Chemical mechanism name.

#### 4.1.2.37 mminlu

```
character(len=19) vars_dat::mminlu
```

Land use input description.

#### 4.1.2.38 moadcenlat

```
real vars_dat::moadcenlat
```

Mother of all domains center latitude.

#### 4.1.2.39 ndims

```
integer vars_dat::ndims
```

Number of dimension in wrfinput file.

#### 4.1.2.40 nh

```
integer, parameter vars_dat::nh =24
```

#### 4.1.2.41 num\_land\_cat

```
real vars_dat::num_land_cat
```

Number of land categories.

#### 4.1.2.42 pollat

```
real vars_dat::pollat
```

The pole latitude.

#### 4.1.2.43 pollon

```
real vars_dat::pollon
```

The pole longitude.

#### 4.1.2.44 radm

```
integer vars_dat::radm =0
```

number of emissions classes

**4.1.2.45 sdim**

```
character (len=19), dimension(:), allocatable vars_dat::sdim
```

Vector of dimensions descriptions.

**4.1.2.46 stdlon**

```
real vars_dat::stdlon
```

Standard longitude.

**4.1.2.47 times**

```
character(len=19), dimension(1,1) vars_dat::times
```

Start date in input emissions file.

**4.1.2.48 title**

```
character(len=38) vars_dat::title
```

Title description input/output files for V4 should have V4.0.

**4.1.2.49 tpob**

```
logical vars_dat::tpob
```

true if input emissions files contains density population

**4.1.2.50 trulat1**

```
real vars_dat::trulat1
```

True latitud lower.

#### 4.1.2.51 trulat2

```
real vars_dat::trulat2
```

True latitud higer.

#### 4.1.2.52 tvar

```
logical, dimension(:), allocatable vars_dat::tvar
```

true if input var is an emissions variable

#### 4.1.2.53 unlimdimid

```
integer vars_dat::unlimdimid
```

ID unlimit variable (time)

#### 4.1.2.54 xlat

```
real, dimension(:,:), allocatable vars_dat::xlat
```

Latitudes in emissions domain nx, ny.

#### 4.1.2.55 xlon

```
real, dimension(:,:), allocatable vars_dat::xlon
```

Longitudes in emissions domain nx, ny.

#### 4.1.2.56 zlev

```
integer vars_dat::zlev
```

Number of emissions layers (1 to 8)



## Chapter 5

# File Documentation

### 5.1 calculos.F90 File Reference

#### Functions/Subroutines

- subroutine [conversion](#)

*It does the interpolation into the new Mesh.*

#### 5.1.1 Function/Subroutine Documentation

##### 5.1.1.1 conversion()

```
subroutine conversion
```

It does the interpolation into the new Mesh.

Interpolates the emissions into new mesh conserving mass uses emission area and the fractional area between the original and new grid to set the emissions.

Computes the mass in the original mesh and compares against the new mesh, if both domains cover the same area the ratio `xemis/ xmas` should be 1

#### Author

Jose Agustin Garcia Reynoso

#### Date

28/08/2012.

#### Version

2.0

## 5.2 Interpola.F90 File Reference

### Functions/Subroutines

- program [interpola](#)  
*Conservative emission interpolation from one mesh to a new mesh.*

### 5.2.1 Function/Subroutine Documentation

#### 5.2.1.1 interpola()

`program interpola`

Conservative emission interpolation from one mesh to a new mesh.

Reads emissions from a wrfchemin file and interpolates to a new mesh provided from wrfinput.

Contains a call for tree subroutines that completes the procedure

##### `file_reading`

Reads Emission inventory and the mesh to interpolate.

##### `conversion`

Computations for emissions mass conservation into the new mesh.

##### `file_out`

Create output file and write results

##### **Author**

Jose Agustin Garcia Reynoso

##### **Date**

2012/06/20

##### **Version**

2.0

##### **Copyright**

Universidad Nacional Autonoma de Mexico 2020



## 5.3 lee\_files.F90 File Reference

### Functions/Subroutines

- subroutine [file\\_reading](#)  
*Reads Emission inventory and the new Mesh to interpolate emissions.*
- subroutine [check](#) (status)  
*Evaluation of netcdf status.*

### 5.3.1 Function/Subroutine Documentation

#### 5.3.1.1 check()

```
subroutine check (
    integer, intent(in) status )
```

Evaluation of netcdf status.

In case of error prints error message description

#### Parameters

<i>status</i>	An error status that might have been returned from a previous call to some netCDF function
---------------	--

#### Date

28/08/2012.

#### 5.3.1.2 file\_reading()

```
subroutine file_reading
```

Reads Emission inventory and the new Mesh to interpolate emissions.

Reads from the emission wrfchemin file the variables and attributes put emissions in `ei` array and coordinates in `xlat`, `xlon`.

reads the new mesh from wrfinput, stores the new coordinates `dlat`, `dlon`

**Author**

Agustin Garcia

**Date**

28/08/2012.

**Version**

2.0

**Copyright**

Universidad Nacional Autonoma de Mexico.

## 5.4 README.md File Reference

## 5.5 salidas.F90 File Reference

### Functions/Subroutines

- subroutine [file\\_out](#)  
*file\_out creates the output file and writes the interpolated emissions from the new mesh*
- subroutine [crea\\_attr](#) (ncid, idm, dimids, svar, cname, cunits, id\_var)  
*creates attributes for gas variables and aerosol variables*

### 5.5.1 Function/Subroutine Documentation

#### 5.5.1.1 crea\_attr()

```
subroutine file_out::crea_attr (  
    integer, intent(in) ncid,  
    integer, intent(in) idm,  
    integer, dimension(idm), intent(in) dimids,  
    character(len=*), intent(in) svar,  
    character(len=*), intent(in) cname,  
    character(len=*), intent(in) cunits,  
    integer, intent(out) id_var )
```

creates attributes for gas variables and aerosol variables

**Author**

Agustin Garcia

**Date**

28/08/2012

**Parameters**

in	<i>ncid</i>	netCDF ID, from a previous call to NF90_OPEN or NF90_CREATE
in	<i>idm</i>	Number of dimensions in <i>dimids</i>
out	<i>id_var</i>	ID from variable to store
in	<i>dimids</i>	Array with ID for each dimension
in	<i>svar</i>	Short name of variable to store
in	<i>cname</i>	Description of variable to store
in	<i>cunits</i>	Units for variable to store

**5.5.1.2 file\_out()**

```
subroutine file_out
```

file\_out creates the output file and writes the interpolated emissions from the new mesh

Uses the attributes from wrfinpout file

Uses current\_date from wrfchemin file

**Author**

Agustin Garcia

**Date**

28/08/2012

**5.6 vars\_dat\_mod.F90 File Reference****Modules**

- module [vars\\_dat](#)

*Set up variables used during the process.*

## Variables

- integer, parameter `vars_dat::nh` =24
- integer `vars_dat::ndims`  
*Number of dimension in wrfinput file.*
- integer `vars_dat::zlev`  
*Number of emissions layers (1 to 8)*
- integer `vars_dat::radm` =0  
*number of emissions classes*
- real, dimension(:, :, :, :), allocatable `vars_dat::ei`  
*emissions input file dimensions nx, ny, level, nh, radm*
- real, dimension(:, :, :, :), allocatable `vars_dat::ed`  
*emissions in new DOMAIN file dimensions nx, ny, level, nh, radm*
- real, dimension(:, :), allocatable `vars_dat::elat`  
*Latitudes from input file emissions.*
- real, dimension(:, :), allocatable `vars_dat::elon`  
*Longitudes from input file emissions.*
- real, dimension(:, :), allocatable `vars_dat::epob`  
*Density population from input file emissions.*
- real, dimension(:, :), allocatable `vars_dat::dlat`  
*Latitudes in new domain nx, ny from new domain ed.*
- real, dimension(:, :), allocatable `vars_dat::dlon`  
*Longitudes in new domain nx, ny from new domain ed.*
- real, dimension(:, :), allocatable `vars_dat::dpob`  
*Density population in new domain nx, ny from new domain ed.*
- real, dimension(:, :, :), allocatable `vars_dat::xlon`  
*Longitudes in emissions domain nx, ny.*
- real, dimension(:, :, :), allocatable `vars_dat::xlat`  
*Latitudes in emissions domain nx, ny.*
- integer `vars_dat::dix`  
*Number of values in longitude in new file.*
- integer `vars_dat::djx`  
*Number of values in latitude in new file.*
- integer `vars_dat::eix`  
*Number of values in longitude in emissions file.*
- integer `vars_dat::ejx`  
*Number of values in latitude in emissions file.*
- integer `vars_dat::grid_id`  
*Domain number (d01, d02, etc.) from wrfinput.*
- integer `vars_dat::julyr`  
*Julian year in emissions file.*
- integer `vars_dat::julday`  
*Julian day in emissions file.*
- integer `vars_dat::mapproj`  
*Map projection type.*
- integer `vars_dat::iswater`  
*Value for land use water.*

- integer `vars_dat::islake`  
*Value for land use lake.*
- integer `vars_dat::isice`  
*Value for land use ice.*
- integer `vars_dat::isurban`  
*Value for land use urban.*
- integer `vars_dat::isoilwater`  
*Value for land use ice.*
- integer `vars_dat::unlimdimid`  
*ID unlimit variable (time)*
- real `vars_dat::cenlat`  
*Central latitude.*
- real `vars_dat::cenlon`  
*Central longitude.*
- real `vars_dat::dx`  
*Grid dimension in m output file x.*
- real `vars_dat::dy`  
*Grid dimension in m output file y.*
- real `vars_dat::dxo`  
*Grid dimension in m emissions file x.*
- real `vars_dat::dyo`  
*Grid dimension in m emissions file y.*
- real `vars_dat::trulat1`  
*True latitud lower.*
- real `vars_dat::trulat2`  
*True latitud higer.*
- real `vars_dat::moadcenlat`  
*Mother of all domains center latitude.*
- real `vars_dat::stdlon`  
*Standard longitude.*
- real `vars_dat::pollat`  
*The pole latitude.*
- real `vars_dat::pollon`  
*The pole longitude.*
- real `vars_dat::gmt`  
*GMT time.*
- real `vars_dat::num_land_cat`  
*Number of land categories.*
- character(len=3) `vars_dat::cday`  
*Day type (lun, mar, mie, jue, vie, sab, dom)*
- character(len=19) `vars_dat::mminlu`  
*Land use input description.*
- character(len=19) `vars_dat::map_proj_char`  
*Map projection description.*
- character(len=19) `vars_dat::itime`  
*Counter for time in file.*
- character(len=38) `vars_dat::title`

*Title description input/output files for V4 should have V4.0.*

- character(len=19), dimension(1, 1) [vars\\_dat::times](#)

*Start date in input emissions file.*

- character(len=19) [vars\\_dat::current\\_date](#)

*Current date in input emissions file.*

- character(len=19) [vars\\_dat::mecha](#)

*Chemical mechanism name.*

- character(len=19), dimension(:), allocatable [vars\\_dat::sdim](#)

*Vector of dimensions descriptions.*

- character(len=11), dimension(:), allocatable [vars\\_dat::ename](#)

*Emissions description long.*

- character(len=50), dimension(:), allocatable [vars\\_dat::cname](#)

*Emissions name variable short.*

- character(len=50), dimension(:), allocatable [vars\\_dat::cunits](#)

*Units in emissions vars.*

- logical, dimension(:), allocatable [vars\\_dat::tvar](#)

*true if input var is an emissions variable*

- logical [vars\\_dat::tpob](#)

*true if input emissions files contains density population*

# Index

calculos.F90, [25](#)  
    conversion, [25](#)  
cday  
    vars\_dat, [14](#)  
cenlat  
    vars\_dat, [14](#)  
cenlon  
    vars\_dat, [14](#)  
check  
    lee\_files.F90, [27](#)  
cname  
    vars\_dat, [15](#)  
conversion  
    calculos.F90, [25](#)  
crea\_attr  
    salidas.F90, [28](#)  
cunits  
    vars\_dat, [15](#)  
current\_date  
    vars\_dat, [15](#)  
  
dix  
    vars\_dat, [15](#)  
dix  
    vars\_dat, [15](#)  
dlat  
    vars\_dat, [15](#)  
dlon  
    vars\_dat, [16](#)  
dpob  
    vars\_dat, [16](#)  
dx  
    vars\_dat, [16](#)  
dx  
    vars\_dat, [16](#)  
dye  
    vars\_dat, [16](#)  
dye  
    vars\_dat, [16](#)  
  
ed  
    vars\_dat, [17](#)  
ei  
    vars\_dat, [17](#)  
eix  
    vars\_dat, [17](#)  
  
ejx  
    vars\_dat, [17](#)  
elat  
    vars\_dat, [17](#)  
elon  
    vars\_dat, [17](#)  
ename  
    vars\_dat, [18](#)  
epob  
    vars\_dat, [18](#)  
  
file\_out  
    salidas.F90, [29](#)  
file\_reading  
    lee\_files.F90, [27](#)  
  
gmt  
    vars\_dat, [18](#)  
grid\_id  
    vars\_dat, [18](#)  
  
interpola  
    Interpola.F90, [26](#)  
Interpola.F90, [26](#)  
    interpola, [26](#)  
isice  
    vars\_dat, [18](#)  
islake  
    vars\_dat, [18](#)  
isoilwater  
    vars\_dat, [19](#)  
isurban  
    vars\_dat, [19](#)  
iswater  
    vars\_dat, [19](#)  
itime  
    vars\_dat, [19](#)  
  
julday  
    vars\_dat, [19](#)  
julyr  
    vars\_dat, [19](#)  
  
lee\_files.F90, [27](#)  
    check, [27](#)  
    file\_reading, [27](#)

map\_proj\_char  
     vars\_dat, 20  
 mapproj  
     vars\_dat, 20  
 mecha  
     vars\_dat, 20  
 mminlu  
     vars\_dat, 20  
 moadcenlat  
     vars\_dat, 20  
  
 ndims  
     vars\_dat, 20  
 nh  
     vars\_dat, 21  
 num\_land\_cat  
     vars\_dat, 21  
  
 pollat  
     vars\_dat, 21  
 pollon  
     vars\_dat, 21  
  
 radm  
     vars\_dat, 21  
 README.md, 28  
  
 salidas.F90, 28  
     crea\_attr, 28  
     file\_out, 29  
 sdim  
     vars\_dat, 21  
 stdlon  
     vars\_dat, 22  
  
 times  
     vars\_dat, 22  
 title  
     vars\_dat, 22  
 tpob  
     vars\_dat, 22  
 trulat1  
     vars\_dat, 22  
 trulat2  
     vars\_dat, 22  
 tvar  
     vars\_dat, 23  
  
 unlimdimid  
     vars\_dat, 23  
  
 vars\_dat, 11  
     cday, 14  
     cenlat, 14  
     cenlon, 14  
     cname, 15  
     cunits, 15  
     current\_date, 15  
     dix, 15  
     dix, 15  
     dlat, 15  
     dlon, 16  
     dpob, 16  
     dx, 16  
     dxe, 16  
     dy, 16  
     dye, 16  
     ed, 17  
     ei, 17  
     eix, 17  
     ejx, 17  
     elat, 17  
     elon, 17  
     ename, 18  
     epob, 18  
     gmt, 18  
     grid\_id, 18  
     isice, 18  
     islake, 18  
     isoilwater, 19  
     isurban, 19  
     iswater, 19  
     itime, 19  
     julday, 19  
     julyr, 19  
     map\_proj\_char, 20  
     mapproj, 20  
     mecha, 20  
     mminlu, 20  
     moadcenlat, 20  
     ndims, 20  
     nh, 21  
     num\_land\_cat, 21  
     pollat, 21  
     pollon, 21  
     radm, 21  
     sdim, 21  
     stdlon, 22  
     times, 22  
     title, 22  
     tpob, 22  
     trulat1, 22  
     trulat2, 22  
     tvar, 23  
     unlimdimid, 23  
     xlat, 23  
     xlon, 23  
     zlev, 23  
 vars\_dat\_mod.F90, 29



---

xlat  
    vars\_dat, [23](#)

xlon  
    vars\_dat, [23](#)

zlev  
    vars\_dat, [23](#)