Documentation of the EMAC submodels `AIRTRAC 1.0` and `CONTRAIL 1.0`

Christine Frömming, Volker Grewe, Sabine Brinkop, and Patrick Jöckel
Institut für Physik der Atmosphäre
DLR-Oberpfaffenhofen

`christine.froemming@dlr.de; volker.grewe@dlr.de`

This manual is available as electronic supplement of our article
"Aircraft routing with minimal climate impact:
The REACT4C climate cost function modelling approach (V1.0)
in Geosci. Model Dev. Disc. (2013),
available at: `http://www.geosci-model-dev.net`.

July 2013

Contents

1 Introduction

This documentation provides an overview on the EMAC submodels AIRTRAC (Section 2) and CONTRAIL (Section 3). Within AIRTRAC the Lagrangian tracers are defined and the evolution of gaseous species caused by a local emission of NO_x and H_2O is simulated on trajectories. The emission location is defined via the submodel TREXP. Physics and chemistry is calculated based on the tagging contribution calculation, i.e. taking input from the submodels MECCA, SCAV, and CLOUD for reaction rates, wash out, and rain rates. The submodel CONTRAIL includes contrail formation and contrail processing in general and additionally from emissions at individual time-regions, which are defned via AIRTRAC. Each section is divided into a subsection describing the general structure (2.1 and 3.1), subroutines, which also includes a description of important variables (2.2 and 3.2), and namelists (2.3 and 3.7).

2 The submodel AIRTRAC

2.1 General structure

The structure of the submodel AIRTRAC follows the MESSY2 coupling standards (Jöckel et al., 2010).

Submodel Interface Layer messy_airtrac_e5.f90	*Base Model Core Layer* messy_airtrac.f90
airtrac_initialize Calls subroutines to read CTRL and CPL namelists	
airtrac_read_nml_cpl Reads coupling relevant parameters	airtrac_read_nml_ctrl Reads core relevant parameters
airtrac_new_tracer Defines Lagrangian tracers with nlgtrac=13 properties for species and contrails	
airtrac_init_memory Allocates memory (tendencies of Lagrangian tracers) and defines stream AIRTRAC with ngptrac=29 grid point fields in Lagrangian representation plus additional scavenging and contrail information	
airtrac_init_coupling Initialise coupling, makes information available from other submodels	
airtrac_global_end Transfers grid point fields to Lagrangian fields, defines start values for Lagrangian integration, checks whether emission takes place, calls core subroutine airtrac_integrate, and updates Lagrangian tendencies	
	airtrac_integrate Calculates processes on trajectories (see Sec. 2.2)
airtrac_free_memory Release memory	

2.2 The subroutine airtrac_integrate

ELEMENTAL SUBROUTINE airtrac_integrate (vars)			
name	type	intent	description
mandatory arguments:			
General information			
dt	REAL(DP)	IN	time step in seconds
Background tracer information from MECCA *(converted from grid space to lagrangian tracers)*			
*X*_lg,	REAL(DP)	IN	background mixing ratio of species *X* in [mol/mol]; *X*∈{NO, NO2, HNO3, OH, HO2}
*X*_lg_te	REAL(DP)	IN	chemical tendencies of background species from individual chemical reactions or groups of reactions in [mol/mol/s]; *X*∈{O3ProdN, O3LossN, O3LossY, NOxLoss, OHProd*i*, OHLoss*j*, HO2Prod1,HO2Loss1,HO2Loss2}; *i*=1,...,3; *j*=1,...,5
HNO3Loss_lg_te	REAL(DP)	IN	tendencies of background HNO3 due to scavenging in mol/mol/s
h2o_lg	REAL(DP)	IN	water vapour in [mol/mol]
h2o_loss_lg	REAL(DP)	IN	water vapour loss in [mol/mol/s]
AIRTRAC *specific information*			
*X*_air_lg	REAL(DP)	INOUT	mixing ratio of *X* due to an emission in a time-region in [mol/mol]; *X*∈{NOx, O3, HNO3, OH, HO2, H2O, CH4}
*X*_air_lg_te	REAL(DP)	OUT	tracer tendencies due to emissions in time-regions in [mol/mol/s]; *X*∈{NOx, O3, HNO3, OH, HO2, H2O, CH4}
*X*_air_lg_te	REAL(DP)	OUT	chemical tendencies of species due to individual reaction (families) in [mol/mol/s]; *X*∈{O3ProdN, O3LossN, O3LossY, NOxLoss, HNO3loss, HNO3scav}
hno3scav_air_lg_te	REAL(DP)	OUT	Scavenging of emitted N-compounds [mol/mol/s]

Note that the dimension of *X*_air_lg_te is nlgtrac=13.

2.3 Namelist

The AIRTRAC submodel has a control namelist, only:

```
!  -*- f90 -*-
&CTRL
N_emis_points=15
/
```

The variable

- n_emis_points is an integer, which defines the number of emission time-regions. The emission locations are defined via TREXP.

3 The submodel CONTRAIL

3.1 General structure

The structure of the submodel CONTRAIL follows the MESSY2 coupling standards (Jöckel et al., 2010).

Submodel Interface Layer `messy_contrail_si.f90`	Base Model Core Layrer `messy_contrail.f90`
`contrail_initialize` Reads `CTRL` and `CPL` namelist and assigns emission fields	
`contrail_read_nml_cpl` Reads coupling information for critical humidity for cloud condensation and emissions	`contrail_read_nml_ctrl` Reads core relevant parameters for fuel, aircraft and contrail parameterization
`contrail_init_memory` Defines stream `contrail` with GP and LG potential contrail coverage fields and actual contrail coverage, and contrail properties such as ice water path, for every time-region and for both GP and LG	
`contrail_init_coupling` Initialise coupling, makes cloud information available from other submodels	
`contrail_physc` Calls subroutine `contrail_pot_cov` for GP and `contrail_calc` for every time-region emission and emission inventory in the GP	
	`contrail_pot_cov` Calculates potential contrail coverage
	`contrail_calc` Calculates actual contrail coverage
`contrail_global_end` Calls subroutine `contrail_pot_cov` and `contrail_uv_grad` for LG. For every time-region emission and emission inventory (`contrail_calc_dev`) is called.	
	`contrail_pot_cov` see above
	`contrail_uv_grad` Calculates wind shear
	`contrail_calc_dev` Calculates development of acutual contrail coverage and contrail ice water (`contrail_calc` see above), water vapour uptake, sublimation and growth of contrail particles, contrail spreading (`contrail_spread`) and sedimentation (`contrail_sedi`)
	`contrail_calc` see above
	`contrail_spread` Calculates contrail spreading based on wind shear
	`contrail_sedi` Calculates sedimentation of ice particles
`contrail_free_memory` Release memory	

3.2 The subroutine `contrail_pot_cov`

ELEMENTAL SUBROUTINE contrail_pot_conv (T,p,q,zqte,b_ci,zrhc,dt,b_cc,potcov,zqsm1,zconpn			
name	type	intent	description
mandatory arguments:			
T	REAL(DP)	IN	temperature [K]
p	REAL(DP)	IN	pressure [Pa]
q	REAL(DP)	IN	water vapour [kg/kg]
zqte	REAL(DP)	IN	water vapour tendency [kg/kg/s]
b_ci	REAL(DP)	IN	cloud cover [fraction]
zrhc	REAL(DP)	IN	critical humidity for nat. clouds [kg/kg]
dt	REAL(DP)	IN	time step length [s]
b_cc	REAL(DP)	OUT	potential contrail cirrus coverage = ISS (contrail+cirrus) [fraction]
potcov	REAL(DP)	OUT	potential persistent contrail cirrus coverage [fraction]
zqsm1	REAL(DP)	OUT	Diagnostic output: saturation water vapour mixing ratio [kg/kg]
zconpn	REAL(DP)	OUT	Water vapour condensate within this timestep [kg/kg]

3.3 The subroutine `contrail_calc`

ELEMENTAL SUBROUTINE contrail_calc (q, b_ci, zconrat, potcov, zconpn, emis, scal,pconcov, pconiwc)			
name	type	intent	description
mandatory arguments:			
q	REAL(DP)	IN	water vapour [kg/kg]
b_ci	REAL(DP)	IN	cloud cover [fraction]
zconrat	REAL(DP)	IN	condensated water vapour within this timestep (=rate from cloud) [kg/kg]
potcov	REAL(DP)	IN	potential persistent contrail cirrus coverage [fraction]
zconpn	REAL(DP)	IN	water vapour condensate within this timestep [kg/kg]
emis	REAL(DP)	IN	emission of water vapour [kg/kg/s]
scal	REAL(DP)	IN	scaling factor, if necessary, for actual contrail coverage to adjust to observations [unitless]
pconcov	REAL(DP)	OUT	contrail-cirrus coverage [fraction]
pconiwc	REAL(DP)	OUT	contrail-cirrus ice water content [kg/kg]

3.4 The subroutine contrail_calc_dev

ELEMENTAL SUBROUTINE contrail_calc_dev (vars)			
name	type	intent	description
mandatory arguments:			
q	REAL(DP)	IN	water vapour [kg/kg]
b_ci	REAL(DP)	IN	cloud cover [fraction]
zconrat	REAL(DP)	IN	condensated water vapour within this timestep (=rate from cloud) [kg/kg]
potcov	REAL(DP)	IN	potential persistent contrail cirrus coverage [fraction]
zconpn	REAL(DP)	IN	water vapour condensate within this timestep [kg/kg]
dt	REAL(DP)	IN	time step [s]
emis	REAL(DP)	IN	emission of water vapour [kg/kg/s]
scal	REAL(DP)	IN	scaling factor for [unitless]
potcov_m1	REAL(DP)	IN	potential persistent contrail cirrus coverage at t-1 [fraction]
pconcov_m1	REAL(DP)	IN	contrail-cirrus coverage at t-1 [fraction]
pconiwc_m1	REAL(DP)	IN	contrail-cirrus ice water content at t-1 [kg/kg]
rho	REAL(DP)	IN	density [kg/m^3]
eta_dot	REAL(DP)	IN	vertical velocity in hybrid coordinates [1/s]
dudz	REAL(DP)	IN	zonal wind shear [1/s]
dvdz	REAL(DP)	IN	meridional wind shear [1/s]
pconcov_sum	REAL(DP)	OUT	contrail coverage of "sum" of aged plus act. contribution [fraction]
pconiwc_sum	REAL(DP)	OUT	ice water content of contrails of "sum" of aged plus actual contribution [kg/kg]
pconcov_now	REAL(DP)	OUT	contrail coverage from emissions at this time step [fraction]
pconiwc_now	REAL(DP)	OUT	ice water content of contrails from emissions at this time step [kg/kg]
concov_te_spread	REAL(DP)	OUT	tendency of pconcov_sum due to spreading [fraction/s]
coniwc_te_sedi	REAL(DP)	OUT	tendency of pconiwc_sum due to sedimentation [kg/kg/s]
coniwc_te_pot	REAL(DP)	OUT	tendency of pconiwc_sum due to growth [kg/kg/s]

3.5 The subroutine contrail_spread

ELEMENTAL SUBROUTINE contrail_spread (concov_te_spread,dudz,dvdz,zconcov_m1,gboxarea)			
name	type	intent	description
mandatory arguments:			
concov_te_spread	REAL(DP)	OUT	change in contrail coverage due to spreading [fraction/s]
dudz	REAL(DP)	IN	zonal wind shear [1/s]
dvdz	REAL(DP)	IN	meridional wind shear [1/s]
zconcov_m1	REAL(DP)	IN	contrail-cirrus coverage at t-1 [fraction]
gboxarea	REAL(DP)	IN	grid box area [m^2]

3.6 The subroutine contrail_sedi

ELEMENTAL SUBROUTINE contrail_sedi (lwi_lg, rho_lg, w_wind, lwi_lg_tte)			
name	type	intent	description
mandatory arguments:			
lwi_lg	REAL(DP)	IN	ice water content [kg/kg]
rho_lg	REAL(DP)	IN	air density [kg/m^3]
w_wind	REAL(DP)	IN	vertical velocity in hybrid coordinates [1/s]
lwi_lg_tte	REAL(DP)	OUT	tendency of lwi due to sedimentation [kg/kg/s]

3.7 Namelist

The CONTRAIL submodel has a control and a coupling namelist:

```
  !  -*- f90 -*-
&CTRL
! Water vapour emission index
EI_H2O = 1.25 !  [kg(H2O)/kg(fuel)]
!  Combustion heat of fuel
Q_fuel = 43.2e6 !  [J/kg] (Schumann et al., 2000)
```

```
!  overall propulsion efficiency of aircraft
eta_ac = 0.31 !  [fraction] (Schumann et al., 2000)
!  see Burkhardt et al., 2008:
a_sac = 0.9,
r_sac = 1.1,
!
/

&CPL
! -----------------------------------------------------------------
!  GRID-POINT
! -----------------------------------------------------------------
L_GP = .TRUE.
r_scal_gp = 1.0, !  scaling factor for contrail coverage
!
!  critical humidity for cloud condensation
C_GP_CLOUD_CRIT = 'cloud', 'rhc',
!  cloud condensate
C_GP_CLOUD_COND = 'cloud', 'condensation',
!
!C_GP_EMIS(1) = 'import_rgt','airc_quantify',
!C_GP_EMIS(2) = 'import_rgt','airc_react4c',
!  ...
!  ----------------------------------------------------------------
!  LAGRANGIAN
! -----------------------------------------------------------------
L_LG = .TRUE.
r_scal_lg = 1.0, !  scaling factor for contrail coverage
L_LG_DIAG_TEND = .TRUE.
!
!  critical humidity for cloud condensation
C_LG_CLOUD_CRIT = 'lggp_lg', 'rhc',
!  cloud condensate
C_LG_CLOUD_COND = 'lggp_lg', 'condensation',
!
!C_LG_EMIS(1) = 'airtrac','emis_001',
!C_LG_EMIS(2) = 'airtrac','emis_002',
!  ...
!
/
```

The control (CTRL) namelist has the following variables:

- EI_H2O = 1.25 is the emission index [kg(H_2O)/kg/fuel].
- Q_fuel = 43.2e6 is the combustion heat of the fuel [J/kg].
- eta_ac = 0.31 is the overall propusion efficiency of the aircraft, i.e. the fraction of the chemical energy, which is used for the propulsion [fraction].
- a_sac = 0.9 and r_sac = 1.1 are parameters in the contrail parameterisation.

The coupling (CPL) namelist has two sets of variables for the grid point (GP) and lagrangian (LG) space. The potential contrail coverage and actual contrail coverage can be calculated for the GP an LG case independently and hence there are two sets of identical parameters.

- L_GP = .TRUE. is a logical switch, which controls whether contrails are calculated in the grid space.
- r_scal_gp = 1.0 is a global scaling factor for the actual contrail coverage, calculated in the grid space.

- C_GP_CLOUD_CRIT = 'cloud', 'rhc' is a two dimensional string array, where the first string gives the channel name and the second the name of the channel object, which defines the critical humidity for cloud condensation for the calculation of the potential contrail coverage in the grid point space. See also zrhc in subroutine contrail_pot_conv.
- C_GP_CLOUD_COND = 'cloud', 'condensation' is a two dimensional string array, where the first string gives the channel name and the second the name of the channel object, which defines the cloud condensate. See also zconrat in subroutine contrail_calc.
- C_GP_EMIS(i) = 'import_rgt','airc_quantify' is a two dimensional string array for every i, where the first string gives the channel name and the second the name of the channel object, which defines the emission dataset.
- L_LG = .TRUE. is a logical switch, which controls whether contrails are calculated in the Lagrangian space.
- r_scal_lg = 1.0 is a global scaling factor for the actual contrail coverage, calculated in the Lagrangian space.
- C_LG_CLOUD_CRIT = 'lggp_lg', 'rhc' is a two dimensional string array, where the first string gives the channel name and the second the name of the channel object, which defines the critical humidity for cloud condensation for the calculation of the potential contrail coverage in the Lagrangian space. See also zrhc in subroutine contrail_pot_conv.
- C_LG_CLOUD_COND = 'lggp_lg', 'condensation' is a two dimensional string array, where the first string gives the channel name and the second the name of the channel object, which defines the cloud condensate. See also zconrat in subroutine contrail_calc.
- C_LG_EMIS(i) = 'import_rgt','airc_quantify' is a two dimensional string array for every i, where the first string gives the channel name and the second the name of the channel object, which defines the emission dataset.
- L_LG_DIAG_TEND is a logical switch, which controls the output of additional diagnostics for Lagrangian calculation, only.

4 References

Jöckel, P., Kerkweg, A., Pozzer, A., Sander, R., Tost, H., Riede, H., Baumgaertner, A., Gromov, S., and Kern, B.: Development cycle 2 of the Modular Earth Submodel System (MESSy2), Geosci. Model Dev., 3, 717-752, doi:10.5194/gmd-3-717-2010, 2010.