The ATTILA based Lagrangian subsystem of the Modular Earth Submodel System

Sabine Brinkop[1,2], Patrick Jöckel[1]

[1]Deutsches Zentrum für Luft- und Raumfahrt, Institut für Physik der Atmosphäre, Oberpfaffenhofen, 82230 Wessling, Germany

[2]Meteorologisches Institut der Universität München, 80333 München, Germany

sabine.brinkop@dlr.de,patrick.joeckel@dlr.de

This manual is part of the electronic supplement of our article "ATTILA 4.0: Lagrangian Advective and Convective Transport of Passive Tracers within the ECHAM5/MESSy (2.53.0) Chemistry Climate Model" in Geosci. Model Dev. Discuss. (2018), available at: http://www.geosci-model-dev-discuss.net

Date: November 20, 2018

Contents

1 Infrastructure submodel TRANSFORM (`messy_main_transform_bi.f90`)

This module contains subroutines for the transposition and transformation of variables (i.e., Fortran arrays) between various representations, such as spectral, grid-point and Lagrangian. Subroutines marked with an asterisk are used specifically within ATTILA.

1.1 The subroutine `trp_gp_fs_3d`

`SUBROUTINE trp_gp_fs_3d`		`(sign, pgp, pfs)`	
name	type	intent	description
mandatory arguments:			
sign	INTEGER,	IN	sign= 1: GP –> FS sign=-1: GP <– FS
pgp	REAL(DP), DIMENSION(:,:,:)	INOUT	grid-point space 3d (GP)
pfs	REAL(DP), DIMENSION(:,:,:)	INOUT	Fourier space (FS)

This subroutine transposes data from grid-point space in parallel domain decomposition into Fourier space (`sign=1`) in parallel decomposition and vice versa (`sign=-1`).

1.2 The subroutine `trp_fs_ls_3d`

`SUBROUTINE trp_fs_ls_3d`		`(sign, fs, ls)`	
name	type	intent	description
mandatory arguments:			
sign	INTEGER,	IN	sign= 1: FS –> LS sign=-1: FS <– LS
fs	REAL(DP), DIMENSION(:,:,:)	INOUT	Fourier space (FS)
ls	REAL(DP), DIMENSION(:,:,:)	INOUT	Legendre space (LS)

This subroutine transposes data from Fourier space in parallel decomposition into Legendre space (`sign=1`) in parallel decomposition and vice versa (`sign=-1`).

1.3 The subroutine `trp_ls_sp_3d`

`SUBROUTINE trp_ls_sp_3d`		`(sign, pls, psp)`	
name	type	intent	description
mandatory arguments:			
sign	INTEGER,	IN	sign= 1: LS –> SP sign=-1: LS <– SP
pls	REAL(DP), DIMENSION(:,:,:)	INOUT	Legrendre space (LS)
psp	REAL(DP), DIMENSION(:,:,:)	INOUT	Spectral space (SP)

This subroutine transposes data from Legendre space in parallel decomposition into spectral space (`sign=1`) in parallel decomposition and vice versa (`sign=-1`).

1.4 The subroutine `trp_fs_as`

`SUBROUTINE trp_fs_as`		`(sign, f, fa, fs)`	
name	type	intent	description
mandatory arguments:			
sign	INTEGER,	IN	sign= 1: Fourier –> antisymmetric+symmetric sign=-1: Fourier <– antisymmetric+symmetric
f	REAL(DP), DIMENSION(:,:,:)	INOUT	Fourier
fa	REAL(DP), DIMENSION(:,:,:,:)	INOUT	Fourier antisymmetric
fs	REAL(DP), DIMENSION(:,:,:,:)	INOUT	Fourier symmetric

This subroutine transposes data from Fourier space in parallel decomposition into the corresponding antisymmetric and symmetric components (`sign=1`) and vice versa (`sign=-1`). The subroutine is twofold overloaded for data (`f`) of rank 2 and 3 (as shown here), respectively.

1.5 The subroutine `trp_gpdc_gpgl*`

`SUBROUTINE trp_gpdc_gpgl`		`(sign, lf, gf [,method])`	
name	type	intent	description
mandatory arguments:			
sign	INTEGER,	IN	sign= 1 decomposed data –> global data sign=-1 decomposed data <– global data
lf	REAL(DP), DIMENSION(:,:,:,:)	INOUT	decomposed data
gf	REAL(DP), DIMENSION(:,:,:,:)	INOUT	global data
optional arguments only for sign=-1:			
method	INTEGER, PARAMETER	IN	M_SUM: point-wise sum over all tasks M_AVE: point-wise average over all tasks M_STD: point-wise standard deviation over all tasks M_LOC: select points directly from global data on corresponding task

This subroutine is for the global domain cloning in grid-point space. It transposes decomposed grid-point data from parallel domain decomposition into global data for each task (`sign=1`) and vice versa (`sign=-1`). For the reduction from the n (number of tasks) global arrays into a joined decomposed data array, four methods are selectable: M_SUM = 0, M_AVE = 1, M_STD = 2, M_LOC = 3. The subroutine is threefold overloaded for data arrays of rank 2, 3 and 4 (as shown above), respectively.

1.6 The subroutine `trf_lti`

`SUBROUTINE trf_lti`		`(fa, fs, ls)`	
name	type	intent	description
mandatory arguments:			
fa	REAL(DP), DIMENSION(:,:,:,:)	OUT	anti-symmetric
fs	REAL(DP), DIMENSION(:,:,:,:)	OUT	symmetric
ls	REAL(DP), DIMENSION(:,:,:)	IN	

This subroutine calculates the inverse Legendre transformation in parallel decomposition and returns the anti-symmetric and the symmetric Fourier coefficients. This subroutine is threefold overloaded for arrays (`ls`) of rank 1, 2, and 3 (as shown above), respectively.

1.7 The subroutine `trf_ltd`

`SUBROUTINE trf_lti`		`(ls, fa, fs)`	
name	type	intent	description
mandatory arguments:			
ls	REAL(DP), DIMENSION(:,:,:)	OUT	
fa	REAL(DP), DIMENSION(:,:,:,:)	IN	anti-symmetric
fs	REAL(DP), DIMENSION(:,:,:,:)	IN	symmetric

This subroutine calculates the (direct) Legendre transformation in parallel decomposition from the anti-symmetric and the symmetric Fourier coefficients. This subroutine is threefold overloaded for arrays (`ls`) of rank 1, 2, and 3 (as shown above), respectively.

1.8 The subroutine `trf_ffti`

`SUBROUTINE trf_ffti`		`(ffs)`	
name	type	intent	description
mandatory arguments:			
ffs	REAL(DP), DIMENSION(:,:,:)	INOUT	

This subroutine calculates the inverse fast Fourier transformation.

1.9 The subroutine `trf_fftd`

`SUBROUTINE trf_ffti`		`(ffs)`	
name	type	intent	description
mandatory arguments:			
ffs	REAL(DP), DIMENSION(:,:,:)	INOUT	

This subroutine calculates the direct fast Fourier transformation.

1.10 The subroutine `trf_ls_dz2uv`

`SUBROUTINE trf_ls_dz2uv`		`(d, z, u, v)`	
name	type	intent	description
mandatory arguments:			
d	REAL(DP), DIMENSION(:,:,:)	IN	divergence
z	REAL(DP), DIMENSION(:,:,:)	IN	vorticity
u	REAL(DP), DIMENSION(:,:,:)	OUT	u * cos(lat)
v	REAL(DP), DIMENSION(:,:,:)	OUT	v * cos(lat)

This subroutine calculates the wind vector components $u \cdot cos(lat)$ and $v \cdot cos(lat)$ by transformation from divergence and vorticity data in Legendre space. This subroutine is twofold overloaded for arrays of 2 and three (as shown above), respectively.

1.11 The subroutine `sp2gp`

`SUBROUTINE sp2gp`		`(sp, gp, lzm0)`	
name	type	intent	description
mandatory arguments:			
sp	REAL(DP), DIMENSION(:,:,:)	IN	spectral data
gp	REAL(DP), DIMENSION(:,:,:)	OUT	grid-point data
lzm0	LOGICAL	IN	wave number m=0?

This subroutine transforms data from (parallel decomposed) spectral space into (parallel decomposed) grid-point space. With `lzm0=.TRUE.` wave number $m = 0$ is set to zero before transformation.

1.12 The subroutine `gp2sp`

`SUBROUTINE gp2sp`		`(gp, sp, lzm0)`	
name	type	intent	description
mandatory arguments:			
gp	REAL(DP), DIMENSION(:,:,:)	IN	grid-point data
sp	REAL(DP), DIMENSION(:,:,:)	OUT	spectral data
lzm0	LOGICAL	IN	wave number m=0?

This subroutine transforms data from (parallel decomposed) grid-point space into (parallel decomposed) spectral space. With `lzm0=.TRUE.` wave number $m = 0$ is set to zero before transformation.

1.13 The subroutine `get_dc_index*`

SUBROUTINE get_dc_index		(N, IDX [,LDO])	
name	type	intent	description
mandatory arguments:			
N	INTEGER	IN	global index range (number)
IDX	REAL(DP), DIMENSION(:,:)	OUT	index ranges per task
optional arguments:			
LDO	LOGICAL, DIMENSION(:)	OUT	task is active?

This subroutine calculates the index range (`IDX(i,1:2)`) of each task $i \in [0, n-1]$ for a parallel decomposition along an arbitrary index space with length `N`. The optional output `LDO(i)` indicates, whether the resulting range of a specific task i is zero (.FALSE.) or not (.TRUE.).

1.14 The subroutine `scatter_glix_1d*`

SUBROUTINE scatter_glix_1d		(gl, lc [,xpe] [,XNG])	
name	type	intent	description
mandatory arguments:			
gl	REAL(DP), DIMENSION(:)	IN	global data
lc	REAL(DP), DIMENSION(:)	OUT	decomposed (local) data
optional arguments:			
xpe	INTEGER	IN	number of task with global data
XNG	INTEGER	IN	size of global data array

This subroutine scatters data of rank 1 as chunks onto the different tasks. The optional parameter `xpe` is for selecting a specific source task, which per default is the I/O task. The optional parameter `XNG` is to specify the length of the global data array `gl`. If this is a priori known, an internal broadcast of `SIZE(gl)` can be avoided.

1.15 The subroutine `scatter_glix_4d`

SUBROUTINE scatter_glix_4d		(gl, lc, index [,xpe] [,xishpg])	
name	type	intent	description
mandatory arguments:			
gl	REAL(DP), DIMENSION(:,:,:,:)	IN	global data
lc	REAL(DP), DIMENSION(:,:,:,:)	OUT	decomposed (local) data
index	INTEGER	IN	along which rank?
optional arguments:			
xpe	INTEGER	IN	number of task with global data
xishpg	INTEGER, DIMENSION(4)	IN	shape of global data array

This subroutine scatters data of rank 4 as chunks along rank `index` onto the different tasks. The optional parameter `xpe` is for selecting a specific source task, which per default is the I/O task. The optional parameter `xishpd` is to specify the shape of the global data array `gl`. If this is a priori known, an internal broadcast of `SHAPE(gl)` can be avoided.

1.16 The subroutine `gather_glix_1d*`

SUBROUTINE gather_glix		(gl, lc [,xpe] [,XNG])	
name	type	intent	description
mandatory arguments:			
gl	REAL(DP), DIMENSION(:)	OUT	global data
lc	REAL(DP), DIMENSION(:)	IN	decomposed (local) data
optional arguments:			
xpe	INTEGER	IN	number of destination task
XNG	INTEGER	IN	size of global data array

This subroutine gathers (or collects) decomposed data of rank 1 from all tasks into one global data array. The default destination task is the I/O task, but any other task can be selected by the optional parameter `xpe`. The optional parameter `XNG` is to specify the length of the global data array `gl`. If this is a priori known, an internal broadcast of `SIZE(gl)` can be avoided.

1.17 The subroutine `gather_glix_4d`

`SUBROUTINE gather_glix`		`(gl, lc [,xpe] [,XNG])`	
name	type	intent	description
mandatory arguments:			
gl	REAL(DP), DIMENSION(:,:,:,:)	OUT	global data
lc	REAL(DP), DIMENSION(:,:,:,:)	IN	decomposed (local) data
index	INTEGER	IN	along which rank?
optional arguments:			
xpe	INTEGER	IN	number of destination task
XNG	INTEGER	IN	size of global data array at rank `index`

This subroutine gathers (or collects) decomposed data of rank 4 from all tasks into one global data array. The default destination task is the I/O task, but any other task can be selected by the optional parameter `xpe`. The optional parameter `XNG` is to specify the size of the global data array `gl` at rank `index`. If this is a priori known, an internal summation of `SIZE(lc,index)` over all tasks can be avoided.

2 Sub-submodel ATTILA_TOOLS (`messy_attila_tools_e5`)

2.1 The subroutine `gp2lg_e5`

`SUBROUTINE gp2lg_e5`		`(gpl ,lgl ,gprl [,lmcons] [,klev] [,llev])`	
name	type	intent	description
mandatory arguments:			
gpl	REAL(DP), DIMENSION(:,:,:,:)	IN	parallel decomposed grid-point data
lgl	REAL(DP), DIMENSION(:,:)	OUT	parallel decomposed Lagrangian data
optional arguments:			
gprl	REAL(DP), DIMENSION(:,:,:,:)	INOUT	parallel decomposed grid-point field with rest of gpl
lmcons	LOGICAL	IN	switch for mass conservation (default: .FALSE.)
optional arguments for xy data only:			
klev	INTEGER	IN	which level (default surface)
llev	LOGICAL, DIMENSION(:)	OUT	true, if box is at this level

This subroutine performs the transformation of data from parallel decomposed grid-point space into parallel decomposed Lagrangian space. The optional argument `lmcons` has three different effects, depending on the presence of the grid-point variable `gprl`:

- `lmcons=.FALSE.` and `.NOT.PRESENT(gprl)`:
  The Lagrangian parcels receive the value of the grid-box corresponding to their locations, independent on other parcels located in the same grid-box. This means, all parcels in a specific grid-box get the same value.
- `lmcons=.TRUE.` and `.NOT.PRESENT(gprl)`:
  The value of the grid-box is distributed as equal share among all Lagrangian parcels located in this grid-box. Information of grid-boxes, where currently no Lagrangian parcel is located, is lost.
- `lmcons=.TRUE.` and `PRESENT(gprl)`:
  The value of the grid-box is distributed as equal share among all Lagrangian parcels located in this grid-box. Information of grid-boxes, where currently no Lagrangian parcel is located, is stored and accumulated in `gprl` until the next Lagrangian parcel comes across.

This subroutine is threefold overloaded for rank 2 (xy), 3 (xyz) and 4 (xyzn, as shown above), respectively. Since `gprl` is used to conserve the total mass, `gprl` should be in kg. However, the time-step length, the ratio of molar masses (air/tracer) etc. are scaling factors, at least as long as the time-step length does not vary during the integration. They are therefore omitted.

The subroutine for rank-2 data (xy) has two additional optional parameters: `klev` is used to select the Lagrangian parcels located at a specific grid-point level (default: surface level). The `llev` (.TRUE. or .FALSE.) returns whether a specific parcel is located in grid-point level `klev`.

2.2 The subroutine `lg2gp_e5`

`SUBROUTINE lg2gp_e5`		`(lgl ,gpl ,method [,lmcons] [,ltm1] [,fill_value] [,fill_field])`	
name	type	intent	description
mandatory arguments:			
lgl	REAL(DP), DIMENSION(:,:)	IN	parallel decomposed Lagrangian data
gpl	REAL(DP), DIMENSION(:,:,:,:)	OUT	parallel decomposed grid-point data
method	INTEGER	IN	LG2GP_AVE, LG2GP_SUM, LG2GP_STD, LG2GP_AVEGT0
optional arguments:			
lmcons	LOGICAL	IN	switch for mass conserving transformation (default .FALSE.)
ltm1	LOGICAL	IN	use positions of parcels at $t - \Delta t$
fill_value	REAL(DP)	IN	optional value to be filled in, where no parcel is located
fill_field	REAL(DP), DIMENSION(:,:,:)	IN	optional field to be filled in, where no parcel is located

This subroutine transforms data from Lagrangian space into grid-point space. A mass-conserving transformation can be requested with `lmcons = .TRUE.`.

Four different methods (`method`) are possible:

- LG2GP_AVE: average over all parcels (on all tasks) in the corresponding grid-box,
- LG2GP_SUM: sum over all parcels (on all tasks) in the corresponding grid-box,
- LG2GP_STD: standard deviation over all parcels (on all tasks) in the corresponding grid-box,
- LG2GP_AVEGT0: average over all parcels (on all tasks) in the corresponding grid-box, where the value is > 0.

Grid-boxes, where currently no parcel is located, are set to zero, or the `fill_value`, or the corresponding value from the `fill_field`. With the optional parameter `ltm1` set to .TRUE., the parcel positions from the previous time step are used for the transformation (default: .FALSE., use actual positions).

This subroutine is twofold overloaded for data of rank 3 (xyz) and 4 (xyzn) as shown above, respectively.

2.3 The subroutine `lggpte2lgte_e5`

`SUBROUTINE lggpte2lgte_e5`		`(gp ,gpte , lg, lgte)`	
name	type	intent	description
mandatory arguments:			
gp	REAL(DP), DIMENSION(:,:,:,:)	IN	grid-point data
gpte	REAL(DP), DIMENSION(:,:,:,:)	IN	grid-point tendency
lg	REAL(DP), DIMENSION(:,:)	IN	Lagrangian data corresponding to gp (above)
lgte	REAL(DP), DIMENSION(:,:)	OUT	resulting Lagrangian tendency

This subroutine transforms a tendency from a grid-point process into a corresponding tendency in Lagrangian space. The grid-point field needs to be previously transformed from the corresponding Lagrangian data. A typical call sequence is:

```
...
CALL lg2gp_e5(lg, gp, LG2GP_AVE, fill_value=0._dp)
! CALL ... process(gp -> gpte) ...
CALL lggpte2lgte_e5(gp, gpte, lg, lgte)
! lgte is the resulting LG process tendency
...
```

This subroutine is twofold overloaded for data of rank 3 (xyz) and 4 (xyzn) as shown above, respectively.

2.4 The subroutine `gpsfemis2lgemis_e5`

`SUBROUTINE gpsfemis2lgemis_e5`		`(gpl, lgl, method [,gprl] [,lmcons])`	
name	type	intent	description
mandatory arguments:			
gpl	REAL(DP), DIMENSION(:,:)	IN	global grid-point data
lgl	REAL(DP), DIMENSION(:)	OUT	Lagrangian data
method	INTEGER	IN	1, 2, 3 or 4
optional arguments:			
gprl	REAL, DIMENSION(:,:)	INOUT	grid-point array with rest of mass (if no parcel resides in grid box)
lmcons	LOGICAL	IN	switch for mass conserving transformation

This subroutine transforms grid-point surface emission fluxes (any intensive quantity per unit time, e.g. mol/mol/s, kg/kg/s, mol/m^2/s etc.) into corresponding Lagrangian fluxes (in the same units). Mass conserving transformation can be requested with the optional parameter `lmcons` set to .TRUE. (default: .FALSE.). Depending on the presence of `gprl`, the fluxes into grid boxes, where no parcels are located, are either lost, or stored in `gprl` and accumulated until the next parcel comes across.

Four different emission methods (`method`) are implemented: the emission flux is distributed among all parcels

1. in the surface layer,
2. located lowest in the boundary layer,
3. in the boundary layer with a negative vertical gradient,
4. in the boundary layer with equal share.

3 ATTILA namelists

The user interface of the submodel ATTILA is the namelist file *attila.nml*. It contains the control namelist `&CTRL` (see Fig. 1), the coupling namelist `&CPL` (see Fig. 2) and the namelist `&TRAJ` (see Fig. 3) for the so-called trajectory mode.

The user can specify in the control namelist `&CTRL`:

- `NCHUNK` (default: 48): the number of chunks into which loops over all parcels are decomposed. This can be used for hybrid parallelization (e.g., OpenMP), however, is currently not implemented.
- `CPGBAVE` (default: 2.2): the average number of parcels per grid-box. From this average number, the total number of parcels is determined, if `I_NCELL` < 0 (see below).
- `LLTINFO` (default: .FALSE.): output of additional information into log-file.
- `I_PBLH_METHOD` (default: 0): method to calculate the planetary boundary layer height (and with that the first layer in the free troposphere); options are 0 (calculation within ATTILA) or 1 (external calculation).
- `ADICO` specifies the diffusion coefficients for the Monte Carlo diffusion:
  - `ADICO(1)` is the horizontal diffusion coefficient for the free atmosphere in m^2 s^{-1},

  - `ADICO(2)` is the horizontal diffusion coefficient for the boundary layer in $m^2\ s^{-2}$, and
  - `ADICO(3)` is the vertical diffusion coefficient in s^{-1}.

  If `ADICO(:) == 0.0` any Monte Carlo diffusion is omitted (see Sect. 2.2.1 in main text).

- `LLTBLTURB` (default: .FALSE.) is the switch for the boundary layer turbulence: If `LLTBLTURB=.TRUE.`, Lagrangian parcels are randomly displaced within the boundary layer by Monte Carlo diffusion.
- `LLCONV` (default: .FALSE.) is the switch for the convective movement of parcels.
- `LLCAT` (default: .FALSE.) is the switch for the turbulent movement of parcels in clear air turbulence areas. This is, however, currently not implemented.
- `LVDIAG` (default: .FALSE.) is a switch for additional velocity diagnostics.
- `I_NCELL` (default: -1) sets the global number of parcels: If `I_NCELL` < 0, the number is calculated from the number of grid-boxes times the average number of particles (`CPGBAVE`, see above), i.e., depending on the grid-point resolution. For `I_NCELL` $>= 0$ the number is set directly.
- `LTRAJEC` (default: .FALSE.) is to select the so-called trajectory mode, which allows to release individual parcels at given times and locations. If this mode is selected (.TRUE.),
  - `LTRAJEC_SAME_DATE` (default: .FALSE.) can be used to overwrite the individual start dates and times (see `&TRAJ` namelist below) with the same date and time specified by
  - `LTRAJEC_DATE` (default: 1978, 1, 1, 1, 0) in the from of Y, M, D, HR, MI, where Y, M, D, HR, MI denote year, month, day, hour and minute (UTC), respectively.
- `I_VERT` (default: 1) selects the vertical coordinate system and the corresponding vertical velocity scheme (see Sect. 2.2.2 in main text):
  1. eta-coordinate system,
  2. theta-sigma hybrid coordinate system,
  3. sigma-coordinate system.
- `press_ref` in Pa is the reference pressure, for `I_VERT=2`, where the theta-sigma hybrid coordinate system changes to a pure theta-coordinate in the stratosphere. If `press_ref=-1`, the reference pressure level is the tropopause level.

In the `&CPL` namelist of ATTILA the user can specify:

- `L_INI_PARALLEL` (default: .TRUE.) to perform the initialization of parcel positions in parallel mode.
- `I_RANDOM_METHOD` (default: 0) to select the pseudo-random number generation method (see Sect. 2.2.1 in main text):
  - 0: Fortran intrinsic,
  - 1: Mersenne Twister,
  - 2: Luxury.
- `I_RANDOM_PARALLEL` (default: 3) to select the parallel generation of pseudo-random numbers:
  - 3: parallel synchronized by jumping ahead,
  - 4: independent by jumping ahead.
- `L_RANDOM_TEST` (default: .FALSE.) to create an additional channel object to output random numbers for testing.
- the channel and channel object names used to drive ATTILA:
  - `C_PBLH_INDEX` to select the externally calculated level index of the planetary boundary layer height (if `I_PBLH_METHOD = 1`, see above in &CTRL namelist).

```
&CTRL
! ########## BASIC SETTINGS ########################################
! NUMBER OF CHUNKS INTO WHICH THE PARALLELIZED LOOPS ARE SPLITTED, DEFAULT=48
NCHUNK=1,
! AVERAGE NUMBER OF PARCELS PER GRID-BOX (DEFAULT 2.2)
CPGBAVE=3.0,
! PRINT OUT SOME MORE INFORMATION, DEFAULT=F
LLTINFO= F
! HOW TO CALCULATE FIRST LAYER IN FREE TROPOSPHERE: 0: ATTILA, 1: EXTERNAL
I_PBLH_METHOD      = 1
! ###################################################################
! ########## PROCESS SETTINGS ########################################
! DIFF COEFFICENTS FOR MONTE CARLO DIFF.
!    ADICO(1) HORIZ. DIFF. COEFF. [m^2/s] IN FREE ATMOSPHERE
!    ADICO(2) HORIZ. DIFF. COEFF. [m^2/s] IN BOUNDARY LAYER
!    ADICO(3) VERTICAL DIFF. COEFF. [1/s]
!    FORMERLY (/  5300./4.,  5300.,  7.E-11 /)
!    SEE DISSERTATION CH. REITHMEIER PAGE 22!!
!    IF ADICO()== 0 --> NO MONTE CARLO DIFF.
ADICO= 0.0, 0.0, 7.E-11,
! BOUNDARY LAYER TURBULENCE
LLTBLTURB = T
! CONVECTIVE TREATMENT OF TRAJECTORIES
LLCONV    = F
! TURBULENT TREATMENT OF TRAJECTORIES IN CLEAR-AIR-TURBULENCE AREAS
LLCAT     = F
! ###################################################################
! ########## ADDITIONAL DIAGNOSTICS ##################################
! ADDITIONAL VELOCITY DIAGNOSTICS
LVDIAG    = F
! ###################################################################
! ########## SPECIAL MODI ############################################
! #### RESOLUTION INDEPENDENT NUMBER OF PARCELS ###
! <=0: DEPENDING ON GRIDPOINT RESOLUTION; >0: RESOLUTION INDEPENDENT
I_NCELL = -1
!I_NCELL = 82944
!I_NCELL = 1000000
! #### TRAJECTORY MODE ##########################
! SWITCH
LTRAJEC = F
! OVERWRITE INDIVIDUAL START DATES ?
LTRAJEC_SAME_DATE = F
! DATE (yyyy, m, d, h) TO INITIALIZE POSITIONS IN TRAJEC MOD
LTRAJEC_DATE = 1978, 1, 1, 1,
! #### SELECT VERTICAL COORDINATE AND CORRESPONDING VERTICAL VELOCITY
I_VERT =  2          ! 1: eta, 2: theta(hybrid with sigma), 3: sigma
press_ref = -1.      ! (Pa) necessary for I_VERT=2
                     ! if press_ref should be the tropopause pressure
                     ! then set press_ref < 0.0
! ###################################################################
/
```

Figure 1: Example &CTRL namelist of the submodel ATTILA.

– `C_CONV_UFLX`, `C_CONV_DFLX` and `C_CONV_TYPE` to select the updraft mass flux, the downdraft mass flux, and the type of convection, respectively (if `LLCONV = T`, see above in &CTRL namelist).

The `&TRAJ` namelist controls the release locations and dates/times of individual parcels in the so-called trajectory mode (if `LTRAJEC=.TRUE.` in the `&CTRL` namelist). These are specified as ATTILA Parcels (with keyword `AP`) with an arbitrary but unique number, with position (latitude, longitude and pressure altitude (in Pa)), release date (year, month, time) and release time (hour, minute (UTC)).

```
&CPL
!
L_INI_PARALLEL = T,                    !# initialisation in parallel mode
!
! RANDOM NUMBERS: 0: F90-INTRINSIC, 1: MERSENNE TWISTER, 2: LUXURY
I_RANDOM_METHOD    = 1
I_RANDOM_PARALLEL  = 3 ! (3 or 4; see messy_main_rnd_bi.f90, RND_MP_[PSJ,PIJ]
L_RANDOM_TEST = .FALSE. ! create ch.object with random numbers (for testing)
!
C_PBLH_INDEX   = 'tropop', 'pblh_i',   !# ONLY IF I_PBLH_METHOD = 1
C_CONV_UFLX    = 'convect','massfu',   !# CONV
C_CONV_DFLX    = 'convect','massfd',   !# CONV
C_CONV_TYPE    = 'convect','conv_type',!# CONV
/
```

Figure 2: Example &CPL namelist of the submodel ATTILA.

```
&TRAJ
! AP( ) = LAT, LON, PRES, YYYY, MM, DD, HH, MI
! LAT : LATITUDE (-90 ... 90 = S-POLE ... N-POLE ),
! LON : LONGITUDE (0 ... 360)
! PRES: PRESSURE HEIGHT [Pa]
! YYYY, MM, DD, HH, MI: START DATE = LTRAJEC_DATE IF OMITTED
AP(1)=   51.00 ,  293.00 ,  95000.00, 1978, 1, 1, 1, 0
AP(2)=   58.00 ,  283.00 ,  85000.00, 1978, 1, 1, 1, 0
AP(3)=   60.00 ,  285.00 ,  95000.00, 1978, 1, 1, 3, 0
AP(4)=   59.00 ,  283.00 ,  95000.00, 1978, 1, 1, 3, 0
AP(5)=   54.00 ,  295.00 ,  95000.00, 1978, 1, 1, 3, 0
AP(6)=   54.00 ,  291.00 ,  85000.00, 1978, 1, 1, 3, 0
AP(7)=   62.00 ,  288.00 ,  95000.00, 1978, 1, 1, 3, 0
AP(8)=   56.00 ,  289.00 ,  85000.00, 1978, 1, 1, 3, 0
AP(9)=   59.00 ,  282.00 ,  85000.00, 1978, 1, 1, 3, 0
AP(10)=   52.00 ,  287.00 ,  85000.00, 1978, 1, 1, 1, 0
AP(11)=   58.00 ,  280.00 ,  85000.00, 1978, 1, 1, 1, 0
AP(12)=   62.00 ,  299.00 ,  95000.00, 1978, 1, 1, 1, 0
AP(13)=   50.00 ,  299.00 ,  85000.00, 1978, 1, 1, 1, 0
AP(14)=   59.00 ,  288.00 ,  85000.00, 1978, 1, 1, 1, 0
AP(15)=   50.00 ,  284.00 ,  95000.00, 1978, 1, 1, 1, 0
AP(16)=   56.00 ,  288.00 ,  95000.00, 1978, 1, 1, 1, 0
AP(17)=   52.00 ,  294.00 ,  95000.00, 1978, 1, 1, 1, 0
AP(18)=   50.00 ,  294.00 ,  95000.00, 1978, 1, 1, 1, 0
AP(19)=   57.00 ,  300.00 ,  85000.00, 1978, 1, 1, 1, 0
AP(20)=   55.00 ,  293.00 ,  85000.00, 1978, 1, 1, 1, 0
AP(21)=   54.00 ,  296.00 ,  85000.00, 1978, 1, 1, 3, 0
AP(22)=   55.00 ,  298.00 ,  85000.00, 1978, 1, 1, 3, 0}
```

Figure 3: Example &TRAJ namelist of the submodel ATTILA.

4 LGGP namelist

The LGGP coupling namelist `&CPL` (example see Fig. 4) allows to define new objects for output, which are either transformed from Lagrangian space into grid-point space (LG2GP) or from grid-point space to Lagrangian space (GP2LG). This renders any hard-wired coding of variable transformations unnecessary, see also Sect. 2. For LG2GP one need to specify:

- the keyword `LG2GP` with an arbitrary but unique number in parentheses,
- the name of the Lagrangian *channel* containing the *object* to be transformed,
- the *object name* representing the Lagrangian variable to be transformed,
- the transformation `method`:
  - 1: sums up the properties of the parcels in each grid-box,
  - 2: averages the properties of the parcels in each grid-box,

  - 3: calculates the standard deviation of the properties of the parcels in each grid-box.

- a logical switch to determine, if the transformation should be mass conserving (.TRUE.) or not (.FALSE.),
- the switch `fill_flag` to select, how grid-boxes, which contain no parcels, should be filled:
  - 0: no filling (filling with zero)
  - 1: a single fill value `fill_value`
  - 2: filling with (the corresponding grid-box value of) a grid-point field.
- the value `fill_value` for `fill_flag=1`,
- the *channel* and the *channel object* name of the grid-pint field, if `fill_flag=2`.

For GP2LG one need to specify:

- the keyword `GP2LG` with an arbitrary but unique number in parentheses,
- the name of the grid-point *channel* containing the *object* to be transformed,
- the *object name* representing the grid-point variable to be transformed,
- a logical switch to determine, if the transformation should be mass conserving (.TRUE.) or not (.FALSE.),
- a logical switch to account for information in grid-boxes, where currently no Lagrangian parcel is located; if .TRUE., this information is stored and accumulated until the next Lagrangian parcel comes across.

```
&CPL
!#############################
!### LAGRANGE -> GRIDPOINT ###
!#############################
!# SYNTAX:
!#   'name', 'lg_channel', 'lg_object', method, mass-conservation ?,
!#   fill_flag, fill_value, 'gp_fill_channel', 'gp_fill_object'
!# NOTES:
!#   method:    1 SUM
!#              2 AVE        (default)
!#              3 STD
!#   fill_flag: 0 no filling (default)
!#              1 fill value
!#              2 fill with GP field
!#############################
!LG2GP(1) = 'IPLAT', 'attila', 'IPLAT', 2, F, 1, -1.E+34, 'tracer_gp', 'O3',
!LG2GP(2) = 'ptrac_v_mass_m04', 'sedi_lg', 'ptrac_v_mass_m04', 2, F, 1, -1.E+34, '', '',
!LG2GP(3) = 'agepbl', 'lgvflux', 'ppbl_inf_clock', 2, F, 1, -1.E+34, '', '',
!LG2GP(4) = 'agetpd', 'lgvflux', 'ptpd_inf_clock', 2, F, 1, -1.E+34, '', '',
!LG2GP(5) = 'age080', 'lgvflux', 'p080_inf_clock', 2, F, 1, -1.E+34, '', '',
LG2GP(6) = 'age100', 'lgvflux', 'p100_inf_clock', 2, F, 1, -1.E+34, '', '',
LG2GP(7) = 'age200', 'lgvflux', 'p200_inf_clock', 2, F, 1, -1.E+34, '', '',
!
!#############################
!### GRIDPOINT -> LAGRANGE ###
!#############################
!# SYNTAX:
!#   'name', 'gp_channel', 'gp_object', mass-conservation ?, account rest ?
!#############################
 GP2LG(1) = 'AIR',  'tracer_gp', 'AIR',  T, F,
 GP2LG(2) = 'tpot', 'ECHAM5',    'tpot', F, F,
!GP2LG(3) = 'qm1',  'ECHAM5',    'qm1',  T, T,
!GP2LG(4) = 'qte',  'ECHAM5',    'qte',  T, T,
!GP2LG(5) = 'tm1',  'ECHAM5',    'tm1',  F, F,
/
```

Figure 4: Example &CPL namelist of the submodel LGGP

```
&CPL
!####################################
!# LAGRANGIAN VERTICAL FLUX DIAGNOSTIC
!####################################
!# SYNTAX:
!#                        GRIDPOINT
!#                        horizontal
!#                        surface [Pa]
!#                       /------^------\
!#          name,       channel,  object,   minimum residence time [s]
!#
vdyn(1)  = 'p080_096', 'viso',   'pp080',   345600.0,
vdyn(2)  = 'p080_inf', 'viso',   'pp080',   3.1536E+10,
!
#
!# SYNTAX:
!#                                        LAGRANGIAN      LAGRANGIAN
!#                                       flux-quantity     tendency
!#                                                         (optional)
!#                             surface   /-----^------\   /-----^------\
!#         name,               name     channel, object, channel, object,
!#

!
vflx(1) = 'air_p080_096',  'p080_096', 'lggp_lg', 'AIR',
vflx(2) = 'air_p080_inf',  'p080_inf', 'lggp_lg', 'AIR',
/
```

Figure 5: Example &CPL namelist of the submodel LGVFLUX

5 LGVFLUX namelist

LGVFLUX is a diagnostic submodel, which calculates on-line vertical fluxes of Lagrangian parcels through horizontal surfaces. The selected horizontal surfaces and flux-quantities must be specified in the LGVFLUX `&CPL` namelist (example see Fig. 5). Horizontal surfaces can for instance be defined with the submodel VISO (Jöckel et al., 2010), e.g., isentropes, pressure levels, levels of constant potential vorticity, etc.. The LGVFLUX `&CPL` namelist contains:

- the keyword `VDYN` with an arbitrary but unique number in parentheses (example vdyn(1)), with
  - a unique *channel object* name (p080_096 in the example),
  - the *channel* name containing the horizontal surface in grid-point space (viso in the example),
  - the *channel object* name of the horizontal surface (2d) in grid-point space (pp080 in the example), and
  - the minimum residence time to account for the flux after the parcels transition through the selected surface (345600.0 s in the example).
- the keyword `VFLX` with an arbitrary but unique number in parentheses (example vflx(1)), with
  - a unique *channel object* name of the vertical flux quantity (air_p080_096 in the example),
  - the *channel object* name of the corresponding `VDYN` object (see above),
  - the *channel* name containing the object of which the flux should be calculated,
  - the *channel object* name of which the flux should be calculated, and
  - optionally the *channel* name and *channel object* name of the corresponding tendency, in case it should be taken into account for prognostic variables (e.g., tracers).

6 LGTMIX namelist

The submodel LGTMIX describes the inter-parcel mixing as a process, where each parcel in a grid-box communicates with the background concentration (the mean over all parcels in that grid-box). The mixing parameter for different regions (layers) of the atmosphere can be specified in the `&CPL` namelist (example see Fig. 6):

- With `l_force=.TRUE.`, the 3d mixing parameter will be defined as a *channel object*, e.g., for output.
- A mixing layer is specified with the keyword `MX` and an arbitrary but unique number in parentheses; it is defined by specifying
  - the lower bound of the vertical (level index) range as a *channel object* with *channel* name (e.g., tropop) and *channel object* name of the vertical level index (e.g., pbh_i),
  - the upper bound of the vertical (level index) range as a *channel object* with *channel* name (e.g., tropop) and *channel object* name of the vertical level index (e.g., tp_i),
  - the mixing parameter (see Sect. 2.4 in the main text), either
    - by a constant, or
    - by a *channel object* (with *channel* name and *channel object* name) and a corresponding value range to be mapped onto the interval [0,1].
- With the keyword `TX` (and an arbitrary, but unique number in parentheses), the mixing parameter can be scaled additionally for each tracer individually; this is defined by (see Jöckel et al., 2008, for details on tracers)
  - the *basename* of the tracer,
  - the *subname* of the tracer, and
  - a list of scaling factors (one for each layer defined by `MX`).

```
&CPL
l_force = F,  !# force channel object of mixing parameter
!#
!# DETAILED LAYERING FOR MIXING OF LAGRANGIAN TACERS:
!#
!# OPTIONS FOR LAYERS:
!#   - channel, object: channel object with level index
!#   - #<n>           : constant level index <n>
!#   - #GND           : ground
!#   - #TOA           : top of the atmosphere
!# OPTIONS FOR MIXING PARAMETERIZATIONS:
!#   - channel, object, mmin, mmax:  channel object with mixing parameter;
!#                                   linear rescaling of interval
!#                                   [mmin, mmax] to [0.0, 1.0]
!#    ------------------------------------------------------------------
!#      | FROM            | TO                | MIXING
!#      | LEVEL INDEX     | LEVEL INDEX       | PARAMETER
!#      |channel|  object | channel | object  | channel   | obj | min| max|
!#    ------------------------------------------------------------------
!# EXAMPLE 1 ######################################################################
!MX(1) = '#GND'  , ''      , 'tropop', 'pblh_i', '=2.0E-03', ''  ,    ,    ,
!MX(2) = 'tropop', 'pblh_i', 'tropop', 'tp_i'  , 'tropop'  , 'PV', 1.0, 0.0,
!MX(3) = 'tropop', 'tp_i'  , '#TOA'  , ''      , '=5.0E-04', ''  ,    ,    ,
!# EXAMPLE 2 ######################################################################
!MX(1) = '#GND'  , ''      , '#10'   , ''      , '=1.0E-03',     ,    ,    ,
!MX(2) = '#10'   , ''      , '#TOA'  , ''      , '=5.0E-04',     ,    ,    ,
!!# EXAMPLE 3 #####################################################################
!MX(1) = '#GND'  , ''      , '#TOA'  , ''      , '=1.0E-04',     ,    ,    ,
!# EXAMPLE 4 ######################################################################
!MX(1) = '#GND'  , ''      , 'tropop', 'pblh_i', '=3.0E-03', ''  ,    ,    ,
!MX(2) = 'tropop', 'pblh_i', 'tropop', 'tp_i'  , '=1.0E-03', ''  ,    ,    ,
!MX(3) = 'tropop', 'tp_i'  , '#TOA'  , ''      , '=5.0E-04', ''  ,    ,    ,
!# EXAMPLE 5 ######################################################################
MX(1) = '#GND',   ''      , 'tropop', 'tp_i'  , '=1.0E-03', ''  ,    ,    ,
MX(2) = 'tropop', 'tp_i'  , '#TOA'  , ''      , '=5.0E-04', ''  ,    ,    ,
!################################################################################
!### SCALE MIXING STRENGTH FOR INDIVIDUAL TRACERS; DEFAULT: 1.0
!# SYNTAX: basename, subname, scaling list for different layers [0,1]
!#
TX(1) = 'SF6',  'nm' ,      1.0, 0.0,
TX(2) = 'AOA',  'nm' ,      1.0, 0.0,
TX(3) = 'SF6',  'AOA_nm' ,  1.0, 0.0,
TX(4) = 'SF6',  'AOAc_nm' , 1.0, 0.0,
TX(5) = 'SF6',  'CCMI_nm' , 1.0, 0.0,
TX(6) = 'CO2',  'nm' ,      1.0, 0.0,
TX(7) = 'SF6',  'CCMI_sm' , 1.0, 2.0,
TX(8) = 'CO2',  'sm' ,      1.0, 2.0,
/
```

Figure 6: Example &CPL namelist of the submodel LGTMIX.

7 DRADON namelist extension

The DRADON submodel and the corresponding namelists are described by Jöckel et al. (2010). For the application with ATTILA, the submodel has been extended. New features can be set in the `&CPL` namelist of DRADON (example see Fig. 7):

- `L_LG` is used to switch the Lagrangian calculations on (.TRUE.) or off (.FALSE.).
- `I_LG_emis_method` selects the treatment of the ^{222}Rn emissions (see Sect. 2.4):
  - 1: emission into parcels in the lowest model layer,
  - 2: emission into parcels located lowest in the boundary layer,
  - 3: emission into parcels located in the boundary layer with negative vertical gradient,
  - 4: emission into parcels located in the boundary layer.

- `L_LG_emis_mcons` is the logical switch for mass conserving transformation of the grid-point emission.
- `I_LG_emis_rest` determines how the emissions from grid-boxes, where currently no parcel is located, are treated:
  - -1 (automatic, default): The accounting of the rest is detemined automatically from `I_LG_emis_method`:
    - : 1: on,
    - : 2,3,4: off.
  - 0 (off): The flux is unaccounted.
  - 1 (on): The flux is stored and accumulated until the next parcel comes across.
- `L_LG_emis_rest_int` determines, if the accumulated rest flux from boxes (where no parcel is located) decays (.TRUE.) or not (.FALSE.).
- `L_LG_chain` selects, if only the ^{222}Rn decay is simulated (.FALSE.), or the decay chain up to ^{210}Pb (.TRUE.).
- If the decay chain is simulated (`L_LG_chain = .TRUE.`), `C_LG_210Pb_aermod` and `I_LG_210Pb_mode` set the aerosol model and mode number for the treatment of the ^{210}Pb tracer, respectively. From these, the corresponding aerosol mean radius and aerosol radius standard deviation are used as tracer properties of ^{210}Pb (for sedimentation and scavenging).

```
&CPL
!
L_GP = T                        ! GRIDPOINT CALCULATION ON/OFF
I_GP_emis_method   = 2          ! emission method for GP (1,2)
L_GP_chain         = T          ! 222Rn -> ... -> 210Pb
C_GP_210Pb_aermod  = 'ptrac',   ! ONLY IN EFFECT IF L_GP_chain = T
!C_GP_210Pb_aermod = 'gmxe',    ! ONLY IN EFFECT IF L_GP_chain = T
I_GP_210Pb_mode    = 4,         ! ONLY IN EFFECT IF L_GP_chain = T

L_LG = T                        ! LAGRANGE
I_LG_emis_method   = 2          ! emission method for LG (1,2,3,4)
L_LG_emis_mcons    = T          ! LG emission mass conserving ?
I_LG_emis_rest     = -1         ! handle LG emission 'rest' (-1 auto, 0 off, 1 on)
L_LG_emis_rest_int = F          ! accumulated LG rest flux decays ?
L_LG_chain         = F          ! 222Rn -> ... -> 210Pb
C_LG_210Pb_aermod  = 'ptrac',   ! ONLY IN EFFECT IF L_LG_chain = T
I_LG_210Pb_mode    = 4,         ! ONLY IN EFFECT IF L_LG_chain = T
!
/
```

Figure 7: Example &CPL namelist of the submodel DRADON with the Lagrangian extension.

References

Jöckel, P., Kerkweg, A., Buchholz-Dietsch, J., Tost, H., Sander, R., Pozzer, A., Technical Note: Coupling of chemical processes with the Modular Earth Submodel System (MESSy) submodel TRACER, Atmos. Chem. Phys., 8, No 6, 1677–1687,http://www.atmos-chem-phys.net/8/1677/2008/,10.5194/acp-8-1677-2008, 2008.

Jöckel, P., A. Kerkweg, 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.