## Overview

The new trajectory code tracks three dimensional variables, with an XZY ordering, from a specified starting point along a lagrangian trajectory. Along any trajectory there may be up to 100 defined variables. Each WRF domain may have up to 1000 individual trajectories.

Each trajectory must have a set of initial spatial coordinates and a start and stop time. Default values, which may apply to all trajectories, may be defined for the start and stop times and the variables associated with a given trajectory. The default values, if defined, are automatically assigned to any individual trajectory that does not explicitly define the start, stop times and/or the associated variables.

There is a second "time" mode for trajectories. If the "stationary" mode is selected then all outputs are fixed at the initial spatial coordinates. In other words the trajectory output represents the time series at a given spatial point in the domain. See is_stationary in the traj_type definition below.

## Trajectory Details

To run a simulation with trajectories enabled you must have the namelist variable 'traj_opt', in the physics namelist group, set to 1; traj_opt = 1. Furthermore, on a domain by domain basis you must set the physics namelist group variable dm_has_traj to .true. for each domain to enable trajectories. The default settings disable trajectories in all simulation domain(s). These namelist trajectory variable settings, traj_opt and dm_has_traj, must remain constant throughout a simulation. This is especially important for restarts.

However, note that you can conduct a "cold start" simulation part way through a simulation by initializing from the WRF output of the desired starting time. For example, if the simulation begins at 00 UTC 1 June and you want the trajectories to begin at 12 UTC 2 June, then the output from the simulation at 12 UTC 2 June can be used as wrfinput data for a new simulation. Restarts will continue the trajectory during the simulation. If the trajectory start time begins after the start of the WRF simulation, then missing values( -9999. ) will be assigned to the trajectory variables (except for i, j, k, latitude, longitude, and altitude).

The trajectories exist only in the domain in which they are defined. If a trajectory exits the domain, missing values are assigned to the trajectory variables for all following times in the simulation.

To initialize trajectories, a trajectory namelist input file, named wrfinput_traj_d<domain>, must be specified for each domain with trajectories. Just as the trajectory namelist variables must not change when restarting a simulation, the trajectory input files wrfinput_traj_d<domain> must be invariant across a restart.

There is an example wrfinput_traj_d01 file at the end of this documentation that can be used as a template. There are two namelist groups, traj_default and traj_spec in each wrfinput_traj_d<domain> file. Please note the following:

(1) There are a maximum of 1000 trajectories per domain

(2) The traj_default namelist group in the wrfinput_traj_d<domain> files is optional but must be included in wrfinput_traj_d<domain> even if you do not want to specify any defaults; in this case you have a "null" default group as in :

```
&traj_default
/
```

(3) the traj_default namelist group is composed of the following variables :

traj_def%start_time, a character, scalar of length 19
traj_def%stop_time. a character, scalar of length 19
traj_def%dyn_name, a character, array of length 100; each is char(len=32)
traj_def%hyd_name, a character, array of length 100; each is char(len=32)
traj_def%chm_name, a character, array of length 100; each is char(len=32)
traj_def%trc_name, a character, array of length 100; each is char(len=32)
traj_def%msc_name, a character, array of length 100; each is char(len=32)
traj_def%dchm_name, a character, array of length 100; each is char(len=32)

(4) the traj_spec namelist group is required

(5) the traj_spec namelist group is composed of traj_type user defined variables. There are a maximum of 100 variables, as defined in the chm_spc, dchm_spc, hyd_spc, trc_spc, dyn_var, and msc_var variables, per traj_type namelist variable. Each traj_type is composed of :

start_time, a character, scalar of length 19
stop_time, a character, scalar of length 19
lon, a real, scalar
lat, a real, scalar
lev, a real, scalar
z_is_agl, a logical scalar
is_stationary, a logical scalar
dyn_var, a character, array of length 100; each is char(len=32)
msc_var, a character, array of length 100; each is char(len=32)
hyd_spc, a character, array of length 100; each is char(len=32)
chm_spc, a character, array of length 100; each is char(len=32)
dchm_spc, a character, array of length 100; each is char(len=32)
trc_spc, a character, array of length 100; each is char(len=32)

REMINDER: if defaults are specified and any of the traj_type entries have unspecified start_time, stop_time, dyn_var, hyd_spc, chm_spc, and/or trc_spc variables the traj_type variable entries will default to the traj_default entries.

**NOTE**: the lon, lat, and lev variables have NO defaults and must be specified for each trajectory.

The lon, lat variables are WRF conforming and are in degrees.

-90 <= lat <= 90
-180 <= lon <= 180

The lev variable is, by default, the trajectory starting height in meters above the ground level (z_is_agl = .true.). The lev variable defines the height, in meters, above sea level if the z_is_agl variable is set to .false. . Overriding the default value for the traj_type variable z_is_agl must be done on a trajectory by trajectory basis. The z_is_agl variable applies to all output variables defined in the trajectory.

**NOTE**: the is_stationary variable in the traj_type variable defaults to .false. . Setting is_stationary to .true. produces a time series at the initial lon,lat, and lev coordinates for all variables defined in traj_type.

The *dyn_var* entries may be any combination of the following:

'p', 'T', 'z', 'u', 'v', 'w', 'rainprod', 'evapprod'

The *msc_var* entries are any 3-dimensional variable, with an XZY memory ordering, in the Registry. Examples are diagnostics like PM2.5, photolysis rates, and tendency chemical diagnostic terms (e.g. advh_tend_o3).

The *hyd_spc* entries must match the names as given in the moist array and used in the cloud physics package in registry.EM.COMMON. For example, these are:

'QVAPOR', 'QCLOUD', 'QRAIN', 'QICE', 'QSNOW', and 'QGRAUP'

The *chm_spc* entries must match the chemical species names as given in a chemical package statement in registry.chem corresponding to the simulation chemical option( chem_opt ). Thus note the use of 'ho', not 'oh', in the example wrfinput_traj_d01 file.

The *dchm_spc* entries are specified in the same manner as the chm_spc entries. However, the dchm_spc variables “track” the chemical change of each species along the trajectory. The value of any dchm_spc variable, units of ppmv, is the difference between the mixing ratio before and after the subroutine that

calculates the chemistry in chem_driver. To get the chemical tendency, per time step, simply divide the result by the time step for the chemistry (chem_dt).

The *trc_spc* entries must match the tracer species names as given in a tracer package statement in registry.chem corresponding to the simulation tracer option (tracer_opt).

REMEMBER: each trajectory may have up to 100 individual variables.
This limit is for the total of all the chm_spc, dchm_spc, hyd_spc, trc_spc, dyn_var, and msc_var variables defined.

**Output**

The trajectory output occurs every time step. Results are written to wrfout_traj_d<domain> files for each domain in your simulation with trajectories specified. These are netcdf files. Even though trajectory variables are calculated every time step trajectory output is efficiently accomplished via the use of internal buffers that hold 1000 time points. Only when the buffer is full or a simulation ends are the buffered trajectory variables written to the wrfout_traj_d<domain> file(s).

Output with values = -9999. signify "missing" output that occurred either because the simulation time is not in the start_time, stop_time interval or the trajectory has exited the domain.

Two-dimensional arrays are created for each variable specified in the namelist. These arrays are dimensioned as the number of time steps by number of trajectories. For example, the hydrometeor QCLOUD will have the following output array:

float QCLOUD_traj(time, traj)

where time = number of time steps and traj = number of trajectories.

The variable 'Times" is a character array containing the time associated with each time step. The location of the trajectory always has a value. There are 5 2-d arrays for the grid location and latitude and longitude:

float traj_i(time, traj) ;
float traj_j(time, traj) ;
float traj_k(time, traj) ;
float traj_lat(time, traj) ;
float traj_long(time, traj) ;

Example NCL scripts can be provided for plotting the trajectories on a map and for gathering frequency distributions upon request.

**Source Code**

The following files have been changed. Note that dyn_em/solve_em.F contains code to advance all trajectory coordinates every time step. The file solve_em.F has been slightly modified to accommodate the “stationary” trajectory time mode.

Registry/ Registry.EM_COMMON
share/solve_interface.F
dyn_em/module_em.F

The following file has been added:
share/module_trajectory.F

**wrfinput_traj_d01 example**

```
&traj_default
traj_def%start_time = '2012-05-29_12:10:00'
traj_def%stop_time  = '2012-05-29_21:00:00'
traj_def%dyn_name(1:8) = 'p', 'T', 'z', 'u', 'v', 'w', 'rainprod', 'evapprod'
traj_def%hyd_name(1:6) = 'QVAPOR', 'QCLOUD', 'QRAIN', 'QICE', 'QSNOW','QGRAUP'
traj_def%chm_name(1:9) = 'co', 'o3', 'no', 'no2', 'hcho', 'h2o2', 'ch3ooh', 'ho', 'ho2'
traj_def%msc_name(1:5) = 't_2', 'pm2_5_dry', 'ph_o31d', 'ph_ch2om', 'ph_ch2or'
traj_def%dchm_name(1:7) = 'co', 'o3', 'no', 'no2', 'hcho', 'h2o2', 'ch3ooh'
 /

 &traj_spec
traj_type(1)%lon = -97.52
traj_type(1)%lat = 35.47
traj_type(1)%lev = 500.
traj_type(1)%z_is_agl = .false.

traj_type(2)%lon = -95.94
traj_type(2)%lat = 36.13
traj_type(2)%lev = 500.
traj_type(2)%msc_var(1:4)    = 'pm2_5_dry', 'ph_o31d', 'ph_ch2om', 'ph_ch2or'
traj_type(2)%dchm_spc(1:5) = 'co', 'o3', 'no', 'no2', 'hcho'

traj_type(3)%lon = -97.80
traj_type(3)%lat = 32.78
traj_type(3)%lev = 500.
traj_type(3)%is_stationary = .true.
/
```

In this example for domain 01 there are 3 trajectories starting from different longitudes, latitudes, and levels. Please note the following:

(a) all three trajectories use the traj_def start_time, stop_time settings
(b) the lev variable in traj_type(1) specifies an initial height of 500 meters above sea level; z_is_agl = .false.
(c) all variables in traj_type(3) will represent time series at the initial lon,lat, and lev values; is_stationary = .true.
(d) trajectories 1 and 3 will use the default variables as defined in the traj_def variable in the traj_default namelist group. Thus trajectories 1 and 3 will have a total of 35 output variables per trajectory.
(e) trajectory 2 will use the default definition for the dyn_var, hyd_spc, and chm_spc variables. However, the default msc_var and dchm_spc variables will not be used as they are explicitly defined in traj_type(2). Thus trajectory 2 will have a total of 30 output variables.
(f) there are no trc_spc variables associated with any trajectory

In the WRF model namelist.input file, the following should be included.

```
&domains
num_traj = 3
/

&physics
traj_opt                        = 1,
dm_has_traj(max_dom)    = .true.,
/
```

If traj_opt = 0 then no matter what the other trajectory namelist variables settings NO trajectory calculations take place. If traj_opt /= 0 then domain by domain the dm_has_traj entries take effect. Note that during initialization if for some reason there are no properly specifed trajectories for a give domain then dm_has_traj will be set to .false. for that domain.

The num_traj variable is only used for internal dimensioning purposes for domains with trajectories. The actual number of active trajectories in any given domain is controlled by the inputs in the wrfinput_traj_<domain> files. The number of trajectories in a given domain is limited by num_traj. It is best to use the num_traj default.