Documentation - Lagrangian part of the TMP-Code

Lukas Muessle

November 24, 2017

1 Overview

In this document you will find all information needed to use the Lagrangian part of the DNS-Code of the Turbulent Mixing Group(TMP) of the Max-Planck-Institute for Meteorology. The document is organized as follows, Section 2 uses the initialization file "dns.ini" to explain the different option to run the Lagrangian part. Section 3 explains the interpolation method. Section 4 presents the statistic tools. Section 5 represents procedures for the visualization and Section 6 presents some useful shell tools.
Current status: in progress
Last change: March 2015

In [Main]	Meaning	Option for value
CalculateParticle	Switch on Lagrange part	’yes’ or ’no’
Lagrange	What kind of Lagrange calculation	see following Table 2.2

Table 1: Options which can be set in [Main] of ”dns.ini”.

Lagrange type	Meaning
None	Lagrange module is switched off
Tracer	Only tracers following the flow
SimpleSettling	Simple settling
BilinearCloud	Simple combination of terms
BilinearCloudTwo	Second version of combination of terms
BilinearCloudThree	Optimized calculation
BilinearCloudFour	Optimized calculation with residence times

Table 2: Options for the Lagrangian calculation method.

2 Options for dns.ini

2.1 Primary

In the ”[Main]” section of the file ”dns.ini” the Lagrangian model can be activated using the flag ”yes”’ for the ”CalculateParticle” option, the default value reads ”no”. Furthermore, the calculation method has to be set as shown in Table 1.

2.2 Options for Lagrange initialize

The different calculation methods for the Lagrangian model are listed in Table 2. Each setting defines the values for certain setup variables presented in Table 3. The according setups of this variables can be found in ”/lagrange/lagrange_type_initialize.f90”. The first two types are self-explaining. The third option ”SimpleSettling” subtracts a constant value of the vertical particle velocity in ”/lagrange/rhs_particle_global.f90” with the flag ”LAG_TYPE_SIMPLE_SETT”. The following ”BilinearCloud” options describe the liquid calculation for the Lagrangian model. The calculation are split over ”/lagrange/rhs_particle_global.f90” and the according interpolation functions like ”/lagrange/rhs_particle_global_interpolation.f90”, ”/lagrange/rhs_particle_global_interpolation_halo_1.f90” and so on. We recommend using BilinearCloudThree for general liquid calculations as this option is the most accurate and fastest one. BilinearCloudFour uses the same method as BilinearCloudThree but with an additional scalar, which we use as a dummy to store the residence time information. To add a new calculation mode we can use the ”search” alias introduced in Chapter 6. For example, we can search for ”LAG_TYPE_BIL_CLOUD_4” to see where we need to add new IF clause when implementing a new calculation method. More information according the residence times can be found in Chapter 2.3.

The ”Mixture” option needs to read ”AirWaterBilinear” or ”AirWaterBilinearStrat” to

Setup variable	**Meaning**	**Used for**
inb_particle_evolution	Number of particle properties calculated time_runge_kutta.f90	index particle array
inb_particle_aux	Number of additional particle properties not looped in time_runge_kutta.f90	index particle array
inb_particle_txc	Number of particle array for temporary calculation	index particle array
inb_lag_aux_field	Number of Eulerian fields for Lagrangian (txc_fields are used)	index Eulerian field array
inb_particle	Total number of particle properties (inb_particle_evolution + inb_particle_aux)	index particle array

Table 3: Setup variables for the particle properties. They are initialized in "modules/lagrange_global.f90" and "modules/dns_global.f90"

solve the liquid calculation in the Eulerian code. The calculation for "AirWaterBilinear" and "AirWaterBilinearStrat" can be found in "/flow/flow_buoyancy.f90" and the setup in "/thermodynamics/thermo_initialize.f90". It is important to know that the third scalar of "AirWaterBilinearStrat" changes the index of scalars. This third scalar is used to create the stratification at the bottom of the domain. The buoyancy and liquid quantity is stored right after the general scalars. This leads to some IF statements at points inside the code where for example the liquid function is updated. To find these parts we can use the "search" alias introduced in Chapter 6 to search for "MIXT_TYPE_BILAIRWATERSTRAT ".

2.3 General options that need to be set

In this section we will describe the general setting that can be made for the Lagrangian scheme. All options are set in "[Lagrange]" and can be found in Table 4. The primary stats contain the total number of particles, the size of the particle array bumper and the initialization mode. The number of particles reads the total number of all simulated particles and the particle array bumper is used to safely communicate particles.
The initialization mode features three options. The first option is to fill the whole domain randomly with particles, the second option is to fill a certain segment in y-direction. For both options we use the particle_rnd_mode=1, the Y_Particle_Pos and the Y_Particle_Width, which is the total width. As a third option, the particles can be distributed according to the first scalar(total water q_t) profile, where the value of the scalar profile is proportional to the particle density. For this option we use particle_rnd_mode=2 and a general limit of Jmax_part and Jmin_part so that particles are not initialized in the stretched regions.
The amount of liquid that each particle features is determined in the function "/tools/in itialiaze/part/particle_random_position.f90". We use the function "FIELD_TO_PARTIC LE" to interpolate the total water q_t and the enthalpy h scalars onto particle buffers and then calculate the liquid content according to the calculation in "/flow/flow_buoyancy.f9 0" but only using the particle buffers with the interpolated scalar quantities instead of the actual scalars.

We implemented a routine to plot trajectories. This routine saves the position data of some chosen particles for every iteration and outputs them together with the binary files. The current version of the trajectory routine "/lagrange/dns_write_trajectories.f90" reads a file, which contains the largest particles (see Chapter 4) and tracks these particular particles. To run the trajectories in the parallel mode there must be a minimum amount of particles, so that every processor features some particles. It is worth mentioning that the routine features the old version of the routine in the commented section at the end of the file. The old version of this routine would track every particle.
The dispersion option is an additional tool, which is not yet fully developed. It can be found in "/tools/statistics/lagrange_dispersion/". It is used to calculate the dispersion of particle pairs in time.
The most important statistic tool is the particle PDF calculations. By default we set up an area over the whole horizontal domain and a span in the vertical direction using "Y_par ticle_PDF_pos" and "Y_particle_PDF_width". The horizontal domain can be constrained with "X_particle_PDF_pos", "Z_particle_PDF_pos", "X_particle_PDF_width" and "Z_parti cle_PDF_width", if desired. Using "Particle_PDF_Max" we can choose the maximum size of the bins according to the liquid field. The size of each bin is set by "Particle_PDF_Inter val".
As already mentioned above, we created a tool, which measures the residence time using BilinearCloudFour. This tool uses an additional part of the l_q and l_hq array. This workaround is used to make sure the residence time for each particle are properly communicated. We use the last part of the l_q array to safe the residence time for particles in the range of 2 $\times$ lambda at the cloud top and the l_hq for half of the cloud domain. We use the variable l_y_lambda and l_y_base for this threshold, the setup of these variables can be found in "/tools/dns/dns_main.f90". Every time-step in "/tools/dns/time_integration.f 90" we calculate the residence times for every particle by adding up the time-steps, therefore we check the location inside the domain and calculate the different residence time for the 2 $\times$ lambda and half of the cloud domain. The residence time of a cloud droplet, which is beneath half of the cloud domain, will be set to zero. Therefore, we account for cloud droplets, which have a great chance to leave the cloud a the cloud base. If it is necessary to continue an old residence measurement it is important to flag "ResidenceReset" with "no", because the default value resets the residence times. Unfortunately the restart of the residence time measurements can only be done for the particles staying in the l_y_lambda region as the l_hq array(buffer for half of the domain residence times) is not saved with the output routine.
The Parameter option is implemented for future studies. Right now it is only used for the settling parameter. The "lagrange_param is allocated in "/modules/lagrange_global.f90 ".

3 Interpolation algorithm

To interpolate the data between the field and the Lagrangian scheme, we developed three main functions. The first takes care of the right hand side calculation for all given scalars. The second interpolates only one Eulerian scalar field to the particles inside the domain.

In [Lagrange]	**Meaning**	**Option for value**
particle_number	Amount of particles	number
particle_bumper	bumper for size of isize_particle	number
particle_rnd_mode	Which mode of initialization	1 = random distribution in y-direciton, 2 = use scalar profile
Y_Particle_Pos, Y_Particle_Width	Boundary values for particle_rnd_mode=1	in percentage
Jmax_part, Jmin_part	Boundary values for particle_rnd_mode=2	in grid units
CalculateTrajectories	Should trajectories be calculated	′yes′ or ′no′
Num_trajectories	How many trajectories	number
Num_dispersion	number of pairs for the statistic tool of dispersion of particles	number
CalculateParticlePDF	Should the particle PDF be calculated	′yes′ or ′no′
Y_particle_PDF_pos	Area for PDF (Y)	percentage
Y_particle_PDF_width		″
X_particle_PDF_pos	Area for PDF (X) - default off	″
X_particle_PDF_width		″
Z_particle_PDF_pos	Area for PDF (Z) - default off	″
Z_particle_PDF_width		″
Particle_PDF_Max	Maximum bin for liquid	value
Particle_PDF_Interval	Size of each bin	value
ResidenceReset	Reset residence timer	′yes′ or ′no′ - default ′yes′
Parameter	Parameters, for example settling	amount of parameters

Table 4: General setup variables for the Lagrangian scheme

The third routine does exactly the opposite, the particle information is extrapolated to the surrounding grid point. In the following section we will discuss the RHS algorithm, which can be found in "/lagrange/rhs_particle_global_interpolation.f90", "/lagrange/rhs _particle_global_interpolation_halo_1.f90" and so on. The second and third routines can be found in "/lagrange/field_to_particle.f90" and "/lagrange/particle_to_field.f90". These two routines will be explained in a second section after the right hand side. To get a brief overview of all functions we present a roadmap in Figure 2 at the end of the document.

3.1 RHS

Before explaining the actual interpolation algorithm we will give an overview about the arrays you need hand over to the subroutines to use the RHS interpolation. "/lagrange/r hs_particle_global_interpolation.f90" needs:

Input

- q [Eulerian fields] $\rightarrow$ information for interpolation
- l_q [Particle property] $\rightarrow$ used to calculate local grid position
- y [Grid] $\rightarrow$ used to calculate local grid position
- wrk1d [1D-Buffer] $\rightarrow$ used for scaling (defined in rhs_particle_global.f90)
- txc [Eulerian buffer field] $\rightarrow$ dummy for liquid calculation
- grid_start [Number] $\rightarrow$ start value in particle array for particles belonging to grid zone (defined in rhs_particle_global.f90)
- grid_field_counter [Number] $\rightarrow$ end value in particle array for in grid zone (defined in rhs_particle_global.f90)

Output

- l_hq [Particle tendencies] $\rightarrow$ here we update the particle tendencies

The subroutines for the halo zones take the "halo fields" instead of "q" and a different start and end position inside the particle array.
The interpolation of the tendencies from the Eulerian scheme to the Lagrangian particle position is fulfilled with a trilinear interpolation which reads

$$\begin{aligned}
f_{\vec{X}_i}(\vec{v},\vec{q_j}) \approx & f(\vec{p}_{111})(x_2-X)(y_2-Y)(z_2-Z)+f(\vec{p}_{211})(X-x_1)(y_2-Y)(z_2-Z) \\
&+f(\vec{p}_{121})(x_2-X)(Y-y_2)(z_2-Z)+f(\vec{p}_{221})(X-x_1)(Y-y_1)(z_2-Z) \\
&+f(\vec{p}_{112})(x_2-X)(y_2-Y)(Z-z_1)+f(\vec{p}_{212})(X-x_1)(Y_2-y)(Z-z_1) \\
&+f(\vec{p}_{122})(x_2-X)(Y-y_1)(Z-z_1)+f(\vec{p}_{222})(X-x_1)(Y-y_1)(Z-z_1),
\end{aligned} \tag{1}$$

where X, Y and Z are the position of the particle, embedded by x_1, x_2, y_1, y_2, z_1 and z_2 the grid points and $\vec{p}_{abc} = (x_a, y_b, z_c)$ represents the quantity at the grid points. The

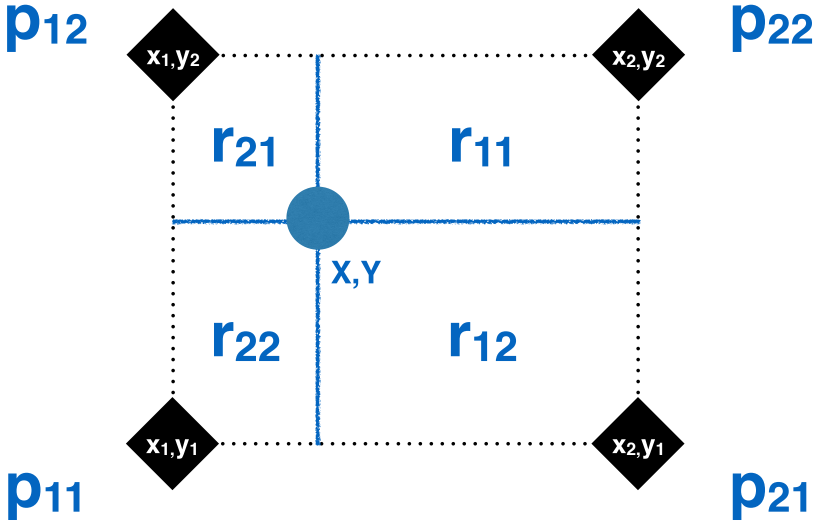

Figure 1: This sketch clarifies the interpolation procedure. The black rectangles represent the Eulerian grid points, whereas the blue point, represents a Lagrangian parcel or particle. The tendencies of the quantities are interpolated for the position of the blue point inside the grid using the calculated rectangles r_{ab}.

interpolation is sketched in a two-dimensional setup in Figure 1, where the black rectangles represent the Eulerian grid points and the blue point represents the Lagrangian particle. For simplicity, the sketch only represents a bilinear interpolation, whereas for a trilinear interpolation a third dimension would be present. The interpolated quantities p_{ab} are weighted by the corresponding rectangles r_{ab} on the opposite side of p_{ab} and then summed up to be used as the interpolated quantity for the particles.
However, the trilinear interpolation has its limit at transitions regions at the boundary of the cloud-top. Steep gradients lead to interpolation errors, so that expressions with expected steep gradients are calculated for each particle individually to minimize the interpolation error.

3.2 Field to particle and vice versa

As mentioned above, we have a second tool to interpolate only a single scalar field to a particle quantity using "/lagrange/field_to_particle.f90". This subroutine uses the subroutines for the development of the halo zones, the sorting algorithm and the right hand side interpolation for only one scalar. This subroutine needs:

Input

- scalar field [Eulerian field] $\rightarrow$ information for interpolation
- wrk1d [1D-Buffer] $\rightarrow$ used for scaling
- wrk2d [2D-Buffer] $\rightarrow$ used for halo_plane_shifting

- wrk3d [3D-Buffer] $\rightarrow$ used for halo_plane_shifting
- x [Grid] $\rightarrow$ information for particle_sort
- y [Grid] $\rightarrow$ used for scaling
- z [Grid] $\rightarrow$ information for particle_sort
- l_tags [Particle ID] $\rightarrow$ for particle_sort
- l_hq [Particle tendencies] $\rightarrow$ for particle_sort
- l_q [Particle properties] $\rightarrow$ to determine local grid position for interpolation and for particle_sort

Output

- particle_property [Buffer] $\rightarrow$ here we store the interpolated values

For the particle_property we mainly use l_txc.
The third subroutine extrapolates a particle quantity back to a field. The interpolation algorithm is customized in a way that the particle information is extrapolated to the eight surrounding grid points, using the trilinear interpolation explained above. The so constructed fields have the same structure as the scalar fields of the Eulerian scheme, so that all statistic tools of the Eulerian scheme can be used on those fields. The subroutine "/lag range/particle_to_field.f90" is split into the actual interpolation using "/lagrange/rhs_pa rticle_to_field.f90" and the sending of the information using "/lagrange/particle_to_field _send_recv_east.f90" and "/lagrange/particle_to_field_send_recv_north.f90". The general subroutine needs:

Input

- l_q [Particle properties] $\rightarrow$ particle position for interpolation
- particle_property [] $\rightarrow$ particular particle property we extrapolate to the field
- x [Grid] $\rightarrow$ not yet used
- y [Grid] $\rightarrow$ used for scaling
- z [Grid] $\rightarrow$ not yet used
- wrk1d [1D-Buffer] $\rightarrow$ used for scaling
- wrk2d [3D-Buffer] $\rightarrow$ used for particle_to_field_send_recv
- wrk3d [3D-Buffer] $\rightarrow$ used for particle_to_field_send_recv

Output

- field_out [Eulerian field] $\rightarrow$ here we store the extrapolated field

4 Statistics

4.1 Fields

All statistic files are processed with the program "tkstat". We implemented two additional "avg" files, which contains the extrapolated field data from the liquid calculation with and without the diffusion term. The Lagrangian files have the index $x = n + l$, where n is the amount of produced avg-files of the eulerian scheme and l=1 is the liquid field with the diffusion term and l=2 the calculation without the diffusion term.
Generally speaking, we have both liquid quantities on avg5s and avg6s, except when we use "AirWaterBilinearStrat". There we have 3 general scalars, which set the buoyancy and liquid field on avg4s and avg5s and the Lagrangian ones to avg6s and avg7s.

4.2 Tools

The folder "/tools/statistics/" contains some additional Lagrangian tools. Most of these tools are in a developing stage. We already mentioned "lagrange_dispersion" and "lagra nge_pdf".
The tool "lagrange_trajec" is used as a post processing tool to calculate the largest cloud droplets, the information are stored in a file called "largest_particle". To find the position of these largest cloud droplets at the start of the simulation, we use "lagrange_pos_traj ec". The data is saved in a file called "pos_largest_particle_start", which is used by "la grange_ini_trajec" to sort these largest particles to the beginning of the particle array as we want to start the simulation again with less particles to track the largest ones for every time-step. These three post processing tools are not fully developed and need to be treated with caution. We discovered that the exact reproduction of the largest trajectories is yet not possible due to some random calculations in the pressure solver in the parallel mode.

5 Visualization

For the visualization we used the DNS tool "ensight" and the visualization programs Paraview and Avizo. We use the tool ensight to create files which can be used with Paraview and Avizo. Inside ensight we use the "particle_to_field" routine to create a general density field of particles and two fields according to the two liquid fields we calculated for the Lagrangian scheme. In the newer versions of the DNS code ensight might be referred as visuals.
For Paraview we use the Format=1 and mainly a slide of the simulation cube. Additionally, we use Paraview to visualize the trajectory data. Therefore, we need to read the different trajectory data as **"VTK Particle Files"**. It is important that the option **"Has Scalar"** in the Paraview **Properties** window is unchecked. To track the particles inside Paraview we add the **Filter** "Temporal Particles To Pathlines" with Mask Points=1, this filter was working with Paraview version 4.3.1. In the Properties window of the path lines we can choose a solid color for the path lines to differentiate between different trajectories. To follow particular trajectories inside the domain it is helpful to go to **View** and

check **Animation View**. In the Animation window we can first choose **Real Time** in the **Mode** section. The **Duration** on the right side of the **Animation View** determines the **Real Time**. Then we want to track a particular trajectory, therefore we need to have a single trajectory data loaded. We click on the actual data vector in the **Pipeline Browser** (its the actual name of the data we loaded), then we need to add a new **Camera** in the **Animation View** panel and we choose **Follow Data**. After we zoomed to our Data, we can now start the animation with the **Play** button. To create movies we used **Save Animation** and combined the .png pictures with a small program (Adapter for OS X) to a movie. To visualize groups or single particles we used the program called ″extract_trajec.py″, which creates additional .vtk files.
For Avizo we basically used a cube where the Subdomain=1,1024,740,840,1,1024 and the Format=2. The Avizo raw files need to be in the .nc format. To create the .nc file we use a small python program, which can be found in the documentation folder and is called ″nc_gen.py″.

6 Useful tools

In this section, we present some useful tools to understand and work with the code. The following tools are aliases for the shell.

- search ′foobar′ =

  ```
  alias search 'grep -r --color --include="*.f90" \!^ .'
  ```

- h_search ′foobar′ =

  ```
  alias h_search 'grep -r --color --include="*.h" \!^ .'
  ```

- txt_search ′foobar′ =

  ```
  alias txt_search 'grep -r --color --include="*.txt" \!^ .'
  ```

The ′search′ command digs recursivly through all *.f90 files, and the ″h_search″ through all *.h and the ″txt_search″ through all *.txt files.

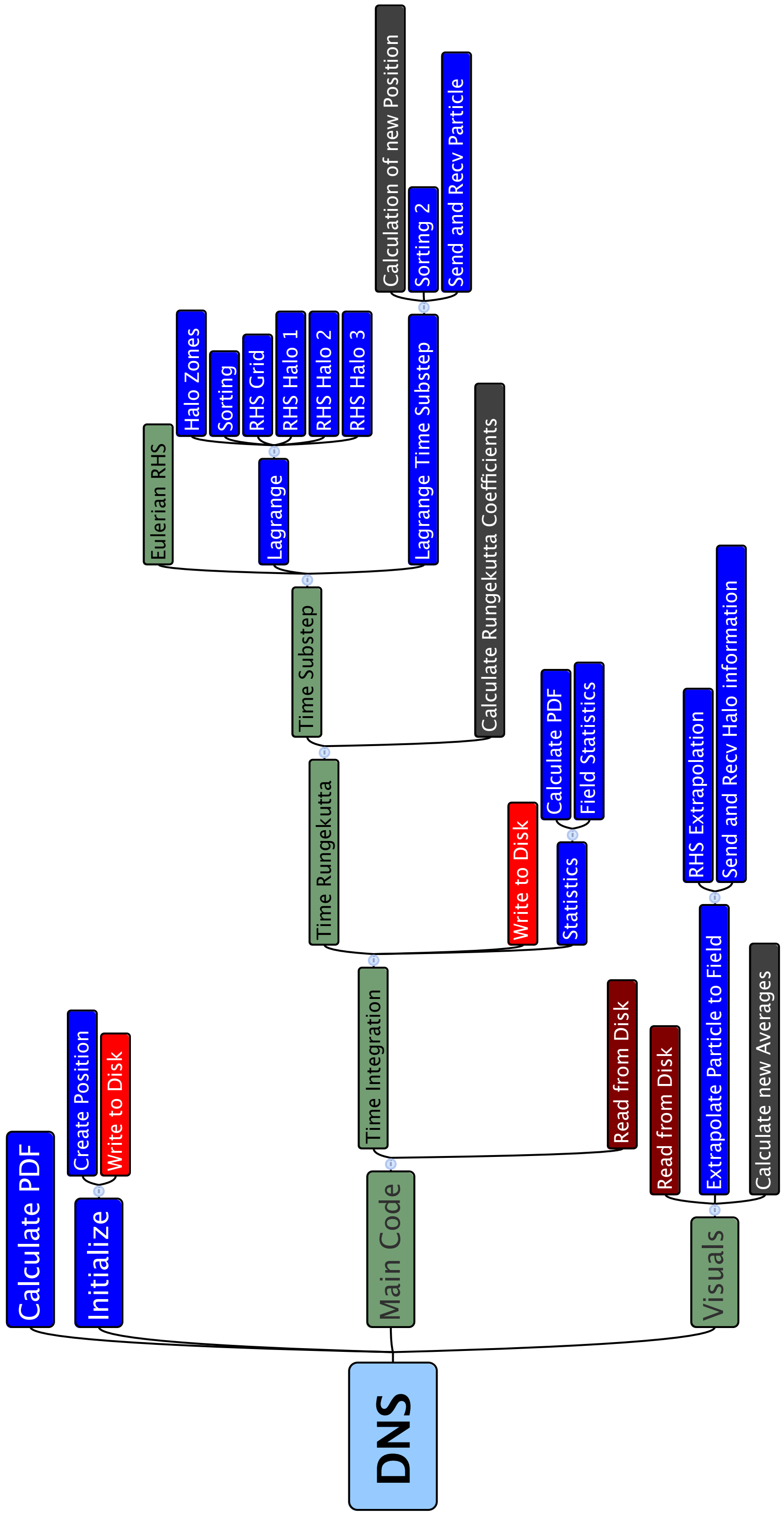

Figure 2: Simplified flow chart of functions used for the Lagrangian implementation. Green colors indicate Eulerian functions, blue represent the Lagrangian functions, red the Lagrangian I/O functions and the dark grey color indicates added calculation inside existing functions.