European Research Council

Established by the European Commission

Slide of the Seminar

Lagrangian tracking in the Pseudo-Spectral code

Dr. Irene Mazzitelli

ERC Advanced Grant (N. 339032) "NewTURB"
(P.I. Prof. Luca Biferale)

Università degli Studi di Roma Tor Vergata
C.F. n. 80213750583 – Partita IVA n. 02133971008 - Via della Ricerca Scientifica, I – 00133 ROMA

Lagrangian tracking in the Pseudo-Spectral code

- Implementation of Tracers
- Dumping flags and parameters
- Implementation of Heavy-Light particles
- Code structure

Tracers

Equation of motion:

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}(t) = \mathbf{u}(\mathbf{x}(t))$$

1st time step $\longrightarrow$ Forward Euler scheme

From 2nd Δt on $\longrightarrow$ 2nd order Adams-Bashforth scheme

ONLY Tracers - No heavy/light families

CmakeLists.mine

```
set(PARTICLE_MICHEL "YES")
set(LAGRANGIAN "YES")
set(LAGRANGIAN_PARTICLE "YES")
set(LAGRANGIAN_PARTICLE_BASETYPE_DOUBLE "YES")
set(LAGRANGIAN_PARTICLE_INIT "YES")
set(LAGRANGIAN_PARTICLE_INIT_RESTART "YES")
set(LAGRANGIAN_PARTICLE_INTERPOLATION "YES")
set(LAGRANGIAN_PARTICLE_INTERPOLATION_LINEAR "YES")
set(LAGRANGIAN_PARTICLE_IO_HDF5 "YES")
set(LAGRANGIAN_PARTICLE_WRITE_SORTED_HDF5 "YES")
set(LAGRANGIAN_DUMP "YES")
```

param.in

```
lagrangian_init_restart_flag 1
lagrangian_dump_rate 25
lagrangian_particle_nfamilies 1
lagrangian_particle_number_total 64
lagrangian_particle_number_fast 32
lagrangian_particle_ndump 50
lagrangian_particle_ndump_fast 5
```

Dumping Flags and Parameters

set(LAGRANGIAN_DUMP "YES")
FILE lagr_out.h5 is written at the end of the run for RESTART

set(LAGRANGIAN_PARTICLE_IO_HDF5 "YES") **SORTED or UNSORTED must be specified**
INPUT/OUTPUT lagrangian FILES are written in hdf5 format with frequency equal to the **lagrangian_dump_rate** parameter (fname: lagr_25.h5, lagr_50.h5 etc.). Those files may be used for RESTART.

SLOW PARTICLES are written with frequency **lagrangian_particle_ndump** (fname: slow.h5)
FAST PARTICLES are written in the subdirectory RUN/FAST with frequency **lagrangian_particle_ndump_fast** (fname: lagrangian_fast_1.h5, lagrangian_fast_5.h5, lagrangian_fast_10.h5, etc.)

set(LAGRANGIAN_PARTICLE_WRITE_SORTED_HDF5 "YES")
FAST PARTICLES are sorted, i.e. as ordinated as they were at the release time.

set(LAGRANGIAN_PARTICLE_WRITE_UNSORTED_HDF5 "YES")
FAST PARTICLES are unsorted (using SORTED doesn't require longer writing time).

In RESTART simulation the input file lagr_in.h5 is read only if flag LAGRANGIAN_PARTICLE_IO_HDF5 is "YES".

Output files of tracers

Files lagrangian_fast_#.h5 contain time, particle position, velocity and acceleration:

ifdef LAGRANGIAN_PARTICLE_OLD_POSITION
xo, yo, zo, else x,y,z, endif

uxo, uyo, uzo, axo, ayo, azo, vxo, vyo, vzo,
particle name

the suffix ‘o’ stands for old. If FLAG “LAGRANGIAN_PARTICLE_OLD_POSITION” is on, all variables correspond to time $t - \Delta t$, otherwise positions correspond to time t and the other variables to time $t - \Delta t$.

The tracer acceleration is computed by forward finite difference scheme.

Heavy-Light Particles: particle forces

set(LAGRANGIAN_PARTICLE_FORCES "YES")

A boolean variable is associated to each force. It is fixed in the param.in and can assume only 0 or 1 values, depending on whether or not the force has effectively to act on the particle.

When Flag LAGRANGIAN_PARTICLE_FORCES is on, also set(LAGRANGIAN_PARTICLE_PROPERTIES "YES") is on. The associated parameter:

lagrangian_particle_properties_tau_values distinguishes between tracers and heavy/light particles.

Tracers have lagrangian_particle_properties_tau_values = 0.0 and lagrangian_particle_forces_stokes_boolean = 0.

Heavy-light particles have lagrangian_particle_properties_tau_values $\neq$ 0.0.

Tracer-CmakeLists.mine.start with FORCES

```
set(PARTICLE_MICHEL "YES")
set(LAGRANGIAN "YES")
set(LAGRANGIAN_PARTICLE "YES")
set(LAGRANGIAN_PARTICLE_BASETYPE_DOUBLE "YES")
set(LAGRANGIAN_PARTICLE_INIT "YES")
set(LAGRANGIAN_PARTICLE_INIT_START "YES")
set(LAGRANGIAN_PARTICLE_INIT_FLUID_VELOCITY "YES")
set(LAGRANGIAN_PARTICLE_INTERPOLATION "YES")
set(LAGRANGIAN_PARTICLE_INTERPOLATION_LINEAR "YES")
set(LAGRANGIAN_PARTICLE_FORCES "YES")
set(LAGRANGIAN_PARTICLE_FORCES_STOKES "YES")
set(LAGRANGIAN_PARTICLE_IO_HDF5 "YES")
set(LAGRANGIAN_PARTICLE_WRITE_SORTED_HDF5 "YES")

set(LAGRANGIAN_DUMP "YES")
```

Tracer-CmakeLists.mine.restart with FORCES

```
set(PARTICLE_MICHEL "YES")
set(LAGRANGIAN "YES")
set(LAGRANGIAN_PARTICLE "YES")
set(LAGRANGIAN_PARTICLE_BASETYPE_DOUBLE "YES")
set(LAGRANGIAN_PARTICLE_INIT "YES")
set(LAGRANGIAN_PARTICLE_INIT_RESTART "YES")
set(LAGRANGIAN_PARTICLE_INTERPOLATION "YES")
set(LAGRANGIAN_PARTICLE_INTERPOLATION_LINEAR "YES")
set(LAGRANGIAN_PARTICLE_FORCES "YES")
set(LAGRANGIAN_PARTICLE_FORCES_STOKES "YES")
set(LAGRANGIAN_PARTICLE_IO_HDF5 "YES")
set(LAGRANGIAN_PARTICLE_WRITE_SORTED_HDF5 "YES")
set(LAGRANGIAN_DUMP "YES")
```

Tracer-param.in with FORCES

lagrangian_init_start_flag 1
lagrangian_dump_rate 25
lagrangian_particle_nfamilies 1
lagrangian_particle_number_total 64
lagrangian_particle_number_fast 32
lagrangian_particle_ndump 50
lagrangian_particle_ndump_fast 1
lagrangian_particle_properties_tau_values0 0
lagrangian_particle_forces_stokes_boolean0 0

Lagrangian tracking in the Pseudo-Spectral code

- Implementation of Tracers
- Dumping flags and parameters
- Implementation of Heavy-Light particles
- Code structure

Derivation of particle equation in the form integrated by the code

$$\rho_p \mathcal{V}_p \frac{d\mathbf{v}}{dt} = (\rho_p - \rho_f)\mathcal{V}_p \mathbf{g} - C_D \frac{\pi a^2}{2} \rho_f |\mathbf{v} - \mathbf{u}|(\mathbf{v} - \mathbf{u})$$
$$+\rho_f \mathcal{V}_p C_M \left(\frac{D\mathbf{u}}{Dt} - \frac{d\mathbf{v}}{dt} \right) + \rho_f \mathcal{V}_p \frac{D\mathbf{u}}{Dt}$$

[Maxey & Riley *Phys. Fluids* (1983)]

Terms on the r.h.s. : gravity+Archimedes - Stokes drag - added mass - fluid acceleration

a particle radius, $\mathcal{V}_p$ particle volume

$C_D = \frac{24}{Re_p} = \frac{12\nu}{a|\mathbf{v}-\mathbf{u}|}$ drag coefficient for a solid sphere, i.e. $\mu_p \gg \mu_f$

$C_M = \frac{1}{2}$ added mass coefficient for a spherical particle

Particle equation integrated by the code

$$\left(1+\frac{1}{2\beta_M}\right)\frac{d\mathbf{v}}{dt}=\left(1-\frac{1}{\beta_M}\right)\mathbf{g}-\frac{1}{\tau_M}(\mathbf{v}-\mathbf{u})+\left(\frac{1}{2\beta_M}+\frac{1}{\beta_M}\right)\frac{D\mathbf{u}}{Dt}$$

density ratio : $\boxed{\beta_M=\frac{\rho_p}{\rho_f}}$

The terms on the r.h.s are computed in the particle_forces.c subroutine:

gravity $\longrightarrow$ set(LAGRANGIAN_PARTICLE_FORCES_GRAVITY "YES")

drag $\boxed{\frac{1}{\tau_M}=\frac{9\nu}{2a^2\beta_M}}$ $\longrightarrow$ set(LAGRANGIAN_PARTICLE_FORCES_STOKES "YES")

added mass $\longrightarrow$ set(LAGRANGIAN_PARTICLE_FORCES_ADDEDM "YES")

fluid pressure term $\longrightarrow$ set LAGRANGIAN_PARTICLE_FORCES_PRESS "YES")

“Standard” particle equation of motion

$$\frac{d\mathbf{v}}{dt} = \beta \frac{D\mathbf{u}}{Dt} - \frac{1}{\tau_s}(\mathbf{v} - \mathbf{u})$$

$$\beta = \frac{3\rho_f}{\rho_f + 2\rho_p}, \qquad \tau_s = \frac{a^2}{3\nu\beta}$$

$\beta < 1 \rightarrow$ heavy particles $\rho_p > \rho_f$

$\beta > 1 \rightarrow$ light particles $\rho_p < \rho_f$

Parameter conversion

$$\boxed{\beta = \frac{3}{1 + 2\beta_M} \quad \tau_s = \tau_M \left(1 + \frac{1}{2\beta_M}\right)}$$

$$\boxed{\beta_M = \frac{1}{2}\left(\frac{3}{\beta} - 1\right) \quad \tau_M = \frac{\tau_s}{3}(3 - \beta)}$$

Heavy-Light Particles equation with rotation

$$\frac{d\mathbf{v}}{dt} = \beta\frac{D\mathbf{u}}{Dt} - \frac{1}{\tau_p}(\mathbf{v} - \mathbf{u}) + 2(\mathbf{v} - \beta\mathbf{u}) \times \mathbf{\Omega} - (1-\beta)\mathbf{\Omega} \times (\mathbf{\Omega} \times \mathbf{r})$$

added mass   Stokes drag   Coriolis   Centrifugal force

Added mass $\longrightarrow$ set(LAGRANGIAN_PARTICLE_FORCES_ADDEDM "YES")+
set(LAGRANGIAN_PARTICLE_FORCES_PRESS "YES")

Drag $\longrightarrow$ set(LAGRANGIAN_PARTICLE_FORCES_STOKES "YES")

Coriolis $\longrightarrow$ set(LAGRANGIAN_PARTICLE_FORCES_CORIOLIS "YES")

Centrifugal $\longrightarrow$ set(LAGRANGIAN_PARTICLE_FORCES_CENTRIPETALROTX "YES")

Heavy-Light Particles: particle forces

set(LAGRANGIAN_PARTICLE_FORCES_STOKES "YES")
lagrangian_particle_forces_stokes_boolean
lagrangian_particle_properties_tau_values $\longrightarrow \tau_M$

set(LAGRANGIAN_PARTICLE_FORCES_ADDEDM "YES")
lagrangian_particle_forces_addedm_boolean
lagrangian_particle_properties_beta_values $\longrightarrow \beta_M$

set(LAGRANGIAN_PARTICLE_FORCES_PRESS "YES")
lagrangian_particle_forces_press_boolean

set(LAGRANGIAN_PARTICLE_FORCES_CORIOLIS "YES")
lagrangian_particle_forces_coriolis_boolean

set(LAGRANGIAN_PARTICLE_FORCES_CENTRIPETALROTX "YES")
lagrangian_particle_forces_centripetalrotx_boolean

ADDEDM & PRESS $\Rightarrow$ set(LAGRANGIAN_PARTICLE_DERIVATIVES "YES")
CENTRIPETALROTX $\Rightarrow$ set(LAGRANGIAN_PARTICLE_OLD_POSITION "YES")

Heavy-Light Particles-CmakeLists.mine.start No Rotation

```
set(PARTICLE_MICHEL "YES")
set(LAGRANGIAN "YES")
set(LAGRANGIAN_PARTICLE "YES")
set(LAGRANGIAN_PARTICLE_BASETYPE_DOUBLE "YES")
set(LAGRANGIAN_PARTICLE_INIT "YES")
set(LAGRANGIAN_PARTICLE_INIT_START "YES")
set(LAGRANGIAN_PARTICLE_INIT_FLUID_VELOCITY "YES")
set(LAGRANGIAN_PARTICLE_INIT_COLLAPSE "YES")
set(LAGRANGIAN_PARTICLE_INTERPOLATION "YES")
set(LAGRANGIAN_PARTICLE_INTERPOLATION_BSPLINE "YES")
set(LAGRANGIAN_PARTICLE_FORCES "YES")
set(LAGRANGIAN_PARTICLE_FORCES_STOKES "YES")
set(LAGRANGIAN_PARTICLE_FORCES_PRESS "YES")
set(LAGRANGIAN_PARTICLE_FORCES_ADDEDM "YES")
set(LAGRANGIAN_PARTICLE_IO_HDF5 "YES")
set(LAGRANGIAN_PARTICLE_WRITE_SORTED_HDF5 "YES")
set(LAGRANGIAN_DUMP "YES")
```

Heavy-Light Particles-CmakeLists.mine.restart No Rotation

```
set(PARTICLE_MICHEL "YES")
set(LAGRANGIAN "YES")
set(LAGRANGIAN_PARTICLE "YES")
set(LAGRANGIAN_PARTICLE_BASETYPE_DOUBLE "YES")
set(LAGRANGIAN_PARTICLE_INIT "YES")
set(LAGRANGIAN_PARTICLE_INIT_RESTART "YES")
set(LAGRANGIAN_PARTICLE_INTERPOLATION "YES")
set(LAGRANGIAN_PARTICLE_INTERPOLATION_BSPLINE "YES")
set(LAGRANGIAN_PARTICLE_FORCES "YES")
set(LAGRANGIAN_PARTICLE_FORCES_STOKES "YES")
set(LAGRANGIAN_PARTICLE_FORCES_PRESS "YES")
set(LAGRANGIAN_PARTICLE_FORCES_ADDEDM "YES")
set(LAGRANGIAN_PARTICLE_IO_HDF5 "YES")
set(LAGRANGIAN_PARTICLE_WRITE_SORTED_HDF5 "YES")
set(LAGRANGIAN_DUMP "YES")
```

1 Tracer + 3 Heavy/Light families param.in

```
lagrangian_particle_nfamilies 4
lagrangian_particle_number_total 640
lagrangian_particle_number_fast 320
lagrangian_dump_rate 200
lagrangian_particle_ndump 100
lagrangian_particle_ndump_fast 10
lagrangian_particle_interpolation_bspline_size_m0 4
lagrangian_particle_interpolation_bspline_size_m1 4
lagrangian_particle_interpolation_bspline_size_m2 4
lagrangian_particle_interpolation_bspline_size_m3 4
lagrangian_particle_properties_tau_values0 0
lagrangian_particle_properties_tau_values1 0.123
lagrangian_particle_properties_tau_values2 0.0885
lagrangian_particle_properties_tau_values3 0.059
lagrangian_particle_properties_beta_values0 0
lagrangian_particle_properties_beta_values1 2.5
lagrangian_particle_properties_beta_values2 0.5
lagrangian_particle_properties_beta_values3 1
```

lagrangian_particle_forces_stokes_boolean0 0
lagrangian_particle_forces_stokes_boolean1 1
lagrangian_particle_forces_stokes_boolean2 1
lagrangian_particle_forces_stokes_boolean3 1
lagrangian_particle_forces_addedm_boolean0 0
lagrangian_particle_forces_addedm_boolean1 1
lagrangian_particle_forces_addedm_boolean2 1
lagrangian_particle_forces_addedm_boolean3 1
lagrangian_particle_forces_press_boolean0 0
lagrangian_particle_forces_press_boolean1 1
lagrangian_particle_forces_press_boolean2 1
lagrangian_particle_forces_press_boolean3 1

Output files of Tracers and Heavy-Light particles

Files lagrangian_fast_#.h5 contain time, particle position, velocity and acceleration, fluid velocity and derivatives:

ifdef LAGRANGIAN_PARTICLE_OLD_POSITION
xo, yo, zo, else x,y,z, endif

uxo, uyo, uzo, axo, ayo, azo, vxo, vyo, vzo,

ifdef LAGRANGIAN_PARTICLE_DERIVATIVES
dvxo/dx, dvyo/dx, dvzo/dx, dvxo/dy, dvyo/dy, dvzo/dy, dvxo/dz, dvyo/dz, dvzo/dz, endif

particle name

the suffix 'o' stands for old. If FLAG "LAGRANGIAN_PARTICLE_OLD_POSITION" is on, all variables correspond to time $t - \Delta t$, otherwise positions correspond to time t and the other variables to time $t - \Delta t$.

The tracer acceleration is computed by:
ifdef LAGRANGIAN_PARTICLE_DERIVATIVES

centered finite difference scheme else forward finite difference scheme endif

INIT FLAGS for particles

LAGRANGIAN_PARTICLE_INIT_START : particles are released randomly in the domain

LAGRANGIAN_PARTICLE_INIT_START_FROZEN : particles are released at fixed position (for testing purposes)

LAGRANGIAN_PARTICLE_INIT_COLLAPSE : particles belonging to different families are initially located at the same position

LAGRANGIAN_PARTICLE_INIT_FLUID_VELOCITY : particles are released with the same velocity of the underlying fluid

LAGRANGIAN_PARTICLE_INIT_ZERO_VELOCITY : particles are released with zero velocity

LAGRANGIAN_PARTICLE_INIT_RESTART : for restart run, independently of the START condition

INIT FLAGS for particles

LAGRANGIAN_PARTICLE_INIT_STARTROTATIONX : particles are released in couples with separation O(dX). The first family is put randomly in the all domain, whereas other families are put on rotation axis parallel to x. The location of the rotation axis depends on the Y processor (mey). Axes are on the diagonal from (y,z) = (0,0) to (y,z) = (euler_sy,euler_sz). Particles belonging to one couple placed on a rotation axis are separated along the x direction only.

When using LAGRANGIAN_PARTICLE_INIT_STARTROTATIONX, the FLAG LAGRANGIAN_PARTICLE_FORCES_CENTRIPETALROTX has to be set to "yes", otherwise rotation axes are not defined.

LAGRANGIAN_PARTICLE_INIT_STARTROTATIONX_ANDUNIFORM : particles are released in couples with separation O(dX). The total number of families is nfamilies = nfamilies1 + nfamilies2, with the two new parameters required in param.in :
"lagrangian_particle_init_startrotationx_anduniform_nfamilies1" and
"lagrangian_particle_init_startrotationx_anduniform_nfamilies2"
there are nfamilies1 families randomly put in the volume and nfamilies2 families on rotation axis parallel to x.

LAGRANGIAN_PARTICLE_INIT_STARTROTATIONX_ANDUNIFORM_TETRAD : same as before but now particles are released in groups of four

INIT FLAGS for particles

LAGRANGIAN_PARTICLE_INIT_STARTROTATIONX_TETRAD : particles are released in groups of four. The location of the first particle is fixed randomly and the other three particles are put at a distance dX/2 from the first, along x, y, and z, respectively. Within each family, the 90% of the tetrads are randomly displaced in the domain and the remaining 10% lies on rotation axis parallel to x.

When using LAGRANGIAN_PARTICLE_INIT_STARTROTATIONX_ANDUNIFORM - LAGRANGIAN_PARTICLE_INIT_STARTROTATIONX_ANDUNIFORM_TETRAD - LAGRANGIAN_PARTICLE_INIT_STARTROTATIONX_TETRAD - also the flag LAGRANGIAN_PARTICLE_FORCES_CENTRIPETALROTX_ANDUNIFORM has to be set to "yes" in order for the particle structure to have two additional variables, ys and zs, that contain the location of the rotation axis (location of particle 1 of the couple or of the tetrad at the release time t=0)

NOTE : Flags LAGRANGIAN_PARTICLE_INIT_STARTROTATIONX_ANDUNIFORM - LAGRANGIAN_PARTICLE_INIT_STARTROTATIONX_ANDUNIFORM_TETRAD are neither under test nor fully debugged

Lagrangian tracking in the Pseudo-Spectral code

- Implementation of Tracers
- Dumping flags and parameters
- Implementation of Heavy-Light particles
- Code structure

Particle structures

The 2 particle structures are defined in file define.h:

particle_lagr_type:

```
ParticleBaseType x, y, z;                /* particle position */
ParticleBaseType uxo, uyo, uzo;          /* old particle velocity */
ParticleBaseType axoo, ayoo, azoo;       /* old old particle acceleration */

ifdef LAGRANGIAN_PARTICLE_DERIVATIVES    /* ADDEDM & PRESS */
ParticleBaseType Vo[12];      /* old fluid velocity with derivatives : vx,vy,vy,
dvx/dx,dvy/dx,dvz/dx, dvx/dy,dvy/dy,dvz/dy, dvx/dz,dvy/dz,dvz/dz */
ParticleBaseType Voo[3];      /* old old fluid velocity */
else
ParticleBaseType Vo[3];       /* old fluid velocity */ endif

ifdef LAGRANGIAN_PARTICLE_OLD_POSITION    /* CENTRIPETALROTX */
xo, yo, zo;      /* old particle position */ endif

ifdef LAGRANGIAN_PARTICLE_FORCES_CENTRIPETALROTX_ANDUNIFORM
ys, zs;          /* Location of particle rotation axis */ endif

unsigned int name;
```

Particle structures

```
particle_lagr_type_extra
double axo, ayo, azo;          /* old particle acceleration */
double ux, uy, uz;             /* particle velocity */
double xn, yn, zn;             /* new particle position */

ifdef LAGRANGIAN_PARTICLE_DERIVATIVES    /* ADDEDM & PRESS */
ParticleBaseType V[12];        /* fluid velocity with derivatives */
else
ParticleBaseType V[3];         /* fluid velocity */ endif

int family_number;

ifdef LAGRANGIAN_PARTICLE_FORCES_STOKES
double dvx, dvy, dvz; endif

ifdef LAGRANGIAN_PARTICLE_FLUID_MAT_DERIVATIVE    /* ADDEDM & PRESS */
double DvxDto, DvyDto, DVzDto; endif
```

suffix V $\Rightarrow$ fluid velocity

suffix u $\Rightarrow$ particle velocity

'oo' refers to time $t - 2\Delta t$, 'o' to time $t - \Delta t$, ' ' to time t, and 'n' to time $t + \Delta t$

Lagrangian integration

main(){

init_lagrangian() {
if (RESTART) read the structure 'particle' from file lagr_in.h5 else
allocate structure 'particle' (of type : particle_lagr_type)
fix particles positions and name depending on the START condition.
All other variables of the structure are set to zero. endif }

nlt() {

call particle_f(){
assign to part_i_extra.V[ ] fluid velocity and derivatives at particle position by making use of the choosen interpolation scheme.

if(START) call particle_start(){
copy : part_i_extra.V[ ] in part_i.Vo[ ]
copy : part_i_extra.V[0]-[1]-[2] in part_i.Voo[0]-[1]-[2]
ifdef LAGRANGIAN_PARTICLE_FORCES_CENTRIPETALROTX
assign : part_i.xo-yo-zo = part_i.x-y-z endif

ifdef LAGRANGIAN_PARTICLE_INIT_FLUID_VELOCITY
copy : part_i_extra.V[0]-[1]-[2] in part_i.uxo-uyo-uzo endif }

Lagrangian integration

```
particle_forces() {
compute :   part_i_ extra->DvxDto-DvyDto-DvzDto
if(not a tracer) compute :   part_i_extra.axo-ayo-azo endif }

ifdef LAGRANGIAN_PARTICLE_IO_HDF5
dump files lagrangian_fast_#.h5   endif

particle_move() {
if(not a tracer){

if(START) {
assign :   part_i.aoox-aooy-aooz  =  part_i_extra.aox-aoy-aoz
assign :   part_i_extra.ux-uy-uz   =  part_i.uxo-uyo-uzo }  else {
the 'aoo's have been fixed in particle_toggle( )
assign :   part_i_extra.ux  =  part_i.uxo + 0.5 Δt ( 3 * part_i_extra.axo - part_i.axoo)    endif }

else { /* it is a tracer */
assign :   part_i_extra.ux-uy-uz  =  part_i_extra.V[0]-[1]-[2]  endif }

part_i_extra.xn = part_i.x + 0.5 Δt ( 3 * part_i_extra.ux-part_i.uxo) } }
```

Lagrangian integration

```
particle_toggle() {
part_i.Voo[0]-[1]-[2]  =  part_i.Vo[0]-[1]-[2]
part_i.Vo[0]-[1]-[2]  =  part_i_extra.V[0]-[1]-[2]

part_i.uxo-uyo-uzo     = part_i_extra.ux-uy-uz
part_i.axoo-ayoo-azoo =  part_i_extra.axo-ayo-azo

ifdef LAGRANGIAN_PARTICLE_OLD_POSITION  part_i.xo-yo-zo  =  part_i.x-y-z endif }
} /* end of nlt( ) */
```