**GUIDE TO PASSIVE TRACERS IN SFINCS**

ELIZABETH PAUL

# 1. Input Options

## 1.1. Input Namelist.

- `--enable-mctracers` Enables Monte-Carlo tracers
- `--enable-vftracers` Enables Velocity Field tracers
- `--with-integrator=vl` Tracers have only been implemented with the VL integrator.

SMR must be disabled (default setting) with tracers.

## 1.2. Preprocessor Directives.

- `#define TOPHAT` Apply top-hat smoothing of the particle output.
- `#define DEBUG` Additional debugging functions are called to test particle algorithms.

# 2. Source Code

The tracer particle modules are located in the `src/tracers/` directory. The following files have been added.

- `bvals_tracer.c` Reflecting, outflow, and periodic boundary conditions have been implemented for the `VFTRACERS` and `MCTRACERS`.
- `init_tracer_grid.c` Tracer grid is initialized. Tracers can be initialized with uniform density (`tracer_init_unif`), in cells above a threshold density (`tracer_init_threshold`), or proportional to the fluid density (`tracer_init_proportional`). The function `tracer_init_xlinflow` initializes tracers in the ghost zone for outflow boundary problems. The function `tracer_debug` is used for testing purposes using assert statements.
- `integrate_tracers.c`
  - `Tracerlist_sweep` Sweeps through tracer list to move tracers that have been flagged for removal.
  - `Tracerlist_sweep_bc` Sweeps through tracer list (on boundary) to move tracers flagged for removal.
  - `prob_iterate_x1` Sweeps through list to flag tracers to be moved in x1 direction.
  - `prob_iterate_x2` Same, but in x2 direction.
  - `prob_iterate_x3` Same, but in x3 direction.
  - `flag_tracer_star` This function is called when a star is created so a tracer can be flagged with `star_id`.
  - `mc_tophat` Tophat algorithm is used to smooth tracer density output.

- `mctracer_out.c` Write tracer output in formatted table, including density, initial density, position, and time. This also includes information of tracer particles within starparticles. Note that this function is currently not being called and probably has not been thoroughly tested.
- `output_tracer_vtk.c` Writes output in vtk format.
- `vfintegrate.c` Contains functions for integrating VF tracers.
  - `Integrate_vf_2nd` Uses a second-order integration method with predicted position at $t + dt/4$.
  - `Integrate_vf_2nd_lower` Uses a second-order integration method without predicted position at $t + dt/4$.
  - `vf_newijk` Sweeps through tracer grid to move VF tracers to new positions.
  - `vf_newpos` This function is called to move VF tracer from a ghost zone to a cell in the active zone.
  - `interp` Uses interpolation weights to interpolate from the new time step.
  - `interp_prev` Uses interpolation weights to interpolate from the previous time step.
- `vfinterp.c` Contains several functions to obtain the interpolation weights for integration of the VF tracers.
  - `getwei_linear` Uses linear interpolation.
  - `getwei_TSC` Uses Traingular Shaped Cloud interpolation.
  - `getwei_QP` Uses quadratic polynomial interpolation.

2.1. **`main.c`.** If `MCTRACERS` or `VFTRACERS` are defined, the tracer grid is initialized (call to `init_tracer_grid`) after the grid and mesh are initialized in Step 4. The boundary conditions are initialized with a call to `bvals_tracer_init`, and the boundary condition is set with a call to `bvals_tracer` during Step 6. The boundary values are set again after the time is updated in Step 9h (another call to `bvals_tracer`). After updating the boundary values, there is a call to a debugging routine (`tracer_debug`) and a top-hat algorithm is called to smooth the output (`mc_tophat`). The tracer memory is freed with a call to `tracer_destruct`.

2.2. **Integrators.** I have implemented the tracer algorithms in the MUSC-Hancock (VL) integrators in 1D, 2D, and 3D.

- `src/integrators/integrate_1d_vl.c` If `MCTRACERS` are defined, the probability flux of Monte Carlo tracer transfer is computed. The list of tracers is iterated through, and some are marked for transfer (call to `prob_iterate_x1`) to adjacent grid cells. The reduced mass is updated. The list of tracers is then swept through again, and those marked for transfer are moved (call to `Tracerlist_sweep`). If `VFTRACERS` is defined, the tracers positions are integrated (call to `Integrate_vf_2nd`). The list of tracers is swept through, and those marked for transfer are moved (call to `Tracerlist_sweep`).
- `src/integrators/integrate_2d_vl.c` If `MCTRACERS` are defined, a similar procedure is followed as in the 1d integrator. There is an additional sweep through the tracers on the boundary if the `inflow_x1` problem is being used. To compute the probability of transfer,

prob_iterate_x2 is called rather than prob_iterate_x1. If VFTRACERS are defined, a call to an integrator is called (Integrate_vf_2nd_lower). The tracers are then moved to the linked list corresponding to their new position (call to vf_newijk).

- src/integrators/integrate_3d_vl.c The implementation is the same as in integrate_2d_vl.c, but a call to prob_iterate_x3 is made.

2.3. **init_mesh.c.** An MPI structure type for the tracer particles is created for communication.