Smooth-Particle Hydrodynamics Assignment
ACSE 4
Applied Computational Science and Engineering
Imperial College London

Team members: Jorge Garcia, Qing Ma, Wenjia Wan, Gabriel Lipkowitz, Jinhui Zhao, Hanchao Chen

This program implements a *Smooth-Particle Hydrodynamics* (SPH) simulation with the following files and methods.

SPH_Snippet.cpp contains the **main()** function for starting the simulation. SPH_2D.cpp contains two classes, one for an individual SPH particle with methods:

void SPH_particle::init_particle() // *initializes a single fluid (or boundary) particle*

void SPH_particle::set_particle_deri(void) *// initializes the particle derivative*

void SPH_particle:: cal_P(void) *// calculates the pressure of a single particle*

void SPH_particle::calc_index(void) *// calculates the index of the particle in the search grid*

SPH_2D.cpp also contains a class for the overall SPH domain, with methods:

void SPH_main::set_values(void) // initializes the particle values for the domain

void SPH_main::initialise_grid(void) // initializes the search grid for finding neighbors

void SPH_main::place_points(double* min, double* max) // assigns points to domain

void SPH_main::fill_domain() // fills domain with particles

void SPH_main::allocate_to_grid(void) // assigns particles to the search grid

void SPH_main::check_if_topped(SPH_particle* part) // determines if particles overlap

void SPH_main::cal_derivative(SPH_particle* part) // calculates the position, velocity, density derivatives for a particles in domain

void SPH_main::smooth_density(SPH_particle* part, int t_step) // smoothing function for time-stepping

void SPH_main::update_parameters_fe(int step,double dt) // uses forward euler explicit method to update parameters for all particles

void SPH_main::get_cfl_time_step(double* part_v, double* other_part_v) // for dynamic time-stepping

void SPH_main::get_tf_ta_time_step(double rho, double* dedv) // for dynamic time-stepping

void SPH_main::update_min_time_step() // for dynamic time-stepping

bool SPH_main::check_inside_domain(SPH_particle* part) // checks if a particle is inside the valid domain

For writing files, file_writer.cpp and file_writer.h are included. The data generated using these programs, which are called in SPH_Snippet.cpp, can be readily visualized with *Paraview*.

In the Github directory pseudo_arbitrary_sloped_boundaries, an added capability is included to allow the user to input a slope, between 0 and 0.5, which specifies the steepness of a beachhead for the simulation. If the slope is too steep to stretch across the entire horizontal domain, then the beachhead is placed to the right hand side of the simulation domain.

Analysis of the converge data is included in Analysis.ipynb.