RRSPH v3.0.19

Asrankulov Alexander

October 2024

1 Experiment

1.1 Experiment directory

Experiment directory is a directory with parameters files and particles data. You should provide specifically prepared directory to solver.

Minimal experiment directory to start an experiment is presented below:

```
\---experiment_00
    |   ModelParams.json
    |   ParticleParams.json
    |
    \---dump
            0.csv
```

Any RRSPH simulation program started from parent directory will produce the following output:

```
Found experiments:
[-1] Change search directory
[0] experiment_00: (0/1) data/dump layers
Type experiment id you want to load:
>
```

You should type '0' and press 'Enter' in order to start simulation with specified parameters and initial state.

Experiment directories with 'ParticleParams.json' and initial state ('dump/0.csv') are usually generated with scripts. Mathematical model's 'ModelParams.json' is user-provided: manually or by SPH2DParamsGenerator app.

1.2 RRSPH package

1.2.1 SPH2D simulation

There are two simulation programs:

- SPH2D_OMP. SPH 2D solver running on CPU with OpenMP or single-threaded.
- SPH2D_CL. SPH 2D solver running on GPU with OpenCL runtime. It's executable file is distributed with CL source code in separate directory:

```
|   SPH2D_CL.exe
|
+---cl
|       ArtificialViscosity.cl
|       AverageVelocity.cl
|       clErrorCodes.txt
|       clparams.h
|       common.h
|       Density.cl
|       EOS.cl
|       ExternalForce.cl
|       GridFind.cl
|       GridUtils.h
|       InternalForce.cl
```

```
|       ParamsEnumeration.h
|       SmoothingKernel.cl
|       SmoothingKernel.h
|       TimeIntegration.cl
|
\---experiment_00
    |   ModelParams.json
    |   ParticleParams.json
    |   SPH2DParams.json
    |
    \---dump
            0.csv
```

1.2.2 SPH2D Post-processing

There's several post-processing programs:

- WaterProfile. This app can be used to compute water profiles in space and time.
- FuncAtPoint. This app can be used to extract field variable at specified point in space by SPH smoothing.
- PartToGridConverter. This app can be used to convert SPH particles output into grid to process as mesh algorithm's output.

2 Parameters Scheme

2.1 Model Params

2.1.1 Target version

- `params_target_version_major`
- `params_target_version_minor`
- `params_target_version_patch`

2.1.2 Smoothing kernel function

Possible values:

- 1: `SKF_CUBIC`.

$$W(q) = \kappa_{\text{cubic}} \begin{cases} \frac{2}{3} - q^2 + \frac{1}{2}q^3 & q \leq 1, \\ \frac{1}{6}(2-q)^3 & 1 < q \leq 2, \\ 0 & q > 2. \end{cases} \tag{1}$$

- 2: `SKF_GAUSS`.

$$W(q) = \kappa_{\text{gauss}} \begin{cases} \frac{1}{h^2\pi} e^{q^{-2}} & q \leq 3, \\ 0 & q > 3. \end{cases} \tag{2}$$

- 3: `SKF_WENDLAND`.

$$W(q) = \kappa_{\text{wendland}} \begin{cases} \left(1 - \frac{q}{2}\right)^4 (2q+1) & q \leq 2, \\ 0 & q > 2. \end{cases} \tag{3}$$

- 3: `SKF_DESBRUN`.

$$W(q) = \kappa_{\text{desbrun}} \begin{cases} (2-q)^3 & q \leq 2, \\ 0 & q > 2. \end{cases} \tag{4}$$

2.1.3 Density

- `density_treatment`. Density integration function. Possible values of enumeration:
  - 0: `DENSITY_SUMMATION`. Use SPH summation over kernel:
$$\rho_j = \sum_i m_i W_{ji}. \tag{5}$$
  - 1: `DENSITY_CONTINUITY`. Use integration of continuity equation:
$$\frac{D\rho_j}{Dt} = \sum_i m_i \vec{v_{ji}} \cdot \vec{\nabla}_j W_{ji}. \tag{6}$$
  - 2: `DENSITY_CONTINUITY_DELTA`. Use integration of continuity equation with density diffussion:
$$\frac{D\rho_j}{Dt} = \sum_i m_i \vec{v}_{ji} \cdot \vec{\nabla}_j W_{ji} + 2\delta_{\text{density}} hc \sum_i m_i \frac{(\rho_i - \rho_j)}{\rho_i} \frac{\vec{r}_{ji}}{|\vec{r}_{ji}|^2} \cdot \vec{\nabla}_j W_{ji}. \tag{7}$$
- `density_normalization`. Enable boundary deficiency correction:
$$\rho_j = \frac{\sum\limits_i m_i W_{ji}}{\sum\limits_i \left(\frac{m_i}{\rho_i}\right) W_{ji}} \tag{8}$$
  Optional. Available if `density_treatment == 0`.
- `density_skf`. See section 2.1.2.
- `density_delta_sph_coef`. δ_{density} parameter in eq. 7.
  Mandatory if `density_treatment == 2`.

2.1.4 Equation of state

- `eos_sound_vel_method`. Method for sound velocity choosing. Possible values of enumeration:
  - 0: `EOS_SOUND_VEL_DAM_BREAK`. Use dam break assumption:
$$c = \sqrt{200 g d c_k}. \tag{9}$$
  - 1: `EOS_SOUND_VEL_SPECIFIC`. Use custom sound velocity.
$$c = c_{\text{user}}. \tag{10}$$
- `eos_sound_vel_coef`. c_k parameter in eq. 9.
  Mandatory if `eos_sound_vel_method == 0`.
- `eos_sound_vel`. User-provided sound velocity c_{user} in eq. 10.
  Mandatory if `eos_sound_vel_method == 1`.

2.1.5 Internal Forces

- `intf_sph_approximation`. SPH momentum equation form. Possible values of enumeration:
  - 1: `INTF_SPH_APPROXIMATION_1`. SPH momentum equation form:
$$\frac{D\vec{v}_j}{Dt} = -\sum_i m_i \left(\frac{p_i + p_j}{\rho_i \rho_j}\right) \vec{\nabla}_j W_{ji}. \tag{11}$$
  - 2: `INTF_SPH_APPROXIMATION_2`. SPH momentum equation form:
$$\frac{D\vec{v}_j}{Dt} = -\sum_i m_i \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2}\right) \vec{\nabla}_j W_{ji}. \tag{12}$$
- `intf_hsml_coef`. h_k in smoothing kernel length equation:
$$h = \delta_0 \cdot h_k, \tag{13}$$
  where δ_0 is initial distance between particles.
- `intf_skf`. See section 2.1.2.

2.1.6 Artificial Pressure

- `artificial_pressure`. Enable additional factor in momentum equation for tensile instability correction:

$$\frac{D\vec{v}_j}{Dt} = -\sum_i m_i \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2} + R\tilde{W}_{ji}^n \right) \vec{\nabla}_j W_{ji}, \tag{14}$$

where $R\tilde{W}_{ji}^n$ is artificial pressure term. SPH approximation function could be of both types.

$$R = R_i + R_j, \tag{15}$$

where any indexed R_x is:

$$R_x = \begin{cases} \dfrac{\epsilon |p_x|}{\rho_x^2} & p_x < 0, \\ 0 & p_x \geq 0, \end{cases} \tag{16}$$

where ϵ is artificial pressure coefficient.

$$\tilde{W}_{ji} = \frac{W(\vec{r}_{ji}, h)}{W(\delta_0, h)}, \tag{17}$$

where δ_0 is initial distance between particles.

- `artificial_pressure_skf`. See section 2.1.2.
- `artificial_pressure_index`. Term n in eq. 14.
- `artificial_pressure_coef`. Term ϵ in eq. 16.

2.1.7 Artificial Viscosity

- `artificial_viscosity`. Enable additional factor in momentum equation for stability correction:

$$\frac{D\vec{v}_j}{Dt} = -\sum_i m_i \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2} + \Pi_{ji} \right) \vec{\nabla}_j W_{ji}, \tag{18}$$

where Π_{ji} is artificial viscosity term. SPH approximation function could be of both types. Π_{ji} is:

$$\Pi_{ji} = \begin{cases} \dfrac{-\alpha_\Pi c_{ji} \phi_{ji} + \beta_\Pi \phi_{ji}^2}{\rho_{ji}^2} & \vec{v}_{ji} \cdot \vec{r}_{ji} < 0, \\ 0 & \vec{v}_{ji} \cdot \vec{r}_{ji} \geq 0, \end{cases} \tag{19}$$

$$\phi_{ji} = \frac{h_{ji} \vec{v}_{ji} \cdot \vec{r}_{ji}}{|\vec{r}_{ji}|^2 + (\epsilon_\Pi h_{ji})^2}, \tag{20}$$

$$\rho_{ji} = \frac{\rho_j + \rho_i}{2}, \quad c_{ji} = \frac{c_j + c_i}{2}, \quad h_{ji} = \frac{h_j + h_i}{2}, \tag{21}$$

$$\vec{v}_{ji} = \vec{v}_j - \vec{v}_i, \quad \vec{r}_{ji} = \vec{r}_j - \vec{r}_i. \tag{22}$$

- `artificial_viscosity_skf`. See section 2.1.2.
- `artificial_shear_visc`. Term α_Π in eq. 19.
- `artificial_bulk_visc`. Term β_Π in eq. 19.

2.1.8 Average Velocity

- `average_velocity`. Enable XSPH velocity smoothing factor:

$$\frac{D\vec{r}_j}{Dt} = \vec{v}_j + \vec{v}_{j\,\text{avg}}, \tag{23}$$

where v_{avg} is:

$$\vec{v}_{j\,\text{avg}} = -\varepsilon_{\text{avg}} \sum_i \frac{m_i}{\rho_i} \vec{v}_{ji} W_{ji}. \tag{24}$$

- `average_velocity_skf`. See section 2.1.2.
- `average_velocity_coef`. Term ε_{avg} in eq. 24.

2.1.9 Time Integration

- `simulation_time`. Total simulation time to stop experiment.
- `dt_correction_method`. Method of Δt choosing. Possible values of enumeration:
  - 0: `DT_CORRECTION_CONST_VALUE`. Use user-provided constant value.

$$\Delta t_\tau = \Delta t_{\text{user}}. \tag{25}$$

  - 1: `DT_CORRECTION_CONST_CFL`. Δt is computed before experiment start with CFL condition:

$$\Delta t_\tau = \Delta t_0 = \text{CFL}\frac{h}{c(1+1.2\alpha_\Pi)} \tag{26}$$

  - 2: `DT_CORRECTION_DYNAMIC`. Δt is computed on every step with CFL condition:

$$\Delta t_\tau = \text{CFL}\min(\Delta t_f, \Delta t_\phi), \tag{27}$$

$$\Delta t_f = \min_i\left(\sqrt{\frac{h}{a_i}}\right), \tag{28}$$

$$\Delta t_\phi = \min_i\left(\frac{h}{c+\phi}\right). \tag{29}$$

- `dt`. Term Δt_{user} in eq. 25.
- `CFL_coef`. Term CFL in eq. 26 and eq. 27.

2.1.10 Boundary Treatment

- `boundary_treatment`. Selects method for boundaries. Possible values of enumeration:
  - 0: `SBT_DYNAMIC`. Boundary particles are the same as fluid but have constant positions.
  - 1: `SBT_REPULSIVE`. Boundary particles are dynamic with additional term in momentum equation:

$$\frac{D\vec{v}_j}{Dt} = -\sum_i\left[m_i\left(\frac{p_j}{\rho_j^2}+\frac{p_i}{\rho_i^2}\right)\vec{\nabla}_j W_{ji} + \vec{\Upsilon}_{ji}\right], \tag{30}$$

    where $\vec{\Upsilon}_{ji}$ is computed if j is fluid particle and i is boundary particle:

$$\vec{\Upsilon}_{ji} = \frac{\vec{r}_{ji}}{|\vec{r}_{ji}|}\begin{cases} D\left[\left(\frac{r_0}{|\vec{r}_{ji}|}\right)^{\alpha_1} - \left(\frac{r_0}{|\vec{r}_{ji}|}\right)^{\alpha_2}\right] & |\vec{r_{ji}}| \le r_0, \\ 0 & |\vec{r_{ji}}| > r_0. \end{cases} \tag{31}$$

    Equation parameters are considered built-in:

$$\begin{aligned} r_0 &= 2h, \\ D &= 5gd, \\ \alpha_1 &= 12, \\ \alpha_2 &= 4, \end{aligned} \tag{32}$$

    where d is initial water depth.

2.1.11 Numerical Waves Maker

- `nwm`. Waves maker method. Possible values of enumeration:
  - 0: `NWM_NO_WAVES`. No waves generator.
  - 2: `NWM_METHOD_DYNAMIC`. First order waves generator. Piston-type wavemaker. Makes waves with surface displacement of type:

$$\eta(x,t) = \frac{H}{2}cos(\omega t - kx + \delta), \tag{33}$$

where H is wave height, x is distance and δ is the initial phase. $\omega = 2\pi/T$ is the angular frequency and $k = 2\pi/L$ is the wave number with T equal to the wave period and L the wave length.

Piston displacement equation is:

$$e_1(t) = \frac{S_0}{2} sin(\omega t + \delta), \tag{34}$$

where S_0 is piston magnitude:

$$S_0 = H \frac{\sinh kd \cosh kd + kd}{2 \sinh kd^2}, \tag{35}$$

where d is depth.

- 4: `NWM_METHOD_WALL_DISAPPEAR`.

Methods `NWM_METHOD_DYNAMIC` and `NWM_METHOD_WALL_DISAPPEAR` require ParticleParams to provide `nwm_particles_start` and `nwm_particles_end`.

- `nwm_time_start`. Simulation time when NWM starts to generate waves.

  Optional parameter.

- `nwm_wave_length`. Wave length L to be generated.

  Mandatory when `nwm == NWM_METHOD_DYNAMIC`.

- `nwm_wave_magnitude`. Wave height H to be generated.

  Mandatory when `nwm == NWM_METHOD_DYNAMIC`.

2.1.12 Output Control

- `save_time`. Program will produce output for further processing every `save_time` seconds of simulation. See 'data' directory of experiment.
- `save_velocity`. Enable velocity saving in output.
- `save_pressure`. Enable pressure saving in output.
- `save_density`. Enable density saving in output.

2.1.13 Dump Control

- `use_dump`. Enable dump dump creation.
- `dump_time`. Program will produce dump every `dump_time` seconds of simulation. You can use that point in time to start from later. See 'dump' directory of experiment.

2.1.14 Time Estimation

- `use_custom_time_estimate_step`. Enables user-set time estimation step. Program will print amount of time left for simulation to finish every k steps. If not set, program will print time estimation with every output.
- `step_time_estimate`. Term k in previous statement.

2.1.15 Consistency Control

- `consistency_check`. Enable consistency control: check if variables are NaN or infinite; check particles are outside of simulation domain.
- `consistency_treatment`. Selects method for consistency control. Possible values of enumeration:
  - 0: `CONSISTENCY_PRINT`. Prints warning message on inconsistent value of variables and continues simulation. Simulation with such values of variables is undefined.
  - 1: `CONSISTENCY_STOP`. Prints error message and stop simulation process.
  - 2: `CONSISTENCY_FIX`. Fixes particles leaving simulation domain (mark them as non-existing). Prints error message and stop simulation process on NaN or infinite values of variables.

2.1.16 Optimization

- `max_neighbours`. Maximum number of neighbours for particle. Affects on allocated memory for grid of neighbours.
- `local_threads`. Local threads number for OpenCL kernels and OpenMP parallel sections.

2.2 Particle Params

2.2.1 Target version

- `params_target_version_major`
- `params_target_version_minor`
- `params_target_version_patch`

2.2.2 Particle count

- `nfluid`. Fluid particles count.
- `nvirt`. Boundary (virtual) particles count.
- `ntotal`. Total particles count (fluid + boundary).

2.2.3 Geometry

These parameters describe simulation domain area.

- `x_mingeom`. Left side of the domain.
- `x_maxgeom`. Right side of the domain.
- `y_mingeom`. Bottom of the domain.
- `y_maxgeom`. Top of the domain.

2.2.4 Particles parameters

- `delta`. Initial distance between particles δ_0.
- `rho0`. Reference density of particles ρ_0.

  Optional parameter. Default value is $\rho_0 = 1000$.

2.2.5 Extra parameters

- `depth`. Reference depth of the fluid d.

  Optional parameter. By default $d = \max_i y_i$ on the start.

2.2.6 Numerical Waves Maker

If `nwm` is piston or disappearing wall, affected boundary particles are from the contiguous block characterized by two indices: [start; end).

- `nwm_particles_start`. Index of the first particle in moving or disappearing block.
- `nwm_particles_end`. Index of the next to the last particle in moving or disappearing block.

2.3 Computing Params

2.3.1 Software version

- `SPH2D_version_major`
- `SPH2D_version_minor`
- `SPH2D_version_patch`

2.3.2 Params module version

- `params_version_major`
- `params_version_minor`
- `params_version_patch`

2.3.3 Common module version

- `SPH2D_common_version_major`
- `SPH2D_common_version_minor`
- `SPH2D_common_version_patch`

2.3.4 Computing module version

- `SPH2D_specific_version_major`
- `SPH2D_specific_version_minor`
- `SPH2D_specific_version_patch`
- `SPH2D_specific_version_name`

2.3.5 Computed constants

- `mass`. Mass of the particle m:
$$m = \rho_0 \delta_0^{\mathrm{dim}}. \tag{36}$$
- `hsml`. Smoothing length h:
$$h = \delta_0 h_k. \tag{37}$$
- `cell_scale_k`. Size of grid cell in terms of h. Equals to 3 if any kernel is gauss. Equals to 2 otherwise.
- `maxn`. Maximum count of particles in simulation:
$$\mathrm{maxn} = 2^{1+\lfloor \log_2 \mathrm{ntotal} \rfloor} \tag{38}$$
- `max_cells`. Count of grid cells to be processed.
- `start_simulation_time`

2.3.6 Numerical Waves Maker

- `nwm_wave_number`. Generated wave number k.
- `nwm_freq`. Generated wave angular frequency ω.
- `nwm_piston_magnitude`. If nwm is dynamic, piston magnitude S_0 computed by eq. 35.

2.3.7 Particle type

- `TYPE_WATER`. Fluid particles type identifier in output.
- `TYPE_BOUNDARY`. Water particles type identifier in output.
- `TYPE_NON_EXISTENT`. Identifier of particles that are not exist.

2.4 Loading Params

It could be useful to load just a part of output data. Any post processing program will load files with respect to 'LoadingParams.json'. Computing programs can't start experiment where 'LoadingParams.json' exists.

- `every_layers`. Load every n^{th} layer.

  Optional parameter. Default value is 1.

- `from`. Skip all layers before t_{from} second.

  Optional parameter. Default value is 0.

- `to`. Skip all layers after t_{to} second.

  Optional parameter. Default value is $t_{\max}$.

2.5 Height Testing Params

2.5.1 Common part

- `mode`. Mode of height testing. Possible values are:
  - "space",
  - "time".
- `y0`. y_0 in eq. 41 and 44.

  Optional parameter. Default value is 0.

- `y_k`. y_k in eq. 41 and 44.
- `search_n`. Smoothing factor s_n: for a given point x_i program will search particles with $\max y$ in interval $[x_i - hs_n; x_i + hs_n]$, where h is smoothing length.
- `particles_type`. Type of particles to consider. See section 2.3.7.

  Optional parameter. Default value is null (all particles are considered).

2.5.2 Space profile

Calculate space profile (y values) for a given points of time:

$$y_i = \max_i(y([x_i - hs_n; x_i + hs_n])), \tag{39}$$

$$i = \lceil \frac{1}{\delta_0}(\texttt{x\_maxgeom} - \texttt{x\_mingeom}) \rceil. \tag{40}$$

Space profile points are transformed by coefficients and terms:

$$\begin{cases} \bar{x} = x_k(x - x_0), \\ \bar{y} = y_k(y - y_0). \end{cases} \tag{41}$$

- `t`. Array of points in time to calculate time profile. Each point will be separate output file.
- `x0`. x_0 in eq. 41.

  Optional parameter. Default value is 0.

- `x_k`. x_k in eq. 41.

  Optional parameter. Default value is 1.

2.5.3 Time profile

Calculate time profile (y values) for a given points in space:

$$y_i = \max_i(y([x - hs_n; x + hs_n])), \tag{42}$$

$$i = \lceil \frac{1}{\Delta t}(\texttt{simulation\_time} - \texttt{save\_time}) \rceil. \tag{43}$$

Time profile points are transformed by coefficients and terms:

$$\begin{cases} \bar{t} = t_k(t - t_0), \\ \bar{y} = y_k(y - y_0). \end{cases} \tag{44}$$

- `x`. Array of points in space to calculate time profile. Each point will be separate output file.
- `t0`. t_0 in eq. 44.

  Optional parameter. Default value is 0.
- `t_k`. t_k in eq. 44.

  Optional parameter. Default value is 1.