SPHCOFEM

Smoothed particle hydrodynamics coupling finite element method

Luděk Hynčík

https://github.com/sphcofem

Contents

1 Theory

The theoretical aspects of the SPHCOFEM code are summarizes in [2, 3, 4, 5, 6]. The code calculates problems, where smoothed particle hydrodynamics (SPH represented by particles) is coupled to the boundary represented by the finite element method (FEM). So further speaking about particles or domain represent the domains and speaking about boundary nodes (or just nodes) and boundary elemets (or just elements) represents the boudary.

2 User manual

2.1 Simulation run

The simulation is run by typing `sphcofem < input_file.in`, which reads the given input file `input_file.in` directly. The solver prints the simulation status to the standard output defined in the input file.

`input_file.in` is a structured text file with the keywords and variables defined in the next section. Three types of variables are taken into account, strings (`%s`), integers (`%d`) and real numbers (`%f`). The default values are in `[]`.

`input_file.in` can be created either by the particular keywords or using MATLAB script `model_save.m` with predefined variables. There can be commented lines beginning by `$`. Character `$` must not be within the function definition section. An existing `input_file.in` can be read using MATLAB script `model_read.m` into the predefined variables.

`sphcofem < input_file.in > output_file.txt` can also save the simulation status to the text file `output_file.txt`, if the standard output is the screen. The simulation results are save to the binary file `output_file.out`.

The basic check on variable consitency and correctness is done at the beginning. If there is an error, it is written to the text message file `output_file.msg`. If there are no initialization errors, the message file `output_file.msg` consists of warnings. The message file is closed after initialization and all further information is written to the standard output. The text file `output_file.log` is being created to summarize the memory integrity.

Simulation can be stopped by copying the signal file `signal` to the simulation folder. Based on the existence of the signal file `signal`, the solver stops the simulation, deletes the signal file `signal` and saves the current state.

2.2 Input file

2.2.1 File name

Keyword `NAME` defines the name of the output file (`%s`). The results are written to the binary file `NAME.out`.

Variable description:

Keyword	Variable
	%s
`NAME`	`name`

Keyword definition:

```
NAME %s
```

MATLAB definition:

```
name = %s;
```

Keyword `NAME` is not necessary in the input file. The default value is:

Variable	`name`
Default value	`[sphcofem]`

2.2.2 Dimension

Keyword `DIM` defines dimensionality of the problem `dim` (`%d`).

Variable description:

Keyword	Variable
	%d
`DIM`	`dim`

Keyword description:

```
DIM %d
```

MATLAB description:

```
dim = %d;
```

Keyword `DIM` is not necessary in the input file. Input `0` for `dim`means a default value. The default value is:

Variable	`dim`
Default value	[3]*

*The solver runs in 3D by default. However, converting problems into 1D or 2D considerably decreases the calculation time comparing to 3D due to the nearest neigbour search in lower dimension. Then, the correct dimension `dim` must be set in order to have correct kernel and smoothing length. 1D problem setting expects user input in the x-axis and 2D problem setting expects expect user input in the x-axis and the y axis. For 1D and 2D problems, the input related to the remaining second and/or third dimension(s) is ignored (set to zero by default) and not saved to the output file.

2.2.3 Termination time

Keyword `TMAX` defines the termination time of the dynamical analysis `t_max` (`%f`).

Variable description:

Keyword	Variable
	%f
`TMAX`	`t_max`

Keyword description:

```
TMAX %f
```

MATLAB description:

```
tnax = %f;
```

Keyword `TMAX` is necessary in the input file.

2.2.4 Time controls

Keyword `TIME` defines the time interval between two saved states `dt_save` (`%f`), the initial time step `dt_init` (`%f`), the maximum time step `dt_max` (`%f`), the Courant number for stabilazing the SPH calculation time step `cour` (`%f`) and the stabilizing coefficient for finite element time step `kstab` (`%f`).

Variable description:

Keyword	Variables				
	%f	%f	%f	%f	%f
TIME	dt_save	dt_init	dt_max	cour	kstab

Keyword description:

```
TIME %f %f %f %f %f
```

MATLAB description:

```
time = [%f, %f, %f, %f, %f];
```

Keyword `TIME` is not necessary in the input file. Input `0` for `dt_init`, `cour` or `kstab` means default values. The defaule values are:

Variable	dt_save	dt_init	dt_max	cour	kstab
Default value	[0.0]*	[0.1]	[0.0]**	[0.9]	[0.9]

*[0.0] means each state is saved.

**[0.0] means no upper bound on the time step.

2.2.5 Global acceleration field

Keyword `GACC` defines the global acceleration field `ax` (`%f`), `ay` (`%f`) and `az` (`%f`) acting to the whole model.

Variable description:

Keyword	Variables		
	%f	%f	%f
`GACC`	`ax`	`ay`	`az`

Keyword description:

```
GACC %f %f %f
```

MATLAB description:

```
gacc = [%f, %f, %f];
```

Keyword `GACC` is not necessary in the input file. Default values are:

Variable	`ax`	`ay`	`az`
Default value	[0.0]	[0.0]	[0.0]

The keyword `GACC` can be used to avoid adding the acceleration field to each particle or node.

2.2.6 Calculation optimization

Keyword `OPTIM` provides additional parameters to optimize the SPH calculation. It defines the switch, if the data will be checked `data_check` (`%d`), the switch, if the output will be printed `data_print` (`%d`), the step between integration cycles output `cycle_print` (`%d`), the nearest neighbour search (NNS) model `nnopt` (`%d`), the radius multiplier for the NNS `opt` (`%f`), the number of cycles between consequent contact search `cycle_contact` (`%d`), the integration scheme `integration` (`%d`) and the memory check option `mem_check` (`%d`).

Variable description:

Keyword	Variables				
	%d	%d	%d	%d	%d
	%d	%d	%d		
`OPTIM`	`data_check`	`data_print`	`cycle_print`	`nnopt`	`opt`
	`cycle_contact`	`integration`	`mem_check`		

Keyword description:

```
OPTIM %d %d %d %d %f %d %d %d
```

MATLAB description:

```
optim = [%d, %d, %d, %d, %f, %d, %d, %d];
```

Keyword `OPTIM` is not necessary in the input file. Input `0.0` for `opt` means a default value. Default values are:

Variable	Value	Description	Remark
`data_check`	0	Inactive	—
	[1]	Active	
`data_print`	0	Output suppressed	—
	[1]	Standard output	
	2	Text file output*	
`cycle_print`	0	Printing suppressed	—
	`N`	Each `N` cycles	Default `N=1`
`nnopt`	[0]	No NNS clustering	`opt` ignored
	1	NNS clustering at the beginning	Default `opt`=1.0
	`N`	NNS clustering each `N` cycles	
`cycle_contact`	`N`	Contact calculation each `N` cycles	Default `N=1`
`integration`	0	Euler method	—
	[1]	Central acceleration	
	2	Predictor-corrector	
	3	Predictor-corrector leapfrog	
`mem_check`	0	Inactive	—
	[1]	Active	

*The text output file is `NAME.txt`.

2.2.7 SPH definitions

Keyword `SPH` defines the parameters for the smoothed particle hydrodynamics (SPH) method. It defines the viscosity model `visc` (`%d`), the artificial viscosity parameters (`alpha`) (`%f`) and (`beta`) (`%f`), the (`eta`) (`%f`) parameter, the artificial stress parameters (`zeta`) (`%f`), (`nas`) (`%f`) and (`theta`) (`%f`), the X-SPH option (`xsph`) (`%d`) and the X-SPH coefficient (`xeps`) (`%f`).

Variable description:

Keyword	Variables								
	%d	%f	%f	%f	%f	%f	%f	%d	%f
SPH	visc	alpha	beta	eta	zeta*	nas*	theta*	xsph	xeps

*Artificial stress parameters are ignored, if material models 7, 8, 9 or 27 are not present.

Keyword description:

```
SPH %d %f %f %f %f %f %f %d %f
```

MATLAB description:

```
sph = [%d, %f, %f, %f, %f, %f, %f, %d, %f];
```

Keyword `SPH` is not necessary in the input file. Default values are:

Variable	visc	alpha	beta	eta	zeta	nas	theta	xsph	xeps
Default value	[0]	[1.2]	[1.5]	[0.1]	[0.3]	[4]	[0.01]	[0]	[0.5]

Variable possible values:

Variable	Value	Description	Remark
Viscosity models			
`visc`	[0]	Artificial viscosity only	Always active
Second order viscous term			
`visc`	1	Kernel second derivative	—
Viscosity approximations			
`visc`	2	Monaghan, Cleary, Gingold (2006)	[8, 1, 7]
	3	Morris et al. (1997)	[9, 7]
	4	Takeda et al. (1994)	[11, 7]
	5	Onderik et al. (2007)	[10, 7]
	6	Monaghan and Gingold (1983)	[8, 7]
XSPH			
`xsph`	0	Inactive	—
	1	Active only for predictor step	
	2	Active only for corrector step	
	3	Active for both predictor and corrector steps	

2.2.8 FEM definitions

Keyword `FEM` provided additional parameters for the finite element (FEM) method. It concerns material damping parameters `c0` and `c1`.

Variable description:

Keyword	Variables	
	%f	%f
`FEM`	`c0`	`c1`

Keyword description:

```
FEM %f %f
```

MATLAB description:

```
fem = [%f, %f];
```

Keyword `FEM` is not necessary in the input file. Default values are:

Variable	`c0`	`c1`
Default value	[0.0]	[0.0]

2.2.9 Saving variables

Keyword `SAVE` defines what variables are save to the output file `FILE.out`. Time `t` and positions of all particles `xs`, `ys`, `zs` and all boundary nodes `xb`, `yb`, `zb` are saved in default (if the particles and/or nodes exist).

Variable description:

	Variables										
Keyword	%d	%d	%d	%d	%d	%d	%d	%d	%d		
	%d	%d	%d	%d	%d	%d	%d	%d	%d	%d	%d
	%d	%d	%d	%d	%d						
	%d	%d	%d	%d							
	%d	%d	%d	%d	%d	%d	%d	%d			
SAVE	dt	kines	innes	potes	kineb	disib	defob	poteb	tote		
	vs	dvs	as	fs	rhos	drhos	us	dus	ps	cs	hs
	es	des	Os	Ds	dDs						
	vb	ab	fb	Fc							
	xr	psir	vr	or	ar	alphar	fr	Mr			

Keyword description:

```
SAVE %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d
%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d
```

MATLAB description:

```
saver = [%d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d,
%d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d,
%d, %d, %d, %d, %d, %d];
```

Keyword `SAVE` is not necessary in the input file. Using `saver` is MATLAB is necesssary due to tthe collision with a standard MATLAB keyword. Default values are:

Quantity	**Variable**		**Value**	**Saving**
	Solver	**MATLAB**		
Time step	`save_dt`	`dt`	[0]	Inactive
			1	Active

Quantity	Variable		Value	Saving
	Solver	MATLAB		
Particles kinetic energy	save_kine_s	`kines`	[0]	Inactive
			1	Active
Particles internal energy	save_inne_s	`innes`	[0]	Inactive
			1	Active
Particles potential energy	save_pote_s	`potes`	[0]	Inactive
			1	Active

Quantity	Variable		Value	Saving
	Solver	MATLAB		
Nodal kinetic energy	save_kine_b	`kineb`	[0]	Inactive
			1	Active
Nodal dissipation energy	save_inne_b	`inneb`	[0]	Inactive
			1	Active
Elements deformation energy	save_defo_b	`defob`	[0]	Inactive
			1	Active
Elements potential energy	save_pote_b	`poteb`	[0]	Inactive
			1	Active

Quantity	Variable		Value	Saving
	Solver	MATLAB		
Total energy	save_tote	`tote`	[0]	Inactive
			1	Active

Quantity	Variable		Value	Saving
	Solver	MATLAB		
Particles velocities	save_v_s	`vs`	[0]	Inactive
			1	Active
Particles velocities derivative	save_dv_s	`dvs`	[0]	Inactive
			1	Active
Particles accelerations	save_a_s	`as`	[0]	Inactive
			1	Active
Particles forces	save_f_s	`fs`	[0]	Inactive
			1	Active

Quantity	Variable		Value	Saving
	Solver	MATLAB		
Particles density	save_rho_s	`rhos`	[0]	Inactive
			1	Active
Particles density derivatives	save_drhodt_s	`drhos`	[0]	Inactive
			1	Active
Particles internal energy	save_u_s	`us`	[0]	Inactive
			1	Active
Particles internal energy derivative	save_dudt_s	`dus`	[0]	Inactive
			1	Active
Particles pressure	save_p_s	`ps`	[0]	Inactive
			1	Active
Particles sound speed	save_c_s	`cs`	[0]	Inactive
			1	Active
Particles smoothing length	save_h_s	`hs`	[0]	Inactive
			1	Active

Quantity	Variable		Value	Saving
	Solver	MATLAB		
Particles deformation	save_e_s	`es`	[0]	Inactive
			1	Active
Particles deformation rate	save_dedt_s	`des`	[0]	Inactive
			1	Active
Particles rotation	save_O_s	`Os`	[0]	Inactive
			1	Active
Particles deviatoric stress	save_S_s	`Ds`	[0]	Inactive
			1	Active
Particles deviatoric stress derivative	save_dSdt_s	`dDs`	[0]	Inactive
			1	Active

Quantity	Variable		Value	Saving
	Solver	MATLAB		
Nodal velocities	save_v_b	`vb`	[0]	Inactive
			1	Active
Nodal accelerations	save_a_b	`ab`	[0]	Inactive
			1	Active
Nodal forces	save_f_b	`fb`	[0]	Inactive
			1	Active

Quantity	Variable		Value	Saving
	Solver	MATLAB		
Rigid bodies positions	save_x_r	`xr`	[0]	Inactive
			1	Active
Rigid bodies rotations	save_psi_r	`psir`	[0]	Inactive
			1	Active
Rigid bodies velocities	save_v_r	`vr`	[0]	Inactive
			1	Active
Rigid bodies rotational velocities	save_omega_r	`omegar`	[0]	Inactive
			1	Active
Rigid bodies accelerations	save_a_r	`ar`	[0]	Inactive
			1	Active
Rigid bodies rotational acceleration	save_alpha_r	`alphar`	[0]	Inactive
			1	Active
Rigid bodies forces	save_f_r	`fr`	[0]	Inactive
			1	Active
Rigid bodies moments	save_M_r	`Mr`	[0]	Inactive
			1	Active

Quantity	Variable		Value	Saving
	Solver	MATLAB		
Contact force	save_f_c	`fc`	[0]	Inactive
			1	Active

2.2.10 Functions

Keyword `FUNCT` defines the function $y = f(x)$. The first line comsists of the function number `num`, the number of function data pairs `N`, the x-multiplier `mx`, the y-multiplier, `my`, the x-shift `dx` and the y-shift `dy`. `N` lines defining the function data pairs `xi` and `xi`, $i \in \{1, \ldots, N\}$ follows.

Variable description:

Keyword	Variables					
	%d	%d	%f	%f	%f	%f
`FUNCT`	`num`	`N`	`mx`	`my`	`dx`	`dy`

%f	%f
x1	y1
x2	y2
...	...
xN	yN

Keyword definition (for more functions repeat the function section):

```
FUNCT %d %d %f %f %f %f
%f %f
%f %f
...
%f %f
FUNCT %d %d %f %f %f %f
%f %f
%f %f
...
%f %f
...
FUNCT %d %d %f %f %f %f
%f %f
%f %f
...
%f %f
```

MATLAB definition (for more material models add lines the material matrix):

```
funct = [%d, %d;
%f, %f;
%f, %f;
%f, %f;
%f, %f;
...;
%f, %f;
%d, %d;
%f, %f;
%f, %f;
%f, %f;
%f, %f;
...;
%f, %f;
...;
%f, %f;
%d, %d;
%f, %f;
%f, %f;
%f, %f;
%f, %f;
...;
%f, %f];
```

Keyword `FUNCT` is necessary in the input file.

2.2.11 Materials

Keyword `MATER` defines the constitutive equation for the implemented material models. The variables depends on the particular material model.

Variable description:

Keyword	Variables											
	%d	%d	%d	%f	%f	%f	%f	%f	%f	%f	%f	%f
`MATER`	`num`	`dom`	`type`	`rho`	`mu`	`T`	`kap`	`gam`	`aux1`	`aux2`	`aux3`	`aux4`

Keyword definition (for more material models repeat the material line):

```
MATER %d %d %d %f %f %f %f %f %f %f %f %f
MATER %d %d %d %f %f %f %f %f %f %f %f %f
...
MATER %d %d %d %f %f %f %f %f %f %f %f %f
```

MATLAB definition (for more material models add lines the material matrix):

```
mater = [%d, %d, %d, %f, %f, %f, %f, %f, %f, %f, %f, %f;
%d, %d, %d, %f, %f, %f, %f, %f, %f, %f, %f, %f;
...;
%d, %d, %d, %f, %f, %f, %f, %f, %f, %f, %f, %f]
```

Keyword `MATER` is necessary in the input file.

Materials implemented:

Dimension	**Type**	**Constitutive equation**	**Remark**
Rigid body			
1D, 2D, 3D	0	Rigid body	Particles and nodes can be combined in single material
Fluid equation of state (EOS)			
1D, 2D, 3D	1	Gas EOS	—
	2	Liquid EOS	Material bulk modulus K
	3	Liquid EOS (SPH)	$K = \rho c^2$
Linear elastic solid FEM (elements with nodes N_1, N_2, N_3, N_4)			
1D	4, 5, 6	Mass point	Takes N_1 into account
2D	4	Bar	Takes N_1 and N_2 into account
	5	Beam	Takes N_1 and N_2 into account
	6	Triangle	$N_4 = N_3$
		Rectangle	—
3D	4	Bar	*Not implemented*
		Triangle membrane	*Only for contact*
		Rectangle membrane	*Not implemented*
	5	Beam	*Not implemented*
		Triangle shell	*Only for contact*
		Rectangle shell*	*Not implemented*
	6	Tetrahedron	—
Solid SPH			
1D, 2D, 3D	7	Linear elastodynamics (Hookean)	—
	8	Elastodynamics (Neo-Hookean)	*Not implemented*
	9	User-defined	In *user_defined_material.c*
Fluid EOS with tension			
1D, 2D, 3D	12	Type 2 with inter-particle tension	
	13	Type 3 with inter-particle tension	
SPH with Mue-Grüneisen EOS			
1D, 2D, 3D	23	Type 3 with Mue-Grüneisen EOS	
	27	Type 7 with Mue-Grüneisen EOS	

*Not possible for rigid body material type 0. For rigid body material type 0, only triangular

elements are possible as 2D representation in 3D.

Variable constitutive parameters (— means that the particular variable is not taken into account):

MATER										
Type	`dom`	`rho`	`mu`	`T`	`kappa`	`gamma`	`aux1`	`aux2`	`aux3`	`aux4`
%d	%d	%f	%f	%f	%f	%f	%f	%f	%f	%f
Rigid body										
0	d	ρ	G	K	—	γ	COG	N_1	N_2	N_3
	d	$-\rho$	m	I_1	I_2	I_3				
Fluid SPH										
1	d	ρ	μ	T	κ	c_V	—	—	—	—
2	d	ρ	μ	K	p_0	γ	—	—	—	—
3	d	ρ	μ	—	p_0	γ	—	—	—	—
Solid FEM										
4	d	ρ	ν	E	A^* (2D, 3D)	b	—	—	—	—
					t^{**} (3D)					
5	d	ρ	ν	E	A^* (2D, 3D)	b	—	—	—	—
					t^{**}					
6	d	ρ	ν	E	A^* (2D, 3D)	b	—	—	—	—
					t^{***} (2D)		—	—	—	—
					— (tertahedron)		—	—	—	—
Solid SPH										
7	d	ρ	G	K	p_0	γ	—	—	—	—
8	d	ρ	—	—	—	—	—	—	—	—
9	d	ρ	c_1^{****}	c_2	c_3	c_4	c_5	c_6	c_7	c_8
Fluid SPH with tension										
12	d	ρ	μ	K	k	γ	—	—	—	—
13	d	ρ	μ	—	k	γ	—	—	—	—
SPH with Mue-Grüneisen EOS										
23	d	ρ	μ	K	p_0	—	s	Γ_0	—	—
27	d	ρ	μ	K	p_0	—	s	Γ_0	—	—

*Cross-sectional area A for bar or beam element in 2D and 3D ($N_2 = N_3 = N_4$).

**Thickness of memrane or shell element in 3D (triangle defined by $N_3 = N_4$).

***Thickness of triangle or rectangle element in 2D (triangle defined by $N_3 = N_4$).

****c = user coefficient.

Variable `dom` represent possibility to divide the model into several domains $d \in \{1, 2, 3, \ldots\}$. Particles, nodes and elements in different domains do not interact (either by smoothing or by contacts).

2.2.12 Contacts

Keyword `CONTACT` defines the contact between two materials. It concerns the contact `number` (`%d`), the contact `type` (`%d`), the `slave`* material number (`%d`), the `master`* material number (`%d`), the contact thickness `ct`** (`%f`), the linear penalty factor `klin` (`%f`), the nonlinear penalty factor `klin` (`%f`), the contact friction coefficient `kf` (`%f`) and the contact damping `kd` (`%f`).

The `slave` material segment motion is driven by the `master` material segment motion.

**For contacts, where particles are active, `ct` can be negative. The contact thickness is then calculated as the negative value of `ct` times the particles smoothing length.

Keyword description (for more contact models repeat the contact line):

```
CONTACT %d %d %d %d %f %f %f %f %f
CONTACT %d %d %d %d %f %f %f %f %f
...
CONTACT %d %d %d %d %f %f %f %f %f
```

MATLAB definition (for more material models add lines the contact matrix):

```
contact = [%d, %d, %d, %d, %f, %f, %f, %f %f;
%d, %d, %d, %d, %f, %f, %f, %f %f;
...
%d, %d, %d, %d, %f, %f, %f, %f %f];
```

Keyword `CONTACT` is not necessary in the input file.

Contacts implemented:

Material		**Segment**	
Slave	**Master**	**Slave**	**Master**
SPH	FEM	Particle	Boundary
FEM	FEM	Node	Boundary
SPH	SPH	Particle	Particle

Contact models for FEM/SPH, FEM/FEM and SPH/SPH contacts:

Dimension	Type	Description	Slave	Master	Remark
1D	0	Null*	Particle	Particle	—
	1	Penalty			
			Node	Particle or node	
	3	Tied			
2D	0	Null	Particle	Particle	
	1	Penalty			
			Particle or node	Line N_1, N_2	Outer normal required
	2	Sliding w/o separation			
	3	Tied			
3D	0	Null	Particle	Particle	—
	1	Penalty			
			Particle or node	Triangles N_1, N_2, N_3 and N_1, N_3, N_4**	Outer normal required
	2	Sliding w/o separation			
	3	Tied			

*Null contact serves for defining area, where no interaction (by contact or by smoothing length interpolation) among particles appears.

**Bar and beam elements in 3D are ignored in the contact calculation.

2.2.13 Particles

Keyword `SNODE` defines the particles. Each row concerns of the particle number `ns` (`%d`), the material number `ms` (`%d`), the represented volume `vs` (`%f`) and the particle coordinates `xs` (`%f`), `ys` (`%f`) and `zs` (`%f`).

Variable description:

Keyword	Variables					
	%d	%d	%f	%f	%f	%f
SNODE	ns	ms	vs	xs	ys	zs

Keyword definition (for more material models repeat the particle line):

```
SNODE %d %d %f %f %f %f
SNODE %d %d %f %f %f %f
...
SNODE %d %d %f %f %f %f
```

MATLAB definition (for more material models add lines the particle matrix):

```
snode = [%d, %d, %f, %f, %f, %f, %f;
%d, %d, %f, %f, %f, %f, %f;
...
%d, %d, %f, %f, %f, %f, %f];
```

The particles numbering must be complementary to the boundary nodes numbering. At least one keyword `SNODE` or `BNODE` is necessary in the input file.

2.2.14 Nodes

Keyword `BNODE` defines the boundary nodes. Each row concerns of the nodal number `nb` (`%d`) and the nodal coordinates `xb` (`%f`), `yb` (`%f`) and `zb` (`%f`).

Variable description:

Keyword	Variables			
	%d	%f	%f	%f
BNODE	nb	xb	yb	zb

Keyword definition (for more material models repeat the node line):

```
BNODE %d %f %f %f
BNODE %d %f %f %f
...
BNODE %d %f %f %f
```

MATLAB definition (for more material models add lines the node matrix):

```
bnode = [%d, %f, %f, %f;
%d, %f, %f, %f;
...
%d, %f, %f, %f];
```

The boundary nodes numbering must be complementary to the particles numbering. At least one keyword `SNODE` or `BNODE` is necessary in the input file.

2.2.15 Elements

Keyword `BELEM` defines the boundary finite elements. Each row concerns of the element number `ne` (`%d`), the element material number `me` (`%d`) and four nodes `n1` (`%d`), `n2` (`%d`), `n3` (`%d`) and `n4` (`%d`). For bars and beams, `n3` and `n4` are not taken into account. If `n3` equals `n4`, the triangular element is considered in 2D.

Variable description:

Keyword	Variables					
	%d	%d	%d	%d	%d	%d
BELEM	n	m	n1	n2	n3	n4

Keyword definition (for more material models repeat the element line):

```
BELEM %d %d %d %d %d %d
BELEM %d %d %d %d %d %d
...
BELEM %d %d %d %d %d %d
```

MATLAB definition (for more material models add lines the element matrix):

```
snode = [%d, %d, %d, %d, %d, %d;
%d, %d, %d, %d, %d, %d;
...
%d, %d, %d, %d, %d, %d];
```

Keyword `BELEM` is not necessary in the input file.

2.2.16 Initial pressure

Keyword `INPRE` defines the initial pressure for particles. Each row concerns of the particle number `n` (`%d`) and the initial pressure `ps` (`%f`).

Variable description:

Keyword	Variables	
	%d	%f
INPRE	n	ps

Keyword definition (for more material models repeat the initial pressure line):

```
INPRE %d %f
INPRE %d %f
...
INPRE %d %f
```

MATLAB definition (for more material models add lines the initial pressure matrix):

```
inpre = [%d, %f;
%d, %f;
...
%d, %f];
```

Keyword `INPRE` is not necessary in the input file.

2.2.17 Initial velocity

Keyword `INVEL` defines the initial velocity for both particles and boundary nodes. Each row concerns of the particle or nodal number `n` (`%d`) and the translational velocities `vx` (`%f`), `vy` (`%f`) and `vz` (`%f`) and the rotational velocities `ox` (`%f`), `oy` (`%f`) and `oz` (`%f`) in the coordinate system frame `frame` (`%d`). Rotational velocities `ox`, `oy`, `oz` and `frame` are ignored if the particle or the node, on which the initial velocity is specified, is not a centre of gravity of a rigid body.

Variable description:

Keyword	Variables							
	%d	%f	%f	%f	%f	%f	%f	%d
INVEL	n	vx	vy	vz	ox	oy	oz	frame

Keyword definition (for more material models repeat the initial velocity line):

```
INVEL %d %f %f %f %f %f %f %d
INVEL %d %f %f %f %f %f %f %d
...
INVEL %d %f %f %f %f %f %f %d
```

MATLAB definition (for more material models add lines the initial velocity matrix):

```
invel = [%d, %f, %f, %f, %f, %f, %f, %d;
%d, %f, %f, %f, %f, %f, %f, %d;
...
%d, %f, %f, %f, %f, %f, %f, %d];
```

Keyword `INVEL` is not necessary in the input file.

2.2.18 Acceleration field

Keyword `ACFLD` defines the acceleration field for both particles and boundary nodes. Each row concerns the particle or nodal number `n` (`%d`), the acceleration field type `type` (`%d`) and the accelerations `ax` (`%f`), `ay` (`%f`) and `az` (`%f`). Coordinate `frame` is ignored if the particle or the node, on which the initial acceleration field is specified, is not a centre of gravity of a rigid body.

Variable description:

Keyword	Variables					
	%d	%d	%f	%f	%f	%d
ACFLD	n	type	ax	ay	az	frame

Keyword definition (for more material models repeat the acceleration field line):

```
ACFLD %d %d %x %x %x %d
ACFLD %d %d %x %x %x %d
...
ACFLD %d %d %x %x %x %d
```

MATLAB definition (for more material models add lines the acceleration field matrix):

```
acfld = [%d, %d, %x, %x, %x;
%d, %d, %x, %x, %x, %d;
...
%d, %d, %x, %x, %x, %d];
```

Keyword `ACFLD` is not necessary in the input file.

Variable possible values:

Description	**Variable**	**Type**	**F**	**Status**
x-acceleration	ax	0	f	Constant a_x = ax
		1	d	Function ax prescribing $a_x(t)$
y-acceleration	ay	0	f	Constant a_y = ay
		1	d	Function ay prescribing $a_y(t)$
z-acceleration	az	0	f	Constant a_z = az
		1	d	Function az prescribing $a_z(t)$

2.2.19 Nodal force

Keyword `FORCE` defines the force acting on the nodes for both particles and boundary nodes. Each row concerns the particle or nodal number `n` (`%d`), the acceleration field type `type` (`%d`) and the forces `fx` (`%f`), `fy` (`%f`) and `fz` (`%f`) and the moments `fx` (`%f`), `fy` (`%f`) and `fz` (`%f`) and the moments `mx` (`%f`), `my` (`%f`) and `mz` (`%f`) in the coordinate system frame `frame` (`%d`). Moments `mx`, `my`, `mz` and `frame` are ignored if the particle or the node, on which the nodal force is specified, is not a centre of gravity of a rigid body.

Variable description:

Keyword	Variables								
	%d	%d	%f	%f	%f	%f	%f	%f	%d
FORCE	n	type	fx	fy	fz	mx	my	mz	frame

Keyword definition (for more material models repeat the nodal force field line):

```
FORCE %d %f %f %f %f %f %f %d
FORCE %d %f %f %f %f %f %f %d
...
FORCE %d %f %f %f %f %f %f %d
```

MATLAB definition (for more material models add lines the nodal force field matrix):

```
force = [%d, %f, %f, %f, %f, %f, %f, %d;
%d, %f, %f, %f, %f, %f, %f, %d;
...
%d, %f, %f, %f, %f, %f, %f, %d];
```

Keyword `FORCE` is not necessary in the input file.

Variable possible values:

Description	**Variable**	**Type**	**F**	**Status**
x-force	`fx`	0	`f`	Constant $f_x =$ `fx`
		1	`d`	Function `ax` prescribing $f_x(t)$
y-force	`fy`	0	`f`	Constant $f_y =$ `fy`
		1	`d`	Function `ay` prescribing $f_y(t)$
z-force	`fz`	0	`f`	Constant $f_z =$ `fz`
		1	`d`	Function `az` prescribing $f_z(t)$
x-moment	`mx`	0	`m`	Constant $m_x =$ `mx`
		1	`d`	Function `ax` prescribing $m_x(t)$
y-moment	`my`	0	`m`	Constant $m_y =$ `my`
		1	`d`	Function `ay` prescribing $m_y(t)$
z-moment	`mz`	0	`m`	Constant $m_z =$ `mz`
		1	`d`	Function `az` prescribing $m_z(t)$

2.2.20 Nodal damping

Keyword `NDAMP` defines the damping of the boundary nodes. Each row concerns of the nodal number `n` (`%d`) and the damping `dx` (`%f`), `dy` (`%f`) and `dz` (`%f`).

Variable description:

Keyword	Variables			
	%d	%f	%f	%f
NDAMP	n	fx	fy	fz

Keyword definition (for more material models repeat the nodal damping field line):

```
NDAMP %d %f %f %f
NDAMP %d %f %f %f
...
NDAMP %d %f %f %f
```

MATLAB definition (for more material models add lines the nodal damping field matrix):

```
ndamp = [%d, %f, %f, %f;
%d, %f, %f, %f;
...
%d, %f, %f, %f];
```

Keyword `NDAMP` is not necessary in the input file.

2.2.21 Boundary conditions

Keyword `BOUNC` defines the boundary conditions for both particles and boundary nodes. Each row concerns of the particle or nodal number `n` (`%d`), the boundary condition type `type` (`%d`), and the conditions `bx` (`%d`), `by` (`%d`) and `bz` (`%d`). Further, rotational boundary conditions for rigid body motion `rx` (`%d`), `ry` (`%d`) and `bz` (`%d`) in the coordinate system frame `frame` (`%d`). Boundary conditions `rx`, `ry`, `bz` and `frame` are ignored if the particle or the node, on which the boundary condition is specified, is not a centre of gravity of a rigid body.

Variable description:

Keyword	Variables								
	%d	%d	%d	%d	%d	%d	%d	%d	%d
BOUNC	n	type	bx	by	bz	rx	ry	rz	frame

Keyword definition (for more material models repeat the boundary condition line):

```
BOUNC %d %d %d %d %d %d %d %d %d
BOUNC %d %d %d %d %d %d %d %d %d
...
BOUNC %d %d %d %d %d %d %d %d %d
```

MATLAB definition (for more material models add lines the boundary condition matrix):

```
bounc = [%d, %d, %d, %d, %d, %d, %d, %d, %d;
%d, %d, %d, %d, %d, %d, %d, %d, %d;
...
%d, %d, %d, %d, %d, %d, %d, %d, %d];
```

Keyword `BOUNC` is not necessary in the input file.

Boundary conditions implemented:

Type	**Boundary condition**	**Remark**
0	Static	Restricted motion
-1*		Keeping translational initial conditions
1	Displacement	Prescribed nodal displacement
2	Velocity	Prescribed nodal velocity
3	Acceleration	Prescribed nodal acceleration

*Boundary condition type -1 moves the particles or nodes using their initial conditions. If the initial condition is not given, it is the same as boundary condition type 0.

Variable possible values:

Description	Variable	Value	Status
Static			
x-translation	bx	0	Free
		1	Fixed
y-translation	by	0	Free
		1	Fixed
z-translation	bz	0	Free
		1	Fixed
Prescribed nodal displacement			
x-translation	bx	0	Free
		-1	Fixed
		$\mathtt{N}_1$	Function $\mathtt{N}_1$ prescribing $x(t)$
y-translation	by	0	Free
		-1	Fixed
		$\mathtt{N}_2$	Function $\mathtt{N}_2$ prescribing $y(t)$
z-translation	bz	0	Free
		-1	Fixed
		$\mathtt{N}_3$	Function $\mathtt{N}_3$ prescribing $z(t)$
Prescribed nodal velocity			
x-translation	bx	0	Free
		-1	Fixed
		$\mathtt{N}_1$	Function $\mathtt{N}_1$ prescribing $v_x(t)$
y-translation	by	0	Free
		-1	Fixed
		$\mathtt{N}_2$	Function $\mathtt{N}_2$ prescribing $v_y(t)$
z-translation	bz	0	Free
		-1	Fixed
		$\mathtt{N}_3$	Function $\mathtt{N}_3$ prescribing $v_z(t)$
Prescribed nodal acceleration			
x-translation	bx	0	Free
		-1	Fixed
		$\mathtt{N}_1$	Function $\mathtt{N}_1$ prescribing $a_x(t)$
y-translation	by	0	Free
		-1	Fixed
		$\mathtt{N}_2$	Function $\mathtt{N}_2$ prescribing $a_y(t)$
z-translation	bz	0	Free
		-1	Fixed
		$\mathtt{N}_3$	Function $\mathtt{N}_3$ prescribing $a_z(t)$

Description	**Variable**	**Value**	**Status**
Static			
x-rotation	`rx`	0	Free
		1	Fixed
y-rotation	`ry`	0	Free
		1	Fixed
z-rotation	`rz`	0	Free
		1	Fixed
Prescribed nodal displacement			
x-rotation	`rx`	0	Free
		-1	Fixed
		$\mathtt{N}_1$	Function $\mathtt{N}_1$ prescribing $\psi(t)$
y-rotation	`ry`	0	Free
		-1	Fixed
		$\mathtt{N}_2$	Function $\mathtt{N}_2$ prescribing $\theta(t)$
z-rotation	`rz`	0	Free
		-1	Fixed
		$\mathtt{N}_3$	Function $\mathtt{N}_3$ prescribing $\phi(t)$
Prescribed nodal velocity			
x-rotation	`rx`	0	Free
		-1	Fixed
		$\mathtt{N}_1$	Function $\mathtt{N}_1$ prescribing $\omega_x(t)$
y-rotation	`ry`	0	Free
		-1	Fixed
		$\mathtt{N}_2$	Function $\mathtt{N}_2$ prescribing $\omega_y(t)$
z-rotation	`rz`	0	Free
		-1	Fixed
		$\mathtt{N}_3$	Function $\mathtt{N}_3$ prescribing $\omega_z(t)$
Prescribed nodal acceleration			
x-rotation	`rx`	0	Free
		-1	Fixed
		$\mathtt{N}_1$	Function $\mathtt{N}_1$ prescribing $\alpha_x(t)$
y-rotation	`ry`	0	Free
		-1	Fixed
		$\mathtt{N}_2$	Function $\mathtt{N}_2$ prescribing $\alpha_y(t)$
z-rotation	`rz`	0	Free
		-1	Fixed
		$\mathtt{N}_3$	Function $\mathtt{N}_3$ prescribing $\alpha_z(t)$

2.2.22 Local coordinate system

Initial condition `INVEL` and boundary conditions `ACFLD`, `FORCE` and `BOUNC` can be defined in local coordinate system of a rigid body, if those are applied to the rigid body centre of gravity. The local coordinate system is defined by `frame` (`%d`) with the following possible values:

Description	**Variable**	**Value**	**Status**
Coordinate system	`frame`	0	Global axes system
		1	Local connected to the rigid body principal inertia axes system only for translation
		2	Local connected to the rigid body principal inertia axes system only for rotation
		3	Local connected to the rigid body principal inertia axes system for translation and rotation

2.2.23 Rigid body

Rigid body is given by material type 0. All particles and nodes belonging to elements from material type 0 are included to a rigid body. Rigid body number is allocated automatically beginning from 1. For rigid body parameters see the material section. Two types of rigid bodies are implemented:

0. For rigid body type 0 (material $\rho > 0$), the mass and moments of inertia are calculated from the mass distribution among the particles and nodes affiliated to the rigid body. Centre of gravity COG node or particle coordinates are calculated from the mass distribution too and particles or nodes N_1, N_2 and N_3 coordinates are set to be coincident with the global coordinate system in time $t = 0$. Original nodes or particles coordinates COG, N_1, N_2 and N_3 are rewritten.

1. For rigid body type 1 (material $\rho < 0$), the mass m and principal moments of inertia I_1, I_2 and I_3 are given and ρ is used only for nearest neighbour smoothing. Nodes or particles N_1, N_2 and N_3 are expected to define the principal inertia axes from the centre of gravity node or particle COG and must form an orthogonal axes system.

Material	Type	`rho`	`mu`	`T`	`kappa`	`gamma`	`aux1`	`aux2`	`aux3`	`aux4`
Rigid body										
0	0	ρ	G	K	p_0	γ	Calculated and rewritten			
	1	$-\rho$	m	I_1	I_2	I_3	COG	N_1	N_2	N_3

The rigid body type 0 constraints the particles and nodes in a similar way as boundary conditions with the advantage that initial and boudary conditions apply only on its centre of gravity. Particles constrained in rigid body type 0 hold the rigid body motion updating state variables including stress and strain. For particles constrained in the rigid body type 1, the state variables including stress and strain are not updated and the rigid body keeps the initial date variables of the particles involved. Nodes constrained in rigid body type 0 or 1 hold the rigid body motion.

2.3 Output file

The results are saved in `FILE.out` based on saved variables defined by `SAVE`. The results can be read and displayed by MATLAB scripts `results_read.m` and `results_view.m`.

The beginning of `FILE.out` file consists of all global definitions and counts followed by the variables saved at the defined time steps. The order of appearance is according to the following tables.

Quantity	**Solver**			**MATLAB**		
	Variable	**Size***	**Format**	**Variable**	**Size**	**Format**
Dimension	`dim`	`1`	`integer`	`dim`	`1`	`integer*4`
X-SPH switch	`ix`	`1`	`integer`	`ix`	`1`	`integer*4`
Number of materials	`n_m`	`1`	`integer`	`n_m`	`1`	`integer*4`
Number of particles	`n_s`	`1`	`integer`	`n_s`	`1`	`integer*4`
Number of nodes	`n_b`	`1`	`integer`	`n_b`	`1`	`integer*4`
Number of elements	`n_e`	`1`	`integer`	`n_e`	`1`	`integer*4`
Number of rigid bodies	`n_r`	`1`	`integer`	`n_e`	`1`	`integer*4`
Number of contacts	`n_c`	`1`	`integer`	`n_c`	`1`	`integer*4`
Material numbers	`num_m`	`[1, n_m]`	`integer`	`num_m`	`[1, n_m]`	`integer*4`
Particle numbers	`num_s`	`[1, n_s]`	`integer`	`num_s`	`[1, n_s]`	`integer*4`
Nodal numbers	`num_b`	`[1, n_b]`	`integer`	`num_b`	`[1, n_b]`	`integer*4`
Element numbers	`num_e`	`[1, n_e]`	`integer`	`num_e`	`[1, n_e]`	`integer*4`
Particle materials	`mat_s`	`[1, n_s]`	`integer`	`mat_s`	`[1, n_s]`	`integer*4`
Elements materials	`mat_e`	`[1, n_e]`	`integer`	`mat_e`	`[1, n_e]`	`integer*4`
Element nodes	`nod_e`	`[1, 4*n_e]`	`integer`	`bele`	`[n_e, 4]`	`integer*4`
Saved variables switch	`variable`**	`1`	`integer`	`saver`	`[1, 37]`	`integer*4`

*`1` means a scalar, `[m, n]` means a matrix of size size $(m \times n)$.

**`variable` represent 37 variables according to the definition in the `SAVE` keyword, particularly `save_dt`, `save_kine_s`, `save_inne_s`, `save_pote_s`, `save_kine_b`, `save_disi_b`, `save_defo_b`, `save_pote_b`, `save_tote`, `save_v_s`, `save_dv_s`, `save_a_s`, `save_f_s`, `save_rho_s`, `save_drhodt_s`, `save_u_s`, `save_dudt_s`, `save_p_s`, `save_c_s`, `save_h_s`, `save_S_s`, `save_dSdt_s`, `save_e_s`, `save_dedt_s`, `save_0_s`, `save_v_b`, `save_a_b`, `save_f_b`, `save_x_r`, `save_psi_r`, `save_v_r`, `save_omega_r`, `save_a_r`, `save_alpha_r`, `save_f_r`, `save_M_r` and `save_f_c`. In the MATLAB script, they are represented by a row vectorof of size (1×37)

The particular output variables defined by `SAVE` are stored in the binary file `FILE.out` as a sequence related to the saved time steps. At each defined time step, firstly the scalar variables are saved. Only variables having `1` in `SAVE` are saved to `FILE.out`. The script `results_read.m` reads the results to the following variables:

Quantity	Solver			MATLAB		
	Variable	Size	Format	Variable	Size	Format
Time	t	1	double	t	[s*, 1]	real*8
Time step	dt	1	double	dt	[s, 1]	real*8
Particles kinetic energy	kines	1	double	kine_s	[s, 1]	real*8
Particles internal energy	ionnes	1	double	inne_s	[s, 1]	real*8
Particles potential energy	potes	1	double	pote_s	[s, 1]	real*8
Nodal kinetic energy	kineb	1	double	kine_b	[s, 1]	real*8
Nodal dissipation energy	disib	1	double	disi_b	[s, 1]	real*8
Nodal deformation energy	defob	1	double	defo_b	[s, 1]	real*8
Total energy	tote	1	double	tote	[s, 1]	real*8

*Variable s is the number of saved states. Variable s is not known in advance, the MATLAB script just reads until the end of file. Thanks to MATLAB, variable s is not neccesary to be pre-allocated.

The vector and matrices follow. The vectors (e.g. particle pressures) are saved as a row. If any variables has more dimensions (e.g. particle coordinates), the x-dimension is the first row, th y-dimension follows and the z-dimension is the last row. The saved number of dimensions depends on dim.

Quantity	Solver			MATLAB		
	Variable	Size	Format	Variable	Size	Format
Particles positions	x_s	[1, dim*n_s]	double	xs	[s, n_s]	real*8
				ys*	[s, n_s]	real*8
				zs**	[s, n_s]	real*8
Particles velocities	v_s	[1, dim*n_s]	double	vxs	[s, n_s]	real*8
				vys	[s, n_s]	real*8
				vzs	[s, n_s]	real*8
Particles velocities derivatives	dv_s	[1, dim*n_s]	double	dvxs	[s, n_s]	real*8
				dvys	[s, n_s]	real*8
				dvzs	[s, n_s]	real*8
Particles accelerations	a_s	[1, dim*n_s]	double	axs	[s, n_s]	real*8
				ays	[s, n_s]	real*8
				azs	[s, n_s]	real*8
Particles forces	f_s	[1, dim*n_s]	double	fxs	[s, n_s]	real*8
				fys	[s, n_s]	real*8
				fzs	[s, n_s]	real*8

*All variables in y-direction are saved only for 2D and 3D problems.

**All variables in z-direction are saved only for 3D problems.

Quantity	Solver			MATLAB		
	Variable	Size	Format	Variable	Size	Format
Particles density	rho_s	1	dlouble	rhos	[s, n_s]	real*8
Particles density derivative	drhodt_s	1	dlouble	drhos	[s, n_s]	real*8
Particles internal energy	u_s	1	dlouble	us	[s, n_s]	real*8
Particles internal energy derivative	dudt_s	1	dlouble	dus	[s, n_s]	real*8
Particles pressure	p_s	1	dlouble	ps	[s, n_s]	real*8
Particles sound speed	c_s	1	dlouble	cs	[s, n_s]	real*8
Particles smoothing length	h_s	1	dlouble	hs	[s, n_s]	real*8

The matrices variables (e.g. deviatoric stress) are usually symetric, taking into account the positions xx, xy, xz, yy, yz, zz, which are saved as six vectors one by one. For matrices, all dimensions are saved regardless `dim`.

Quantity	Solver			MATLAB		
	Variable	Size	Format	Variable	Size	Format
Particles deformation	e_s	[1, dime*n_s]*	dlouble	es	[s, dime*n_s]	real*8
Particles deformation rate	dedt_s	[1, dime*n_s]	dlouble	des	[s, dime*n_s]	real*8
Particles rotation	O_s	[1, dimo*n_s]*	dlouble	Os	[s, dimo*n_s]	real*8
Particles deviatoric stress	S_s	[1, dime*n_s]	dlouble	Ss	[s, dime*n_s]*	real*8
Particles deviatoric stress derivative	dSdt_s	[1, dime*n_s]	dlouble	dSs	[s, dime*n_s]	real*8

*`dime` and `dimo` depend on `dim` as only an upper triangular matrix for the particular dimension is always saved. For 1D problems, `Os` and `Ss` are not save at all as they are irrelevant in 1D. For 2D problems, `dime = 3` and `dimo = 1`, for 3D problems, `dime = 6` and `dimo = 3`. The reason is that `es` and `Ss` are symmetric matrices (so only 3 elements needed in 2D and only 6 elements needed in 3D) and `Os` is an antisymmetric matrix with zeros on diagonal (so only 1 element needed in 2D and only 3 elements needed in 3D).

Quantity	Solver			MATLAB		
	Variable	Size	Format	Variable	Size	Format
Nodal positions	x_b	[1, dim*n_b]	double	xb	[s, n_b]	real*8
				yb*	[s, n_b]	real*8
				zb**	[s, n_b]	real*8
Nodal velocities	v_b	[1, dim*n_b]	double	vxb	[s, n_b]	real*8
				vyb	[s, n_b]	real*8
				vzb	[s, n_b]	real*8
Nodal accelerations	a_b	[1, dim*n_b]	double	axb	[s, n_b]	real*8
				ayb	[s, n_b]	real*8
				azb	[s, n_b]	real*8
Nodal forces	f_b	[1, dim*n_b]	double	fxb	[s, n_b]	real*8
				fyb	[s, n_b]	real*8
				fzb	[s, n_b]	real*8

*All variables in y-direction are saved only for 2D and 3D problems.

**All variables in z-direction are saved only for 3D problems.

Quantity	Solver			MATLAB		
	Variable	Size	Format	Variable	Size	Format
Rigid bodies positions	x_r	[1, dim*n_r]	double	xr	[s, n_r]	real*8
				yr*	[s, n_r]	real*8
				zr**	[s, n_r]	real*8
Rigid bodies rotations	psi_r	[1, dimr*n_r]***	double	psixr	[s, n_r]	real*8
				psiyr	[s, n_r]	real*8
				psizr	[s, n_r]	real*8
Rigid bodies velocities	v_r	[1, dim*n_r]	double	vxr	[s, n_r]	real*8
				vyr	[s, n_r]	real*8
				vzr	[s, n_r]	real*8
Rigid bodies rotational velocities	o_r	[1, dimr*n_r]***	double	oxr	[s, n_r]	real*8
				oyr	[s, n_r]	real*8
				ozr	[s, n_r]	real*8
Rigid bodies accelerations	a_r	[1, dim*n_r]	double	axr	[s, n_r]	real*8
				ayr	[s, n_r]	real*8
				azr	[s, n_r]	real*8
Rigid bodies rotational accelerations	alpha_r	[1, dimr*n_r]***	double	alphaxr	[s, n_r]	real*8
				alphayr	[s, n_r]	real*8
				alphazr	[s, n_r]	real*8
Rigid bodies forces	f_r	[1, dim*n_r]	double	fxr	[s, n_r]	real*8
				fyr	[s, n_r]	real*8
				fzr	[s, n_r]	real*8
Rigid bodies moments	M_r	[1, dimr*n_r]***	double	Mxr	[s, n_r]	real*8
				Myr	[s, n_r]	real*8
				Nzr	[s, n_r]	real*8

*All variables in y-direction are saved only for 2D and 3D problems.

**All variables in z-direction are saved only for 3D problems.

***`dimr` depends on `dim`. For 1D problems, `psir`, `or` and `alphar` are not save at all as they are irrelevant in 1D. For 2D problems, `dimr = 1` and for 3D problems, `dimr = 3`. The reason is that there is only a single rotation in 2D and 2 rotations in 3D.

Quantity	Solver			MATLAB		
	Variable	Size	Format	Variable	Size	Format
Contact force	F_c	[1, dim*n_c]	double	fxc	[s, n_b]	real*8
				fyc*	[s, n_b]	real*8
				fzc**	[s, n_b]	real*8

*Force in y-direction are saved only for 2D and 3D problems.

**Force in z-direction are saved only for 3D problems.

For displaying particular options, the script `results_read.m` uses particular switches :

Variable	Switch	Value	Description
`nt`	Movie step	`vector`	`nt` is vector of saved time steps
`iw`	Save movie	[0]	No
		1	Yes
`id`	Movie composition	[0]	Movie only
		1	Movie with a variable*
`ip`	Draw	0	Contours (*only in 2D*)
		[1]	Particles
		2	Particles with velocity (*only in 2D*)
`ik`	Draw particular particles	`vector`	`ik` is vector of particle numbers (empty `vector` draws all particles)
`ib`	Draw boundary elements	[0]	No
		1	Yes
		2	Element with edges

*The variable is defined in `c2` and described in `c2val`. Variable `c2` is one of the output variables in MATLAB. E.g. `c2 = exx;` and `c2val = e_{xx};` show colour contours for deformation ϵ_{xx}.

3 Validation

The validation for 1D, 2D and 3D cases is summarized in [2, 3, 4, 5].

References

[1] P. W. Cleary. Modelling confned multi-material heat and mass flows using SPH. *Applied Mathematical Modelling*, 22:981–993, 1998.

[2] L. Hynčík. *Biomechanical model of abdominal organs and tissues for crash test purposes*. PhD thesis, University of West Bohemia, 2002.

[3] L. Hynčík. Smoothed particle hydrodynamics: Theory and practice. *Engineering Mechanics*, 9(6):1–15, 2002.

[4] L. Hynčík. Interaction of flowing liquid with deformable boundary by coupling SPH to FE. In J.F. Silva Gomes & S.A. Meguid, editor, *6th International Conference on Mechanics and Materials in Design*, pages 959–966, 2015.

[5] L. Hynčík. SPHCOFEM: Solver for coupling SPH and FE. In Springer, editor, *7th WACBE World Congress on Bioengineering 2015*, volume 52, pages 162–165, 2015.

[6] L. Hynčík. On modelling ballistoic gelatine by SPH. In J.F. Silva Gomes & S.A. Meguid, editor, *9th International Conference on Mechanics and Materials in Design*, pages 1–6, 2022.

[7] F. Macià, J. M. Sánchez, Souto-Iglesias A., and González L. M. Wcsph viscosity diffusion processes in vortex flows. *International Journal for Numerical Methods i Fluids*, pages 1–25, 2011.

[8] J. J. Monaghan and R. A. Gingold. Shock simulation by the particle method. *Journal of Computational Physics*, 52:374–389, 1983.

[9] J. P. Morris, P. J. Fox, and Y. Zhu. Modeling low reynolds number incompressible flows using sph. *Journal of Computational Physics*, 136:214–226, 1997.

[10] J. Onderik and R. Ďurikovič. Efficient neighbor search for particle-based fluids. *Journal of the Applied Mathematics, Statistics and Informatics*, 2(3):1–15, 2007.

[11] H. Takeda, S. M. Miyama, and M. Sekiya. Numerical simulation of viscous flow by smoothed particle hydrodynamics. *Progress of Theoretical Physics*, 92(5):939–960, 1999.