

# PARMEC MANUAL

January 17, 2017

# Contents

<b>1</b>	<b>Running</b>	<b>2</b>
<b>2</b>	<b>Input commands</b>	<b>3</b>
2.1	RESET . . . . .	3
2.2	MATERIAL . . . . .	3
2.3	SPHERE . . . . .	3
2.4	MESH . . . . .	4
2.5	ANALYTICAL . . . . .	4
2.6	OBSTACLE . . . . .	5
2.7	SPRING . . . . .	5
2.8	GRANULAR . . . . .	6
2.9	CONSTRAIN . . . . .	7
2.10	PRESCRIBE . . . . .	7
2.11	VELOCITY . . . . .	7
2.12	GRAVITY . . . . .	7
2.13	DAMPING . . . . .	8
2.14	CRITICAL . . . . .	8
2.15	HISTORY . . . . .	8
2.16	OUTPUT . . . . .	8
2.17	DEM . . . . .	10
<b>3</b>	<b>Output files</b>	<b>11</b>
<b>4</b>	<b>Output viewers</b>	<b>12</b>
<b>5</b>	<b>Tutorial</b>	<b>13</b>

# Chapter 1

## Running

PARMEC is a command line program. Typical usage:

1. Place PARMEC in a globally accessible path (e.g. suitably extend the PATH variable on a unix system).
2. Create a directory where your input file and output files will be stored (e.g. *mkdir test*).
3. Edit your Python input file in this directory (e.g. *test.py*); Chapter 2 documents all input commands.
4. Run PARMEC (e.g. *PARMEC path/to/test/test.py*).
5. Time histories can be generated during analysis using the HISTORY command; see Section 2.15.
6. Upon termination a \*.dump file and/or a \*.vtk.\* file(s) is/are created in the same directory (e.g. *path/to/test/test.dump*) their format is documented in Chapter 3.
7. The output files can be viewed with OVITO and/or ParaView, as documented in Chapter 4.

A tutorial is provided in Chapter 5.

## Chapter 2

# Input commands

PARMEC input language extends Python. Subroutines related to input processing are listed below.

### 2.1 RESET

Erase all data.

**RESET ()**

### 2.2 MATERIAL

Create material.

**matnum = MATERIAL (density, young, poisson)**

- **matnum** - material number
- **density** - mass density
- **young** - Young modulus
- **poisson** - Poisson ratio

### 2.3 SPHERE

Create a spherical particle.

**parnum = SPHERE (center, radius, material, color)**

- **parnum** - particle number
- **center** - tuple  $(x, y, z)$  defining the center
- **radius** - radius
- **material** - material number
- **color** - positive integer surface color

## 2.4 MESH

Create a meshed particle.

**parnum = MESH (nodes, elements, material, colors)**

- **parnum** - particle number
- **nodes** - list of nodes:  $[x0, y0, z0, x1, y1, z1, \dots]$
- **elements** - list of elements:  $[e1, n1, n2, \dots, ne1, me1, e2, n1, n2, \dots, ne2, me2, \dots]$ , where  $e1$  is the number of nodes of the first element,  $n1, n2, \dots, ne1$  enumerate the element nodes, and  $me1$  is the material number. Similarly for the second and all remaining elements. Supported numbers of nodes per element are 4, 5, 6, and 8 for respectively *tetrahedron*, *pyramid*, *wedge*, and *hexahedron*, cf. Figure 2.1.
- **material** - material number
- **colors** - list of positive integer face colors:  $[gcolor, f1, n1, n2, \dots, nf1, c1, f2, n1, n2, \dots, nf2, c2, \dots]$ , where  $gcolor$  is the global color for all not specified faces,  $f1$  is the number of nodes in the first specified face,  $n1, n2, \dots, nf1$  enumerate the face nodes, and  $c1$  is the surface color of that face. Similarly for the second and all remaining faces. If only the global color is required, it can be passed as  $[gcolor]$  or as  $gcolor$  alone.

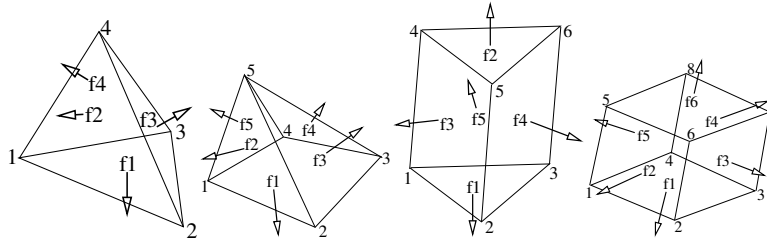


Figure 2.1: Element types.

## 2.5 ANALYTICAL

Create an analytical particle. Analytical particles have no shapes and are not involved in contact.

**parnum = ANALYTICAL ( | inertia, mass, rotation, position, material, particle)**

Note, that all parameters are optional.

- **parnum** - particle number
- **inertia** - inertia tensor passed as a list  $[Ixx, Iyy, Izz, Ixy, Ixz, Iyz]$ ; optional, if **particle** parameter is used; default  $[1, 1, 1, 0, 0, 0]$
- **mass** - scalar mass; optional, if **particle** parameter is used; default 1
- **rotation** - optional orientation matrix passed as a list  $[e1x, e1y, e1z, e2x, e2y, e2z, e3x, e3y, e3z]$ , where vectors  $e1, e2, e3$  are orthonormal; default  $[1, 0, 0, 0, 1, 0, 0, 0, 1]$
- **position** - optional position vector passed as a tuple  $(x, y, z)$ ; default  $(0, 0, 0)$

- **material** - material number; default 0
- **particle** - optional; if specified, an existing particle is converted into an analytical particle; its properties are inherited or overwritten, depending on whether any of the **inertia**, **mass**, **rotation**, **position** parameters are used; if initially specified, particle shape is inherited and its animated motion is included into the results

## 2.6 OBSTACLE

Create an obstacle.

### OBSTACLE (triangles, color | point, linear, angular)

- **triangles** - list of triangle tuples  $[(t1x1, t1y1, t1z1, t1x2, t1y2, t1z2, t1x3, t1y3, t1z3), (t2x1, t2y1, t2z1, t2x2, t2y2, t2z2, t2x3, t2y3, t2z3), \dots]$  defining the obstacle
- **color** - positive integer surface color or a list  $[color1, color2, \dots]$  of colors for each individual triangle
- **point** - spatial reference point
- **linear** - linear velocity history callback:  $(v_x, v_y, v_z) = \mathbf{linear}(t)$
- **angular** - spatial angular velocity history callback:  $(\omega_x, \omega_y, \omega_z) = \mathbf{angular}(t)$

## 2.7 SPRING

Create a translational spring constraint. The applied force formula reads

$$\text{force}(t) = \text{direction}(t) \cdot [\text{spring}(\text{stroke}(t)) + \text{dashpot}(\text{velocity}(t)) \cdot |\text{sign}(\text{spring}(\text{stroke}(t)))|]$$

where

$$\text{direction}(t) = (\text{point2}(t) - \text{point1}(t)) / |\text{point2}(t) - \text{point1}(t)| \text{ or constant } (d_x, d_y, d_z) \text{ or tangent}$$

$$\text{stroke}(t) = \text{direction}(t) \cdot [\text{point2}(t) - \text{point1}(t)] - \text{direction}(0) \cdot [\text{point2}(0) - \text{point1}(0)]$$

$$\text{velocity}(t) = \text{direction}(t) \cdot \frac{d}{dt} [\text{point2}(t) - \text{point1}(t)]$$

$$\text{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

The  $\text{spring}(\text{stroke})$  and  $\text{dashpot}(\text{velocity})$  relationships are defined by means of lookup tables;  $\text{force}(t)$  is applied at  $\text{point2}(t)$ , and  $-\text{force}(t)$  is applied at  $\text{point1}(t)$ ; dashpot force is not applied when spring force is zero.

**sprnum = SPRING (part1, point1, part2, point2, spring | dashpot, direction, planar, unload, ylim)**

- **sprnum** - spring number
- **part1** - first particle number
- **point1** - tuple  $(x, y, z)$  defining a point moving with the first particle
- **part2** - second particle number;  $-1$  can be used to indicate a single-particle constraint
- **point2** - tuple  $(x, y, z)$  defining a second point, either moving with the second particle, or a spatial point
- **spring** - spring force lookup table  $[\text{stroke}_1, \text{force}_1, \text{stroke}_2, \text{force}_2, \dots, \text{stroke}_n, \text{force}_n]$ ; used for both loading and unloading when the **unload** table and the **yield** limits are not given
- **dashpot** - optional dashpot force lookup table  $[\text{velocity}_1, \text{force}_1, \text{velocity}_2, \text{force}_2, \dots, \text{velocity}_m, \text{force}_m]$ ; default:  $[-\infty, 0, +\infty, 0]$
- **direction** - optional constant direction  $(d_x, d_y, d_z)$
- **planar** - optional planar spring flag; when 'ON' spring direction

$$(\text{point2}(t) - \text{point1}(t)) / |\text{point2}(t) - \text{point1}(t)|$$

is projected onto a plane orthogonal to  $(d_x, d_y, d_z)$ ; default: 'OFF'

- **unload** - spring unloading lookup table  $[\text{stroke}_1, \text{force}_1, \text{stroke}_2, \text{force}_2, \dots, \text{stroke}_n, \text{force}_n]$ ; must be monotonically increasing
- **ylim** - tuple  $(f_{yc}, f_{yt})$  defining the compression,  $f_{yc} < 0$ , and tension,  $f_{yt} > 0$ , yield limits; the unloading curve begins to be used once either of these limits is crossed; default:  $(0, 0)$

## 2.8 GRANULAR

Define surface pairing for the granular contact interaction model.

**GRANULAR (color1, color2, spring | damper, friction, rolling, drilling, kskn)**

- **color1** - first color (positive, or  $\text{color1} = 0$  and  $\text{color2} = 0$  to redefine default parameters)
- **color2** - second color (positive, or  $\text{color1} = 0$  and  $\text{color2} = 0$  to redefine default parameters)
- **spring** - normal spring constant
- **damper** - optional normal damping ratio; default: 1.0
- **friction** - optional Coulomb's friction coefficient; default: 0.0; tuple  $(\mu_s, \mu_d)$  can be used to specify respectively static and dynamic friction coefficients
- **rolling** - optional rolling friction coefficient; default: 0.0
- **drilling** - optional drilling friction coefficient; default: 0.0
- **kskn** - optional ratio of normal to tangential spring and dashpot parameters; default: 0.5

## 2.9 CONSTRAIN

Constrain particle motion.

**CONSTRAIN** (**parnum** | **linear**, **angular**)

- **parnum** - particle number
- **linear** - list  $[x_1, y_1, z_1]$ ,  $[x_1, y_1, z_1, x_2, y_2, z_2]$ , or  $[x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3]$  defining directions of constrained linear motion; default:  $[0, 0, 0]$
- **angular** - list  $[x_1, y_1, z_1]$ ,  $[x_1, y_1, z_1, x_2, y_2, z_2]$ , or  $[x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3]$  defining directions of constrained spatial rotation; default:  $[0, 0, 0]$

## 2.10 PRESCRIBE

Prescribe particle motion. Prescribed motion overwrites this resulting from dynamics and constraints.

**PRESCRIBE** (**parnum** | **linear**, **angular**, **kind**)

- **parnum** - particle number
- **linear** - linear velocity or acceleration history callback:  $(v_x, v_y, v_z) = \mathbf{linear}(t)$ ; default: *not prescribed*
- **angular** - spatial angular velocity or acceleration history callback:  $(\omega_x, \omega_y, \omega_z) = \mathbf{angular}(t)$ ; default: *not prescribed*
- **kind** - string 'vv', 'va', 'av', or 'aa' indicating interpretation of respectively **linear** and **angular** time histories as either velocity or acceleration; default: 'vv'

## 2.11 VELOCITY

Set particle velocity.

**VELOCITY** (**parnum** | **linear**, **angular**)

- **parnum** - particle number
- **linear** - linear velocity tuple  $(v_x, v_y, v_z)$ ; default:  $(0, 0, 0)$  at  $t = 0$
- **angular** - angular velocity tuple  $(\omega_x, \omega_y, \omega_z)$ ; default:  $(0, 0, 0)$  at  $t = 0$

## 2.12 GRAVITY

Set gravity.

**GRAVITY** (**gx**, **gy**, **gz**)

- **gx** - constant x component or callback  $gx(t)$
- **gy** - constant y component or callback  $gy(t)$
- **gz** - constant z component or callback  $gz(t)$



## 2.13 DAMPING

Set global damping, applied as

$$\text{force} = -m \begin{bmatrix} -d_{vx}v_x \\ -d_{vy}v_y \\ -d_{vz}v_z \end{bmatrix}, \text{ torque} = -\mathbf{\Lambda}\mathbf{J}\mathbf{\Lambda}^T \begin{bmatrix} -d_{\omega x}\omega_x \\ -d_{\omega y}\omega_y \\ -d_{\omega z}\omega_z \end{bmatrix}$$

where  $m$  is scalar mass,  $v$  is linear velocity,  $\mathbf{\Lambda}$  is the rotation matrix,  $\mathbf{J}$  is the referential inertia matrix, and  $\omega$  is spatial angular velocity.

### DAMPING (linear, angular)

- **linear** - linear damping curve callback:  $(d_{vx}, d_{vy}, d_{vz}) = \mathbf{linear}(t)$
- **angular** - angular damping curve callback:  $(d_{\omega x}, d_{\omega y}, d_{\omega z}) = \mathbf{angular}(t)$

## 2.14 CRITICAL

Estimate critical time step.

**h = CRITICAL ()**

- **h** - critical time step

## 2.15 HISTORY

Before running a simulation, request time history output.

**list = HISTORY (entity | source, point)**

- **list** - output time history list (empty upon initial request, populated during simulation)
- **entity** - entity name; global entities: (output time) 'TIME'; particle entities: (position) 'PX', 'PY', 'PZ', 'P|', (displacement) 'DX', 'DY', 'DZ', 'D|', (linear velocity) 'VX', 'VY', 'VZ', 'V|', (angular velocity) 'OX', 'OY', 'OZ', 'O|', (body force) 'FX', 'FY', 'FZ', 'F|', (body torque) 'TX', 'TY', 'TZ', 'T|'; spring entities: (spring stroke) 'STROKE', (spring total force) 'STF', (spring force without damping) 'SF';
- **source** - particle number  $i$ , or a list of particle numbers  $[i, j, \dots]$ , or a spatial sphere defined as tuple  $(x, y, z, r)$ , or a spatial box defined as tuple  $(x_{\min}, y_{\min}, z_{\min}, x_{\max}, y_{\max}, z_{\max})$ ; in case of a list of particle numbers the output entity is averaged over the set of particles; in case of a spatial sphere or box the output entity is averaged over the set of particles passing through it; default: 0 (useful when entity is 'TIME'); spring number or a list of numbers can be used as a source in case of spring entities
- **point** - optional referential point used in case of a single particle source; default: particle mass centre

## 2.16 OUTPUT

Before running a simulation, define scalar and/or vector entities included into the output file(s). PARMEC outputs:

- \*.dump files for spherical particles

- `*0.vtk.*` files for obstacles and mesh based particles **not** specified as a **subset** in the OUTPUT command
- `*1.vtk.*`, `*2.vtk.*`, ... files for mesh based particles specified as subsets, where numbers 1, 2, ... match consecutive OUTPUT calls
- `*0rb.vtk.*` for rigid body data of particles **not** specified as a **subset** in the OUTPUT command
- `*1rb.vtk.*`, `*2rb.vtk.*`, ... files for rigid body data of particles specified as subsets, where numbers 1, 2, ... match consecutive OUTPUT calls
- `*0cd.vtk.*` for contact data including particles **not** specified as a **subset** in the OUTPUT command
- `*1cd.vtk.*`, `*2cd.vtk.*`, ... files for contact data including particles specified as subsets, where numbers 1, 2, ... match consecutive OUTPUT calls
- `*0sd.vtk.*` for spring data including particles **not** specified as a **subset** in the OUTPUT command
- `*1sd.vtk.*`, `*2sd.vtk.*`, ... files for spring data including particles specified as subsets, where numbers 1, 2, ... match consecutive OUTPUT calls

## OUTPUT (entities | subset, mode)

- **entities** - list of output entities; default: ['NUMBER', 'COLOR', 'DISPL', 'ORIENT', 'LINVEL', 'ANGVEL', 'FORCE', 'TORQUE', 'F', 'FN', 'FT', 'SF', 'AREA', 'PAIR'] where:
  - 'NUMBER' - scalar field of particle numbers (modes: 'SPH', 'MESH', 'RB'), or scalar field of spring numbers (modes: 'SD')
  - 'COLOR' - scalar field of surface colors (modes: 'SPH', 'MESH'), or 2-component vector field of contact surface colors (modes: 'CD')
  - 'DISPL' - 3-component vector field of displacements (modes: 'SPH', 'MESH', 'RB'), or scalar field of contact depths (modes: 'CD'), or scalar field of spring strokes (modes: 'SD')
  - 'ORIENT' - three 3-component vector fields representing columns of rigid rotation matrix (orientation vectors) (modes: 'RB'), or 3-component vector field of spring orientations (modes: 'SD')
  - 'LINVEL' - 3-component vector field of linear velocity (modes: 'SPH', 'MESH', 'RB')
  - 'ANGVEL' - 3-component vector field of (spatially constant) angular velocity (modes: 'SPH', 'MESH', 'RB')
  - 'FORCE' - 3-component vector field of (spatially constant) total body force (modes: 'SPH', 'MESH', 'RB')
  - 'TORQUE' - 3-component vector field of (spatially constant) total body torque (modes: 'SPH', 'MESH', 'RB')
  - 'F' - 3-component vector field of total contact forces (modes: 'CD'), or scalar field of total spring forces (modes: 'SD')
  - 'FN' - 3-component vector field of normal contact forces (modes: 'CD')
  - 'FT' - 3-component vector field of tangential contact forces (modes: 'CD')
  - 'SF' - scalar field of spring force magnitude, without dashpot contribution (modes: 'CD', 'SD')
  - 'AREA' - scalar field of contact area (modes: 'CD')
  - 'PAIR' - 2-component vector field of particle pair numbers (modes: 'CD', 'SD')
- **subset** - optional particle number  $i$ , or a list of particle numbers  $[i, j, \dots]$ , to which this specification is narrowed down
- **mode** - optional output mode or list of output modes: 'SPH' for sphere output, 'MESH' for mesh output, 'RB' for rigid body output, 'CD' for contact data output, 'SD' for spring data output; default: ['SPH', 'MESH', 'RB', 'CD', 'SD']

## 2.17 DEM

Run DEM simulation.

**t = DEM (duration, step | interval, prefix, adaptive)**

- **t** - simulation runtime in seconds
- **duration** - simulation duration
- **step** - time step; initial if **adaptive** is used or constant otherwise
- **interval** - output interval (default: time step); tuple  $(dt_{\text{files}}, dt_{\text{history}})$  can be used to indicate different output frequencies of output files and time histories, respectively; callback functions can also be used, e.g.  $dt_{\text{files}} = dt\_files(t)$  and  $dt_{\text{history}} = dt\_history(t)$  prescribing variable interval frequencies, depending on current time;
- **prefix** - output file name prefix (default: input file name without the “.py” extension)
- **adaptive** - adaptive time step reduction factor; zero turns off adaptive time stepping, values  $> 0.0$  and  $\leq 1.0$  turn it on; default: 0.0 (**experimental**)

## Chapter 3

# Output files

## Chapter 4

# Output viewers

## Chapter 5

## Tutorial