

# SOFLEC 2.0 USER MANUAL

July 16, 2019

# Contents

<b>1</b>	<b>Installation</b>	<b>2</b>
<b>2</b>	<b>Running</b>	<b>3</b>
2.1	Version string . . . . .	3
<b>3</b>	<b>Input commands</b>	<b>4</b>
3.1	ARGV . . . . .	4
3.2	RESET . . . . .	4
3.3	SPLINE . . . . .	4
3.4	MATERIAL . . . . .	5
3.5	MESH . . . . .	5
3.6	FRICTION . . . . .	6
3.7	RESTRAIN . . . . .	6
3.8	PRESCRIBE . . . . .	6
3.9	VELOCITY . . . . .	7
3.10	GRAVITY . . . . .	7
3.11	HISTORY . . . . .	7
3.12	OUTPUT . . . . .	8
3.13	RUN . . . . .	8
<b>4</b>	<b>Output files</b>	<b>10</b>

# Chapter 1

## Installation

Clone Solfec-2.0 sources from [GitHub](https://github.com/parmes/solfec-2.0):

```
git clone https://github.com/parmes/solfec-2.0.git
```

Enter directory:

```
cd solfec-2.0
```

Edit Config.mak file variables:

```
# ISPC (http://ispc.github.io) and MPICXX compilers are assumed to be in the PATH

# Python paths (used to parse input files)
PYTHONINC=-I/usr/include/python2.7
PYTHONLIB=-L/usr/lib -lpython2.7

# HDF5 paths (used to write output files)
HDF5INC=-I/usr/include
HDF5LIB=-L/usr/lib -lhdf5 -lhdf5_hl

# Debug version
DEBUG=no
```

Compile sources:

```
make
```

Solfec-2.0 executables are:

```
solfec4 (single precision)
solfec8 (double precision)
```

To update the executables type:

```
make clean
git pull
make
```

## Chapter 2

# Running

Solfec-2.0 is a command line program utilizing MPI runtime environment and shared memory parallelism. For optimum performance, for an input model and a hardware platform at hand, you may want to test which degree of MPI parallelism per cluster node results in shortest runtimes. Typical usage:

1. Include solfec-2.0 directory into your PATH variable.
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 3 documents all input commands.
4. Run Solfec-2.0 (e.g. `mpirun -np N solfec4 path/to/test/test.py`, or `mpirun -np N solfec8 path/to/test/test.py`, or use a batch script).
5. Time histories can be generated during analysis using the HISTORY command; see Section 3.11.
6. Upon termination output files are created in the same directory (e.g. `path/to/test/test.h5`, `path/to/test/test.xmlf`); see Section 3.12 and Chapter 4.

### 2.1 Version string

Running Solfec-2.0 without parameters, e.g.

```
./solfec4
```

results in the hint

```
VERSION: 2.4a49378 (2019-07-10)
SYNOPSIS: [mpirun -np N] solfec4 path/to/input/file.py
```

The version string has syntax “2.” (denoting Solfec-2.0’s primary version number) followed by a shorthand hexadecimal number (“4a49378” in this case) of a [GitHub commit](#) that most recently modified Solfec-2.0’s source code. This is followed by the date of that most recent modification of the source code.

## Chapter 3

# Input commands

Solfec-2.0 input language extends [Python](#) by subroutines listed below. In all cases, when an object number is returned, **indexing starts at 0** and increments on each call. Only **rank 0** MPI process reads the input file.

### 3.1 ARGV

List command line arguments.

**list** = **ARGV** (| **nonsolfec**)

- **list** - Python list (possibly empty) of command line arguments
- **nonsolfec** - optional boolean flag enabling filtering out Solfec-2.0 arguments; default: True

### 3.2 RESET

Erase all data.

**RESET** ( | **outname**)

- **outname** - optional output file name (default: input file name without the “.py” extension); output name can include sub-directories, e.g. ‘an/alternative/path’ will result in output files ‘an/alternative/path.xmlf’ and ‘an/alternative/path.h5’ (relative to whence Solfec-2.0 was run) when XDMF output is used (see the OUTPUT command)

### 3.3 SPLINE

Create a linear spline based on series of 2-points.

**splnum** = **SPLINE** (**points** | **cache**)

- **splnum** - spline number
- **points** - a list  $[x0, y0, x1, y1, \dots]$  or  $[[x0,y0], [x1,y1], \dots]$  or  $[(x0,y0), (x1,y1), \dots]$  of points (where  $t_i < t_j$ , when  $i < j$ ), or a path to a file storing pairs of  $(x, y)$ s in format:

```
# comment 1 ...
# comment 2 ...
x0 y0
x1 y1
# comment 3 ...
x2 y2
...
```

- **cache** - optional partial cache size; if **points** = file path and **cache** > 0 then only the cache size of points is stored in memory at any given time; this helps to save memory in case of a need for many large spline objects; default: 0 (entire spline is stored in memory)

### 3.4 MATERIAL

Create material.

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

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

### 3.5 MESH

Create a meshed body.

**bodnum** = MESH (nodes, elements, matnum, colors | transform)

- **bodnum** - body 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*, see Figure 3.1.
- **matnum** - 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.

- **transform** - optional transformation tuple, or a list of transformation tuples, e.g. [translate1, rotate1, translate2, scale2, rotate2, ...], where  $\text{translate} = (tx, ty, tz)$ ;  $\text{rotate} = (px, py, pz, ax, ay, az, angle)$ , where  $px, py, pz$  define the point and  $ax, ay, az$  define the direction of the rotation axis while  $angle$  defines the angle of rotation, or alternatively  $\text{rotate} = (r_{11}, r_{21}, r_{31}, r_{12}, r_{22}, r_{32}, r_{13}, r_{23}, r_{33})$  is the rotation matrix (orthogonality is not checked, hence this matrix can be used for more general transformations);  $\text{scale} = (px, py, pz, factor)$ , where  $px, py, pz$  define the scaling centre point and  $factor$  is the scaling factor; transformations are applied in the list order; default: *not specified*

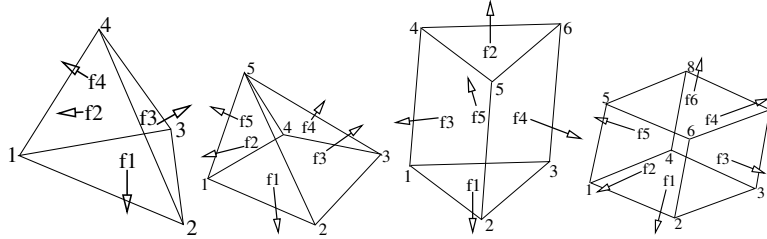


Figure 3.1: Mesh element types in Solfec-2.0.

## 3.6 FRICTION

Define surface pairing contact friction. Default values, for unspecified pairings, are: **static** = 0, **dynamic** = 0.

### FRICTION (color1, color2 | static, dynamic)

- **color1** - first color (positive, or color1 = 0 and color2 = 0 to redefine default parameters for unspecified pairings)
- **color2** - second color (positive, or color1 = 0 and color2 = 0 to redefine default parameters for unspecified pairings)
- **static** - optional static Coulomb's friction coefficient; default: 0.0
- **dynamic** - optional dynamic Coulomb's friction coefficient; default: 0.0

## 3.7 RESTRAIN

Restrain body point or surface motion.

### RESTRAIN (bodnum | point, color)

- **bodnum** - body number
- **point** - optional referential point  $(px, py, pz)$  or list of points [point1, point2, point3, ...]; default: *not specified*
- **color** - optional surface color; default: *not specified*

## 3.8 PRESCRIBE

Prescribe body, point, or surface motion. Prescribed motion overwrites this resulting from dynamics and restraints.

**PRESCRIBE (bodnum | point, color, linear, angular)**

- **bodnum** - body number
- **point** - optional referential point  $(px, py, pz)$  or list of points  $[point1, point2, point3, \dots]$ ; default: *not specified*
- **color** - optional surface color; default: *not specified*
- **linear** - a tuple  $(i, j, k)$  of SPLINE numbers or floating point constant values, or a callback:  $(v_x, v_y, v_z) = \mathbf{linear}(t)$ , defining linear velocity history; default: *not prescribed*
- **angular** - a tuple  $(i, j, k)$  of SPLINE numbers or floating point constant values, or a callback:  $(\omega_x, \omega_y, \omega_z) = \mathbf{angular}(t)$ , defining spatial angular velocity history with respect to the entity (body or surface) mass center; default: *not prescribed*

**3.9 VELOCITY**

Set body velocity.

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

- **bodnum** - body number
- **linear** - linear velocity tuple  $(v_x, v_y, v_z)$ ; default:  $(0, 0, 0)$  at  $t = 0$
- **angular** - angular velocity tuple  $(\omega_x, \omega_y, \omega_z)$ , applied with respect to the referential mass center; default:  $(0, 0, 0)$  at  $t = 0$

**3.10 GRAVITY**

Set gravity.

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

- **gx** - constant  $x$  float number, or callback  $\mathbf{gx}(t)$ , or SPLINE number
- **gy** - constant  $y$  float number, or callback  $\mathbf{gy}(t)$ , or SPLINE number
- **gz** - constant  $z$  float number, or callback  $\mathbf{gz}(t)$ , or SPLINE number

**3.11 HISTORY**

Before running a simulation, request time history output; or read history from an existing output file.

**list = HISTORY (entity | point, bodnum, filepath)**

- **list** - output time history list (empty upon initial request, populated during simulation)
- **entity** - entity name; global entities: (output time) 'TIME', (number of contacts) 'CONTACTS'; body entities: (position) 'PX', 'PY', 'PZ', '|P|', (displacement) 'DX', 'DY', 'DZ', '|D|', (linear velocity) 'VX', 'VY', 'VZ', '|V|', (Cauchy stress) 'SX', 'SY', 'SZ', 'SXY', 'SXZ', 'SYZ', '|S|' (von Mises norm);
- **point** - referential point; default: *not specified*



- **bodnum** - body number; default: *not specified*
- **filepath** - optional file path to store the history in a text file; default: *not specified*

## 3.12 OUTPUT

Before running a simulation, define scalar and/or vector entities included into the output file(s). Solfec-2.0 outputs:

- **\*0.vtk.\* and/or (\*0.h5, \*0.xmf)** files for meshed bodies **not** specified as a **subset** in the OUTPUT command
- **\*1.vtk.\*, \*2.vtk.\*, ... and/or (\*1.h5, \*1.xmf, \*2.h5, \*2.xmf, ...)** files for meshed bodies specified as subsets, where numbers 1, 2, ... match consecutive OUTPUT calls
- **\*0cd.vtk.\* and/or (\*0cd.h5, \*0cd.xmf)** files for contact data including bodies **not** specified as a **subset** in the OUTPUT command
- **\*1cd.vtk.\*, \*2cd.vtk.\*, ... and/or (\*1cd.h5, \*1cd.xmf, \*2cd.h5, \*2cd.xmf, ...)** files for contact data including bodies specified as subsets, where numbers 1, 2, ... match consecutive OUTPUT calls

### OUTPUT ( | entities, subset, modes, formats)

- **entities** - list of output entities; default: ['NUMBER', 'COLOR', 'DISPL', 'LINVEL', 'STRESS', 'CF', 'CFN', 'CFT', 'AREA', 'BPAIR', 'CPAIR'] where:
  - 'NUMBER' - scalar field of body numbers (modes: 'MESH')
  - 'COLOR' - scalar field of surface colors (modes: 'MESH')
  - 'DISPL' - 3-component vector field of displacements (modes: 'MESH')
  - 'LINVEL' - 3-component vector field of linear velocity (modes: 'MESH')
  - 'STRESS' - 6-component tensor field representing Cauchy stress (modes: 'MESH')
  - 'CF' - 3-component vector field of total contact forces (modes: 'CD')
  - 'CFN' - 3-component vector field of normal contact forces (modes: 'CD')
  - 'CFT' - 3-component vector field of tangential contact forces (modes: 'CD')
  - 'AREA' - scalar field of contact area (modes: 'CD')
  - 'BPAIR' - 2-component vector field of body pair numbers (modes: 'CD')
  - 'CPAIR' - 2-component vector field of color pair numbers (modes: 'CD')
- **subset** - optional body number  $i$ , or a list of body numbers  $[i, j, \dots]$ , to which this specification is narrowed down
- **modes** - optional output mode or list of output modes: 'MESH' for mesh output, 'CD' for contact data output; default: ['MESH', 'CD']
- **formats** - optional output format, e.g. 'VTK' or 'XDMF', or a list, e.g. ['VTK', 'XDMF'], see Chapter 4; default: 'XDMF'

## 3.13 RUN

Run simulation.

**t = RUN (duration, step | interval)**

- **t** - simulation runtime in seconds
- **duration** - simulation duration; 0 can be used to output the initial model state
- **step** - simulation time step
- **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 or SPLINE numbers can also be used, e.g.  $dt_{\text{files}} = dt\_fies(t)$  and  $dt_{\text{history}} = splnum$ , prescribing variable interval frequencies, depending on current time;

## Chapter 4

# Output files

Currently Solfec-2.0 supports the following output file formats:

- [\\*.vtk](#) text based legacy [VTK format](#).
- [\\*.xmf](#) HDF5/XML based [XDMF format](#).

Both of the above formats can be viewed with [ParaView](#) or [VisIt](#). See also the OUTPUT command for details on outputted entities and file kinds.