LS-Reader Tutorial

Introduction

This document describes the application functions interface of LS-Reader using Python.

The LS-Reader is designed to read LS-DYNA results and it supports C, C++ and Python languages. It supports both Windows(vs2010, vs2015, vs2017, vs2019) and Linux(GCC>=4.1.2). Because of the simplicity of the LS-Reader, using the libraries is very convenie

Python

D3plotReader

API Functions (*Recommended*)

```
class D3plotReader():
    def __init__(self, path):
        pass
```

❖ Purpose: Constructor.

❖ Input: path: d3plot name.

❖ Return: D3plotReader object.

Example: `dr = D3plotReader("d3plot/file/path")`

```
    def get_data(self, type, param):
        pass
```

❖ Purpose: Extract data.

❖ Input: type: type - enum the data variables' name in d3plot.

param: structure of description which is the advance setting for getting special data in d3plot.

❖ Return: data.

Example:

```
dr = D3plotReader("d3plot/file/path")
p = D3P_Parameter()
p.ist = 11
p.ipt = 0
shell_stress = dr.get_data(DataType.D3P_SHELL_STRESS, p)
```

Or

```
dr = D3plotReader("d3plot/file/path")


shell_stress = dr.get_data(

DataType.D3P_SHELL_STRESS, ist=11, ipt=0

)
```

API Functions (*Deprecated*)

```
class D3plotReader():
    def __init__(self, path):
        pass
```

- Purpose: Constructor.
- Input: path: d3plot name.
- Return: D3plotReader object.

  Example:`dr = D3plotReader("d3plot/file/path")`

```
    def GetDataInt(self, type, param):
        pass
```

Deprecated. Use the get_data(...) instead.

- Purpose: Get an integer value.
- Input: type - enum the data variables' name in d3plot.

  Param - structure of description which is the advance setting for getting special data in d3plot.
- Return: int

```
    def GetDataFloat(self, type, param):
        pass
```

Deprecated. Use the get_data(...) instead.

- Purpose: Get a float value.
- Input: type - enum the data variables' name in d3plot.

  param - structure of description which is the advance setting for getting special data in d3plot.

❖ Return: float.

```
def GetDataString(self, type, param):
    pass
```

Deprecated. Use the get_data(...) instead.

❖ Purpose: Get a string value.

❖ Input: type - enum the data variables' name in d3plot.

param - structure of description which is the advance setting for getting special data in d3plot.

❖ Return: string.

```
def GetDataIntArray(self, type, param):
    pass
```

Deprecated. Use the get_data(...) instead.

❖ Purpose: Get a int array.

❖ Input: type - enum the data variables' name in d3plot.

param - structure of description which is the advance setting for getting special data in d3plot.

❖ Return: int array.

```
def GetDataFloatArray(self, type, param):
    pass
```

Deprecated. Use the get_data(...) instead.

❖ Purpose: Get a float array.

❖ Input: type - enum the data variables' name in d3plot.

param - structure of description which is the advance setting for getting special data in d3plot.

❖ Return: float array.

```
def GetDataVectorArray(self, type, param):
    pass
```

Deprecated. Use the get_data(...) instead.

❖ Purpose: Get a vector array.

❖ Input: type - enum the data variables' name in d3plot.

param - structure of description which is the advance setting for getting special data in d3plot.

❖ Return: vector array.

```
def GetDataTensorArray(self, type, param):
    pass
```

Deprecated. Use the get_data(...) instead.

❖ Purpose: Get a tensor array.

❖ Input: type - enum the data variables' name in d3plot.

param - structure of description which is the advance setting for getting special data in d3plot.

❖ Return: tensor array.

```
def GetDataSolidArray(self):
    pass
```

Deprecated. Use the get_data(...) instead.

❖ Purpose: Get solid elements array.

❖ Return: solid elements array.

```
def GetDataTshellArray(self):
    pass
```

Deprecated. Use the get_data(...) instead.

❖ Purpose: Get tshell elements array.

❖ Return: tshell elements array.

```
def GetDataBeamArray(self):
    pass
```

Deprecated. Use the get_data(...) instead.

❖ Purpose: Get beam elements array.

❖ Return: beam elements array.

```
def GetDataShellArray(self):
    pass
```

Deprecated. Use the get_data(...) instead.

❖ Purpose: Get shell elements array.

❖ Return: shell elements array.

```
def GetDataSphArray(self):
    pass
```

Deprecated. Use the get_data(...) instead.

❖ Purpose: Get sph elements array.

❖ Return: sph elements array.

D3P_Parameter

parameter to call D3plotReader::get_data*, only specific those member variables you are interested, otherwise, ignore this.

```
class D3P_Parameter:

    def __init__(self):

        self.ist = -1

        self.ipt = -1

        self.ipart = -1

        self.ipart_user = -1

        self.i_rigid_wall = -1

        self.ides = -1

        self.ihv = -1

        self.index_multisolver = -1

        self.id_var_multisolver = -1

        self.iuser = -1

        self.var_name = ""
```

1. **ist**: Specify the state number, starting from 0, as follows:

```
shell_thickness = dr.get_data(DataType.D3P_SHELL_THICKNESS, ist=11)
```

Or

```
p = D3P_Parameter()

p.ist = 11

shell_thickness = dr.get_data(DataType.D3P_SHELL_THICKNESS, p)
```

2. **ipt**: Specify the integration point, ranging in [0, MAXINT), as follows:

```
shell_stress = dr.get_data(DataType.D3P_SHELL_STRESS, ist=11, ipt=0)
```

Or

```
p = D3P_Parameter()
```

```
p.ist = 11

p.ipt = 0

shell_stress = dr.get_data(DataType.D3P_SHELL_STRESS, p)
```

3. **ipart**: Specify the index of part, starting from 0, as follows:

```
part_name = dr.get_data(DataType.D3P_PART_NAME, ipart=0)

Or

p = D3P_Parameter()

p.ipart = 0

part_name = dr.get_data(DataType.D3P_PART_NAME, p)
```

4. **ipart_user:** Specify the user id of part, as follows:

```
num_shells = dr.get_data(DataType.D3P_NUM_SHELL, ipart_user=3)
```

5. **i_rigid_wall**: Specify the index of rigid wall, starting from 0, as follows:

```
r_wall_f = dr.get_data(

      DataType.D3P_RIGID_WALL_FORCE, ist=11, i_rigid_wall=0

)

Or

p = D3P_Parameter()

p.ist = 11

p.i_rigid_wall = 0

r_wall_f = dr.get_data(DataType.D3P_RIGID_WALL_FORCE, p)
```

6. **ides**: Specify the index of the des data, starting from 0, as follows:

```
num_des = dr.get_data(DataType.D3P_NUM_DES_PART_IN_GEOM, ides=0)

Or

p = D3P_Parameter()

p.ides = 0

num_des = dr.get_data(DataType.D3P_NUM_DES_PART_IN_GEOM, p)
```

7. **ihv**: Specify the index of history variables, starting from 0, as follows:

```
solid_hsvar = dr.get_data(
      DataType.D3P_SOLID_HISTORY_VAR, ist=11, ipt=0, ihv=5
)
Or
p = D3P_Parameter()
p.ist = 11
p.ipt = 0
p.ihv = 5
solid_hsvar = dr.get_data(DataType.D3P_SOLID_HISTORY_VAR, p)
```

8. **index_multisolver**: Specify the index of the multisolver domain, start from 0 and default is 0 also:

```
ms_id = dr.get_data(DataType.D3P_MS_DOMAIN_ID, index_multisolver=0)
Or
p = D3P_Parameter()
p.index_multisolver = 0
ms_id = dr.get_data(DataType.D3P_MS_DOMAIN_ID, p)
```

9. **id_var_multisolver**: Specify the index of the multisolver var, start from 0 and default is 0 also:

```
ms_varn = dr.get_data(DataType.D3P_MS_VAR_NAME, id_var_multisolver=0)
Or
p = D3P_Parameter()
p.id_var_multisolver = 0
ms_varn = dr.get_data(DataType.D3P_MS_VAR_NAME, p)
```

10. **iuser:** Specify the index of the user id:

```
internal_id = dr.get_data(DataType.D3P_SHELL_INTERNAL_ID, iuser=354)
```

11. **var_name**: Specify name of output variables, currently used by DES and CPM data, default is empty

```
cpm_geodt = dr.get_data(DataType.D3P_CPM_GEOM_DATA, var_name='cpm1')
```

```
Or

p = D3P_Parameter()

p.var_name = 'cpm1'

cpm_geodt = dr.get_data(DataType.D3P_CPM_GEOM_DATA, p)
```

DataType

```
class D3P_Vector():
    def x(self):
        pass
    def y(self):
        pass
    def z(self):
        pass


class D3P_VectorDouble():
    def x(self):
        pass
    def y(self):
        pass
    def z(self):
        pass


class D3P_Tensor():
    def x(self):
        pass
    def y(self):
        pass
    def z(self):
        pass
    def xy(self):
        pass
```

```
    def yz(self):
        pass
    def zx(self):
        pass

class D3P_Solid():
    # return the value of nodal index(start from 0) plus one
    def node(self, index):
        pass
    def mat(self):
        pass

class D3P_Tshell():
    # return the value of nodal index(start from 0) plus one
    def node(self, index):
        pass
    def mat(self):
        pass

class D3P_Beam():
    # return the value of nodal index(start from 0) plus one
    def node(self, index):
        pass
    def mat(self):
        pass

class D3P_Shell():
```

```
    # return the value of nodal index(start from 0) plus one
    def node(self, index):
        pass
    def mat(self):
        pass

class D3P_Sph():
    def id(self):
        pass
    def mat(self):
        pass

class D3P_Var():
    def type(self):
        pass
    def name(self):
        pass

class D3P_Des():
    def id(self):
        pass
    def mat(self):
        pass
    def radius(self):
        pass
    def mass(self):
        pass
```

```
def inertia(self):
    pass
```

---

name	conversion	length	parameters
D3P_NUM_STATES	int	1	ignore
D3P_TIMES	float	D3P_NUM_STATES	ignore
D3P_TITLE	Char		ignore
Global			
D3P_GLOBAL_KINETIC_ENERGY	float	1	ist
D3P_GLOBAL_INTERNAL_ENERGY	float	1	ist
D3P_GLOBAL_TOTAL_ENERGY	float	1	ist
D3P_GLOBAL_VELOCITY	D3P_Vector	1	ist
Part			
D3P_NUM_PARTS	int	1	ignore
D3P_PART_IDS	int	D3P_NUM_PARTS	ignore
D3P_PART_NAME	char	80	ipart
D3P_PART_INTERNAL_ENERGY	float	1	ist, ipart
D3P_PART_KINETIC_ENERGY	float	1	ist, ipart
D3P_PART_VELOCITY	D3P_Vector	1	ist, ipart
D3P_PART_MASS	float	1	ist, ipart
D3P_PART_HOURGLASS	float	1	ist, ipart
D3P_PART_INTERNAL_ID	int	1	iuser
RIGID WALL			
D3P_NUM_RIGID_WALL	int	1	ignore
D3P_RIGID_WALL_FORCE	float	1	ist, i_rigid_wall
D3P_RIGID_WALL_POSITION	D3P_Vector	1	ist, i_rigid_wall
NODE			
D3P_NUM_NODES	int	1	ignore
D3P_NODE_INITIAL_COORDINATES	D3P_Vector	D3P_NUM_NODES	ignore

D3P_NODE_IDS	int	D3P_NUM_NODES	ignore
D3P_NODE_TEMPERATURE	float	D3P_NUM_NODES	ist
D3P_NODE_COORDINATES	D3P_Vector	D3P_NUM_NODES	ist
D3P_NODE_VELOCITIES	D3P_Vector	D3P_NUM_NODES	ist
D3P_NODE_ACCELERATIONS	D3P_Vector	D3P_NUM_NODES	ist
D3P_NODE_COORDINATES_DOUBLE	D3P_VectorDouble	D3P_NUM_NODES	ist
D3P_NODE_VELOCITIES_DOUBLE	D3P_VectorDouble	D3P_NUM_NODES	ist
D3P_NODE_ACCELERATIONS_DOUBLE	D3P_VectorDouble	D3P_NUM_NODES	ist
D3P_NODE_INTERNAL_ID	int	1	iuser
SOLID			
D3P_NUM_SOLID	int	1	ignore
D3P_SOLID_MAXINT	int	1	ignore
D3P_SOLID_CONNECTIVITY_MAT	D3P_Solid	D3P_NUM_SOLID	ignore
D3P_SOLID_IDS	int	D3P_NUM_SOLID	ignore
D3P_SOLID_STRESS	D3P_Tensor	D3P_NUM_SOLID	ist, ipt if necessary
D3P_SOLID_EFFECTIVE_PLASTIC_STRAIN	float	D3P_NUM_SOLID	ist, ipt if necessary
D3P_SOLID_STRAIN	D3P_Tensor	D3P_NUM_SOLID	ist, ipt if necessary
D3P_SOLID_HISTORY_VAR	float	D3P_NUM_SOLID	ist, ipt, ihv
D3P_SOLID_INTERNAL_ID	int	1	iuser
TSHELL			
D3P_NUM_TSHELL	int	1	ignore
D3P_TSHELL_MAXINT	int	1	ignore
D3P_TSHELL_CONNECTIVITY_MAT	D3P_Tshell	D3P_NUM_TSHELL	ignore

D3P_TSHELL_IDS	int	D3P_NUM_TSHELL	ignore
D3P_TSHELL_STRESS	D3P_Tensor	D3P_NUM_TSHELL	ist, ipt
D3P_TSHELL_EFFECTIVE_PLASTIC_STRAIN	float	D3P_NUM_TSHELL	ist, ipt
D3P_TSHELL_STRAIN	D3P_Tensor	D3P_NUM_TSHELL	ist, ipt
D3P_TSHELL_HISTORY_VAR	float	D3P_NUM_TSHELL	ist, ipt, ihv
D3P_TSHELL_INTERNAL_ID	int	1	iuser
BEAM			
D3P_NUM_BEAM	int	1	ignore
D3P_BEAM_MAXINT	int	1	ignore
D3P_BEAM_CONNECTIVITY_THIRD_MAT	D3P_Beam	D3P_NUM_BEAM	ignore
D3P_BEAM_IDS	int	D3P_NUM_BEAM	ignore
D3P_BEAM_AXIAL_FORCE	float	D3P_NUM_BEAM	ist
D3P_BEAM_S_SHEAR_RESULTANT	float	D3P_NUM_BEAM	ist
D3P_BEAM_T_SHEAR_RESULTANT	float	D3P_NUM_BEAM	ist
D3P_BEAM_S_BENDING_MOMENT	float	D3P_NUM_BEAM	ist
D3P_BEAM_T_BENDING_MOMENT	float	D3P_NUM_BEAM	ist
D3P_BEAM_TORSIONAL_RESULTANT	float	D3P_NUM_BEAM	ist
D3P_BEAM_RS_SHEAR_STRESS	float	D3P_NUM_BEAM	ist, ipt
D3P_BEAM_TR_SHEAR_STRESS	float	D3P_NUM_BEAM	ist, ipt
D3P_BEAM_AXIAL_STRESS	float	D3P_NUM_BEAM	ist, ipt
D3P_BEAM_AXIAL_PLASTIC_STRAIN	float	D3P_NUM_BEAM	ist, ipt
D3P_BEAM_AXIAL_STRAIN	float	D3P_NUM_BEAM	ist, ipt
D3P_BEAM_HISTORY_VAR	float	D3P_NUM_BEAM	ist, ipt, ihv
D3P_BEAM_INTERNAL_ID	int	1	iuser
SHELL			

D3P_NUM_SHELL	int	1	ignore
D3P_SHELL_MAXINT	int	1	ignore
D3P_SHELL_CONNECTIVITY_MAT	D3P_Shell	D3P_NUM_SHELL	ignore
D3P_SHELL_IDS	int	D3P_NUM_SHELL	ignore
D3P_SHELL_STRESS	D3P_Tensor	D3P_NUM_SHELL	ist, ipt
D3P_SHELL_EFFECTIVE_PLASTIC_STRAIN	float	D3P_NUM_SHELL	ist, ipt
D3P_SHELL_STRAIN	D3P_Tensor	D3P_NUM_SHELL	ist, ipt
D3P_SHELL_HISTORY_VAR	float	D3P_NUM_SHELL	ist, ipt, ihv
D3P_SHELL_MX	float	D3P_NUM_SHELL	ist
D3P_SHELL_MY	float	D3P_NUM_SHELL	ist
D3P_SHELL_MXY	float	D3P_NUM_SHELL	ist
D3P_SHELL_QX	float	D3P_NUM_SHELL	ist
D3P_SHELL_QY	float	D3P_NUM_SHELL	ist
D3P_SHELL_NX	float	D3P_NUM_SHELL	ist
D3P_SHELL_NY	float	D3P_NUM_SHELL	ist
D3P_SHELL_NXY	float	D3P_NUM_SHELL	ist
D3P_SHELL_INTERNAL_ID	int	1	iuser
DELETION			
D3P_HAS_DELETION	bool	1	ist
D3P_SOLID_DELETION	float	D3P_NUM_SOLID	ist
D3P_TSHELL_DELETION	float	D3P_NUM_TSHELL	ist
D3P_SHELL_DELETION	float	D3P_NUM_SHELL	ist
D3P_BEAM_DELETION	float	D3P_NUM_BEAM	ist
SPH			
D3P_NUM_SPH	int	1	ignore
D3P_SPH_NODE_MAT	D3P_Sph	D3P_NUM_SPH	ignore

D3P_SPH_RADIUS	float	D3P_NUM_SPH	ist
D3P_SPH_PRESSURE	float	D3P_NUM_SPH	ist
D3P_SPH_STRESS	D3P_Tensor	D3P_NUM_SPH	ist
D3P_SPH_PLASTIC_STRAIN	float	D3P_NUM_SPH	ist
D3P_SPH_DENSITY	float	D3P_NUM_SPH	ist
D3P_SPH_INTERNAL_ENERGY	float	D3P_NUM_SPH	ist
D3P_SPH_NUMBER_OF_PARTICLE_NEIGHBORS	int	D3P_NUM_SPH	ist
D3P_SPH_STRAIN	D3P_Tensor	D3P_NUM_SPH	ist
D3P_SPH_MASS	float	D3P_NUM_SPH	ist
D3P_SPH_INTERNAL_ID	int	1	iuser
DES			
D3P_HAS_DES_DATA	bool	1	ignore
D3P_NUM_DES_DATA	int	1	ignore
D3P_NUM_DES_PART_IN_GEOM	int	1	ides if necessary
D3P_NUM_DES_ELEM_IN_GEOM	int	1	ides if necessary
D3P_NUM_DES_PART_IN_STATE	int	1	ides if necessary
D3P_NUM_DES_ELEM_IN_STATE	int	1	ides if necessary
D3P_NUM_DES_PART_VAR_IN_GEOM	int	1	ides if necessary
D3P_DES_PART_VAR_LIST_IN_GEOM	D3P_Var	D3P_NUM_DES_PART_VAR_IN_GEOM	ides if necessary
D3P_NUM_DES_ELEM_VAR_IN_GEOM	int	1	ides if necessary
D3P_DES_ELEM_VAR_LIST_IN_GEOM	D3P_Var	D3P_NUM_DES_ELEM_VAR_IN_GEOM	ides if necessary
D3P_NUM_DES_PART_VAR_IN_STATE	int	1	ides if necessary

D3P_DES_PART_VAR_LIST_IN_STATE	D3P_Var	D3P_NUM_DES_PART_VAR_IN_STATE	ides if necessary
D3P_NUM_DES_ELEM_VAR_IN_STATE	int	1	ides if necessary
D3P_DES_ELEM_VAR_LIST_IN_STATE	D3P_Var	D3P_NUM_DES_ELEM_VAR_IN_STATE	ides if necessary
D3P_DES_NODAL_MAT_RADIUS_MASS_INERTIA	D3P_Des	D3P_NUM_DES_ELEM_IN_GEOM	ides if necessary
D3P_DES_DATA_IN_STATE	int/float/vector/tensor..depends	D3P_NUM_DES_ELEM_IN_STATE	var_name, ist, ides if necessary
CPM			
D3P_HAS_CPM_DATA	bool	1	ignore
D3P_CPM_NUM_AIRBAGS	int	1	ignore
D3P_CPM_NUM_PARTICLES	int	1	ignore
D3P_CPM_NUM_GEOM_VAR	int	1	ignore
D3P_CPM_GEOM_VAR_LIST	D3P_Var	D3P_CPM_NUM_GEOM_VAR	ignore
D3P_CPM_GEOM_DATA	D3P_Var	D3P_CPM_NUM_GEOM_VAR	ignore
D3P_CPM_NUM_STATE_VAR	int	1	ignore
D3P_CPM_STATE_VAR_LIST	D3P_Var	D3P_CPM_NUM_STATE_VAR	ignore
D3P_CPM_STATE_DATA	int/float...depends	D3P_CPM_NUM_PARTICLES	var_name, ist
D3P_CPM_NUM_STATE_GEOM_VAR	int	1	ignore
D3P_CPM_STATE_GEOM_VAR_LIST	D3P_Var	D3P_CPM_NUM_STATE_GEOM_VAR	ignore
D3P_CPM_STATE_GEOM_DATA	int/float...depends	D3P_CPM_NUM_AIRBAGS	var_name, ist
Multisolver			

D3P_HAS_MS_DATA	bool	1	ignore
D3P_MS_NUM_DOMAINS	int	1	ignore
D3P_MS_DOMAIN_ID	int	1	index_multisolver
D3P_MS_DOMAIN_NAME	char	80	index_multisolver
D3P_MS_DOMAIN_VAR_NUM	int	1	index_multisolver
D3P_MS_DOMAIN_VARS_LIST	int	D3P_MS_DOMAIN_VAR_NUM	index_multisolver
D3P_MS_VAR_NAME	char	80	id_var_multisolver
D3P_MS_VAR_IS_VECTOR	bool	1	id_var_multisolver
D3P_MS_VAR_IS_SCALAR	bool	1	id_var_multisolver
D3P_MS_VAR_IS_TENSOR	bool	1	id_var_multisolver
D3P_MS_DOMAIN_VAR_LENGTH	int	1	ist, index_multisolver
D3P_MS_DOMAIN_IS_SOLID	bool	1	ist, index_multisolver
D3P_MS_DOMAIN_IS_SHELL	bool	1	ist, index_multisolver
D3P_MS_DOMAIN_IS_BEAM	bool	1	ist, index_multisolver
D3P_MS_DOMAIN_ELEM_NUM_IN_STATE	int	1	ist, index_multisolver
D3P_MS_SOLID_CONNECTIVITY_MAT_IN_STATE	D3P_Solid	D3P_MS_DOMAIN_ELEM_NUM_IN_STATE	ist, index_multisolver

D3P_MS_SHELL_CONNECTIVITY_MAT_IN_STATE	D3P_Shell	D3P_MS_DOMAIN_ELEM_NUM_IN_STATE	ist, index_multi solver
D3P_MS_BEAM_CONNECTIVITY_MAT_IN_STATE	D3P_Beam	D3P_MS_DOMAIN_ELEM_NUM_IN_STATE	ist, index_multi solver
D3P_MS_DOMAIN_NODE_NUM_IN_STATE	int	1	ist, index_multi solver
D3P_MS_DOMAIN_COORD_IN_STATE	D3P_Vector	D3P_MS_DOMAIN_NODE_NUM_IN_STATE	ist, index_multi solver
D3P_MS_DOMAIN_DATA_IN_STATE	float or D3P_Vector or D3P_Tensor	D3P_MS_DOMAIN_VAR_LENGTH	ist, index_multi solver,id_var_multisolver
D3P_MS_DOMAIN_DATA_IS_ON_STRUCTURE_ELEMENT	bool	1	index_multi solver
D3P_MS_DOMAIN_DATA_IS_ON_MS_NODE	bool	1	index_multi solver
D3P_MS_DOMAIN_DATA_IS_ON_MS_ELEMENT,	bool	1	index_multi solver
D3P_MS_DOMAIN_IS_FOLLOW_SURFACE_METHOD	bool	1	index_multi solver
D3P_MS_DOMAIN_NODE_NUM_ONSURFACE_IN_STATE	int	1	ist, index_multi solver
D3P_MS_DOMAIN_SURFACE_IDS_IN_STATE	int	D3P_MS_DOMAIN_NODE_NUM_ONSURFACE_IN_STATE	ist, index_multi solver

How to use

Sample1.py

Purpose: obtain resultant displacement for all the nodes and find maximum value.

3D scatterplot(x=shell_nodes_x, y=shell_nodes_y, z=shell_nodes_z, c=resultant displacement of shell nodes)

ist: last.

```
from lsreader import D3plotReader, DataType as dt
import os
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
from math import pow


d3plot = os.path.join(os.getcwd(), 'd3plot')
dr = D3plotReader(d3plot)


num_states = dr.get_data(dt.D3P_NUM_STATES)
nodes_init_coor = dr.get_data(
    dt.D3P_NODE_INITIAL_COORDINATES, ist=num_states-1
)
nodes_coor = dr.get_data(dt.D3P_NODE_COORDINATES, ist=num_states-1)


# obtain resultant displacement for all nodes and find maximum
nodes_res_disp = []
for i in range(nodes_coor.__len__()):
    disp_x = nodes_coor[i].x() - nodes_init_coor[i].x()
    disp_y = nodes_coor[i].y() - nodes_init_coor[i].y()
    disp_z = nodes_coor[i].z() - nodes_init_coor[i].z()
```

```
    tmp = pow(disp_x, 2) + pow(disp_y, 2) + pow(disp_z, 2)
    nodes_res_disp.append(pow(tmp, 0.5))

print(
"""
Maximum resultant displacement of nodes is: {0}, index is: {1}
""".format(
    max(nodes_res_disp), nodes_res_disp.index(max(nodes_res_disp))
    )
)

# nodes coordinates of shell elements when ist=last
shells = dr.get_data(dt.D3P_SHELL_CONNECTIVITY_MAT)
nodes_shell = []
for shell in shells:
    nodes_shell.append(shell.node(0))
    nodes_shell.append(shell.node(1))
    nodes_shell.append(shell.node(2))
    nodes_shell.append(shell.node(3))
nodes_shell = list(set(nodes_shell))
nodes_shell.sort()
nodes_x, nodes_y, nodes_z, res = [], [], [], []
for node_shell in nodes_shell:
    nodes_x.append(nodes_coor[node_shell-1].x())
    nodes_y.append(nodes_coor[node_shell-1].y())
    nodes_z.append(nodes_coor[node_shell-1].z())
    res.append(nodes_res_disp[node_shell-1])
```

```
# plotting
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1, projection='3d')
scat = ax.scatter3D(
    nodes_x, nodes_y, nodes_z, c=res, s=15,
)
fig.colorbar(scat, label='Resultant Displacement')
ax.set_zlim3d(-50, 50)
plt.show()
```

Sample2.py

Purpose: extract Variable data for Multisolver.

State: 2

```
import lsreader

from lsreader import D3plotReader

from lsreader import DataType as dt

from lsreader import D3P_Parameter as dp

import os


d3plot = os.path.join(os.getcwd(), 'd3plot')

dr = D3plotReader(d3plot)


has_ms_data = dr.get_data(dt.D3P_HAS_MS_DATA)

if not has_ms_data:

    print("No Multisolver Data")

num_ms_datasets = dr.get_data(dt.D3P_MS_NUM_DOMAINS)

for dataset in range(num_ms_datasets):

    domain_var_ids = dr.get_data(dt.D3P_MS_DOMAIN_VARS_LIST, index_multisolver=da
taset)


    for var in range(domain_var_ids.__len__()):

        sizevar = dr.get_data(dt.D3P_MS_DOMAIN_VAR_LENGTH, index_multisolver=data
set, ist=2)

        is_scalar = dr.get_data(dt.D3P_MS_VAR_IS_SCALAR, id_var_multisolver=domai
n_var_ids[var])

        is_vector = dr.get_data(dt.D3P_MS_VAR_IS_VECTOR, id_var_multisolver=domai
n_var_ids[var])
```

```
        is_tensor = dr.get_data(dt.D3P_MS_VAR_IS_TENSOR, id_var_multisolver=domai
n_var_ids[var])

        p = dp()

        p.ist=2

        p.index_multisolver = dataset

        p.id_var_multisolver = domain_var_ids[var]

        if is_scalar:

            svalue = dr.get_data(dt.D3P_MS_DOMAIN_DATA_IN_STATE, p)

            print("Value type: scalar, value[0]={}".format(svalue[0]))

        if is_vector:

            vvalue = dr.get_data(dt.D3P_MS_DOMAIN_DATA_IN_STATE, p)

            print(

                   "Value type: vector, value[0].X()={}"

                   .format(vvalue[0].x())

             )

        if is_tensor:

            tvalue = dr.get_data(dt.D3P_MS_DOMAIN_DATA_IN_STATE, p)

            print(

                   "Value type: tensor, value[0].X()={}"

                   .format(tvalue[0].x())

             )
```

Sample3.py

Purpose: extract stress of shells by part.

State: 2

Ipt: 0

Part User Id: 3

```
from lsreader import D3plotReader, DataType as dt


d3plot = os.path.join(os.getcwd(), 'd3plot')

dr = D3plotReader(d3plot)


num_shells = dr.get_data(dt.D3P_NUM_SHELL)

shell_stress = dr.get_data(

    dt.D3P_SHELL_STRESS, ist=1, ipt=0, ipart_user=3

)

# do something with shell_stress
```

BinoutReader

API Functions

```
class BinoutReader():
    def __init__(self, path):
        pass
```

❖ Purpose: Constructor.

❖ Input: path: binout name.

❖ Return: BinoutReader object.

Example: `br = BinoutReader("binout/file/path")`

```
@staticmethod
def is_valid(path):
    pass
```

❖ Purpose: Check if the path is correct

❖ Input: path: binout name(full path).

❖ Return: True or False.

```
@staticmethod
def write(path, x_array, y_array):
    pass
```

❖ Purpose: Output the x_array and y_array to path.

❖ Input: path: binout name(full path).

x_array: The array of X direction.

y_array: The array of Y direction.

❖ Return: True.

```
def get_branch(self):
    pass
```

❖ Purpose: Get branches.

❖ Input: void.

❖ Return: The array of branches.

```
def set_branch(self, branch):
    pass
```

❖ Purpose: Set current branch.

❖ Input: branch: The name of the branch to set.

❖ Return: True.

```
def set_id(self, id, master):
    pass
```

❖ Purpose: Set current id.

❖ Input: id: The id to set. It can be string or integer.

master: choose master or slave. It can be ignored.

❖ Return: True.

```
def get_id(self):
    pass
```

❖ Purpose: Get ids.

❖ Input: void.

❖ Return: The array of ids.

```
def set_component(component):
    pass
```

❖ Purpose: Set current component.

❖ Input: branch: The name of the component to set.

❖ Return: True.

```
def get_component():
    pass
```

❖ Purpose: Get components.

❖ Input: void.

❖ Return: The array of components.

```
def get_x_array():
    pass
```

❖ Purpose: Get the array of X direction.

❖ Input: void.

❖ Return: The array of X direction.

```
def get_y_array():
    pass
```

❖ Purpose: Get the array of Y direction.

❖ Input: void.

❖ Return: The array of Y direction.

How to use

Sample1.py

Purpose: obtain branches and component, and get x_array, y_array.

Branch: nodout.

Component: x_acceleration.

Id: 1787

Ouput: nodoutPy.dat

```
br = BinoutReader(data_path)


res = BinoutReader.is_valid(data_path)
print(res)


branches = br.get_branch()
for branch in branches:
    print(branch, end=',')


br.set_branch('nodout')
br.set_id(1787)
br.set_component('x_acceleration')
x_array = br.get_x_array()
y_array = br.get_y_array()
out_path = os.path.join(cwd, 'nodoutPy.dat')
BinoutReader.write(out_path, x_array, y_array)
```