

# SUNFLUIDH FILE READER

---

You find below a python package you can use or include in your own post-processing program to load data computed by Sunfluidh. The data files are :

- The instantaneous fields (res\*.d)
- The statistical files (rst\*.d)
- The slice files (slice\*.d)
- The space-averaged data file (spav\*.d)

Information about these files can be found here ([Sunfluidh output files](#)). Information about the file reader is included in the following file.

[fct\\_readsunfluidhdata.py](#)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#=====
=====
# Author   : Yann Fraigneau, CNRS-LIMSI (yann.fraigneau@limsi.fr)
# date      : July 2020
# comments  : Function to read the binary data provided by the code
SUNFLUIDH
#
#           - ReadSunfluidhDataset : main function to read the binary
data of sunfluidh
#           - Check_Dict           : To read a python dictionnary
#                                   useful to read file information
provided by 'ReadSunfluidhDataset'
#           (see an example at the end)
#
#           Keep in mind :
#
#           - Fields have been computed on a staggered grid
(MAC)
#           (to see some scketches :
https://sunfluidh.lisn.upsaclay.fr/doku.php?id=sunfluidh:sunfluidh_tool
s)
#           - scalar quantities are defined at the
cell-centers
#           - for snapshot files only the velocity
components are located at the cell-faces. For other types of file, they
are located at the cell-centers
#           - domain ends are placed at the cell-faces
#           - Cells surrounding the computational
domain are "ghost-cells" only used to define the boundary conditions
#           They can be considered outside of the
computational domain
```

```
# - Coordinates provided in files are associated to
the cell-center points
# - Velocity components can be easily placed at the
cell-centers (as scalar quantities).
# For that, set flag_center= True (see the example
at the end)

#=====
=====

#-----
#--- modules
#-----

import numpy as np
import os
import sys
import glob

#-----
#--- Global variables
#-----

_lst_filetype=['snapshots','slices','statistics','spavslices']

#-----
#--- Functions
#-----

#-----
-----
#-----
-----

def _BuildDataAccess
(dset_type,dir_name,id_subdom,id_time,slice_id,slice_dir,flag_checkexis
t=False):
    """ Private function used by ' ReadSunfluidhDataset'

        Build the data file name and its directory access

        Args IN :
            dset_type (str) : Type of data
            ("snapshots","slices","statistics")
            dir_name (str) : Acces path (where files are stored)
            id_subdom (str) : subdomain rank : "0" or "1" or ... "0"
            if no MPI domain decomposition
            id_time (str) : Time ID ( "0" or "1" or ...)
```

```

        slice_id (str) : slice ID    (for slice files only)
        slice_dir (str) : file orientation  (for slice files only)

    return : data acces (directory/file name) , data file name

"""

if dset_type not in _lst_filetype :
    print(" Define the dataset type to read : \n - slices \n -
snapshots \n - statistics \n - \spavslides")
    print(" --- Stop ---")
    exit()
elif dset_type == _lst_filetype[0] :
    file_name='res_'+id_subdom.zfill(5)+'_'+id_time.zfill(7)+'.d'
elif dset_type == _lst_filetype[1] :
    if slice_id == '0' :
        print("The slice ID is not set")
        print(" -- Stop --")
        exit()
    if slice_dir == '0' :
        print("The slice orientation is not set")
        exit()
    if id_subdom == '0' :
f_name=dir_name+'/'+slice_'+slice_id.zfill(2)+'_'+slice_dir.zfill(1)+'
_'+id_time.zfill(7)+'.d'
        if os.path.exists(f_name) :
file_name='slice_'+slice_id.zfill(2)+'_'+slice_dir.zfill(1)+'_'+id_time
.zfill(7)+'.d'
        else :
file_name='slice_'+slice_id.zfill(2)+'_'+slice_dir.zfill(1)+'_'+id_subd
om.zfill(5)+'_'+id_time.zfill(7)+'.d'
        else :
file_name='slice_'+slice_id.zfill(2)+'_'+slice_dir.zfill(1)+'_'+id_subd
om.zfill(5)+'_'+id_time.zfill(7)+'.d'
    elif dset_type == _lst_filetype[2] :
        file_name='rst_'+id_subdom.zfill(5)+'_'+id_time.zfill(7)+'.d'
    elif dset_type == _lst_filetype[3] :
file_name='spav_slice_'+slice_dir.zfill(1)+'_'+id_time.zfill(7)+'.d'

    data_access=dir_name+'/'+file_name

    flag= True
    if not os.path.exists(data_access) and flag_checkexist:
        print("The data file : ",data_access," does not exist. It is
ignored.")
        flag= False
        exit()

    return data_access,file_name

#-----

```

```
-----
#-----
-----

def _CenteredVelocity (fieldname,field) :

    """ Computation of velocity components at the cell-centre

    IN :
        fieldname (list of str) : list of field names
        field      (numpy array)  : fields

        centered velocity fields are stored in the array "field"
        (overwriting previous velocity fields)

    """

    ib=0 ; ie=field.shape[0]
    jb=0 ; je=field.shape[1]
    kb=0 ; ke=field.shape[2]

    for i,n in enumerate(fieldname):
        if n == "U" :
            field[ib+1:ie,jb:je,kb:ke,i]=(field[ib:ie-1,jb:je,kb:ke,i]
\
+field[ib+1:ie,jb:je,kb:ke,i])*5.e-01
            elif n == "V" :
                field[ib:ie,jb+1:je,kb:ke,i]=(field[ib:ie,jb:je-1,kb:ke,i]
\
+field[ib:ie,jb+1:je,kb:ke,i])*5.e-01
            elif n == "W" and ke > 1 :
                field[ib:ie,jb:je,kb+1:ke,i]=(field[ib:ie,jb:je,kb:ke-1,i]
\
+field[ib:ie,jb:je,kb+1:ke,i])*5.e-01

    return

#-----
#-----
#-----

def ReadSunfluidhDataset
(dset_type="snapshots",dir_name=".",id_subdom=0,id_time=-1,slice_id=1,s
lice_dir=1,selected_fields=[],flag_centered_vel=False) :
    """
        Read the full binary dataset from the suitable file

    * Synopsis :
```

```

    file_info,coordinates,field= ReadSunfluidhDataset (dset_type=
(str), dir_name= (str), id_subdom= (str), id_time(str), .... )

```

\* Keyword arguments (IN) :

```

- dset_type (str, optional) : type of dataset considered
("snapshots", "slices", "statistics"). Default value= "snapshots"
- dir_name (str) : directory where files are located
- id_subdom (int, optional) : ID number of the split data file
(MPI computation with domain decomposition)

```

The default value is zero

```

- id_time (int,optional) : ID number associated to timestep
of file recording. If id_time is < 0 -- or omitted--,
is selected the last file created
(with the greatest value of id_time)

```

```

- slice_id (int,optional) : ID number of slice dataset.
Useful for 'slice' files only. Default value is zero

```

```

- slice_dir (int,optional) : normal direction of slice dataset
(1= I,2= J or 3=K). Useful for 'slice' file only. Default value is
zero

```

```

- selected_fields (list of str, optional) : list of selected
fields

```

```

(default : all fields are selected)

```

see the sunfluidh  
wiki (sunfluidh.limsi.fr) for more details on field names

```

- flag_centered_vel (boolean) : if True, the velocity field
components are interpolated at at the cell-center (as scalar
qunatities)

```

The default value is False

\* Return (OUT) :

```

- fil_info (dict) : relevant data on the
data file (grid size, fields name, ...)

```

contents of this  
dictionnary can be checked with the function "check\_dict"

```

- coordinates (list of 1D numpy array) : coordinates
coordinates[0] :
cell-center coordinates along the I-direction

```

```

coordinates[1] :
cell-center coordinates along the J-direction

```

```

coordinates[2] :
cell-center coordinates along the K-direction

```

```

- field (4D numpy arrays) : arrays of fields
field (i,j,k,n) : with

```

i,j,k : grid indices

n : field ID ranked as the field names in file\_info['field names']

"""

```
file_info={}

#-----
-----
#---- Data access
#-----
-----

if not isinstance(id_time,int) :
    print("Variable 'id_time' must be a integer")
    print("--- Program stops ---")
    exit ()
if not isinstance(id_subdom,int) :
    print("Variable 'id_subdom' must be a integer")
    print("--- Program stops ---")
    exit ()
if not isinstance(slice_id,int) :
    print("Variable 'slice_id' must be a integer")
    print("--- Program stops ---")
    exit ()
if not isinstance(slice_dir,int) :
    print("Variable 'slice_dir' must be a integer")
    print("--- Program stops ---")
    exit ()

if id_time >= 0 :
    data_access,filename=_BuildDataAccess
(dset_type,dir_name,str(id_subdom),str(id_time),str(slice_id),str(slice_dir))
else :
    lst_file=_ExistingFileList (dir_name=dir_name,dset_type=dset_type)
    filename=lst_file[-1]
    data_access=dir_name+'/'+filename
    print()
    print("Automatic selection of the last file created
{}".format(data_access))
    print()

file_info['file name']=filename
file_info['dataset type']=dset_type

#-----
-----
# Structure and size of each data type
#-----
-----
```

```

dt_int32=np.dtype('>i4')
dt_int64=np.dtype('>i8')
dtflt32=np.dtype('>f4')
dtflt64=np.dtype('>f8')
dt_str06=np.dtype('a6')

with open(data_access,"rb") as f :
    #-----
    -----
    #--- Extract The three 1st integer
    #   (Binary file version, number of fields, number of elements
in the file header)
    #-----
    -----

    data=np.fromfile(f,dtype=dt_int32,count=3,offset=0)
    file_info['file version']=np.abs(data[0])
    file_info['number of fields']=data[1]
    #file_info['number of useless elmt']=data[2]

    #-----
    -----
    #--- Extract the data location (useless data at present)
    #-----
    -----

data=np.fromfile(f,dtype=dt_int64,count=data[1]+data[2],offset=0)
#print("useless = ",data)

    #-----
    -----
    #--- Extract the mesh size
    #-----
    -----

    data=np.fromfile(f,dtype=dt_int32,count=3,offset=0)
    file_info['mesh size']=data[0:3]

    #-----
    -----
    #--- Extract some simulation features
    #   single or double real precision, cylindrical geometry,
centered velocity or not
    #-----
    -----

    data=np.fromfile(f,dtype=dt_int32,count=3,offset=0)

    if dset_type == _lst_filetype[1] : data[2] = 1 #--- correction
for slice (already centered 17/01/2023)

```

```
    file_info['real precision']=data[0]
    if file_info['real precision'] == 1 : #--- Single float
precision
        dt_flt= dt_flt32
    else:    #--- Double float precision
        dt_flt= dt_flt64

    file_info['geometry type']=data[1]
    if data[2] == 1 :
        file_info['centered velocity']= True
    else:
        file_info['centered velocity']= False

    #-----
    #--- Extract list of field name
    #-----

    if file_info['geometry type'] == 0:
        file_info['coordinate names']= ["X","Y","Z"]
    elif file_info['geometry type'] == 1:
        file_info['coordinate names']= ["R","Theta","Z"]
    elif file_info['geometry type'] == 2:
        file_info['coordinate names']= ["Z","R","Theta"]
    elif file_info['geometry type'] == 3:
        file_info['coordinate names']= ["Theta","Z","R"]
    data=np.fromfile(f,dtype=dt_str06,count=file_info['number of
fields'],offset=0)
    fieldname=[ data[i].decode('utf-8').replace(" ","") for i in
range(len(data))]

    #-----
    #--- Extract Time
    #-----

    data=np.fromfile(f,dtype=dt_flt,count=1,offset=0)
    file_info['time']= data[0]

    #-----
    #--- Extract coordinates
    #-----

    if file_info['mesh size'][2] == 1 :
        file_info['geometry dimension']=2
```



```

        else:
            file_info['geometry dimension']=3

            coordinates=[0,0,0]
            for n in range(0,3):
coordinates[n]=np.fromfile(f,dtype=dt_flt,count=file_info['mesh
size'])[n],offset=0)
                #print(file_info['coordinate names'][n] , ' start coord :
',coordinates[n][0],\
                #
                'end coord : ',coordinates[n][-1], ' size :
',np.size(coordinates[n]))

            #-----
-----

            #--- Extract specific data related to the dataset type
            #-----
-----

            if dset_type == _lst_filetype[1] :
                data=np.fromfile(f,dtype=dt_int32,count=2,offset=0)
                file_info['slice direction' ]= data[0]
                file_info['slice index' ]= data[1]

            elif dset_type == _lst_filetype[2] :
                data=np.fromfile(f,dtype=dt_int32,count=3,offset=0)
                file_info['average type' ]= data[0]
                file_info['sample' ]= data[1]
                file_info['range number' ]= data[2]
                data=np.fromfile(f,dtype=dt_flt,count=1,offset=0)
                file_info['statistic time' ]= data[0]

            #-----
-----

            #---          FIELDS
            #-----
-----

            nmax=file_info['mesh size'][0]*file_info['mesh
size'][1]*file_info['mesh size'][2] \
                *file_info['number of fields']
            data=np.fromfile(f,dtype=dt_flt,count=nmax,offset=0)
            field=data.reshape(file_info['mesh size'][0],file_info['mesh
size'][1],file_info['mesh size'][2],file_info['number of
fields'],order='F')

            #--- Collect fields selected by the user

            if len(selected_fields) != 0 :
                fieldname,field= SunfluidhData.FieldSelection
(fualllist=fieldname,selectlist=selected_fields,fields=field)

```

```
        file_info['number of fields']= len(fieldname)

    #--- Velocity location at the cell-centre
    if flag_centered_vel and not file_info['centered velocity'] :
        _CenteredVelocity (fieldname=fieldname,field=field)
        file_info['centered velocity']= True

    file_info['field names']= fieldname

    #print("NEW READ")
    #print("SYS byte order      ", sys.byteorder)
    #print("CHECK name          : ",field.dtype.name)
    #print("CHECK byte order : ",field.dtype.byteorder)

    #for n in range(file_info['number of fields']) :
    #    print ("var {}, min {}, max {}".format(fieldname[n],np.min(field[:, :, :, n]),np.max(field[:, :, :, n]))
    #)

    #exit ()

    return file_info,coordinates,field

#-----
#-----
#-----
#-----

def Check_Dict (title= "", dic={}) :
    """ check dictionnary contents """

    print("\nCheck dictionnary : ", title, "\n")
    for key,value in dic.items():
        print('{0:30} : {1:}'.format(key,value))

#-----
#-----
#          TEST
#-----
#-----

if __name__ == "__main__" :

    #-----
    #-----
    #--- Parameters to build the file name to read (arguments for the
    function "ReadSunfluidhDataset")
    #    See here for information about files created by Sunfluidh
```

```

#
(https://sunfluidh.lisn.upsaclay.fr/doku.php?id=sunfluidh:sunfluidh_out
put)
#-----
-----

dirname="/Users/yann/Projets/WORK_SUNFLUIDH/DATABASE_LUCOR/CAS_SQUARECY
L/SNAPSHOTS"      #--- Path to the directory where files are stored
                  #   for example :
dirname="/Users/path_to_datset"
                  #   This parameter is optional and it could be
omitted.
                  #   In this case, default value="." and the
python script must be run in the directory where files are stored

    dsetype="snapshots" #--- Type of file : "snapshots"  -->
instantaneous 3D or 2D fields.
                  #                               Name :
res_idsdom_idtime.d ,
                  #                               "slices",  --> this file
corresponds to a plane extracted from a 3D field (3D simulation only)
                  #                               Name :
slice_sliceid_slicedir_idsdom_idtime.d
                  #                               "statistics" --> (3D or 2D)
averaged fields
                  #                               Name :
rst_idsdom_idtime.d

    idsdom = 0      #--- subdomain ID (if the computation has been
performed in MPI domain decomposition,
                  #                               fields are split in space.
id_sdom is the ID value related to the MPI subdomain ID)
                  #
                  #   This parameter is optional and it could be
omitted. In this case, default value=0
                  #
(unica domain ID for simulation without MPI domain decomposition)

    idtime= 250     #--- time ID of the file (time ID of file).
                  #   This parameter is optional and it could be
omitted.

    sliceid=0       #--- For slice files only : define the rank ID
of the extracted plan as it has been defined in the input data file of
sunfluidh
                  #   This parameter is optional and it could be
omitted (default value=1).

    slicedir=1      #--- For slice files only : orientation of the
extracted plan defined by its normal vector (1 : I-direction, 2 : J-
direction, 3: K-direction)

```

```
#      This parameter is optional and it could be
omitted (default value=1).

    lst_fields=[]      #--- Allow you to select the fields to read in
the file. When the list is empty, all fields are read by default
                        #      ex : lst_fields ['T'] --> select the
temperature only (if it is present)
                        #      see the sunfluidh wiki
(sunfluidh.limsi.fr) for more details on field names
                        #      This parameter is optional and it could be
omitted (default value=[] empty list).

    flag_center= True  #--- The velocity field components are
interpolated at the cell-center (as the scalar quantities)
                        #      This parameter is optional and it could be
omitted (default value=False).

#--- Read file

#..... Put here loops on the variables idtime (or/and idsdom if
needed) according to files to read

    file_info,coordinates,field= ReadSunfluidhDataset
(dset_type=dsetype,dir_name=dirname,id_subdom=idsdom,id_time=idtime, \
slice_id=sliceid,slice_dir=slicedir,selected_fields=lst_fields,
flag_centered_vel= flag_center)

# OUTPUTS :
#--- file_info      : dictionary          : It contains information
describing the data : grid size , fields ...
#--- coordinates : list of 1D arrays : each element of the list
contains the coordinates at the center of cell in a given direction
#                                     : coordinates[0] : in the x-
direction
#                                     : coordinates[1] : in the y-
direction
#                                     : coordinates[2] : in the y-
direction
#--- field          : 4D array field(i,j,k,n)
#                   i,j,k refers the grid indices. For 2D fields,
k=0
#                   n refers the field rank (in the same order as
field names expressed in file_info['field names'])
#                   BEWARE : the array size about spatial indices
(i,j,k) includes ghost-cells
#                   Ghost-Cells surround the discretized
computational(sub)domain,
#                   they are used to treat the boundary
conditions during the computation
```

```
#                                for 'snapshot' or 'statistics' files,
the ghost-cells are the first and last cells in each direction
#                                for 'slice' files, the ghost-cells are
only the last cells in each direction
#                                These ghost-cells are usually not
important except in the case of periodic direction
#                                you can remove these ghost-cells for
data processing
#                                be careful in cases where these ghost-
cell can be relevant : i.e. periodic domains

#--- check the dictionnary file_info which contains information
about the file

Check_Dict (title= "Information collected in dictionnary
'file_info' ", dic=file_info)
```

From:

<https://sunfluidh.lisn.upsaclay.fr/> - Documentation du code de simulation numérique SUNFLUIDH

Permanent link:

[https://sunfluidh.lisn.upsaclay.fr/doku.php?id=sunfluidh:python\\_read\\_sunfluidh\\_files&rev=1706784897](https://sunfluidh.lisn.upsaclay.fr/doku.php?id=sunfluidh:python_read_sunfluidh_files&rev=1706784897)

Last update: 2024/02/01 11:54

