TimeSeriesData

Version 0.1.0

Generated by Doxygen 1.8.13

Contents

1 Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

2 Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

object

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

5 Namespace Documentation

5.1 NH3_H2O_data_processing Namespace Reference

Script to read in all folders of NH3 + H2O Catalyst data.

Functions

- def help_message ()
  Define a help message to display.
- def perform_standard_processing (name_and_path)
  Define a function that reads all data in a given subdirectory and performs standard processing.
- def main (argv)
  Define the 'main' function.

5.1.1 Detailed Description

Script to read in all folders of NH3 + H2O Catalyst data.

Python script to read in CLEERS transient data for for all sets of folders of the NH3 and H2O transient data and perform a series of analyses, compress the data, output the data into a new compressed format, and prepare a series of plots to visualize all data. This script works in conjunction with the transient_data_sets.py script and is meant to be uninteractive (i.e., the user does not need to provide any live inputs beyond calling the script)

Author

Austin Ladshaw

Date

02/27/2020

Copyright

5.1.2 Function Documentation

5.1.2.1 help_message()

```
def NH3_H2O_data_processing.help_message ( )
```

Define a help message to display.

5.1.2.2 perform_standard_processing()

```
def NH3_H2O_data_processing.perform_standard_processing (
              name_and_path )
```

Define a function that reads all data in a given subdirectory and performs standard processing.

name_and_path = input_folder + "/" + data_folder

5.1.2.3 main()

```
def NH3_H2O_data_processing.main (
              argv )
```

Define the 'main' function.

argv is the list of arguments pass to the script at the command line

Accepted arguments include...

```
-h          ==>  display help information
-i dir/     ==>   path and name of the folder than contains other folders of data
-o dir/     ==>   path and name of the folder to place output into (Unsupported)
```

5.2 transient_data Namespace Reference

Read TransientData from CLEERS team.

Classes

- class PairedTransientData

  PairedTransientData This is the object that can automatically pair bypass data with corresponding run data.
- class TransientData

  TransientData This is the basic object to read, operate on, plot, and save transient CLEERS data.

5.2.1 Detailed Description

Read TransientData from CLEERS team.

Python script to read in CLEERS transient data for NH3 storage on Cu-SSZ-13. This script will store the orginal data as is and provide other functions to redistribute, print, or parse that data as needed.

Author

Austin Ladshaw

Date

02/07/2020

Copyright

Note

The CLEERS data files are very, very large, so I am saving them as $*$.dat files. The reasoning behind this is so that I can direct 'git' to ignore files that end with a $*$.dat file extension. This prevents the repository from becoming bloated. The $*$.dat files behave exactly like regular text files.

5.3 transient_data_sets Namespace Reference

Script to read in all sets of CLEERS data of a particular folder.

Classes

- class TransientDataFolder

  TransientDataFolder This object creates a map of other transient data objects (paired or unpaired)

5.3.1 Detailed Description

Script to read in all sets of CLEERS data of a particular folder.

Python script to read in CLEERS transient data for for a particular folder or folders. This script works in tandem with the transient_data.py script which contains the necessary objects for storing and operating on a set of time series data. What this script does is create new objects and methods for dealing with folders filled with similar transient data and performing a series of like actions on all that data and creating output files in sub-folders with the newly processed data.

Author

Austin Ladshaw

Date

02/24/2020

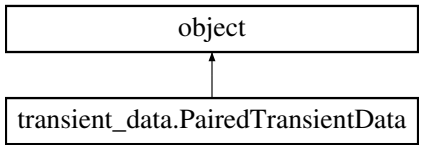
Copyright

6 Class Documentation

6.1 transient_data.PairedTransientData Class Reference

PairedTransientData This is the object that can automatically pair bypass data with corresponding run data.

Inheritance diagram for transient_data.PairedTransientData:



Public Member Functions

- def __init__ (self, bypass_file, result_file)

  Constructor for the paired object requires the bypass and result file.
- def __str__ (self)

  Function to print file information message to console.
- def displayColumnNames (self)

  Function to display the column names to console.
- def compressColumns (self)

  Function to compress columns in both data sets.
- def appendBypassColumn (self, column_name, data_set)

  Function to append a column to by-pass data.
- def appendResultColumn (self, column_name, data_set)

  Function to append a column to result data.
- def extractBypassColumns (self, column_list)

  Function to extract a map of columns from the by-pass data.
- def extractResultColumns (self, column_list)

  Function to extract a map of columns from the result data.
- def extractBypassRows (self, min_time, max_time)

  Function to extract a set of rows from the by-pass data.
- def extractResultRows (self, min_time, max_time)

  Function to extract a set of rows from the result data.
- def getBypassDataPoint (self, time_value, column_name)

  Function to get a data point from the by-pass data.
- def getResultDataPoint (self, time_value, column_name)

  Function to get a data point from the result data.
- def getMaximum (self, column_name)

  Function to get the maximum value of a column (should only use after calling alignData())
- def getMinimum (self, column_name)

  Function to get the minimum value of a column (should only use after calling alignData())
- def getAverage (self, column_name)

  Function to get the average value of a column (should only use after calling alignData())
- def getDataRange (self, column_name)

  Function to get the range of values of a column (should only use after calling alignData())
- def getTimeFrames (self)

  Function to return the list of time ranges.

- def getNumRows (self)

  Function to return the number of rows.
- def getNumCols (self)

  Function to return the number of columns.
- def deleteColumns (self, column_list)

  Function to delete a set of columns from both data sets.
- def retainOnlyColumns (self, column_list)

  Function to delete a set all columns from both data sets, except for the specified set.
- def alignData (self, addNoise=True)

  Function to align the bypass and results data so each has the same number of rows at the appropriate time values.
- def compressRows (self, factor=2)

  Function will compress the rows of data based on the given compression factor.
- def calculateRetentionIntegral (self, column_name, normalized=False, conv_factor=1, input_col_name="")

  Function will compute a mass retention integral for the given column.
- def mathOperation (self, column_name, operator, value_or_column, append_new=False, append_name="")

  This function is used to perform a number of operations using a given column.
- def printAlltoFile (self, file_name="")

  Function will print out the results to an sinle output file.
- def printColumnstoFile (self, column_list, file_name="")

  Function to print only specific columns to an output file.
- def createPlot (self, column_list=[ ], range=None, display=False, save=True, file_name="", file_type=".png", subdir="")

  Function to a create plot from the data_map.
- def savePlots (self, range=None, folder="", file_type=".png")

  Quick use function for saving all processed data plots in a series of output files.
- def saveTimeFramePlots (self, folder="", file_type=".png")

  Function to iteratively save all plots in all time frames separately.

Public Attributes

- bypass_trans_obj

  object for bypass data
- result_trans_obj

  object for result data
- aligned

  Flag used to determine whether or not the data sets are aligned in time.
- file_errors

6.1.1 Detailed Description

PairedTransientData This is the object that can automatically pair bypass data with corresponding run data.

This object is used when you want to 'pair' inlet and outlet data sets together, as well as perform some post-processing such as integrals over data sets. To initialize the data set, you must pass a data file that contains the inlet data. For the CLEERS data sets, an inlet data file is denoted by a "-bp" at the end of the file name as opposed to an isothermal temperature.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 __init__()

```
def transient_data.PairedTransientData.__init__ (
              self,
              bypass_file,
              result_file )
```

Constructor for the paired object requires the bypass and result file.

When creating an instance of this object, you must pass to files to the constructor

(i) A file for the input data (or by-pass run data)
(ii) A file for the output data (or non-by-pass run data)

The constructor will check the file names to make sure that the given information aligns so that the given by-pass and non-by-pass data sets should actually be paired. Because of this check, maintaining the same file name conventions as before is necessary.

Parameters

bypass_file	name of the bypass file
result_file	name of the data run file that needs to be paired with the bypass file

6.1.3 Member Function Documentation

6.1.3.1 __str__()

```
def transient_data.PairedTransientData.__str__ (
              self )
```

Function to print file information message to console.

6.1.3.2 displayColumnNames()

```
def transient_data.PairedTransientData.displayColumnNames (
              self )
```

Function to display the column names to console.

6.1.3.3 compressColumns()

```
def transient_data.PairedTransientData.compressColumns (
              self )
```

Function to compress columns in both data sets.

6.1.3.4 appendBypassColumn()

```
def transient_data.PairedTransientData.appendBypassColumn (
              self,
              column_name,
              data_set )
```

Function to append a column to by-pass data.

6.1.3.5 appendResultColumn()

```
def transient_data.PairedTransientData.appendResultColumn (
              self,
              column_name,
              data_set )
```

Function to append a column to result data.

6.1.3.6 extractBypassColumns()

```
def transient_data.PairedTransientData.extractBypassColumns (
              self,
              column_list )
```

Function to extract a map of columns from the by-pass data.

6.1.3.7 extractResultColumns()

```
def transient_data.PairedTransientData.extractResultColumns (
              self,
              column_list )
```

Function to extract a map of columns from the result data.

6.1.3.8 extractBypassRows()

```
def transient_data.PairedTransientData.extractBypassRows (
              self,
              min_time,
              max_time )
```

Function to extract a set of rows from the by-pass data.

6.1.3.9 extractResultRows()

```
def transient_data.PairedTransientData.extractResultRows (
              self,
              min_time,
              max_time )
```

Function to extract a set of rows from the result data.

6.1.3.10 getBypassDataPoint()

```
def transient_data.PairedTransientData.getBypassDataPoint (
              self,
              time_value,
              column_name )
```

Function to get a data point from the by-pass data.

6.1.3.11 getResultDataPoint()

```
def transient_data.PairedTransientData.getResultDataPoint (
              self,
              time_value,
              column_name )
```

Function to get a data point from the result data.

6.1.3.12 getMaximum()

```
def transient_data.PairedTransientData.getMaximum (
              self,
              column_name )
```

Function to get the maximum value of a column (should only use after calling alignData())

6.1.3.13 getMinimum()

```
def transient_data.PairedTransientData.getMinimum (
              self,
              column_name )
```

Function to get the minimum value of a column (should only use after calling alignData())

6.1.3.14 getAverage()

```
def transient_data.PairedTransientData.getAverage (
              self,
              column_name )
```

Function to get the average value of a column (should only use after calling alignData())

6.1.3.15 getDataRange()

```
def transient_data.PairedTransientData.getDataRange (
              self,
              column_name )
```

Function to get the range of values of a column (should only use after calling alignData())

6.1.3.16 getTimeFrames()

```
def transient_data.PairedTransientData.getTimeFrames (
              self )
```

Function to return the list of time ranges.

6.1.3.17 getNumRows()

```
def transient_data.PairedTransientData.getNumRows (
              self )
```

Function to return the number of rows.

6.1.3.18 getNumCols()

```
def transient_data.PairedTransientData.getNumCols (
              self )
```

Function to return the number of columns.

6.1.3.19 deleteColumns()

```
def transient_data.PairedTransientData.deleteColumns (
              self,
              column_list )
```

Function to delete a set of columns from both data sets.

6.1.3.20 retainOnlyColumns()

```
def transient_data.PairedTransientData.retainOnlyColumns (
              self,
              column_list )
```

Function to delete a set all columns from both data sets, except for the specified set.

6.1.3.21 alignData()

```
def transient_data.PairedTransientData.alignData (
              self,
              addNoise = True )
```

Function to align the bypass and results data so each has the same number of rows at the appropriate time values.

Data is aligned such that the file that contains the most points in time is considered the 'master_set'. In most cases, the result data is the 'master_set' and we are trying to align the bypass data to it. Alignment is done on each time frame within the sets. This function is REQUIRED prior to performing any processing actions (except for compressColumns(), deleteColumns(), and retainOnlyColumns()). Object contains a flag variable to denote whether or not data has been aligned.

RECALL:

```
Time frames are denoted in each time series data file by a repeat of the column names. The time
stamp values at which those repeated column names occur marks the end of the previous time frame.
The bypass file and result file MUST have the same number of time frames in order to align the data.
```

NOTE:

This function is riddled with comment lines and print statements that are used for debugging DO NOT REMOVE ANY COMMENTS UNLESS THE FUNCTION IS FULLY TESTED AND APPROVED (still developing right now) This function is exceedingly complicated, do not modify unless you know what you are doing

NOTE 2:

```
Data in the bypass file is always misaligned in the x-axis, but may also be
misaligned in the y-axis. This was demonstrated in the results data for
NH3 storage at 350 oC where the bypass ppmv measurements were as high as
50 ppmv above the outlet measurements during the actual run. This causes
significant errors when trying to interpret mass retention data. To fix
this issue, data will also be aligned in the y-axis by comparing the
autoregChangedInput() for each column for both bypass and results and
using that information to scale the y-axis bypass information to match
the expected outlet information from each data run.
```

Parameters

addNoise	if True, then the gaps in data are filled in with random noise to simulate the missing information if False, then the gaps in data are filled in with the average value of the last few points of the current time frame.

6.1.3.22 compressRows()

```
def transient_data.PairedTransientData.compressRows (
              self,
              factor = 2 )
```

Function will compress the rows of data based on the given compression factor.

NOTE: Before you can compress the rows of data, each data set must be aligned

6.1.3.23 calculateRetentionIntegral()

```
def transient_data.PairedTransientData.calculateRetentionIntegral (
              self,
              column_name,
              normalized = False,
              conv_factor = 1,
              input_col_name = "" )
```

Function will compute a mass retention integral for the given column.

The align data function must have already been called. That function ensures the data sets for bypass and results are aligned and appends the aligned data from bypass to the results with "[bypass]" added to the end of the file name.

By default, the calculated integral will have the same units as the data in the given column, however, the user may specify to have the integral normalized (thus making the result unitless) and/or may specify a unit conversion factor to multiple the results by in order to get a specific unit outcome.

NOTE:

```
    As an option, the user may specify an input_col_name if that name is
    expected to be different from the given column_name suffixed with
    "[bypass]". May be useful if creating new columns by performing
    unit conversions or other mathOperation() functions.
```

6.1.3.24 mathOperation()

```
def transient_data.PairedTransientData.mathOperation (
              self,
              column_name,
              operator,
              value_or_column,
              append_new = False,
              append_name = "" )
```

This function is used to perform a number of operations using a given column.

The default setting is to operate on the given column to change it's values, however, the user can specify that the result should create a new column to append to the map if desired.

Supported Operations:

```
multiply by a scalar ==> this_column, "*", constant

divide by a scalar   ==> this_column, "/", constant

add a scalar         ==> this_column, "+", constant

subtract a scalar    ==> this_column, "-", constant

multiply by a column ==> this_column, "*", other_column

divide by a column   ==> this_column, "/", other_column

add a column         ==> this_column, "+", other_column

subtract a column    ==> this_column, "-", other_column
```

NOTE: ‘ Multiplying and dividing by a given column will just take all corresponding rows of the first column and perform the operation using the data in the other corresponding row of the other column.

EXAMPLE USAGE:

```
obj.mathOperation("A","*","B")

        this will compute the result of the multiplication of every
            row of A times every row of B and store the result in A

obj.mathOperation("A","+",273,True)

        this will compute the result of every row of A plus 273 and
            store the result in a new column in the map named "A+273"

obj.mathOperation("A","/","B",True,"Res")

        this will compute the result of the division of every row
            of A divided by every row of B and store the result in a
            new column in the map named "Res"
```

6.1.3.25 printAlltoFile()

```
def transient_data.PairedTransientData.printAlltoFile (
              self,
              file_name = "" )
```

Function will print out the results to an sinle output file.

Data must already be aligned. This allows us to only print out information in the results object, since after alignment the results object is linked to the columns in the bypass object.

6.1.3.26 printColumnstoFile()

```
def transient_data.PairedTransientData.printColumnstoFile (
              self,
              column_list,
              file_name = "" )
```

Function to print only specific columns to an output file.

6.1.3.27 createPlot()

```
def transient_data.PairedTransientData.createPlot (
              self,
              column_list = [],
              range = None,
              display = False,
              save = True,
              file_name = "",
              file_type = ".png",
              subdir = "" )
```

Function to a create plot from the data_map.

Options:

```
- column_list: list of columns to create plots of (default is all columns of plottable data)

- range: tuple of the minimum to maximum time values that you want plotted (default is full range)

- display: if True, the images will be displayed once complete

- save: if True, the images will be saved to a file

- file_name: name of the file to save the plot to

- file_type: type of image file to save as (default = .png)
               allowed types: .png, .pdf, .ps, .eps and .svg
```

6.1.3.28 savePlots()

```
def transient_data.PairedTransientData.savePlots (
              self,
              range = None,
              folder = "",
              file_type = ".png" )
```

Quick use function for saving all processed data plots in a series of output files.

Function will automatically pair result data and bypass data together File names will be automatically generated and plots will not be displayed live

6.1.3.29 saveTimeFramePlots()

```
def transient_data.PairedTransientData.saveTimeFramePlots (
              self,
              folder = "",
              file_type = ".png" )
```

Function to iteratively save all plots in all time frames separately.

6.1.4 Member Data Documentation

6.1.4.1 bypass_trans_obj

`transient_data.PairedTransientData.bypass_trans_obj`

object for bypass data

6.1.4.2 result_trans_obj

`transient_data.PairedTransientData.result_trans_obj`

object for result data

6.1.4.3 aligned

`transient_data.PairedTransientData.aligned`

Flag used to determine whether or not the data sets are aligned in time.

6.1.4.4 file_errors

`transient_data.PairedTransientData.file_errors`

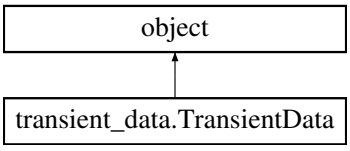
The documentation for this class was generated from the following file:

- transient_data.py

6.2 transient_data.TransientData Class Reference

TransientData This is the basic object to read, operate on, plot, and save transient CLEERS data.

Inheritance diagram for transient_data.TransientData:



Public Member Functions

- def __init__ (self, file)

  Constructor for the class Initialize data object by passing the current file to it Each key in the data_map represents a column label Each key maps to a data list The input file should have a specific convention for naming a file e.g., 20160209-CLRK-BASFCuSSZ13-700C4h-NH3H2Ocomp-30k-0_2pctO2-11-3pctH2O-400ppmNH3-150C.dat.
- def __str__ (self)

  Function to print object information to the console.
- def displayColumnNames (self)

  Function to display the column names to the console.
- def readFile (self)

  Function to read in the data file User does not need to call this function separately, the constructor calls this function for you.
- def closeFile (self)

  Function to manually close the open data file.
- def defineVolume (self, vol)

  Function to set the system volume.
- def defineVoidFraction (self, frac)

  Function to set the system porosity.
- def appendColumn (self, column_name, data_set)

  This function will add a column to the data map given the column name and associated data.
- def extractColumns (self, column_list)

  This function will extract column sets from the data_map and return a new, reduced map.
- def mathOperation (self, column_name, operator, value_or_column, append_new=False, append_name="")

  This function is used to perform a number of operations using a given column The default setting is to operate on the given column to change it's values, however, the user can specify that the result should create a new column to append to the map if desired.
- def extractRows (self, min_time, max_time)

  This function will extract a row of data (or set of rows) based on the value of Elapsed time provided.
- def getDataPoint (self, time_value, column_name)

  This function will get a particular data point based on the given elapsed time and column name.
- def getMaximum (self, column_name)

  Function to retrive the maximum value in a given column.
- def getMinimum (self, column_name)

  Function to retrive the minimum value in a given column.
- def getAverage (self, column_name)

  Function to calculate and retrive the average value in a given column.
- def getDataRange (self, column_name)

  Function to retrive the range of the data in a given column (i.e., max - min point)
- def getTimeFrames (self)

  Function to return the list of time ranges.
- def getNumRows (self)

  Function to return the number of rows.
- def getNumCols (self)

  Function to return the number of columns.
- def compressColumns (self)

  This function will iterate through all columns to find data that can be compressed or eliminated.
- def deleteColumns (self, column_list)

  This function will delete the given columns from the map.
- def retainOnlyColumns (self, column_list)

  This function will delete all columns in the data_map except for the ones specified to retain.
- def printAlltoFile (self, file_name="")

This function is used to print processed data to an output file.

- def printColumnstoFile (self, column_list, file_name="")

  This function is used to print select columns of data to a file.

- def createPlot (self, column_list=[ ], range=None, display=False, save=True, file_name="", file_type=".png", subdir="")

  Function to a create plot from the data_map.

- def savePlots (self, range=None, folder="", file_type=".png")

  Quick use function for saving all processed data plots in a series of output files File names will be automatically generated and plots will not be displayed live.

- def saveTimeFramePlots (self, folder="", file_type=".png")

  Function to iteratively save all plots in all time frames separately.

- def compressRows (self, factor=2)

  This function compresses the rows of the data map New columns are created to store compressed data and old columns are deleted to reserve space.

- def registerChangedInput (self, data_key, input_list)

  This function will take in a data_map key name and a list of changed values to add to the map of input_change for each input that corresponds to an output in data_map for the size of the change_time list.

- def autoregChangedInput (self, data_key, avg_points=10, non_neg=True)

  This function automates the above function by utiliizing the corresponding output information of the given data_key to automatically approximate the input data for each change_time.

- def createStepChangeInputData (self, data_key, avg_points=10, non_neg=True)

  This function will create a new column in the data_map by creating a step-wise set of input data based on the change_time and corresponding input_change information.

- def calculateRetentionIntegral (self, inlet_column, outlet_column, normalized=False, conv_factor=1)

  This function will perform a trapezoid rule integration between two given curves in the data_map versus the time_key set of data.

Public Attributes

- input_file_name
- material_name
- aging_condition
- flow_rate
- have_flow_rate
- inlet_data
- isothermal_temp
- data_file

  Contains data file we are digitizing.

- exp_header

  Contains the first line of the data file.

- data_map

  Contains a map of all the data by column.

- num_rows

  Contains the number of rows of data.

- ordered_key_list

  Contains an ordered list of column names.

- change_time

  Contains an ordered list of the times when experimental inputs changed.

- time_frames

  Contains a list of tuples that hold the start and end time for each instance in the data where inputs were changed.

- input_change

  Contains a map of the inputs values that correspond to change_time.

- time_key

  Contains the name of the time key for the data_map.

- sys_vol

  Volume of the system over which data was gathered (in L)

- void_frac

  Void volume fraction for the total system volume.

6.2.1 Detailed Description

TransientData This is the basic object to read, operate on, plot, and save transient CLEERS data.

This python class is responsible for reading in a set of tab delimited time series data files generated by the CLEERS Team. Currently, the class is designed to look for some key specific formatting quirks in the data file that is common to all CLEERS data sets (at the moment). Those formatting quirks are the following...

(1) The first line of every file contains a generic header that contains no data (2) All data contained within the file starts at the second line (3) All data is in a series of columns and each column has a unique header name That header name is used to identify what the data below it is and is used in this object to create a mapping of the data. (4) Every so often, the column names are reinserted into the rows of data. This is used to indicate when a change in the input conditions of a data run has occurred. This class is aware that these changes can and do happen, so it uses that information to parse each set of data (in a same run) under a series of data frames. (5) When a column has a header/name, but no data beneath it, that column is ignored (6) At least one column in the data file must be labeled as "Elapsed Time (unit)" Where 'unit' is any time units.

EXAMPLE:

first header contains ignored text
Elapsed Time (min) NH3 (300) NH3 (3000) T (C)
0.0 0.01 1.2 151.2
0.16 -0.9 2.0 150.5
Elapsed Time (min) NH3 (300) NH3 (3000) T (C)
0.32 30.1 34.2 151.6
0.64 35.6 41.0 149.7

In this example, there are 2 columns containing NH3 data, but is measured by instruments calibrated with different tolerances labeled by the numbers in parentheses. This set also includes the "Elapsed Time (min)" column name as required, then the set of column names repeats halfway down the file. This repeat is how changes in input conditions are marked. Notice that the NH3 values suddenly jump at this point. That sudden jump is correlated with the repeat of the column names. The data frames created are then from time 0.0 to 0.16 (as the first frame) and from time 0.32 to 0.64 (as the second frame).

6.2.2 Constructor & Destructor Documentation

6.2.2.1 __init__()

```
def transient_data.TransientData.__init__ (
              self,
              file )
```

Constructor for the class Initialize data object by passing the current file to it Each key in the data_map represents a column label Each key maps to a data list The input file should have a specific convention for naming a file e.g., 20160209-CLRK-BASFCuSSZ13-700C4h-NH3H2Ocomp-30k-0_2pctO2-11-3pctH2O-400ppmNH3-150C.dat.

Each important piece of information is split by a "-" character. We then use this to parse the file name to obtain particular information. However, that file name convention is not necessarily consistent for all files. So we are limited in what can be interpreted from the names. The most important information is as follows...

```
item[2] = Name of the catalyst material
item[3] = Aging condition
item[5] = flow rate information (in kilo-volumes per hour)
item[-1] = either a temperature or "bp"
            temperature is irrelevant, but bp indicates that this is inlet information
            (Also note, item[-1] will carry the file extension with it)
```

Parameters

file	the name of the data file we are reading

6.2.3 Member Function Documentation

6.2.3.1 __str__()

```
def transient_data.TransientData.__str__ (
              self )
```

Function to print object information to the console.

6.2.3.2 displayColumnNames()

```
def transient_data.TransientData.displayColumnNames (
              self )
```

Function to display the column names to the console.

This method is particularly useful when using this python script interactively as it will tell you all the columns by name that you can have access to. To get a specific column, or other functions that require a specific column, you use the names of the columns displayed by this function.

6.2.3.3 readFile()

```
def transient_data.TransientData.readFile (
              self )
```

Function to read in the data file User does not need to call this function separately, the constructor calls this function for you.

6.2.3.4 closeFile()

```
def transient_data.TransientData.closeFile (
              self )
```

Function to manually close the open data file.

6.2.3.5 defineVolume()

```
def transient_data.TransientData.defineVolume (
              self,
              vol )
```

Function to set the system volume.

6.2.3.6 defineVoidFraction()

```
def transient_data.TransientData.defineVoidFraction (
              self,
              frac )
```

Function to set the system porosity.

6.2.3.7 appendColumn()

```
def transient_data.TransientData.appendColumn (
              self,
              column_name,
              data_set )
```

This function will add a column to the data map given the column name and associated data.

NOTE: Appending the column does NOT copy the column into the map. It merely directs the map to point to the given data_set. If you change the data_set that you pass to this function, then the data in this object's map will also change.

6.2.3.8 extractColumns()

```
def transient_data.TransientData.extractColumns (
              self,
              column_list )
```

This function will extract column sets from the data_map and return a new, reduced map.

NOTE:

```
Column list must be a list of valid keys in the data_map
```

6.2.3.9 mathOperation()

```
def transient_data.TransientData.mathOperation (
              self,
              column_name,
              operator,
              value_or_column,
              append_new = False,
              append_name = "" )
```

This function is used to perform a number of operations using a given column The default setting is to operate on the given column to change it's values, however, the user can specify that the result should create a new column to append to the map if desired.

Supported Operations:

```
multiply by a scalar ==> this_column, "*", constant

divide by a scalar   ==> this_column, "/", constant

add a scalar         ==> this_column, "+", constant

subtract a scalar    ==> this_column, "-", constant

multiply by a column ==> this_column, "*", other_column

divide by a column   ==> this_column, "/", other_column

add a column         ==> this_column, "+", other_column

subtract a column    ==> this_column, "-", other_column
```

NOTE:

```
    Multiplying and dividing by a given column will just take all
    corresponding rows of the first column and perform the operation
    using the data in the other corresponding row of the other column.
```

EXAMPLES:

```
obj.mathOperation("A","*","B")

        this will compute the result of the multiplication of every
            row of A times every row of B and store the result in A

obj.mathOperation("A","+",273,True)

        this will compute the result of every row of A plus 273 and
            store the result in a new column in the map named "A+273"

obj.mathOperation("A","/","B",True,"Res")

        this will compute the result of the division of every row
            of A divided by every row of B and store the result in a
            new column in the map named "Res"
```

6.2.3.10 extractRows()

```
def transient_data.TransientData.extractRows (
              self,
              min_time,
              max_time )
```

This function will extract a row of data (or set of rows) based on the value of Elapsed time provided.

6.2.3.11 getDataPoint()

```
def transient_data.TransientData.getDataPoint (
              self,
              time_value,
              column_name )
```

This function will get a particular data point based on the given elapsed time and column name.

6.2.3.12 getMaximum()

```
def transient_data.TransientData.getMaximum (
              self,
              column_name )
```

Function to retrive the maximum value in a given column.

6.2.3.13 getMinimum()

```
def transient_data.TransientData.getMinimum (
              self,
              column_name )
```

Function to retrive the minimum value in a given column.

6.2.3.14 getAverage()

```
def transient_data.TransientData.getAverage (
              self,
              column_name )
```

Function to calculate and retrive the average value in a given column.

6.2.3.15 getDataRange()

```
def transient_data.TransientData.getDataRange (
              self,
              column_name )
```

Function to retrive the range of the data in a given column (i.e., max - min point)

6.2.3.16 getTimeFrames()

```
def transient_data.TransientData.getTimeFrames (
              self )
```

Function to return the list of time ranges.

6.2.3.17 getNumRows()

```
def transient_data.TransientData.getNumRows (
              self )
```

Function to return the number of rows.

6.2.3.18 getNumCols()

```
def transient_data.TransientData.getNumCols (
              self )
```

Function to return the number of columns.

6.2.3.19 compressColumns()

```
def transient_data.TransientData.compressColumns (
              self )
```

This function will iterate through all columns to find data that can be compressed or eliminated.

For instance,

```
            if a column contains no data, then delete it

            if there are multiple columns that carrier similar info, then combine them
```

USAGE:

```
            When we have sets of data that are measuring the same thing, but at
            different levels of tolerances, we can use this function to select the
            best information to keep based on those tolerances.
```

EXAMPLE:

```
A column NH3 (300) and another column NH3 (3000) would be combined into a single
column named NH3 (300,3000) and use the data from either of the two columns depending
on whether or not the recorded data went over the tolerance of the instrument and which
of the recordings were closest to that tolerance level.
```

6.2.3.20 deleteColumns()

```
def transient_data.TransientData.deleteColumns (
              self,
              column_list )
```

This function will delete the given columns from the map.

6.2.3.21 retainOnlyColumns()

```
def transient_data.TransientData.retainOnlyColumns (
              self,
              column_list )
```

This function will delete all columns in the data_map except for the ones specified to retain.

6.2.3.22 printAlltoFile()

```
def transient_data.TransientData.printAlltoFile (
              self,
              file_name = "" )
```

This function is used to print processed data to an output file.

6.2.3.23 printColumnstoFile()

```
def transient_data.TransientData.printColumnstoFile (
              self,
              column_list,
              file_name = "" )
```

This function is used to print select columns of data to a file.

6.2.3.24 createPlot()

```
def transient_data.TransientData.createPlot (
              self,
              column_list = [],
              range = None,
              display = False,
              save = True,
              file_name = "",
              file_type = ".png",
              subdir = "" )
```

Function to a create plot from the data_map.

Options:

```
- column_list: list of columns to create plots of (default is all columns of plottable data)

- range: tuple of the minimum to maximum time values that you want plotted (default is full range)

- display: if True, the images will be displayed once complete

- save: if True, the images will be saved to a file

- file_name: name of the file to save the plot to

- file_type: type of image file to save as (default = .png)
                allowed types: .png, .pdf, .ps, .eps and .svg
```

6.2.3.25 savePlots()

```
def transient_data.TransientData.savePlots (
              self,
              range = None,
              folder = "",
              file_type = ".png" )
```

Quick use function for saving all processed data plots in a series of output files File names will be automatically generated and plots will not be displayed live.

6.2.3.26 saveTimeFramePlots()

```
def transient_data.TransientData.saveTimeFramePlots (
              self,
              folder = "",
              file_type = ".png" )
```

Function to iteratively save all plots in all time frames separately.

6.2.3.27 compressRows()

```
def transient_data.TransientData.compressRows (
              self,
              factor = 2 )
```

This function compresses the rows of the data map New columns are created to store compressed data and old columns are deleted to reserve space.

The 'factor' argument is optional and is used to determine how much compression to use

```
Default is 2x compression: ==>  Cuts number of rows in half
```

6.2.3.28 registerChangedInput()

```
def transient_data.TransientData.registerChangedInput (
              self,
              data_key,
              input_list )
```

This function will take in a data_map key name and a list of changed values to add to the map of input_change for each input that corresponds to an output in data_map for the size of the change_time list.

NOTE:

```
    Make sure you give the same units for the data in the input_list
    as the units provided in the corresponding output in data_map
```

6.2.3.29 autoregChangedInput()

```
def transient_data.TransientData.autoregChangedInput (
              self,
              data_key,
              avg_points = 10,
              non_neg = True )
```

This function automates the above function by utiliizing the corresponding output information of the given data_key to automatically approximate the input data for each change_time.

That input data is estimated by averaging the last few output data points within the corresponding time range. By default, the last few data points are taken as the last 10 data points, however, you can override this by simply calling this function with a different value for avg_points. You may also specify whether or not the inlet conditions for this data set should be non-negative (e.g., for things such as inlet concentrations or molefractions)

6.2.3.30 createStepChangeInputData()

```
def transient_data.TransientData.createStepChangeInputData (
              self,
              data_key,
              avg_points = 10,
              non_neg = True )
```

This function will create a new column in the data_map by creating a step-wise set of input data based on the change_time and corresponding input_change information.

When calling this function, it is unnecessary to call registerChangedInput as this function will automatically perform the associated actions of that function.

```
NOTE: data_key can be a single column or a list of columns
```

6.2.3.31 calculateRetentionIntegral()

```
def transient_data.TransientData.calculateRetentionIntegral (
              self,
              inlet_column,
              outlet_column,
              normalized = False,
              conv_factor = 1 )
```

This function will perform a trapezoid rule integration between two given curves in the data_map versus the time_key set of data.

The first value of the integrated curve is always assumed to be zero. The units for the given columns do not matter, resulting curve will have same units as the given columns for inlet and outlet. Generally, this function is used to create a data column for Mass Retained in the catalyst.

User may pass an additional argument to determine whether or not they want the integral to be normalized. When normalized, the resulting column becomes unitless and is normalized to the magnitude of the maximum integrated value.

In addition to normalization, user may also provide a conversion factor to the calculated integral. The conversion factor can be used to scaled the normalized curve to a desired maximum or minimum, or can be used as a way to convert the units of the result from it's starting units to whatever the user desires.

For instance, if the units of the given column are in ppmv, but you want the result to come out to mol/L, then your conversion factor would be...

```
conv_factor = (1/10^6)*P/8.314/T

            where P is total pressure in kPa and T is temperature in K
```

As another unit conversion example, let's say we want the result to come out in typical adsorption units: mol adsorbed / L catalyst. Then, the conversion factor would be like above, but we would also multiple by the total system volume and divide by the solid volume of the catalyst.

```
conv_factor = (1/10^6)*P/8.314/T*(V_tot/V_cat)
```

The following relationship is assumed...

d(MR)/dt = Q∗(Min - Mout)

Min = Mass in (given data column to represent inlet mass) Mout = Mass out (given data column to represent outlet mass) Q = flow rate (usually as space velocity [hr^-1]) MR = Mass retained in the catalyst (Representative of adsorbed mass)

6.2.4 Member Data Documentation

6.2.4.1 input_file_name

```
transient_data.TransientData.input_file_name
```

6.2.4.2 material_name

transient_data.TransientData.material_name

6.2.4.3 aging_condition

transient_data.TransientData.aging_condition

6.2.4.4 flow_rate

transient_data.TransientData.flow_rate

6.2.4.5 have_flow_rate

transient_data.TransientData.have_flow_rate

6.2.4.6 inlet_data

transient_data.TransientData.inlet_data

6.2.4.7 isothermal_temp

transient_data.TransientData.isothermal_temp

6.2.4.8 data_file

transient_data.TransientData.data_file

Contains data file we are digitizing.

6.2.4.9 exp_header

transient_data.TransientData.exp_header

Contains the first line of the data file.

6.2.4.10 data_map

`transient_data.TransientData.data_map`

Contains a map of all the data by column.

6.2.4.11 num_rows

`transient_data.TransientData.num_rows`

Contains the number of rows of data.

6.2.4.12 ordered_key_list

`transient_data.TransientData.ordered_key_list`

Contains an ordered list of column names.

6.2.4.13 change_time

`transient_data.TransientData.change_time`

Contains an ordered list of the times when experimental inputs changed.

NOTE:

```
this is just used to initialize data in the map, it does not
change or update if the map changes
```

6.2.4.14 time_frames

`transient_data.TransientData.time_frames`

Contains a list of tuples that hold the start and end time for each instance in the data where inputs were changed.

This item may be slightly redundant with change_time, but will be more useful to end users who might want to create separate plots for each time frame

6.2.4.15 input_change

```
transient_data.TransientData.input_change
```

Contains a map of the inputs values that correspond to change_time.

Keys of this map are modifications to the keys of data_map that correspond to output values corresponding to the input values given

Specific values on input can be user specified or auotmatically generated based on the last few data points in the corresponding columns.

NOTE:

```
When pairing bypass data with result data, this object will not be
used unless the data alignment function needs it.
```

6.2.4.16 time_key

```
transient_data.TransientData.time_key
```

Contains the name of the time key for the data_map.

Key must contain "Elapsed Time (min)" in the data file unit can be any time units, but the first part of the string must alwas exist as "Elapsed Time (..."

```
(NOTE: We assume that elapsed time has units of minutes)
```

6.2.4.17 sys_vol

```
transient_data.TransientData.sys_vol
```

Volume of the system over which data was gathered (in L)

Space volume of the column for the catalyst User must manually assign a value, if needed

6.2.4.18 void_frac

```
transient_data.TransientData.void_frac
```

Void volume fraction for the total system volume.

This would represent the overall bulk porosity of the catalyst. User must manually override this value if needed.

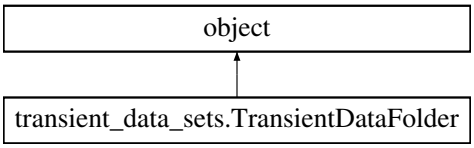
The documentation for this class was generated from the following file:

- transient_data.py

6.3 transient_data_sets.TransientDataFolder Class Reference

TransientDataFolder This object creates a map of other transient data objects (paired or unpaired)

Inheritance diagram for transient_data_sets.TransientDataFolder:



Public Member Functions

- def __init__ (self, folder, addNoise=True)

  Initialize the object by reading in all readable data.
- def __str__ (self)

  Print object attributes to the console.
- def displayColumnNames (self)

  Display names of all columns for all data sets.
- def deleteColumns (self, column_list)

  Delete specific columns from all data sets that we do not need.
- def retainOnlyColumns (self, column_list)

  Retain on specific columns from all data sets.
- def grabDataObj (self, file)

  Function to return an instance of the TransientData or PairedTransientData object given a file name.
- def getTotalDataProcessed (self)

  Function to report total data processed.
- def compressAllRows (self, num_rows_target=1000, max_factor=10)

  Function to compress all rows of data for each data object according to size of rows.
- def mathOperations (self, column_name, operator, value_or_column, append_new=False, append_name="")

  Function to perform a math operation to all like columns of data in all data objects.
- def isPaired (self, file_name)

  Function to determine whether or not the given file name is paired or unpaired.
- def calculateRetentionIntegrals (self, column_name, normalized=False, conv_factor=1, input_col_name="")

  Function to compute a mass retention integral for all columns of the given name.
- def printAlltoFile (self, subdir="")

  Function to print all results (paired and unpaired) to a series of output files in a sub-directory.
- def createPlot (self, obj_name, column_list=[ ], range=None, display=False, save=True, file_name="", file_↩ type=".png", subdir="")

  Function to create plots from columns of data.
- def savePlots (self, range=None, file_type=".png")

  Function to save all plots of data to several files.
- def saveTimeFramePlots (self, folder="", file_type=".png")

  Function to iteratively save all plots in all time frames separately.

Public Attributes

- folder_name
- file_names
  List of file names for non-bypass files.
- bypass_names
  List of file names for bypass files.
- file_pairs
  Map of paired files: key ==> file_base_name –> (bypass_file, result_file)
- unpaired_data
  Map of Transient Data objects by file_name (unpaired)
- paired_data
  Map of Paired Transient Data objects by file_name (paired)
- has_unpaired
  Flag to denote whether or not folder contains unpaired data.
- has_paired
  Flag to denote whether or not folder contains paired data (note: CAN have both)
- unread
  List to correlate with all file_names to determine if a file has been read or not.
- was_compressed
  Flag to denote whether or not user has requested row compression of data sets.
- total_data_processed
  Running total of the number of data points processed (based on rows and columns)

6.3.1 Detailed Description

TransientDataFolder This object creates a map of other transient data objects (paired or unpaired)

This object will iteratively read all data files in a given folder and store that information into TransientData or Paired↩ TransientData objects depending on the typical file name flags used to distinguish between bypass runs and actual data runs. The data in each file of the folder can have a bypass pairing or can be solitary. HOWEVER, all data files should have all the same common column names.

NOTE:

```
ALL data files in the folder should have the same column name formatting!
 This is so that we can process entire folders of data en masse by performing
 the same set of actions across all data sets to a particular column.
 IF DATA HAS DIFFERENT COLUMN NAMES, IT SHOULD BE IN DIFFERENT FOLDER!
```

For instance:

```
If we want to perform a data reduction analysis and extract only
 a specific sub-set of columns from all the files, then those
 column names need to be all the same. That way we can use a single
 function call/operation to perform the analysis on all data.
```

6.3.2 Constructor & Destructor Documentation

6.3.2.1 __init__()

```
def transient_data_sets.TransientDataFolder.__init__ (
              self,
              folder,
              addNoise = True )
```

Initialize the object by reading in all readable data.

Parameters

folder	name of the folder that contains sets of data files
addNoise	whether or not to add random noise for missing data emulation

NOTE:

```
The code expects that the folder only contains a set of CLEERS data files.
If there are non-CLEERS data files or sub-folders, then this may cause errors.
```

6.3.3 Member Function Documentation

6.3.3.1 __str__()

```
def transient_data_sets.TransientDataFolder.__str__ (
              self )
```

Print object attributes to the console.

6.3.3.2 displayColumnNames()

```
def transient_data_sets.TransientDataFolder.displayColumnNames (
              self )
```

Display names of all columns for all data sets.

NOTE:

ASSUMES ALL DATA SETS HAVE SAME COMMON COLUMN NAMES

6.3.3.3 deleteColumns()

```
def transient_data_sets.TransientDataFolder.deleteColumns (
              self,
              column_list )
```

Delete specific columns from all data sets that we do not need.

6.3.3.4 retainOnlyColumns()

```
def transient_data_sets.TransientDataFolder.retainOnlyColumns (
              self,
              column_list )
```

Retain on specific columns from all data sets.

6.3.3.5 grabDataObj()

```
def transient_data_sets.TransientDataFolder.grabDataObj (
              self,
              file )
```

Function to return an instance of the TransientData or PairedTransientData object given a file name.

NOTE:

```
This function returns different data types depending on the argument it gets
```

6.3.3.6 getTotalDataProcessed()

```
def transient_data_sets.TransientDataFolder.getTotalDataProcessed (
              self )
```

Function to report total data processed.

6.3.3.7 compressAllRows()

```
def transient_data_sets.TransientDataFolder.compressAllRows (
              self,
              num_rows_target = 1000,
              max_factor = 10 )
```

Function to compress all rows of data for each data object according to size of rows.

NOTE:

```
This function should be called before printing to a file, but after everything else
```

This function accepts 2 optional arguments:

```
(i) num_rows_target
(ii) max_factor
```

num_rows_target:

```
        is used to represent the number of rows of data we
        want to compress the data to. For instance, num_rows_target = 1000
        means that after compression, we want the data to fit within about
        1000 rows of data.
```

max_factor:

```
         puts a cap on the maximum level of compression we will allow,
        regardless of the num_rows_target. For intance, a max_factor of 10
        means that the number of data rows will be reduced by a factor of 10
        at most, and no more than that.
```

6.3.3.8 mathOperations()

```
def transient_data_sets.TransientDataFolder.mathOperations (
              self,
              column_name,
              operator,
              value_or_column,
              append_new = False,
              append_name = "" )
```

Function to perform a math operation to all like columns of data in all data objects.

6.3.3.9 isPaired()

```
def transient_data_sets.TransientDataFolder.isPaired (
              self,
              file_name )
```

Function to determine whether or not the given file name is paired or unpaired.

Returns True if paired, False if unpaired or doesn't exist

6.3.3.10 calculateRetentionIntegrals()

```
def transient_data_sets.TransientDataFolder.calculateRetentionIntegrals (
              self,
              column_name,
              normalized = False,
              conv_factor = 1,
              input_col_name = "" )
```

Function to compute a mass retention integral for all columns of the given name.

NOTE:

```
    This function SHOULD automatically handle both paired and unpaired data sets
```

6.3.3.11 printAlltoFile()

```
def transient_data_sets.TransientDataFolder.printAlltoFile (
              self,
              subdir = "" )
```

Function to print all results (paired and unpaired) to a series of output files in a sub-directory.

6.3.3.12 createPlot()

```
def transient_data_sets.TransientDataFolder.createPlot (
              self,
              obj_name,
              column_list = [],
              range = None,
              display = False,
              save = True,
              file_name = "",
              file_type = ".png",
              subdir = "" )
```

Function to create plots from columns of data.

Options:

```
- obj_name: name of the file/obj for which the data we are plotting is held

- column_list: list of columns to create plots of (default is all columns of plottable data)

- range: tuple of the minimum to maximum time values that you want plotted (default is full range)

- display: if True, the images will be displayed once complete

- save: if True, the images will be saved to a file

- file_name: name of the file to save the plot to

- file_type: type of image file to save as (default = .png)
               allowed types: .png, .pdf, .ps, .eps and .svg
```

6.3.3.13 savePlots()

```
def transient_data_sets.TransientDataFolder.savePlots (
              self,
              range = None,
              file_type = ".png" )
```

Function to save all plots of data to several files.

Function will automatically pair result data and bypass data together File names will be automatically generated and plots will not be displayed live Folder names are choosen automatically as well

6.3.3.14 saveTimeFramePlots()

```
def transient_data_sets.TransientDataFolder.saveTimeFramePlots (
              self,
              folder = "",
              file_type = ".png" )
```

Function to iteratively save all plots in all time frames separately.

6.3.4 Member Data Documentation

6.3.4.1 folder_name

transient_data_sets.TransientDataFolder.folder_name

6.3.4.2 file_names

transient_data_sets.TransientDataFolder.file_names

List of file names for non-bypass files.

6.3.4.3 bypass_names

transient_data_sets.TransientDataFolder.bypass_names

List of file names for bypass files.

6.3.4.4 file_pairs

transient_data_sets.TransientDataFolder.file_pairs

Map of paired files: key ==> file_base_name –> (bypass_file, result_file)

NOTE:

file_base_name is taken as the name of the file without the "-bp.dat" suffix

6.3.4.5 unpaired_data

transient_data_sets.TransientDataFolder.unpaired_data

Map of Transient Data objects by file_name (unpaired)

6.3.4.6 paired_data

transient_data_sets.TransientDataFolder.paired_data

Map of Paired Transient Data objects by file_name (paired)

6.3.4.7 has_unpaired

`transient_data_sets.TransientDataFolder.has_unpaired`

Flag to denote whether or not folder contains unpaired data.

6.3.4.8 has_paired

`transient_data_sets.TransientDataFolder.has_paired`

Flag to denote whether or not folder contains paired data (note: CAN have both)

6.3.4.9 unread

`transient_data_sets.TransientDataFolder.unread`

List to correlate with all file_names to determine if a file has been read or not.

6.3.4.10 was_compressed

`transient_data_sets.TransientDataFolder.was_compressed`

Flag to denote whether or not user has requested row compression of data sets.

6.3.4.11 total_data_processed

`transient_data_sets.TransientDataFolder.total_data_processed`

Running total of the number of data points processed (based on rows and columns)

The documentation for this class was generated from the following file:

- transient_data_sets.py

7 File Documentation

7.1 NH3_H2O_data_processing.py File Reference

Namespaces

- NH3_H2O_data_processing

  Script to read in all folders of NH3 + H2O Catalyst data.

Functions

- def NH3_H2O_data_processing.help_message ()

  Define a help message to display.
- def NH3_H2O_data_processing.perform_standard_processing (name_and_path)

  Define a function that reads all data in a given subdirectory and performs standard processing.
- def NH3_H2O_data_processing.main (argv)

  Define the 'main' function.

7.2 transient_data.py File Reference

Classes

- class transient_data.TransientData

  TransientData This is the basic object to read, operate on, plot, and save transient CLEERS data.
- class transient_data.PairedTransientData

  PairedTransientData This is the object that can automatically pair bypass data with corresponding run data.

Namespaces

- transient_data

  Read TransientData from CLEERS team.

7.3 transient_data_sets.py File Reference

Classes

- class transient_data_sets.TransientDataFolder

  TransientDataFolder This object creates a map of other transient data objects (paired or unpaired)

Namespaces

- transient_data_sets

  Script to read in all sets of CLEERS data of a particular folder.

Index