PKP Code Documentation

Release 1.0.0

Martin Pollack

December 12, 2012

CONTENTS

This is the code documentation of the PKP program. The general structur of the classes is the following:

- There are pyrolysis program (like CPD and FG-DVC) specific ones to write the configuration files, launch their program and read the specific output files and support the other classes with the pyrolysis program specific data. Every of the programs has also it's own class calculating the species and energy balance.
- The pyrolysis program independent Fitting classes.
- The IO-classes reading the user input and the classes generaing the output, e.g. for the FG-DVC coal file.
- The different user input classes reading the input files or the GUI input.

The manual is also structured the same way. Firstly the specific classes for CPD and FG-DVC are introduced, followed by the fitting classes.

Required additional python libraries for using PKP are:

- scipy
- numpy
- pylab
- pyevolve
- PyQt4

CHAPTER

ONE

THE CPD SPECIFIC CLASSES

The CPD classes are contained in the files:

- CPD_SetAndLaunch.py (Writes the CPD input file and launches the program)
- Compos_and_Energy.py (The species and energy balance)
- CPD_Result.py (Reads the specific output format of CPD)

1.1 The Class writing the CPD instruction File and launching CPD

class CPD_SetAndLaunch.**SetterAndLauncher**

This class is able to write the CPD input file and launch the CPD program. Before writing the CPD input file (method 'writeInstructFile') set all parameter using the corresponding methods (SetCoalParameter, SetOperateCond, SetNumericalParam, CalcCoalParam). After writing the instruct file, the .Run method can be used.

CalcCoalParam()

Calculates the CPD coal parameter mdel, mw, p0, sig and sets the as an attribute of the class.

Run(*CPD_exe*, *Input_File*, *LogFile*)

Launches CPD_exe and inputs Input_File. If the CPD executable is in another directory than the Python script enter the whole path for CPD_exe.

SetCoalParameter(*fcar*, *fhyd*, *fnit*, *foxy*, *VMdaf*)

Set the mass fraction of carbon (UA), hydrogen (UA), nitrogen (UA), oxygen (UA) and the daf fraction of volatile matter (PA).

SetNumericalParam(*dt*, *timax*)

Sets the numerical parameter and the maximum simulation time. dt is a vector with the following information: [dt_initial,print-increment,dt_max]

SetOperateCond(*pressure*, *TimeTemp*)

Stes the operating condtions pressure and the time-temperature array. TimeTemp is an 2D-Array. One line of this array is [time_i,temp_i].

writeInstructFile(*Dirpath*)

Writes the File 'CPD_input.dat' into the directory Dirpath.

1.2 The Class making the Species and the Energy Balance

class Compos_and_Energy.**SpeciesBalance**

This is the parent Class for the CPD and FG-DVC specific Species Balances, containing general methods like the Dulong formular.

Dulong()

Uses the Dulong formular to calculate the Higher heating value. The output is in J/kg.

SpeciesIndex(*species*)

Returns the column number of the input species.

correctUA()

Scale Ultimate Analysis to have sum=1

class Compos_and_Energy.**CPD_SpeciesBalance**(*CPD_ResultObject*, *UAC*, *UAH*, *UAN*, *UAO*, *UAS*, *PAVM*, *PAFC*, *PAmoist*, *PAash*, *HHV*, *MTar*, *RunNr*)

This class calculates the Species and the Energy balance for CPD. See the manual for the formulas and more details.

_CPD_SpeciesBalance__CheckOthers()

If the yield of nitrogen (is equal the UA of nitrogen) is lower than the species 'Other' the difference is set equal Methane.

_CPD_SpeciesBalance__CheckOxygen()

Checks weather the amount of Oxygen in the light gases is lower than in the Ultimate Analysis. If not, the amount of these species decreases. If yes, the tar contains oxygen.

_CPD_SpeciesBalance__QPyro()

Calculates the heat of the pyrolysis process, assuming heat of tar formation is equal zero.

_CPD_SpeciesBalance__Q_React()

Calculates the Heat of Reaction of the coal cumbustion.

_CPD_SpeciesBalance__TarComp()

Calculates and returns the tar composition.

_CPD_SpeciesBalance__closeResultFile()

Closes the CPD Composition file.

_CPD_SpeciesBalance__correctYields()

Correct the amount of the yields 'Other'.

_CPD_SpeciesBalance__hfRaw()

Calculates the heat of formation of the coal molecule and writes it into the output file.

_CPD_SpeciesBalance__hfTar()

Calculates the heat of formation of tar and writes it into the CPD Composition file.

_CPD_SpeciesBalance__nysEq15()

Calculates the stoichiometric coefficients of the devolatilization reaction.

1.3 The Reading Class containg the CPD specific output information

class CPD_Result.**CPD_Result**(*FilePath*)

Reads the CPD input and saves the values in one array. The results include the yields and the rates. The rates were calculated using a CDS. This class also contains the dictionaries for the columns in the array - the name of

the species. These dictionaries are CPD-Version dependent and the only thing which has to be changed for the case of a new release of CPD with a new order of species in the result files.

DictCols2Yields()
Returns the whole Dictionary Columns of the matrix to Yield names

DictYields2Cols()
Returns the whole Dictionary Yield names to Columns of the matrix

FinalYields()
Returns the last line of the Array, containing the yields at the time=time_End

Name()
returns 'CPD' as the name of the Program

Rates_all()
Returns the whole result matrix of the Rates.

Yields_all()
Returns the whole result matrix of the yields.

CHAPTER

TWO

THE FG-DVC SPECIFIC CLASSES

The FG-FVC classes are contained in the files:

- FGDVC_SetAndLaunch.py (Writes the FG-DVC 'instruct.ini' and launches FG-DVC)
- Compos_and_Energy.py (The species and energy balance)
- FGDVC_Result.py ('FGDVC_Result' reads the FG-DVC output and contains the output specific information.)

2.1 The Class writing the FG-DVC instruction file and launches FG-DVC

class FGDVC_SetAndLaunch.**SetterAndLauncher**

This class is able to write the 'instruct.ini' and launch the 'fgdvcd.exe'. Before writing the instruct.ini (method 'writeInstructFile') set all parameter using the corresponding methods (at least necessary: set1CoalLocation, set2KinLocation, set3PolyLocation, set5Pressure, set7Ramp). After writing the instruct file, the .Run method can be used.

Run(*PathFromEXE*)

Lauchnes fgdvcd.exe. The 'PathFromEXE' should not include the .exe and end with a backslash.

set1CoalLocation(*PathCoalFile*)

Sets the coal composition file directory.

set2KinLocation(*PathKinFile*)

Sets the coal kinetic file directory.

set3PolyLocation(*PathPolyFile*)

Sets the coal polymer file directory.

set4RunID(*pyrolysisORgeology*)

Sets weather the pyrolysis process or the geolegy process shall be modeled. For more information see FG-DVC manual.

set5Pressure(*PressureIn_atm*)

Sets the operating pressure (float).

set6Theorie(*Theorie*, *ResidenceTime*)

Sets the theory: 13 for no or partial tar cracking, 15 for full tar cracking. The residence input should be 0.0 for no tar cracking or a time greater zero for partial tar pressure. Full tar pressure also requires 0.0 as input, as it is the characteristic input of FG-DVC (although writing here 0.0, the full tar cracking is activated).

set7File(*THistoryFileLocation*)

For the case a temperature history shall be imported, this method should be used. 'THistoryFileLocation'

is the directory of the .txt file containing two columns: first column the time in seconds, the second the temperature indegree Celsius-

set7Ramp(*timeTotal*, *dt*, *dT*, *finalPyrolysisTemp*, *initialT*, *HeatingRate*)
Sets the following time relevant and temperature history relevant parameter: the total simulation time 'timeTotal', the constant numerical time step 'dt', the maximum temperture step 'dT', the final pyrolysis temperature 'finalPyrolysisTemp', the temperature at t=0 'initialT', and the constant heating rate 'HeatingRate'. All these parameter are required for the case a linear heating rate should be modeled.

set9AshMoisture(*AshContent*, *MoistureContent*)
Sets the amount of ash and moisture in the coal. By initializing the SetterAndLauncher object, both of these values are setted equal zero.

setTRamp_or_TFile(*selectRamp_or_File*)
Select weather the time should be defined using a linear ramp ('selectRamp_or_File'='Ramp') or with a input file ('selectRamp_or_File'='File').

writeInstructFile(*Filepath*)
Writes the File 'instruct.ini' into the directory 'Filepath', which should end with FGDVC_8-2-3/FGDVC .

2.1.1 The Class generating the Coal File

class InformationFiles.**WriteFGDVCCoalFile**(*CoalGenFile*)
writes the file, which will be inputted into the FG-DVC coal generator

setCoalComp(*Carbon*, *Hydrogen*, *Oxygen*, *Nitrogen*, *Sulfur*, *SulfurPyrite*)
Enter the coal composition with values in percent which have to sum up to 100

write(*CoalsDirectory*, *CoalResultFileName*)
writes the FG-DVC coal generator input file

2.2 The Class making the Species and the Energy Balance

class Compos_and_Energy.**SpeciesBalance**
This is the parent Class for the CPD and FG-DVC specific Species Balances, containing general methods like the Dulong formular.

Dulong()
Uses the Dulong formular to calculate the Higher heating value. The output is in J/kg.

SpeciesIndex(*species*)
Returns the column number of the input species.

correctUA()
Scale Ultimate Analysis to have sum=1

class Compos_and_Energy.**FGDVC_SpeciesBalance**(*FGDVC_ResultObject*, *UAC*, *UAH*, *UAN*, *UAO*, *UAS*, *PAVM*, *PAFC*, *PAmoist*, *PAash*, *HHV*, *MTar*, *RunNr*)
This class calculates the Species and the Energy balance for FG-DVC. See the manual for the formulars and more details.

_FGDVC_SpeciesBalance__EnergyBalance()
Calculates the heat of formation of tar.

_FGDVC_SpeciesBalance__TarComp()
Calculates the Tar composition using analyisis of the Ultimate Analysis.

_FGDVC_SpeciesBalance__closeFile()
Closes the FG-DVC Composition file.

_FGDVC_SpeciesBalance__correctYields()
Further, only the yields of char, tar, CO, CO2, H2O, CH4, N2, H2, O2 are considered. Modifies the yields, merge species like Olefins parafins, HCN to tar.

_FGDVC_SpeciesBalance__writeEnergyResults()
Writes the Energy results into the result file.

_FGDVC_SpeciesBalance__writeSpeciesResults()
Writes the Species Balance results into the result file.

2.3 The Reading Class containg the CPD specific output information

class FGDVC_Result.**FGDVC_Result**(*FilePath*)
Reads the FG-DVC input and saves the values in one array. The results include the yields (from 'gasyields.txt') and the rates. The rates for all species except the solids (here a CDS is used) are imported from 'gasrates.txt'. The H_2 yields were calculated by subtract all other species except parafins and olefins from the total yields (see FG-DVC manual). This H_2-yield curve was smoothed and derived using a CDS to generate the H_2 rates. The parafins and olefins are added into the tar. This class also contains the dictionaries for the columns in the array - the name of the species. These dictionaries are FG-DVC-Version dependent and the only thing which has to be changed for the case of a new release of FG-DVC with a new order of species in the result files (this was made for Versions 8.2.2. and 8.2.3.).

DictCols2Yields()
Returns the whole Dictionary Columns of the matrix to Yield names

DictYields2Cols()
Returns the whole Dictionary Yield names to Columns of the matrix

FilePath()
Returns the FG-DVC File path

FinalYields()
Returns the last line of the Array, containing the yields at the time=time_End

Name()
returns 'FG-DVC' as the name of the Program

Rates_all()
Returns the whole result matrix of the Rates.

Yields_all()
Returns the whole result matrix of the yields.

CHAPTER

THREE

THE FITTING CLASSES

3.1 The General Fitting Support Class

class FitInfo.**Fit_one_run**(*ResultObject*)

Imports from the Result objects the arrays. It provides the fitting objects with the yields and rates over time for the specific species. This class futher offers the option to plot the generated fitting results.

Dt()

Returns the vector with the time steps dt.

DtC()

Returns the vector with the time steps dt_C. This time steps are for points between the original points, so the lenght of this vector is the lenght of the time vector minus one.

Interpolate(*Species*)

Outputs the interpolation object (e.g.Species(time)).

LineNumberMaxRate(*Species*)

Returns the line with the maximum Rate of the inputted species.

MassCoal()

returns the Vector with the solid mass(t)

MassVM_s()

Returns the Vector with the mass of the volatile Matter over time

NPoints()

returns number of Points for each species over time. Is equal the number of time points.

Name()

Returns the Name of the imported Result object (e.g. 'CPD')

Rate(*species*)

Returns the Vector of the species rate(t). The species can be inputted with the Column number (integer) or the name corresponding to the dictionary saved in the result class (string).

RateSingleSpec(*NameSpecies*)

Returns the Rate of the species (inputted as string) by calculate it from the yields by using a CDS

SpeciesIndex(*species*)

Returns the species column number (integer) of the recieved species name (string)

SpeciesName(*ColumnNumber*)

Returns the species name (string) of the recieved column number (integer)

SpeciesNames()

Returns a list with all species names (including time and temperature).

Time()
Returns the time vector

Yield(*species*)
Returns the Vector of the species yield(t). The species can be inputted with the Column number (integer) or the name corresponding to the dictionary saved in the result class (string).

plt_InputVectors(*xVector*, *y1Vector*, *y2Vector*, *y3Vector*, *y4Vector*, *y1Name*, *y2Name*, *y3Name*, *y4Name*)
Plots the y input Vectors vs. the x input vector.

plt_RateVsTime(*ColumnNumber*)
Plots the original rates output of the pyrolysis program (as e.g. CPD) of the species marke with the columns number

plt_YieldVsTime(*ColumnNumber*)
Plots the original yield output of the pyrolysis program (as e.g. CPD) of the species marke with the columns number

3.2 A simple two point estimator for constant rate

class Fitter.**TwoPointEstimator**
Solves the devolatilization reaction analytically using two arbitrary selected points and the constant rate model. Unprecise. Should only be used for tests.

3.3 The Least Square Optimization Class

class Fitter.**LeastSquarsEstimator**
Optimizes the Fitting curve using the Least Squares for the Yields and the Rates.

Deviation()
Returns the Deviation after the optimization procedure.

estimate_T(*fgdvc_list*, *model*, *Parameter_Vector*, *Name*, *preLoopNumber=0*)
The main optimization method. Optimizes the Fitting curve using the Least Squares for the weighted Yields and the weighted Rates considering the temperatur history. Requires at input: The corresponding Fit_one_run object, the Model object, the kinetic parameter list, a name (e.g. the species). preLoopNumber is the number of running the improve_E and improve_a routines. So the standard setting of preLoopNumber is equal zero. It may be used if there is only a very bad convergence when optimize all three parameter.

improve_E(*fgdvc*, *model*, *t*, *T*, *Parameter_Vector*, *Name*)
Additional option: Only the Activation Energy in the Arrhenius Equation is optimized. Actual not necessary.

improve_a(*fgdvc*, *model*, *t*, *T*, *Parameter_Vector*, *Name*)
Additional option: Only the preexponential factor in the Arrhenius Equation is optimized. Actual not necessary.

minLengthOfVectors(*fgdvc_list*)
Returns the minimum lenght of a all vectors from the several runs.

setMaxIter(*MaxiumNumberOfIterationInMainProcedure*)
Sets the maximum number of iteration oin the optimizer as a abort criterion for the fitting procedure.

setOptimizer(*ChosenOptimizer*)
Select one optimizer of the scipy.optimizer library: 'fmin','fmin_cg','fmin_bfgs','fmin_ncg','fmin_slsqp' or 'leastsq'. According to experience 'fmin' (or also 'leastsq') generates at best the results.

setPreMaxIter(*MaxiumNumberOfIterationInPreProcedure*)
Sets the maximum number of iteration oin the optimizer as a abort criterion for the prefitting procedure (if preLoopNumber in estimate_T is not equal zero).

setPreTolerance(*ToleranceForFminFunction*)
Sets the tolerance as a abort criterion for the prefitting procedure (if preLoopNumber in estimate_T is not equal zero).

setTolerance(*ToleranceForFminFunction*)
Sets the tolerance as a abort criterion for the fitting procedure.

setWeights(*WeightMass*, *WeightRates*)
Sets the weights for the yields and the rates for the fitting procedure. See manual for equation.

3.4 The Global Optimizer Class

class Fitter.**GlobalOptimizer**(*localOptimizer*, *KineticModel*, *Fit_one_runObj*)
Makes runs over a defined range to look for global optimum. Local Optimizer is an LeastSquarsEstimator object, KineticModel is e.g. an constantRate model object, Fit_one_runObj is the List containing the Objects supporting the local fitting procedure with data.

GenerateOptima(*Species*, *IndexListofParameterToOptimize*, *ArrayOfRanges*, *ListNrRuns*)
This method makes a several number of runs and returns the Parameter having the lowest deviation of all local minima. The list contains 3 or 4 parameters. If e.g., the second and the third Parameter have to be optimized: IndexListofParameterToOptimize=[1,2]. If e.g. the range of the second is 10000 to 12000 and for the third 1 to 5, ArrayOfRanges=[[10000,12000],[1,5]]. When ListNrRuns is e.g. [0,10,5,0] the the second Parameter will be optimizted eleven times between 10000 and 12000, the third six times between 1 and 5. Attention, the number of runs grows by NrRuns1*NrRuns2*NrRuns3*NrRuns4, which can lead to a very large number needing very much time!

ParamList()
Returns the Parameter list.

setParamList(*ParameterList*)
Sets the Parameter list.

3.5 The Evolution algorithm Class

class Evolve.**GenericOpt**(*KineticModel*, *Fit_one_runObj*, *Species*)
Class which uses the pyevolve module to search for the global optimum.To initialize use the Kinetic model (e.g. Kobayashi) and the Fit one run object list, which supports the fitting process with the informations.

_GenericOpt__UpdateParam()
Updates the non-scaled Parameter. Updates the non-scaled parameter vector with the values of the sclaed vector

mkResults()
Generates the result.

setNrGenerations(*NrOfGenerations*)
Defines the number of generations for the generic optimization.

setNrPopulation (*NrOfPopulation*)
Defines the size of the population for the generic optimization.

setParamRanges (*InitialGuess*, *minimum*, *maximum*)
Sets the range where to evolve.

setScaledParameter (*Parameter*)
Sets Sclaed Parameter equal to the input parameter.

setWeights (*WeightY*, *WeightR*)
Sets the yield and the weight rate for the optimization equation.

3.6 The Model parent Class

class Models.**Model**
Parent class of the children ConstantRateModel, the three Arrhenius Models (notations) and the Kobayashi models.

ErrorRate (*fgdvc*, *Species*)
Returns the absolute deviation per point between the fitted and the original rate curve.

ErrorYield (*fgdvc*, *Species*)
Returns the absolute deviation per point between the fitted and the original yield curve.

ParamVector ()
Returns the Vector containing the kinetic parameter of the Model (refering to the child model).

calcRate (*fgdvc*, *time*, *T*, *Name*)
Generates the Rates using the yields vector and a CDS.

deriveC (*fgdvc*, *yVector*, *maxVectorLenght=None*)
Returns a CDS of the inputted yVector.

minLengthOfVectors (*fgdvc_list*)
Returns the minimum lenght of a all vectors from the several runs.

mkSimpleResultFiles (*fgdvc_list*, *Species*)
Simple result file if no fitting is carried out. Writes only the transformed results into a file.

plot (*fgdvc_list*, *Species*)
Plot the yield and the rates over time with two curves: one is the original data, the other the fitting curve. Also file 'PyrolysisProgramName-Species.out' (e.g. 'CPD-CO2.out') containing the time (s), yields (kg/kg), rates (kg/(kg s)).

pltRate (*fgdvc_list*, *xValueToPlot*, *yValueToPlot*)
Plots the rates (to select with yValueToPlot) over Time or Temperature (to slect with xValueToPlot).

pltYield (*fgdvc_list*, *xValueToPlot*, *yValueToPlot*)
Plots the yields (to select with yValueToPlot) over Time or Temperature (to slect with xValueToPlot).

setParamVector (*ParameterList*)
Sets the Vector containing the kinetic parameter of the Model (refering to the child model).

3.7 The Model children Classes

3.7.1 The Constant Rate Model

class Models.**ConstantRateModel**(*InitialParameterVector*)

The model calculating the mass with m(t)=m_s0+(m_s0-m_s,e)*e**(-k*(t-t_start)) from the ODE dm/dt = -k*(m-m_s,e). The Parameter to optimize are k and t_start.

3.7.2 The Arrhenius Model

class Models.**ArrheniusModel**(*InitialParameterVector*)

The Arrhenius model in the standart notation: dm/dt=A*(T**b)*exp(-E/T)*(m_s-m) with the parameter a,b,E to optimize.

ConvertKinFactors(*ParameterVector*)

Dummy. Function actual has to convert the parameter into the standart Arrhenius notation.

calcMass(*fgdvc*, *time*, *T*, *Name*)

Outputs the mass(t) using the model specific equation.

3.7.3 The Arrhenius Model with no correction Term

class Models.**ArrheniusModelAlternativeNotation2**(*InitialParameterVector*)

Arrhenius model with a notation having a better optimization behaviour: dm/dt=exp[c*(b1*(1/T(t)-1/T_min)-b2*(1/T(t)-1/T_max))]*(ms-m); with c=(1/T_max-1/Tmin)**(-1). See the documentation for the reference. The parameters to optimize are b1 and b2.

ConvertKinFactors(*ParameterVector*)

Converts the own kinetic factors back to the standard Arrhenius kinetic factors.

ConvertKinFactorsToOwnNotation(*ParameterVector*)

Converts the standard Arrhenius kinetic factors back to the factors of the own notation.

calcMass(*fgdvc*, *time*, *T*, *Name*)

Outputs the mass(t) using the model specific equation.

setMinMaxTemp(*Tmin*, *Tmax*)

Sets the temperature constants, see the equation.

3.7.4 The Kobayashi Models

The first of these two models has the Parameter A1, A2, E1, and E2 to optimize. The second one has like PC Coal Lab the four parameter A1,A2,E1 and alpha1 to optimize.

class Models.**Kobayashi**(*InitialParameterVector*)

Calculates the devolatilization reaction using the Kobayashi model. The Arrhenius equation inside are in the standard notation.

calcMass(*fgdvc*, *time*, *T*, *Name*)

Outputs the mass(t) using the model specific equation. The input Vector is [A1,E1,A2,E2,alpha1,alpha2]

class Models.**KobayashiPCCL**(*InitialParameterVector*)

Calculates the devolatilization reaction using the Kobayashi model. The Arrhenius equation inside are in the standard notation. The fitting parameter are as in PCCL A1,A2,E1,alpha1.

ConvertKinFactors(*ParameterVector*)
Outputs the Arrhenius equation factors in the shape [A1,E1,A2,E2]. Here where the real Arrhenius model is in use only a dummy function.

E2Diff()
Returns the dE in E2=E1+dE.

KobWeights()
Returns the two Kobayashi weights alpha2.

calcMass(*fgdvc*, *time*, *T*, *Name*)
Outputs the mass(t) using the model specific equation.

setE2Diff(*DifferenceE1E2*)
Sets the dE in E2=E1+dE.

setKobWeights(*alpha2*)
Sets the two Kobayashi weights alpha2.

CHAPTER

FOUR

THE INPUT- OUTPUT CLASSES

4.1 The General User Input File Class

class InformationFiles.**ReadFile**(*InputFile*)
general parent class for the reading objects CPDFile and FGDVCFile

Fitting(*FileNote*)
outputs the Fitting mode for Pyrolysis Program output (string: 'constantRate','Arrhenius','Kobayashi'). Possible input: 'constantRate', 'Arrhenius' or 'Kobayashi'

UsePyrolProgr(*FileNote*)
gets the information, whether Pyrolsis Program will be in use. Enter 'Yes' or 'True' for the case it should be used

getText(*FileNote*)
output the data of the line below the FileNote as a string

getValue(*FileNote*)
output the data of the line below the FileNote as a float

readLines()
reads the input File line by line

4.2 The Class writing the FG-DVC Coal Generator Input File

class InformationFiles.**WriteFGDVCCoalFile**(*CoalGenFile*)
writes the file, which will be inputted into the FG-DVC coal generator

setCoalComp(*Carbon*, *Hydrogen*, *Oxygen*, *Nitrogen*, *Sulfur*, *SulfurPyrite*)
Enter the coal composition with values in percent which have to sum up to 100

write(*CoalsDirectory*, *CoalResultFileName*)
writes the FG-DVC coal generator input file

4.3 The Class reading the Operating Condition Input File

class InformationFiles.**OperCondInput**(*InputFile*)
Reads the input file for the operating conditions and also writes the temperature-history file, required by FG-DVC.

getTimePoints(*FileNoteBegin*, *FileNoteEnd*)
reads the time points in the shape 'time, temperature' for the lines between the line with the FileNoteBegin and the line with the FileNoteEnd

writeFGDVCtTHist(*tTPoints*, *dt*, *OutputFilePath*)
Writes output file for FG-DVC containing in first column time in s, in the second tempearure in degree Celsius. FG-DVC will import this file. The time-temperature array has to be a numpy.array, dt a float, OutputFilePath a string.

CHAPTER

FIVE

THE GUI CLASSES

5.1 The Main Window

class PKPgui.**Ui_PKP**

LoadTtFile1()
Loads the temperature history nr 1 file via file browser

LoadTtFile2()
Loads the temperature history nr 2 file via file browser

LoadTtFile3()
Loads the temperature history nr 3 file via file browser

LoadTtFile4()
Loads the temperature history nr 4 file via file browser

LoadTtFile5()
Loads the temperature history nr 5 file via file browser

OpenManual()
Opens the manual.

Plot1()
Plots the temperature over time history (temperature history nr 1) and saves temperatuer history in "TempHist1.dat".

Plot2()
Plots the temperature over time history (temperature history nr 2) and saves temperatuer history in "TempHist2.dat".

Plot3()
Plots the temperature over time history (temperature history nr 3) and saves temperatuer history in "TempHist3.dat".

Plot4()
Plots the temperature over time history (temperature history nr 4) and saves temperatuer history in "TempHist4.dat".

Plot5()
Plots the temperature over time history (temperature history nr 5) and saves temperatuer history in "TempHist5.dat".

ReOpenResultWindow()
If the calculation were done once, the window showing the results can be opened again.

SaveInfos()
Saves the Information when the save or the run -option is used.

WriteRun()
Writes the *.inp files and launch the process.

5.2 The Class Saving the information from the GUI

class PKPgui.**InfosFromGUI**
Saves the information from the GUI.

ArrhSpec()
Returns which species shall be fitted for Arrhenius.

ArrhSpecReverse(*SpeciesName*)
returns the UI columns bar index of species fitted for Arrhenius.

FGCoalProp()
Defines the way of the FG-DVC Coal Fitting and the Tar Modeling.

MwsHHV()
Retruns the Molar Weight of Tar and sets the Higher Heating Value.

OperCond()
Sets the pressure and the time step.

PA()
Returns the coal PA properties.

RunPyrolProg()
Returns which options of the three Pyrolysis programs are used.

RunPyrolProgReverse(*ModelName*)
Returns the GUI column bar index of the models selected.

SetArrhSpec(*SpeciesIndex*)
Sets which species shall be fitted for Arrhenius.

SetRunPyrolProg(*CPDIndex*, *FGDVCIndex*, *PCCLIndex*)
Saves which options of the three Pyrolysis programs are used.

TimeHistories()
Returns the number of time history.

UA()
Returns the coal UA properties.

WeightYR()
Returns the weghts for Yields and Rates

setFGCoalProp(*FGCoalFit*, *FGTarModeling*)
Defines the way of the FG-DVC Coal Fitting and the Tar Modeling.

setMwsHHV(*MoleWweightTar*, *HHV*)
Sets the Molar Weight of Tar and sets the Higher Heating Value.

setOperCond(*pressure*, *timestep*)
Sets the pressure and the time step.

setPA(*FixedCarbon*, *VolatileMatter*, *Moisture*, *Ash*)
Saves the coal PA properties.

setTimeHistories(*NrOfTs*)
Saves the Number time history

setUA(*UAC*, *UAH*, *UAN*, *UAO*, *UAS*)
Saves the coal UA properties.

setWeightYR(*Y*, *R*)
Sets the weghts for Yields and Rates

5.3 The Classes writing the Info Files

5.3.1 The Coal File

class writeInfoFiles.**WriteCoalFile**(*InfosFromGUIObject*)
Writes the Coal.inp file using the output of the GUI.

5.3.2 The CPD File

class writeInfoFiles.**WriteCPDFile**(*InfosFromGUIObject*)
Writes the CPD.inp file using the output of the GUI. The number of print increment is imorted from the previous version of CPD.inp.

5.3.3 The FG-DVC File

class writeInfoFiles.**WriteFGFile**(*InfosFromGUIObject*)
Writes the FGDVC.inp file using the output of the GUI. The filepaths are imorted from the previous version of FGDVC.inp.

5.3.4 The Operating Condition File

class writeInfoFiles.**WriteOCFile**(*InfosFromGUIObject*)
Writes the OperCond.inp file using the output of the GUI.

CHAPTER

SIX

THE MAIN CLASS

This is the class controlling the whole process. It initialize the other classes and use their methods. If accesed as main file, it launches itself the whole process. But it is also connetced by the GUI, there the GUI tells to apply the MainProcess' methods.

class PKP.**MainProcess**

Controls the whole process of generating input files, fitting etc.

CheckFGdt()

Aborts, if FG-DVC is selected and the timestep is lower than 1.e-3 (which is FG-DVC not able to read):

DAF(*PAFC_asRecieved*, *PAVM_asRecieved*)

calculates PAFC, PAVM from the as recieved state to the daf state of coal

MakeResults_Arrh(*PyrolProgram*, *File*, *Fit*)

Generates the results for Arrhenius Rate.

MakeResults_ArrhNoB(*PyrolProgram*, *File*, *Fit*)

Generates the results for Arrhenius Rate with no correction term T**b.

MakeResults_CPD()

generates the result for CPD

MakeResults_CR(*PyrolProgram*, *File*, *Fit*)

Generates the results for constant Rate.

MakeResults_DEAM(*PyrolProgram*, *File*, *Fit*)

Generates the results for DAEM model.

MakeResults_FG()

generates the result for FG-DVC

MakeResults_Kob(*PyrolProgram*, *File*, *Fit*)

Generates the results for Kobayashi Rate.

ReadInputFiles()

get parameters from input files

SpeciesEnergy(*PyrolProgram*, *File*)

Carries out the species and Energy balance.

INDEX

Symbols

A

C

R

S

T

U

W

Y