HPFC

User guide

Version: 1.0.0

2020-02-21

Table of contents

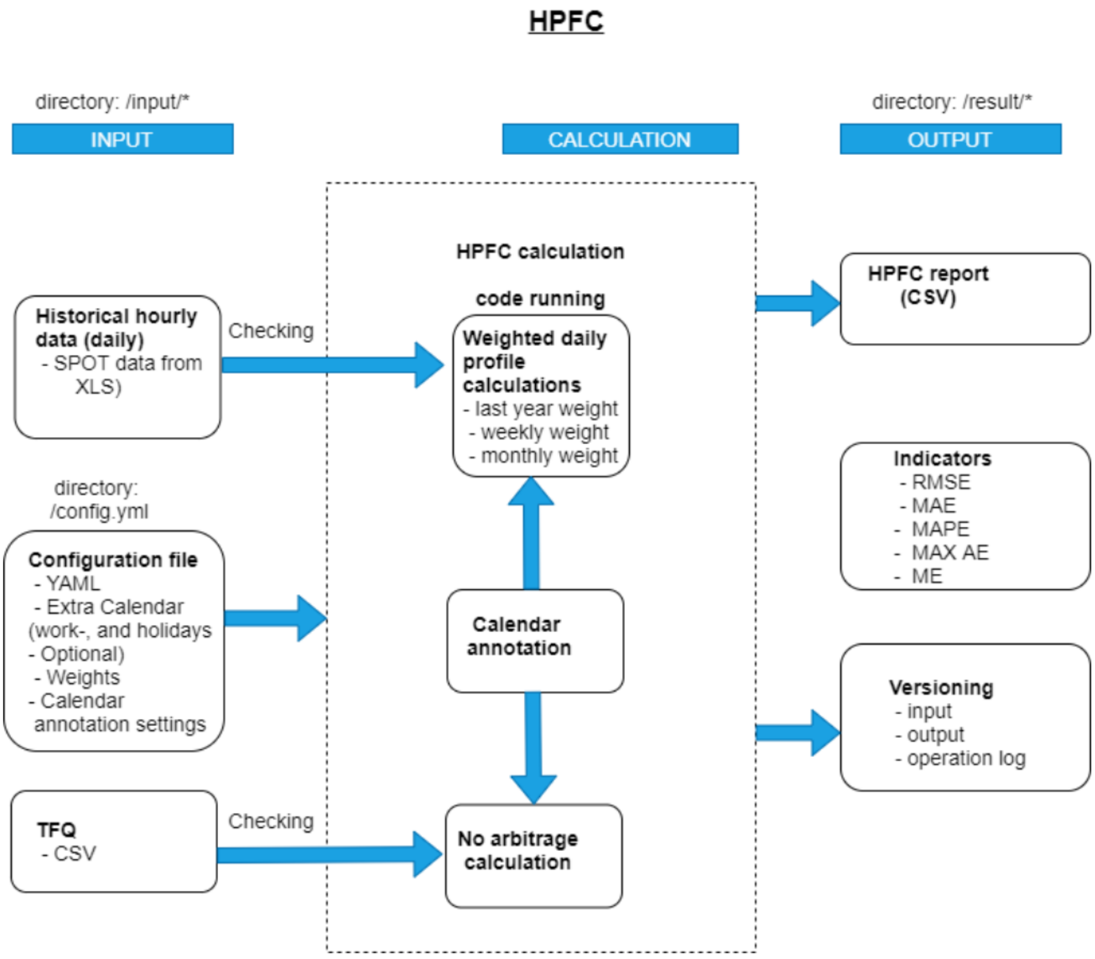
1. Overview

The application provides HPFC calculation and it estimates future prices on an hourly basis, based on past spot and current weekly / monthly / yearly forward (TFQ) data. HPFC will be generated automatically in case of changes in the TFQ data.

An hourly price forward curve (HPFC) is an approximated price curve constructed from past spot data thus capturing seasonalities, on future date intervals continuing the known market prices. The forward curve construction combines two different approaches. A statistical method examines, how spot prices had moved in the past, and a fundamental model shows how the price changes due to supply and demand (respectively, fuel prices on the merit order curve, and load).

It is highly difficult to specify what exactly makes a HPFC, which should be arbitrage-free for products traded on an exchange and should also reflect the seasonality of spot prices.

Overview about logical flow and directories:



The application monitors a directory, and when a new input is detected or an existing input file has been modified, it runs the generator with the changed inputs. So, when the TFQ export finishes, the new HPFC calculation will be automatically started.

2. Installation

The application is portable, which means it only needs to be extracted and after that, it is ready to run.

On Windows extract the compressed archive to a folder, open a command line and navigate to the extracted hpfc.exe, where it can be used with its arguments.

On Linux/Unix systems the installation is the same procedure, however the GCC compiler cannot be older, than the one used to build the application. In that case, update the compiler according to the distribution in use. The application can be started with a runnable binary called hpfc.

3. Command line application

Command line application

A command line application is a type of software that is run from the command prompt on windows and from the terminal on Unix-based systems by calling its executable file. The application can be started by typing the executable file's name (and extension in case of Windows) and additional arguments and options into the shell and pressing enter.

Starting the application

Before starting the application, it is advised to navigate to the folder where the application was originally installed and start the application from there. In this case the executable file's name starts the process immediately. In another case, an absolute or relative path is to be referenced to invoke the application. When referencing relative paths, more caution is needed, since the current folder functions as the working directory. This means, that every file path in the configuration file (also the configuration file’s path) must be specified relative to the working directory.

Command line window

The command line application accepts multiple command line arguments and options. For available options and their usage, consult the 'HPFC commands' section of this manual. A command line application 'takes hold of' the terminal it's running in, meaning that no other application can be started while the application is running in that exact terminal. While running, the program outputs information about certain processes within the application to the terminal window.

Example output

```
Loading the following configuration file: ./config/config.yml
Calculation started by: TestUser
Input data validation started..
Input data validation finished!
Loading the following Forward data file: 20191104_TfQ_ver1.csv
Forward data file loaded
HPFC generation's start date is: 2019-11-05
HPFC generation's end date is: 2022-12-31
...
HPFC calculation started..
HPFC calculation finished!
HPFC generated: data\result\HU_HPFC_2020-02-06_V7.csv
Calculation finished!
```

Running multiple instances of the application

At a time, only one instance of the application can be in a running state. Trying to run a second instance will result in an error message, stating that an instance is already running, even when trying to start the second instance in an independent terminal or command line window. This effect is achieved by a lock file. When the application is started, a lock file is created. The first action the application does after execution is to check for this lock file. If the lock file is present, another instance cannot be started until the current process finishes and deletes it after completion. It is important to note, that the lock file is deleted upon successful HPFC generation and failure alike.

4. HPFC commands

Generate

Usage: hpfc generate [OPTIONS]

HPFC calculation can be started with the generate command. By running this command, the application loads the necessary input files, such as configuration, spot data, tfq, and so on, then validates these files and command line arguments. Upon successful validation, the HPFC calculation starts. It calculates the daily profiles and combines them with the forward data, then finally saves the calculated curve. For more information, check chapter 8. “HPFC results”.

generate command options:

-c, --calendar TEXT

Imports an exported csv calendar. The file consists of two columns: date and day_feature. The date column follows the ‘yyyy-mm-dd' format.

-e, --end-date TEXT

Endpoint of HPFC generation's interval, can be as late as the date of the last forward data's date or earlier.

-s, --spot-end-date TEXT

Sets the last historic date, which is considered in the process of creating day profiles. If “spot_end_date” was specified in the configuration file too, the command line parameter will be used anyway.

-a, --archive-folder-base-path TEXT

Sets the archive folder base path, in this folder a separate folder will be created. If “archive_base_path” was specified in the configuration file too, the command line parameter will be used anyway.

--forward-data-file TEXT

Sets the forward data (TFQ) file location. If “forward_data_file_name” was specified in the configuration file too, the command line parameter will be used anyway.

--config-file TEXT

Specify a configuration file. [default: ./config/config.yml]

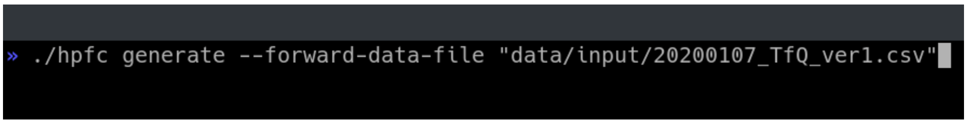
-f, --hpfc-format TEXT

Specify the file format of the generated HPFC. Valid options are: [default | elmu]

--help

Show help message and exit.

Example:



```
» ./hpfc generate --forward-data-file "data/input/20200107_TfQ_ver1.csv"
```

Compare

Usage: hpfc compare [OPTIONS] [FIRST_CURVE_PATH] [SECOND_CURVE_PATH]

The compare command compares two price curves. The application gets these two curves from csv files, so the path of these files must be specified by the user as command line arguments. Upon successful comparison, the following stats will be calculated: MAE, RMSE, MAPE, MAX AE, ME.

compare command options:

-s, --start-date TEXT

Start date for the comparing: yyyy-mm-dd

-e, --end-date TEXT

End date for the comparing: yyyy-mm-dd

-i, --ignore-nan

Ignore NaN's in the data.

--config-file TEXT

Specify a configuration file. [default: ./config/config.yml]

-f, --hpfc-format-first TEXT

Specify the file format of the first compared HPFC. Valid options are: [default | elmu]

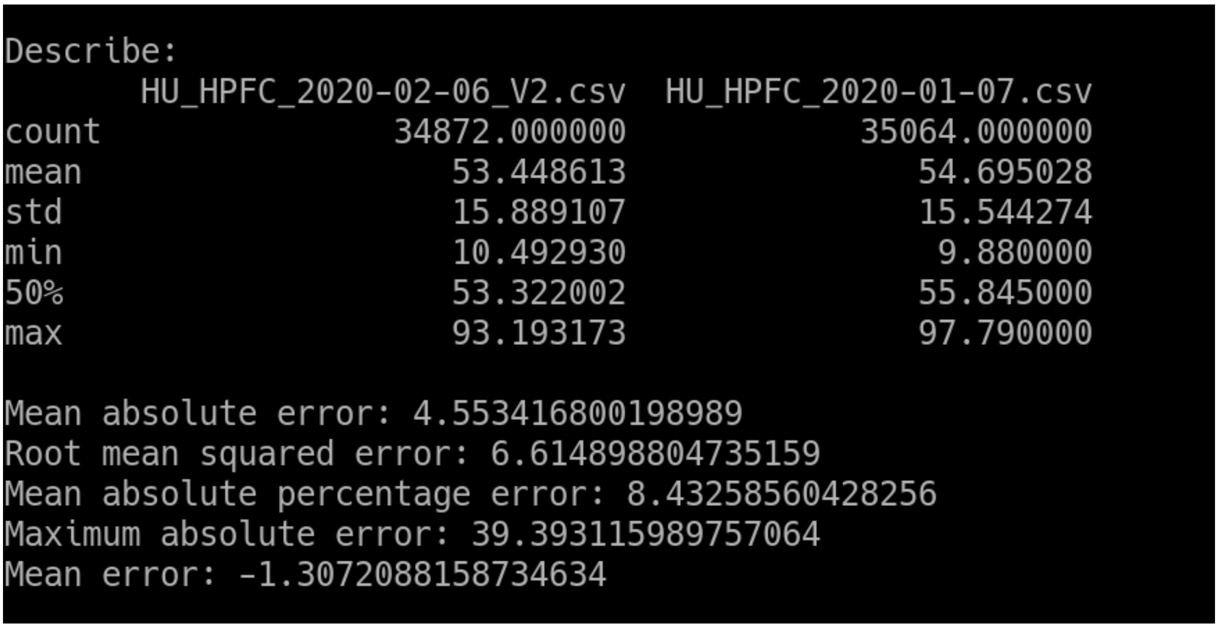
-z, --hpfc-format-second TEXT

Specify the file format of the second compared HPFC. Valid options are: [default | elmu]

--help

Show help message and exit.

Example output:



```
Describe:
       HU_HPFC_2020-02-06_V2.csv  HU_HPFC_2020-01-07.csv
count               34872.000000            35064.000000
mean                   53.448613               54.695028
std                    15.889107               15.544274
min                    10.492930                9.880000
50%                    53.322002               55.845000
max                    93.193173               97.790000

Mean absolute error: 4.553416800198989
Root mean squared error: 6.614898804735159
Mean absolute percentage error: 8.43258560428256
Maximum absolute error: 39.393115989757064
Mean error: -1.3072088158734634
```

Validate

Usage: hpfc validate [OPTIONS]

The validate command validates all input files specified in the configuration file (and the configuration file itself). It looks for all sorts of errors, such as empty values, wrong column names, inconsistencies in the data in some cases and so on. The validation process runs as part of the generate process as well but invoking the validate command on its own provides more insight into the results of the validation phase.

validate command options:

-c, --calendar TEXT

Imports and validates an exported csv calendar.

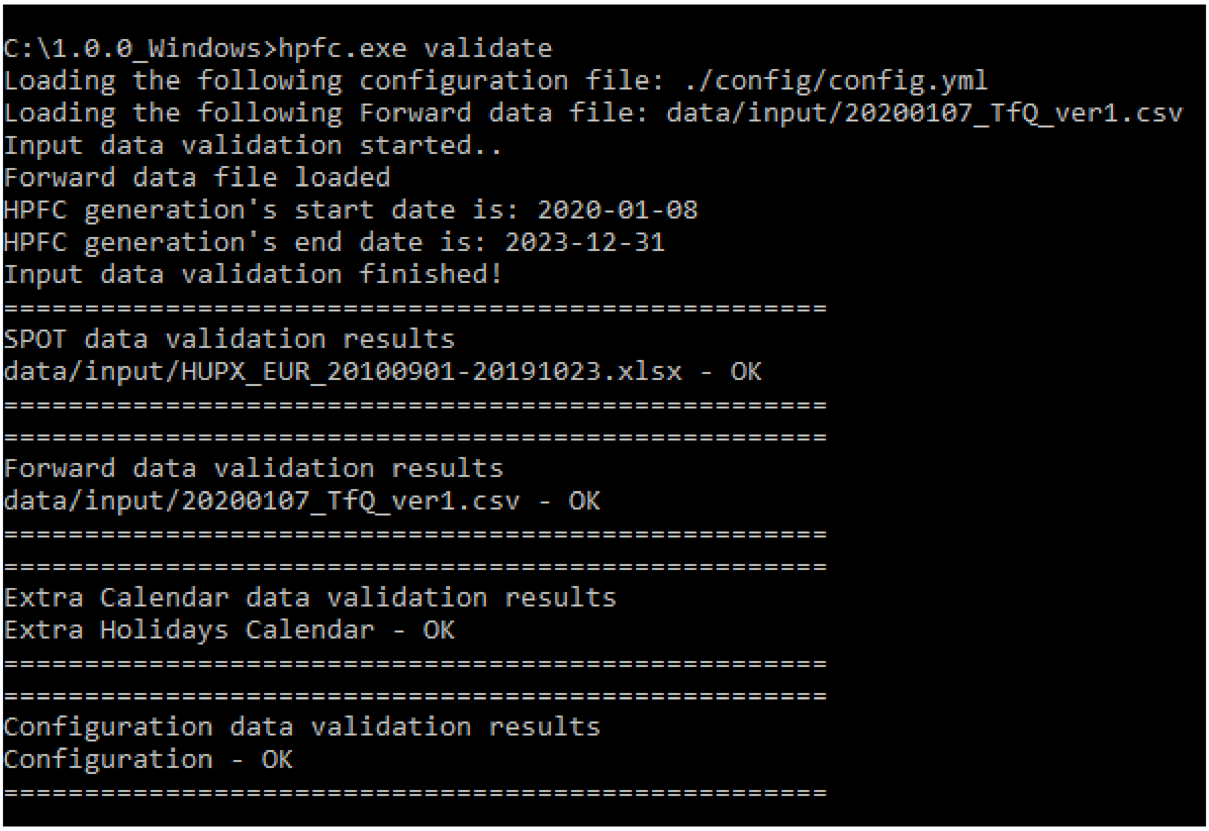
--config-file TEXT

Specify a configuration file. [default: ./config/config.yml]

--help

Show help message and exit.

Example:



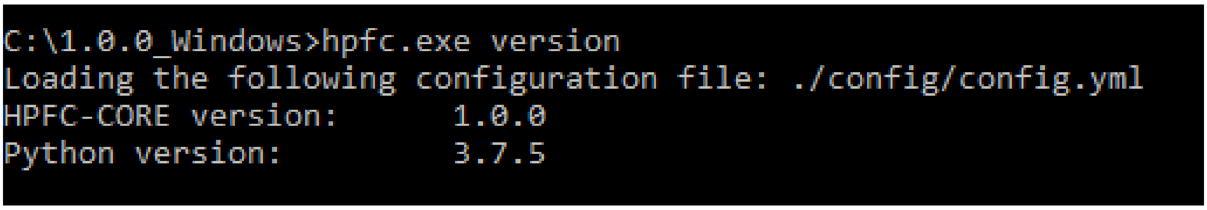
```
C:\1.0.0_Windows>hpfc.exe validate
Loading the following configuration file: ./config/config.yml
Loading the following Forward data file: data/input/20200107_TfQ_ver1.csv
Input data validation started..
Forward data file loaded
HPFC generation's start date is: 2020-01-08
HPFC generation's end date is: 2023-12-31
Input data validation finished!
===================================================
SPOT data validation results
data/input/HUPX_EUR_20100901-20191023.xlsx - OK
===================================================
===================================================
Forward data validation results
data/input/20200107_TfQ_ver1.csv - OK
===================================================
===================================================
Extra Calendar data validation results
Extra Holidays Calendar - OK
===================================================
===================================================
Configuration data validation results
Configuration - OK
===================================================
```

Version

Usage: hpfc version

Invoking the version command prints the application’s version number and the python version, that powers the application.

Example:



```
C:\1.0.0_Windows>hpfc.exe version
Loading the following configuration file: ./config/config.yml
HPFC-CORE version:      1.0.0
Python version:         3.7.5
```

Export

Usage: hpfc export [OPTIONS] TARGET

The export command makes it possible to export various data generated by the application.

Available targets:

calendar: Export the current calendar in csv format. The file consists of two columns: date and day_feature. The date column follows the ‘yyyy-mm-dd' format.

export command options:

-t, --target-file TEXT

Path to export calendar (e.g.:/users/_user_/calendar.csv). [default: calendar.csv]

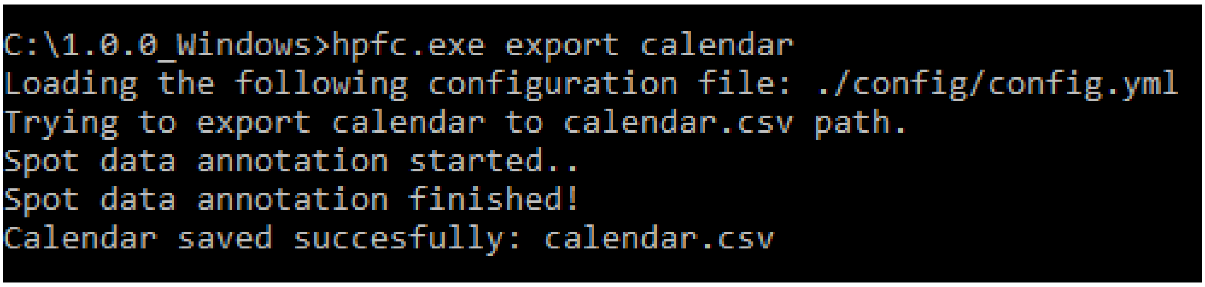
--config-file TEXT

Specify a configuration file. [default: ./config/config.yml]

--help

Show help message and exit.

Example:



```
C:\1.0.0_Windows>hpfc.exe export calendar
Loading the following configuration file: ./config/config.yml
Trying to export calendar to calendar.csv path.
Spot data annotation started..
Spot data annotation finished!
Calendar saved succesfully: calendar.csv
```

5. Configuration file

In the Configuration file the user can set multiple parameters for HPFC generation:

File name or path

- **spot_data_file_name**

  The path to the file containing spot data

  spot_data_file_name: data/input/HUPX_EUR_20100901-20191023.xlsx

- **forward_data_file_name**

  The path to the file containing forward data, there is also a cmd line parameter for this: forward-data-file

  forward_data_file_name: example/20191104_TfQ_ver1.csv

- **result_path**

  The path to the generated HPFC file (versioned). There is an inner Log directory for log files.

  for example: data/result/

- **archive_base_path**

  The path to the base archive folder, where a versioned separate folder will be created for every calculation, and all the results, partial results, logs, input data, and parameters will be saved there

  archive_base_path: data/archive/

- **timezone**

  The timezone where the result hpfc will be interpreted

  Note: Daylight saving time for the input data will be corrected: hour duplications will be deleted, and the missing hours will have the average of the surrounding hours. In the hpfc output daylight saving time changes will be added with the necessary hour duplications and omissions.

  for example: Europe/Budapest

- **country**

The country whose holidays and other non-working days will be taken into consideration during the hpfc calculation.

You can specify the country either with its name or with the corresponding country code, which are listed in the configuration.

Note: Currently this holiday data comes from a library called holidays 0.9.12

- **Spot end date**

  Optional, sets the last historic date, which is considered in the process of creating day profiles. If command line argument is also given it will be used!

  spot_end_date: 2019-10-01

Peak hours configuration

Peak hours configuration to set start and end hours, (0 <= start <= end <= 24) these must be integers

peak_hour

start: 8

end: 20

Price

These must be integers or floating-point numbers, these values will be considered during the data validation process

spot_data

- max_price_threshold: 400
- min_price_threshold: -400

Calendar (labelling the calendar days with day type)

- Used by the calendar annotation configuration to annotate workdays (non-holidays/weekends/bridge days).

  Possible similarities are:

  1. All weekdays (MON-FRI) are treated as similar (which means day have more or less similar characteristics)

2. Middle days: TUE/WED/THU are similar, MON and FRI are different
3. All weekdays (MON-FRI) are treated as different (which means day have different characteristics)

weekday_similarity: 3

- Used by the calendar annotation configuration to annotate weekend days.

Possible similarities are:

1. All weekend days (SAT-SUN) are treated as similar (which means these days have similar characteristics)
2. All weekend days (SAT, SUN) are treated as different (which means day have somewhat different characteristics)

weekend_similarity: 2

- Use holidays as a separate day type (bool: possible values are **true** and **false**, meaning yes and no)

use_holidays_as_feature: true

- Use the days before and after a holiday as a separate day type (bool)

use_holidays_neighbors_as_feature: true

- Use the bridge days (a day between two holidays or between a holiday and a weekend day) as a separate day type (bool)
  use_bridgedays_as_feature: true
- Use the working days between Christmas and New Year's Day as a separate day type (bool)

use_days_between_christmas_and_newyear_as_feature: true

- If a holiday falls on a weekend, treat it as a weekend day (bool)

treat_weekend_holidays_as_weekend_days: false

- If a holiday neighbor falls on a weekend, treat it as a weekend day (bool)

treat_weekend_holiday_neighbors_as_weekend_days: false

- If a bridge day falls on a weekend, treat it as a weekend day (bool) treat_weekend_bridgedays_as_weekend_days: false
- Use state holidays and working saturdays (bool)

use_state_holidays_and_working_saturdays: false

Day Profiler Annotation config

day_profiler:

Used by the day profiler configuration to set resolution

Possible resolutions are:

1: MONTH

2: WEEK

base_resolution: 1

The current month/week is at the centre, and we sample the 1, 2, ...nth (n = steps) month/week in the past and future. Eg.: if base_resolution is MONTH and the actual month is JAN and steps = 2 then we sample in NOV, DEC (past) and FEB, MAR (future)

base_weights: [1, 3, 1]

The actual weights of the relative years. Eg.: if the spot end date is 2020-02-10 and the number of year weights are 2 ([0.75, 0.25]) that means that we will use data from 2019-02-10 to 2020-02-10 with 0.75 weight and data from 2018-02-10 to 2019-02-10 with 0.25 weight.

year_weights: [9, 6, 5, 4, 3, 2, 1, 1, 1]

Day Profile Replacement config

- Used by the day profile calculation to fill missing day types.
- If some day profiles are missing during or after the weighting process, they will be replaced with another data, from another node or another day type.

node_range

Specifies the number of subsequent and previous nodes, that will be checked to replace missing day types. First, the closer nodes will be checked. If the base_resolution is set to MONTH, the node_range setting means monthly distance, if the base_resolution is seto to WEEK, then the node_range setting means weekly distance. Recommended setting: 1 if base_resolution is month, 5 if base_resolution is week.

node_range: 1

execution_order

The place of the replacement during the day profile calculation process. The replacement after yearly weighting is always part of the process. The extra replacement, that can be executed before or after the node weighting, is optional.

Possible values are:

1: BEFORE_WEIGHTING

2: AFTER_WEIGHTING

3: ONLY_AFTER_YEARLY_WEIGHTING

Default: 3

- The BEFORE_WEIGHTING option means, that the replaced values are used in the weekly/monthly weighting. This could improve the result, for day types with missing data, but can also misfit the curve in the adjacent nodes in some cases.
- The AFTER_WEIGHTING option means, that the replaced values are not used in the weekly/monthly weighting, but they are used in the yearly weighting. This could improve the result, for day types with less data than other day types.
- The AFTER_YEARLY_WEIGHTING option means, that the replaced values are not used in the weekly/monthly weighting nor the yearly weighting. They are just filling the holes in the future data frame.

```
  execution_order: 3
```

filling_by_type_first

If set to true, the filling by day types step happens before filling by nodes step.

This means, the missing profiles (except if the day feature is holidays) will be filled with another day feature, if possible. (Holidays are always filled with adjacent nodes first.)

This can result in a "smoother" curve, if some weekdays are missing because of special day features.

Default: false

```
  filling_by_type_first: false
```

Issued holidays and Saturday workdays

For Hungarian state issued holidays and Saturday workdays the key is the date in this format:

YYYY-MM-DD

the value is:

2: WORKDAY

3: HOLIDAY

The workdays are validated, they can only be Saturdays

```
extra_holidays_calendar:
  2010-12-11: 2
  2010-12-24: 3
  2011-03-14: 3
  2011-03-19: 2
  2011-10-31: 3
```

2011-11-05: 2

…

6. Spot data

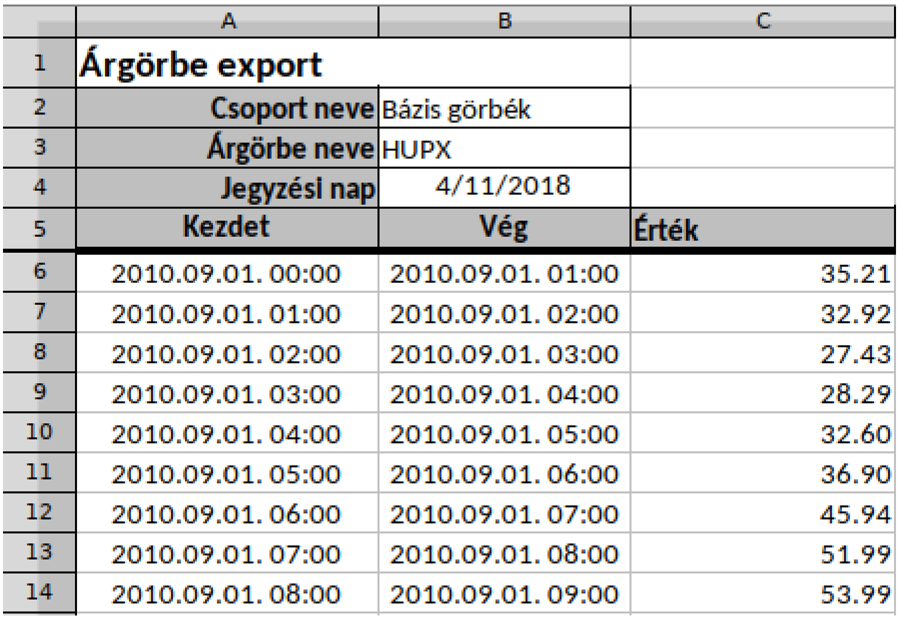
The path of the spot data file is to be specified in the configuration file. The file must have .xlsx as its extension. Spot data is presented as hourly data. Starting from the 6th row, column A holds the datetimes complementing the prices in column C. Datetimes are in the "yyyy.MM.dd hh:mm" format, prices use the dot notation for decimals.

The file must have yearly data for every weight in the day_profiler.year_weights specified in the configuration file.

Handling daylight saving time:

The missing hour caused by the clock shift in March should not be present in the excel file. The extra hour caused by the clock shift in October should be present in the excel file. This means that 02:00 and its price should appear twice on October's clock shift day.

Example:



	A	B	C
1	Árgörbe export		
2	Csoport neve	Bázis görbék	
3	Árgörbe neve	HUPX	
4	Jegyzési nap	4/11/2018	
5	Kezdet	Vég	Érték
6	2010.09.01. 00:00	2010.09.01. 01:00	35.21
7	2010.09.01. 01:00	2010.09.01. 02:00	32.92
8	2010.09.01. 02:00	2010.09.01. 03:00	27.43
9	2010.09.01. 03:00	2010.09.01. 04:00	28.29
10	2010.09.01. 04:00	2010.09.01. 05:00	32.60
11	2010.09.01. 05:00	2010.09.01. 06:00	36.90
12	2010.09.01. 06:00	2010.09.01. 07:00	45.94
13	2010.09.01. 07:00	2010.09.01. 08:00	51.99
14	2010.09.01. 08:00	2010.09.01. 09:00	53.99

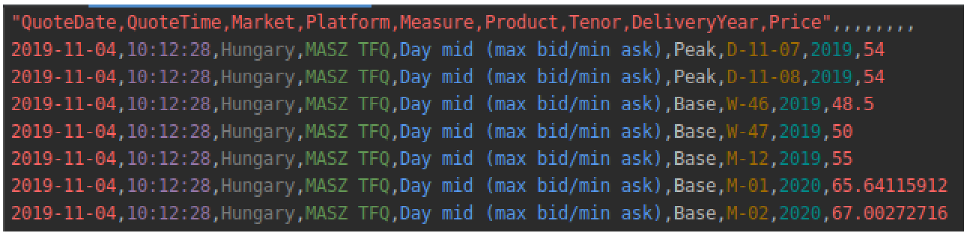
7. Forward data

The forward data file’s path can be specified in the configurations file under ‘forward_data_file_name’. This file must have .csv as its file extension. In this file data should be separated with commas in 9 columns, from which the last four is important as far as the data loading is concerned:

- Product: Base, Peak.
- Tenor: D-{MM-dd}, W-{ww}, M-{mm} (e.g. D-01-31)
- Delivery Year
- Price: dots are used for decimal values.

This csv must contain arbitrage-free price values, and it can be generated with this condition from the Hungary_Table_for_Quotes_import_fajl_keszito_{yyyy}.xlsm file: certain cells on the “table for quotes” worksheet must be filled in order to generate a valid .csv file. Clicking the export button creates the .csv file in the folder specified in the “Input” worksheet under “path to save import file”.

Example:



```
"QuoteDate,QuoteTime,Market,Platform,Measure,Product,Tenor,DeliveryYear,Price",,,,,,,,
2019-11-04,10:12:28,Hungary,MASZ TFQ,Day mid (max bid/min ask),Peak,D-11-07,2019,54
2019-11-04,10:12:28,Hungary,MASZ TFQ,Day mid (max bid/min ask),Peak,D-11-08,2019,54
2019-11-04,10:12:28,Hungary,MASZ TFQ,Day mid (max bid/min ask),Base,W-46,2019,48.5
2019-11-04,10:12:28,Hungary,MASZ TFQ,Day mid (max bid/min ask),Base,W-47,2019,50
2019-11-04,10:12:28,Hungary,MASZ TFQ,Day mid (max bid/min ask),Base,M-12,2019,55
2019-11-04,10:12:28,Hungary,MASZ TFQ,Day mid (max bid/min ask),Base,M-01,2020,65.64115912
2019-11-04,10:12:28,Hungary,MASZ TFQ,Day mid (max bid/min ask),Base,M-02,2020,67.00272716
```

8. HPFC results

HPFC Formats

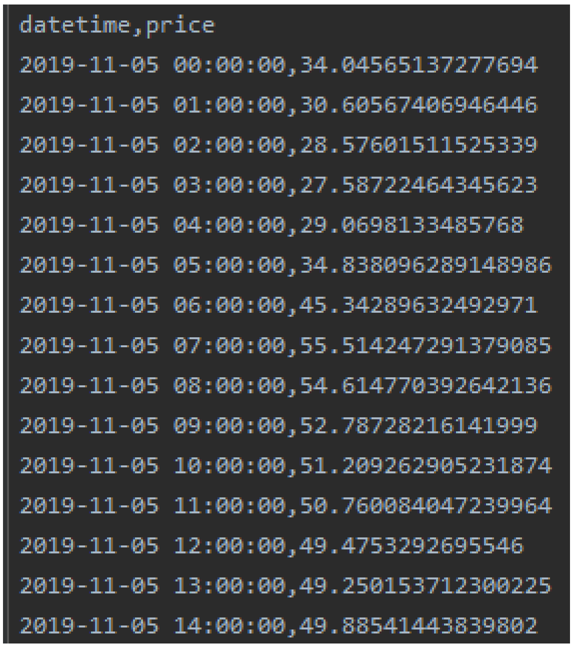
The application can generate two types of HPFCs as far as formatting is concerned:

- default format
- elmu-specific format.

The main differences between the two formats are:

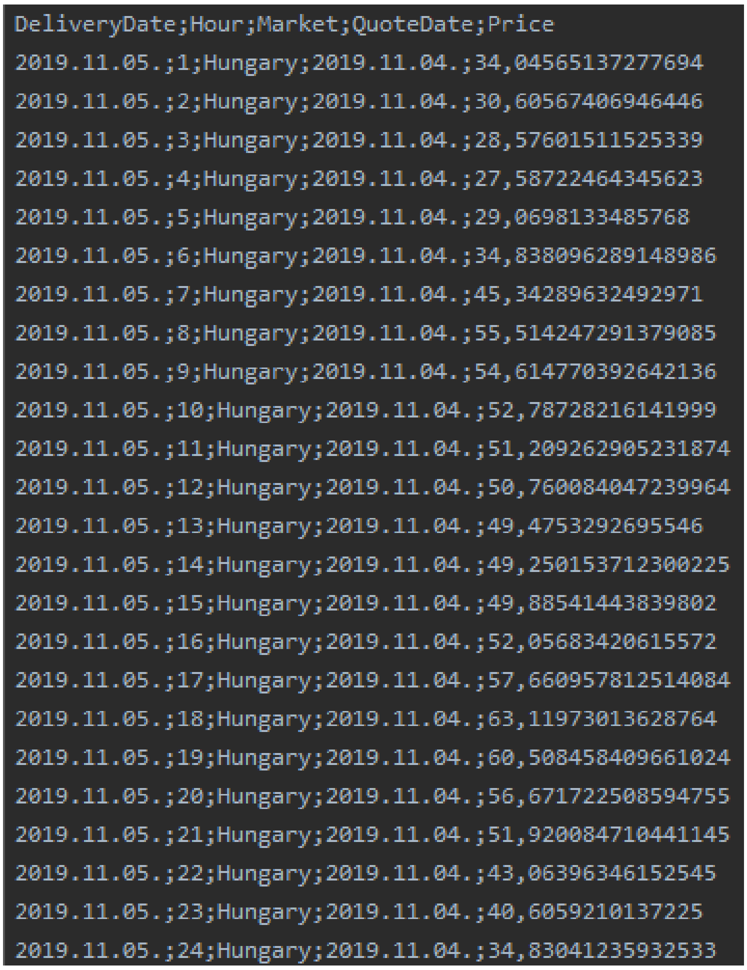
- The default format has two columns, datetime and price.
- The elmu format has five columns, DeliveryDate, Hour, Market, QuoteDate and Price.
- The default option uses dot notation in the prices column, while elmu uses commas.
- Default's datetime column is ‘yyyy-MM-dd hh:mm:ss’ formatted, while elmu's uses the ‘yyyy.MM.dd’ format, without hour, minute and second. (Also, the case with QuoteDate)

Default Format:



```
datetime,price
2019-11-05 00:00:00,34.04565137277694
2019-11-05 01:00:00,30.60567406946446
2019-11-05 02:00:00,28.57601511525339
2019-11-05 03:00:00,27.58722464345623
2019-11-05 04:00:00,29.0698133485768
2019-11-05 05:00:00,34.838096289148986
2019-11-05 06:00:00,45.34289632492971
2019-11-05 07:00:00,55.514247291379085
2019-11-05 08:00:00,54.614770392642136
2019-11-05 09:00:00,52.78728216141999
2019-11-05 10:00:00,51.209262905231874
2019-11-05 11:00:00,50.760084047239964
2019-11-05 12:00:00,49.4753292695546
2019-11-05 13:00:00,49.250153712300225
2019-11-05 14:00:00,49.88541443839802
```

Elmu Format:



```
DeliveryDate;Hour;Market;QuoteDate;Price
2019.11.05.;1;Hungary;2019.11.04.;34,04565137277694
2019.11.05.;2;Hungary;2019.11.04.;30,60567406946446
2019.11.05.;3;Hungary;2019.11.04.;28,57601511525339
2019.11.05.;4;Hungary;2019.11.04.;27,58722464345623
2019.11.05.;5;Hungary;2019.11.04.;29,0698133485768
2019.11.05.;6;Hungary;2019.11.04.;34,838096289148986
2019.11.05.;7;Hungary;2019.11.04.;45,34289632492971
2019.11.05.;8;Hungary;2019.11.04.;55,514247291379085
2019.11.05.;9;Hungary;2019.11.04.;54,614770392642136
2019.11.05.;10;Hungary;2019.11.04.;52,78728216141999
2019.11.05.;11;Hungary;2019.11.04.;51,209262905231874
2019.11.05.;12;Hungary;2019.11.04.;50,760084047239964
2019.11.05.;13;Hungary;2019.11.04.;49,4753292695546
2019.11.05.;14;Hungary;2019.11.04.;49,250153712300225
2019.11.05.;15;Hungary;2019.11.04.;49,88541443839802
2019.11.05.;16;Hungary;2019.11.04.;52,05683420615572
2019.11.05.;17;Hungary;2019.11.04.;57,660957812514084
2019.11.05.;18;Hungary;2019.11.04.;63,11973013628764
2019.11.05.;19;Hungary;2019.11.04.;60,508458409661024
2019.11.05.;20;Hungary;2019.11.04.;56,671722508594755
2019.11.05.;21;Hungary;2019.11.04.;51,920084710441145
2019.11.05.;22;Hungary;2019.11.04.;43,06396346152545
2019.11.05.;23;Hungary;2019.11.04.;40,6059210137225
2019.11.05.;24;Hungary;2019.11.04.;34,83041235932533
```

Archiving, versions

Along with the generated HPFC file, other files and data are archived also in separate files, such as:

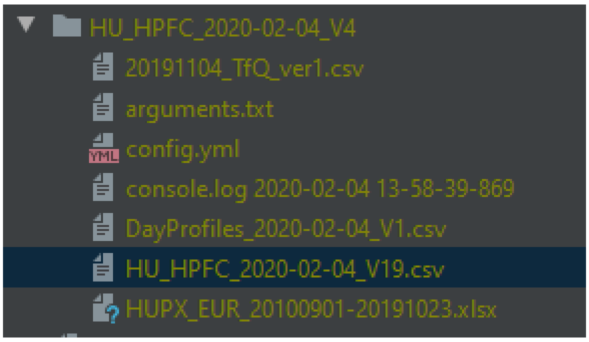
- the application's console output,
- the config file,
- the TFQ file,
- the spot data file,
- date profiles,
- command line arguments, that were used for HPFC generation.

The naming conventions for the output files are the following:

Every time a HPFC generation gets past the validation stage (so the data and config are in the correct format), a new folder is created. This folder's name is 'HU_HPFC__DATE__V__VERSION'.

For example, a valid folder name is HU_HPFC_2020-02-04_V1. It is important to note, that versioning restarts every day, since the generation's date is considered unique. Within this folder are the aforementioned output files.

Archive Folder:



```
HU_HPFC_2020-02-04_V4
    20191104_TfQ_ver1.csv
    arguments.txt
    config.yml
    console.log 2020-02-04 13-58-39-869
    DayProfiles_2020-02-04_V1.csv
    HU_HPFC_2020-02-04_V19.csv
    HUPX_EUR_20100901-20191023.xlsx
```

As far as versioning is concerned for these files, the console.log file uses the generation's start datetime with milliseconds precision. The generated HPFC file uses the same principles as the folder itself, but the version of these two objects can differ. The reason for this is that the generated HPFC csv file is created only on successful runs, and not on user interruption or failure, but the folder is always created after the validation stage. This means, we can end up with a folder with the version number 7, but with the generated HPFC file with version number 5.

9. Error messages

Export

Wrong argument, choices: calendar.

This error means, that the **'target'** argument of the export command was either mistyped or given an invalid value. The export command only accepts calendar as its parameter.

Target_file has wrong file extension.

This error means, that the **target-file** command line option has wrong file extension. This option only accepts .csv files as parameters.

Something wrong with the given target file value! Please check the given value!

This error means, that the folder or the drive specified in the **target-file** option does not exist.

Compare

Invalid parameter: hpfc-format-first/second... Accepted values are: ...

This error means, that either the hpfc-format-first or the hpfc-format-second option's value is not in the valid pool of choices. Valid choices are default and elmu.

Missing file path! / file is not found!

These errors mean, that either the first or second curve's path wasn't specified properly.

Abort comparing!

This error means, that either the start and end dates are wrong, or some problem occurred with either of the curves. Please, make sure, that the HPFCs are formatted correctly. For the correct formatting, please consult the 'HPFC Results' section of this manual.

Validate

The input data contains not date value(s)! Please check the values of the following values in your configuration file under extra_holidays_calendar.

Check the first column of the **'extra_holidays_calendar'** section of the configuration file for potential missing values or typos.

The extra calendar contains invalid day type(s)! Please check the values of the following values in your configuration file under extra_holidays_calendar.

Check the second column of the **'extra_holidays_calendar'** section of the configuration file for potential missing or invalid values or typos.

The extra calendar contains workday that is not Saturday! Please check the values of the following values in your configuration file under extra_holidays_calendar.

Check both columns of the **'extra_holidays_calendar'** section of the config for date - day type mismatches.

Invalid column names in the created data structure! Expected columns: {correct_columns} Columns after parse: {wrong_columns}.

Check the calendar csv's columns. Valid column names are **datetime** and **day_feature**.

File does not exist!

Specify a path to a file, that exists.

The input data contains empty value(s)! Please check the values of the following rows in the input file.

Check the rows mentioned in the input file for potential missing or mistyped values.

The calendar contains invalid day type(s)! Please check the values of the following rows in the input file.

Check the **day_feature** column of the calendar for incorrect values.

Valid values are: 0: MON, 1: TUE, ... 6: SUN, 7: Every weekday treaded as similar days, 8: TUE, WED, THU treated as similar days (with MON, FRI being significantly different), 9: SAT and SUN treated as similar days, 10: any holiday, 11: any bridge day (a day between a holiday and a weekend day, or between two holidays.) 12: any day before or after a holiday, 13: a separate day type for working days between Christmas and the New Year’s Day.

The calendar contains invalid date! Please check the values of the following rows in the input file.

Check the calendar csv for mistyped dates.

Generated date values are not the same with the input data! Please check the values of the following rows in the input file

A value mismatch occurred when trying to compare the original values from the calendar and the generated values from the first and last piece of data in the calendar csv.

There are empty values in the input Forward data. Please verify the following input rows.

Check the rows for empty cells in the forward data file.

There are invalid values in the input Forward data Product column. Please check the following rows in the input file.

Valid product values are Peak and Base.

There are invalid values in the input Forward data 'DeliveryYear' column. Please check the following rows in the input file.

Check for invalid date formats in the forward data file.

There are invalid values in the input Forward data 'Price' column. Please check the following rows in the input file.

Check for letters in the price column.

Invalid column names in the created data structure! Expected columns: {correct_columns} Columns after parse: {wrong_columns}

Check the spot data file for invalid column names. Valid column names are **datetime** and **price**.

There are price values that exceed the threshold limit! Max threshold: {max}, Min threshold: {min}, Invalid rows: {wrong_rows}.

Either set the prices threshold settings in the config to fit the data or adjust the input data's price column accordingly.

Could not find historic spot data for spot date: ... adjusted by historical weight count.

Check the year_weights settings in the config and check whether the input data is sufficient. Adjust the number of weights accordingly.

Invalid column names in the created data structure! Expected columns: {correct_columns} Columns after parse: {wrong_columns}.

Check the spot data file's columns. Valid columns are datetime and price.

The input data contains empty value(s)! Please check the values of the following rows in the input file.

The spot data file's datetime and/or price column contains empty cells, values.

There are price values that exceed the threshold limit! Max threshold: {max}, Min threshold: {min}, Invalid rows: {wrong_rows}.

The spot data file's price column contains a higher or lower value than the specified threshold in the configuration file. Adjust **spot_data.min/max_price_threshold** in the configuration file accordingly.

Could not find historic spot data for spot date: ... adjusted by historical weight count...

For every weight specified in the configuration file under **day_profiler.year_weights**, there must be historic spot data for years going backwards.

Generate

Invalid parameter: hpfc-format. Accepted values are: ...

When specifying the **hpfc-format** command line option, accepted values are **default** and **elmu**.

HPFC generation cannot proceed with the given input parameters! Valid date formats are: YYYY-mm-dd, YYYY/mm/dd, YYYYY.mm.dd.

When specifying the end-date and spot-end-date parameters, only the aforementioned date formats are accepted.

The archive base location cannot be None, it should be specified in the config or via CMD parameter.

A folder must be specified either in the configuration file or input manually as a command line argument when trying to generate HPFC.

HPFC generation aborted, because of invalid input data!

Please, consult the validation error part of this section.

Start date cannot be later than End date.

Check the input data files and the end-date command line argument.

The tenor presented in the Forward data file is not valid. Please check these indexes in the Forward data file.

The forward data file might contain typos in value cells.

Archive folder cannot be created.

The archive folder might already exist, or the application tries to create it in a folder, where the current user doesn't have the necessary permission level.

Check the **archive_base_path** setting in the configuration file for potential typos and specify a folder, that is on a valid drive and in a folder with write permission.

Script is already running!

Trying to start the application while another instance is running results in this error. Either terminate the running instance, or wait for it, to complete.

The program has run into an error during HPFC calculation.

Any unspecified error will produce this message, and it will come with the details of the original exception.

10. Process of HPFC generation

Algorithm overview

The model consists of two major steps:

- Firstly, the calculation of the daily profiles, from the historical spot prices. This will ensure the shape of the HPFC curve.
- Secondly, the calibration of these forecasted values to the desired price level, which is based on the given forward data.

Daily Profiles Calculation

For creating the shape of the HPFC curve the given historical hourly spot prices are used.

At first, every historical price must be labelled according to its date's type. These day feature categories can be defined in the configuration file as described in the previous chapters.

Before averaging, the correct base resolution must be configured. In this model two types of weighting are possible, monthly and weekly.

The first step of creating the daily profiles is to calculate an average of these annotated historical prices for every hour, day feature, chosen base resolution (week or month) and year.

The average is calculated as follows:

$$\alpha_{h,d,b,y} = \frac{1}{\left|I_{h,d,b,y}\right|} \sum_{i \in I_{h,d,b,y}} s^{i}_{h,d,b,y}$$

where

- $\alpha_{h,d,b,y}$ is the base average for hour h, day type d within base resolution (month/week) b and year y,
- $s^{i}_{h,d,b,y}$ is the i-th observed value for hour h, day type d within base resolution (month/week) b and year y,
- $I_{h,d,b,y}$ denotes the set of indexes of the spot prices for hour h, day type d, base resolution (month/week) b and year y,
- $|.|$ stands for the total number of set elements.

Weighting:

The next step is to calculate the weighted average by the chosen base resolution. This feature provides the user the possibility of considering other months / weeks as well. The weights also can be defined in the configuration file. This weighting scheme uses only one central node therefore an odd number of weights should be given.

It is important to note, that with the month weighting and as well as with the week weighting it is quite typical to take values from adjacent months and sometimes even from the adjacent years.

This weighted average is calculated in following way:

$$\beta_{h,d,b,y} = \sum_j u^j_{h,d,b,y}\ \alpha^j_{h,d,b,y}$$

where

- $\beta_{h,d,b,y}$ is the weighted average for hour h, day type d within base resolution (month/week) b and year y,
- j denotes the base resolution node and takes 0 for the central node,
- $u^j_{h,d,b,y}$ is the weight given to node j so that $u^j_{h,d,b,y} \in [0,1]$ and $\sum_j u^j_{h,d,b,y} = 1$ If the required values do not exist, the weights for the existing values are linearly re-scaled to sum up to 1.
- $\alpha^j_{h,d,b,y}$ is the base average for hour h, day type d within base resolution (month/week) b and year y, for node j. Calculated by the previous formula for node j.

Year weighting:

The last step is the weighted average calculation by years. The used weights can be given in the configuration file as well. The first weight in the list will correspond to the nearest (usually present) year.

For hour h, day type d within the base resolution (month/week) b, the forecasted value is:

$$f_{h,d,b} = \sum_y w_y\ \beta_{h,d,b,y}$$

where

- $f_{h,d,b}$ is the forecasted value, the yearly weighted average for hour h, day type d within base resolution (month/week) b,
- w_y is the weight assigned to the year y so that $w_y \in [0,1]$ and $\sum_y w_y = 1$ If needed the weights are linearly rescaled to sum up to 1.
- $\beta_{h,d,b,y}$ is the weighted average for hour h, day type d within base resolution (month/week) b and year y. Calculated with the previous formula.

Finally, as a result for every day feature, we have a forecasted value for every hour in every week/month for a whole year.

Missing Day Profile filling

In some cases, there could be day types in the future, that fall on weeks/months that did not occur in the past years. In this case, there would be holes in the future data frame. During the Daily Profiles calculation, the Missing Day Profile filling prevents this situation.

Optionally, an extra filling can be executed before or after the weighting, with the “execution_order” setting of the "day_profile_replacement” configuration. When using the extra filling option, the weighting and/or the year weighting use the replaced proxy data too. Executing the extra filling before the weighting usually improves the result for day types with less data, but it can also produce malformed curves in the adjacent nodes.

The application has four steps to fill missing data.

- christmas_newyear handling
    - If christmas_newyear day type is in use, the application copies this profile to all nodes. This option is only used when it’s necessary, but this is usually the easiest way out.
- holiday handling
    - In this step the algorithm is looking for 5-5 adjacent weeks or 1-1 adjacent months for fill the missing holidays with another holiday profile. The reason for this is that Easter may fall either in March or in April.
- filling_by_types
    - In this step, the missing profiles are replaced with another day type’s profiles from the same node. The replacing orders are implemented in the code. Later they may be added to the configuration file.
- filling_by_nodes
    - In this step, the missing profiles are replaced with from the adjacent nodes with the same day type. The range of the adjacent nodes can be set in the configuration.

Extra profile filling:

The extra profile filling process can be executed before or after the weighting. First, the application executes the christmas_newyear handling and the holiday handling steps. Then come the filling_by_types and filling_by_nodes steps. The order of these two can be set in the configuraton.

Executing the filling_by_types step first may result in a more accurate curve, in case some weekday or special profiles are missing. Then the filling_by_types and filling_by_nodes steps are executed.

Required profile filling:

This filling is always executed after the yearly weighting. This ensures that the future data frame has all the necessary data.

If the extra profile filling was not executed, the application executes the christmas_newyear handling and the holiday handling steps. This time the filling_by_nodes is executed first.

Combining Daily Profiles with Forward Data

Based on the daily profiles the hourly forward curve shape is generated for the future dates of the HPFC interval.

The desired forward price level is given in the TFQ forward data file. This contains daily, weekly and monthly arbitrage-free average prices. The previously obtained forecasted values are calibrated to achieve these averages for every specified time period.

Days

In this case the considered time period is exactly one day, which is specified in the TFQ file.

Let us use the following notation:

- f_t denotes the previously calculated forecasted values for this day, for every hour t in the specified time period.
- P is the set of peak hours during this day. (This set can be empty.)
- O is the set of off-peak hours during this day. (This set also can be empty.)
- $q_{o,p}$ is the given peak or off-peak forward quote from the TFQ file.
- And $\hat{f}_t$ stands for the corresponding calibrated values, for every hour t in the specified time period.

For every hour t in the specified time period:

$$\hat{f}_t = f_t\, m_{o,p},$$

where $m_{p,o}$ is the peak or off-peak multiplier.

These are calculated by the following formula:

$$m_p = \frac{q_p\ |P|}{\sum_{t \in P} f_t}\ , \qquad m_o = \frac{q_o\ |O|}{\sum_{t \in O} f_t}$$

Weeks

Technically, this case is the same as the days was before. The time period is still specified by the given forward quotes in the TFQ file, but it is a whole week instead of one day.

The notation and the formulas are the same as well.

Months

In case of the monthly periods we must pay attention for the overlapping days. This can occur when the first day of the first monthly tenor is not Monday. The previous formulas need to be modified to handle this situation. These possible overlapping days have already been calibrated, so they cannot be modified again during the monthly calibration.

Hence, we need to introduce some new notation:

- $\hat{P}$ is the set of the peak hours during the specified time period without the previously calibrated peak hours.
- $\hat{O}$ is the set of the off-peak hours during the specified time period without the previously calibrated off-peak hours.

For every hour t, which is not yet calibrated in the specified time period:

$$\hat{f}_t = f_t \, \widehat{m}_{o,p},$$

where $\widehat{m}_{p,o}$ is the modified peak or off-peak multipliers, calculated by the following formula:

$$\widehat{m}_p = \frac{q_p \, |P| - \sum_{t \in P \setminus \hat{P}} \hat{f}_t}{\sum_{t \in \hat{P}} f_t} \,, \qquad m_o = \frac{q_o \, |O| - \sum_{t \in O \setminus \hat{O}} \hat{f}_t}{\sum_{t \in \hat{O}} f_t}$$

In the end, the $\hat{f}_t$ values are calculated for every hour in the desired HPFC interval.

11. FAQ

How can I check the HPFC calculation results?

Running HPFC generation either locally, or on a server, it is the user’s responsibility to provide the necessary input files and to check the results of the generated HPFC. Users should check for potential missing values or outliers in the generated HPFC, which is found in the Output folder. The Input and TFQ folders, and the files within these folders should be checked the same way.

Where can state holidays and workdays be set?

Holidays and workdays can be set in the configuration file.

Where to check for errors and their causes in the calculation process?

If the HPFC file is not created at the end of the calculation process, then looking for a possible cause in the log file can help to solve the issue. The log file for every run can be found in the archive folder for every respective run.

What happens, when a file, other than the TFQ file, is modified?

Modifying files, other than the TFQ file will not trigger an automatic calculation and HPFC generation. In this case, the process must be started manually. Modifying the TFQ file will trigger HPFC generation automatically with the modified data.

How to distribute the generated HPFCs?

Distributing generated HPFCs can be done in the usual communication channels, such as email. Another option can be, to generate the HPFCs to a shared folder, so all users with permission to the shared folder can access the generated HPFCs immediately. The system does not support notifications, it does not send messages.

How does the system handle multiple instances of the application at the same time on the same server/computer? (e.g. Two people trying to generate HPFC at the same time?)

Only one instance of the application can be in a running state. This is achieved by a lock file, which is when present, the application exists immediately with an error message, stating that another instance of the application is already running.

How to load new spot data into the system?

New spot data files are added manually to the input folder.

Is it possible, to start HPFC generation manually?

Beside automatic HPFC generation, triggered by new TFQ data, users can start the HPFC generation process manually on their computer. To be able to do this, the application must be installed on a local computer with every necessary folders and files to a folder with read and write permissions. Also, the input folder with the necessary files for HPFC generation must be created. The installation makes it possible, to generate HPFC locally, even with custom input files.

Important! Running HPFC generation locally with incorrect values in the configurations file, and wrong permissions can result in overwritten files in the servers output folder. (This means it overwrites the current HPFC version in the server's output folder, so it generates a new file.)

What's the reason behind the potential difference between the version numbers of the generated HPFC file, and its folder?

The reason for this, is that the creation time for the archive folder and the HPFC file is different. The archive folder is created right after the validation phase, but the HPFC file is only generated at the end of the calculation. Errors or user interruption may cause version mismatch between the folder and the generated file.

Can the HPFC generation process be started by double-clicking the hpfc.exe?

Double clicking the exe opens a command prompt, which closes shortly after. The reason for this is opening the exe runs the script without any arguments and options. Running the application this way lists the possible command line arguments and hints for them, then exits the application. Typing hpfc.exe in the command prompt while in the installation folder causes the same effect, without closing the command prompt.

Can the HPFC generation process be stopped?

Pressing the keyboard combination CTRL-C halts the script execution immediately. The User is notified by a message. Depending on which stage did the user stop the script, there might or might not be an archived folder. Script execution can be stopped any time while the application is running.