Introduction

This is a brief documentation for Matlab code simulating OFR responses to changes in exhaust gas concentrations and calculating the SOA production factors with different methods.

The documentation does not cover the classes and functions in detail. The best way to learn the use of the code is to read the paper and check the examples in 'plotScripts' folder.

Classes

Ofr

This class contains:

- The name of the OFR
- OFR transfer function (called 'rtd' in the class)
- Parameters t_{peak} and t_{mean} of the transfer function. The mean residence time is t_{peak} and t_{mean} is the residence time corresponding to the maximum value of the transfer function.

Once the class instance is created, it is possible to add the transfer function with function 'setRtd'. The class automatically calculates t_{peak} and t_{mean} based on the transfer function.

Cycle

This class contains:

- The original gas concentrations, SOA formation potential, speed profile and exhaust flow rate of a driving cycle.
- The gas concentrations sampled from CVS.
- Cumulative emissions, momentary SOA production factor and cumulative SOA production factor (i.e., the production factor between the beginning of the cycle and time *t*).
- Function 'soaPfBetween' to calculate the SOA production factor between any two points in the driving cycle.

Once an OFR class has been added to the Cycle with function 'addOfr', the class contains also:

- Gas concentrations and SOA formation potential measured downstream the OFR sampling from tailpipe.
- Gas concentrations and SOA formation potential measured downstream the OFR sampling from CVS.
- Cumulative emissions, momentary SOA production factors and cumulative SOA production factors calculated with different methods that are presented in the paper.
    - Depending on the method used, the OFR-specific emissions do not necessarily represent anything measured downstream of the OFR. For example, in 'standard' method, the cumulative OFR CO_2 emission for the OFR is identical to the original CO_2 emission.
- Exhaust flow rate convolved with the OFR transfer function (to be used when calculating the emissions with 'convolution' method).
- Note that simulating concentration at OFR outlet at t = 0 s requires knowledge of inlet concentration at t < 0 s because of the residence time of the OFR. To make the simulation at t = 0 s possible, it is assumed (in function 'simulateRtd') that the OFR has been sampling

stable concentration between -700 s < t < 0 s. The value of the stable concentration in this period is assumed to be the inlet concentration at t = 0 s.

Other functions:

- deconvolveOfrData
  - Deconvolves the SOA concentration that is simulated at OFR outlet when sampling directly from tailpipe (i.e., the vector cycle.ofr.tailpipe.soa_g_per_m3).
- setOfrDelayCorrection
  - By default, the OFR data is delay-corrected with the value of t_{peak}. The delay-corrected concentration data is denoted with '_shifted'. This function changes the delay correction value, and also recalculates the emission data for the specific OFR since the delay-corrected concentration data is used in most PF calculation methods.
- soaPfBetween
  - Calculates the SOA PF for a specific interval within the driving cycle based on the true data or concentrations measured at OFR outlet. In case of OFR, the user must choose which method will be used.

Examples

Example 1

Create a cycle (with name 'A') where gas concentrations are constant and exhaust flow rate follows a step function, the speed data is not available, SOA yield is assumed to be 0.15, and adding an end buffer of 600 s where all concentrations and the exhaust flow rate are zero:

```
c.time = (1:1:100)';
c.hc_ppm = 10.*ones(size(c.time));
c.co2_percent = 5.*ones(size(c.time));
c.exhFlow_m3_per_s = 1e-3.*ones(size(c.time));
c.exhFlow_m3_per_s(50:end) = 1e-2;
c.speed = NaN;
soaYield = 0.15;
endBufferLength = 600;

cycle = Cycle('A', c, soaYield, endBufferLength);
```

Note that all data must have 1 s time resolution.

Next, compare the concentration of HC at tailpipe to the concentration in CVS (the tailpipe concentration is constant but the CVS concentration not because of the changing exhaust flow rate):

```
figure;
plot(cycle.original.time, cycle.original.tailpipe.hc_ppm);
hold on;
plot(cycle.original.time, cycle.original.cvs.hc_ppm);
```

Example 2

Create a cycle with a square pulse of 10 ppm HC and constant exhaust flow rate:

```
c.time = (1:1:100)';
c.hc_ppm = zeros(size(c.time));
c.hc_ppm(10:20) = 10;
c.co2_percent = 5.*ones(size(c.time));
c.exhFlow_m3_per_s = 1e-3.*ones(size(c.time));
c.speed = NaN;
```

```
soaYield = 0.15;
endBufferLength = 600;

cycle = Cycle('A', c, soaYield, endBufferLength);
```

Create a class for DOFR and add the DOFR transfer function (corresponding to ‘low’ UV lamp setting and length of 700 s) to it. Add the DOFR class to the cycle:

```
dofr = Ofr('DOFR');
dofr.setRtd(calcDofrRtd('low',700));
cycle.addOfr(dofr);
```

Plot the true HC concentration, HC concentration downstream DOFR and the delay-corrected HC concentration downstream DOFR:

```
figure;
plot(cycle.original.tailpipe.hc_ppm);
hold on;
plot(cycle.DOFR.tailpipe.hc_ppm);
plot(cycle.DOFR.tailpipe.hc_ppm_shifted);
```