Measuring a Flame transfer function (FTF)

Table of Contents

1. Before we start

Before proceeding we will `clear` the workspace, and run the `dir` command to ensure that we are in the correct workspace. You should be able to see a folder named `tutorial_functions`.

```
clear
dir
```

```
.                   P8kW                WK03_FTM.mlx
..                  WK01_MMM.mlx        tutorial_functions
P12kW               WK02_FTF.mlx
```

```
addpath(genpath('tutorial_functions'))
```

2. Introduction

In this tutorial we will learn how to obtain flame transfer functions (FTFs). FTFs are essential for:

1. Studying how flames respond to acoustic forcing.
2. Linear stability analyses where the heat release rate is not modelled from first principles.

In this tutorial we will use the same combustor used for the Multi-Microphone-Method, but now we will add a flame. First let's take a look at the combustor.

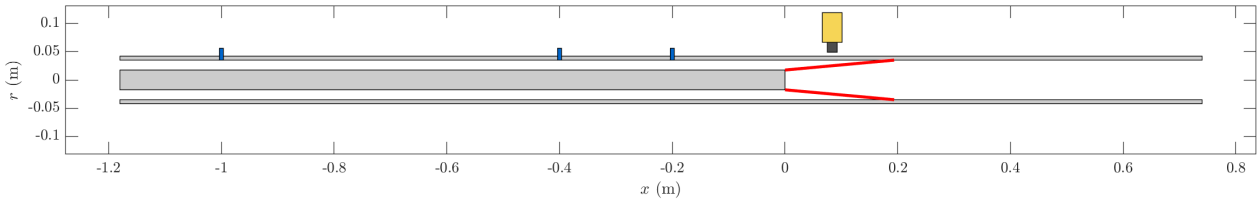


Fig. 1: Schematic of the combustor.

The configuration that we will be using is equipped with 3 differential pressure transducers (blue probes), a photomultiplier (PMT) (yellow box) and a speaker, which will be used in parts of the tutorial to force the flame. The flame that we will be analysing is a "V" flame attached to the rod at the centre of the configuration. As in previous parts of the tutorial the reference point is set such that the flame in Fig. (1) anchors at $x = 0$.

In this tutorial we are interested in studying the response of the flame to acoustic perturbations, through a single quantity named the Flame Transfer Function (FTF). The acoustic perturbations can be in the form of pressure (p') or velocity (u') fluctuations. However, it is well known [1] that in low Mach number applications, flames respond much strongly to velocity fluctuations than to pressure fluctuations. Therefore in this tutorial we will be focusing in the following definition for the FTF:

$$\mathrm{FTF}\left(\omega\right) = \frac{\hat{Q}/\bar{Q}}{\hat{u}/\bar{u}} \qquad \text{Eq. (1)}$$

Here, $\hat{Q}$ and $\hat{u}$ are the first Fourier coefficients at frequency ω of the unsteady rate of heat release (Q') and the velocity fluctuations (u') respectively. We assume that $\bar{Q}$ and $\bar{u}$ are known (for instance, by a simple calculation involving the mass flow rate times the low heating value of the fuel).

The FTF is by definition a complex quantity, and it is often plotted in terms of a gain and a phase, such that $\mathrm{FTF}\left(\omega\right) = G\left(\omega\right)\ \exp\left(i\ \varphi\left(\omega\right)\right)$, where

$$\begin{aligned} G\left(\omega\right) &= \left|\mathrm{FTF}\left(\omega\right)\right| \\ \varphi\ \left(\omega\right) &= \mathrm{angle}\left(\mathrm{FTF}\left(\omega\right)\right) \end{aligned}$$

The most basic example of an FTF is the so called $n-\tau$ model in which

$$\begin{aligned} G\left(\omega\right) &= n \\ \varphi\ \left(\omega\right) &= \tau \end{aligned}$$

n is an interaction index and τ the time delay. This FTF is usually valid for a small range of frequencies (where the gain and phase are approximately constant).

[1] Lieuwen, T. (2003). Modeling Premixed Combustion-Acoustic Wave Interactions: A Review. Journal of Propulsion and Power, 19(5), 765–781. doi:10.2514/2.6193

3. Objectives

For the configuration given in Fig. (1) the tasks will be:

1. Ensure that the operating conditions chosen represent a stable point.
2. With a stable operating point compute the Flame Transfer Function.

4. Setting up the simulation

For the configuration shown in Fig. 1 we will assume that inlet of the configuration is choked (in experiments this type of boundary condition is useful to decouple the acoustics upstream of the combustor, i.e., fuel/oxidizer lines), and we will specify the velocity of the flow $\bar{u}_1$ and its temperature $\bar{T}_1$. We will also assume that the outlet of the configuration is open to the atmosphere and specify the outlet pressure $\bar{p}_2$. Since there is a flame, we will consider a mean heat release of $\bar{Q} = 100$ kW . Notice that we are using the subscripts 1 and 2 to denote the mean flow conditions before and after the area expansion at $x = 0$ respectively as shown in Fig. 2:

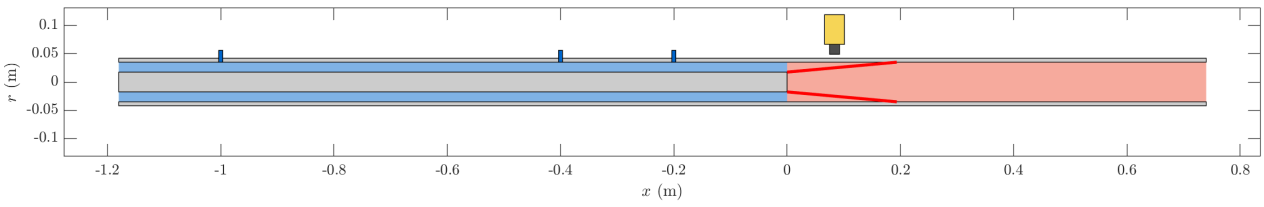


Fig. 2: Mean flow regions. In blue we show regions 1 (before the area expansion, cold flow), and in red we show region 2 (after the area expansion, hot flow).

```
% Define inlet and outlet conditions:
u1 = 30;       % Inlet velocity (m/s)
T1 = 293;      % Inlet temperature (m/s)
p2 = 101300;   % Outlet pressure (Pa)
Qbar = 100000; % Heat release rate (W)
```

In the presence of heat release, the flow can become self-excited. We need to ensure that when we are measuring the acoustic response of a flame we are using operating conditions which are stable (not self-excited). Therefore, to begin with, we will not consider any forcing:

```
% Define Forcing Conditions:
Forcing.Speaker = 'OFF';
Forcing.Speaker_Downstream = 'OFF';
```

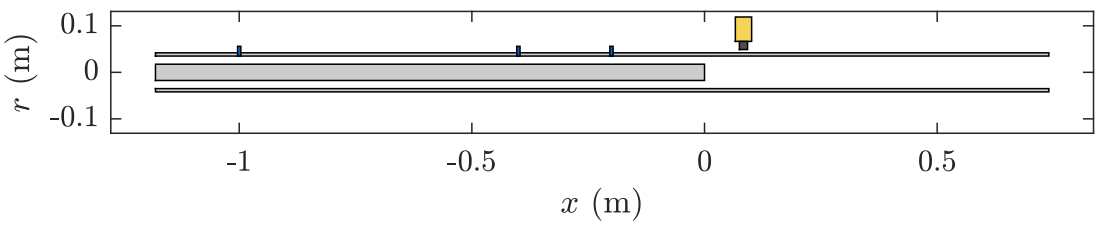
In the last step we need to set up the measurement probes as well as the sampling frequency and length of the measurements. This time we will turn the PMT on, and for this tutorial we will use pressure transducers to reconstruct the velocity field as we did in the MMM tutorial.

```
% Sampling frequency:
Measurement.Fs = 5000; % (Hz)
% Length of the measurements:
Measurement.Time = 2; % (s)

% Photomultiplier:
Measurement.PMT = 'ON';
% Differential Microphones:
x_mic =  [-0.2,-0.4,-1]; % (m)
Measurement.Mic = 'ON';
Measurement.Mic_Pos  = x_mic;
% Hot wire anemometry:
Measurement.HWA = 'OFF';
```

Before proceeding with the simulation, let's preview the set up.

```
% Preview set up:
figure
fn_plot_config(Forcing,Measurement)
```



5. Running the simulation

Now let's, run the simulation, make sure you wait until it says "Finished". It takes about 3 mins!

```
[~,~,Signals,Sim] = Run_Simulation(u1,T1,p2,Qbar,Forcing,Measurement);
```

```
Progress: 0%, 5%, 10%, 15%, 20%, 25%, 30%, 35%, 40%, 45%, 50%, 55%, 60%, 65%,
 70%, 75%, 80%, 85%, 90%, 95%,
 Finished
```

In the mean time take a look at an animation of the simulation:

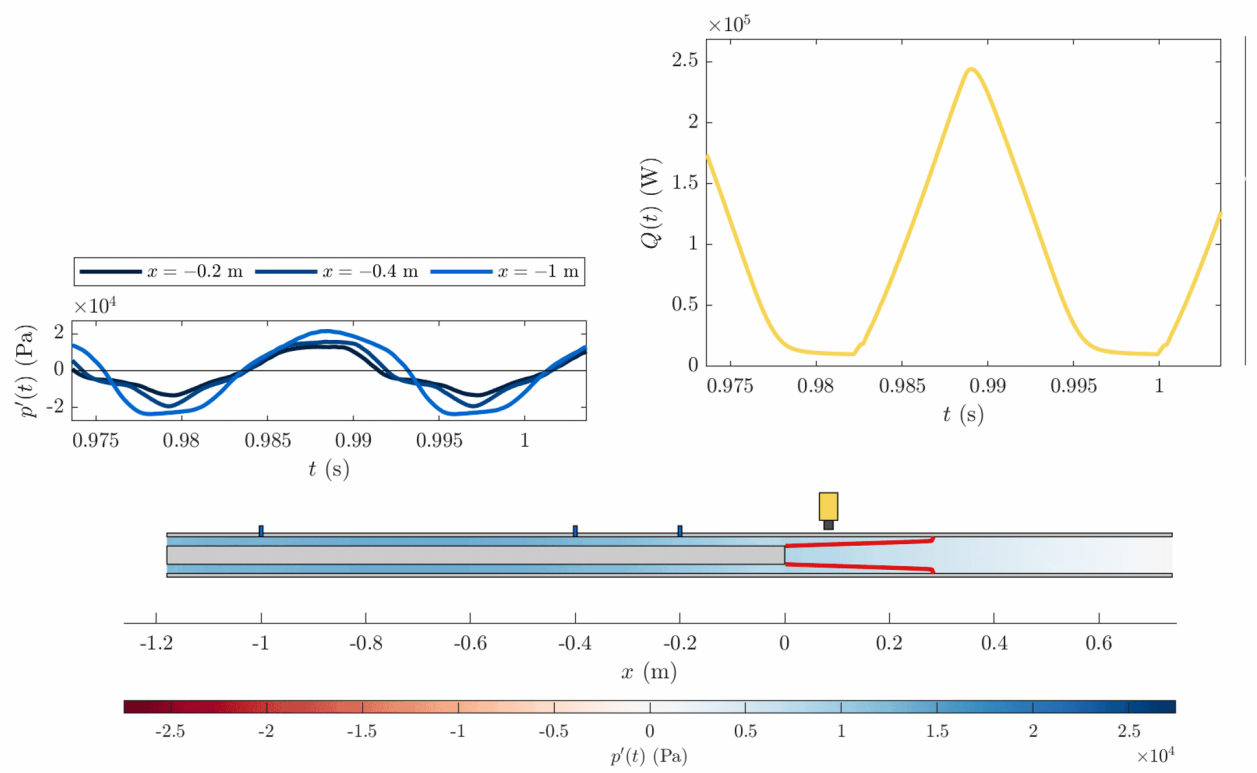


Fig. 3: Simulation of the combustor during a limit cycle oscillation.

From this animation, it is clear that the system is self-excited. At this stage the thermoacoustic feedback loop has been established and the system is oscillating in a limit cycle.

- In this simulation the rate of heat release is proportional to the flame surface area. Can you spot what happens to the flame when the heat release is almost zero?
- The pressure oscillations are quite strong, on the order of 20,000 Pa, that is about 20% of the mean pressure. Can acoustics still be considered linear?
- At this stage, can you tell if the signals are harmonic or just periodic?

Once the simulation is finished, let's continue to analyse these signals.

Optional:

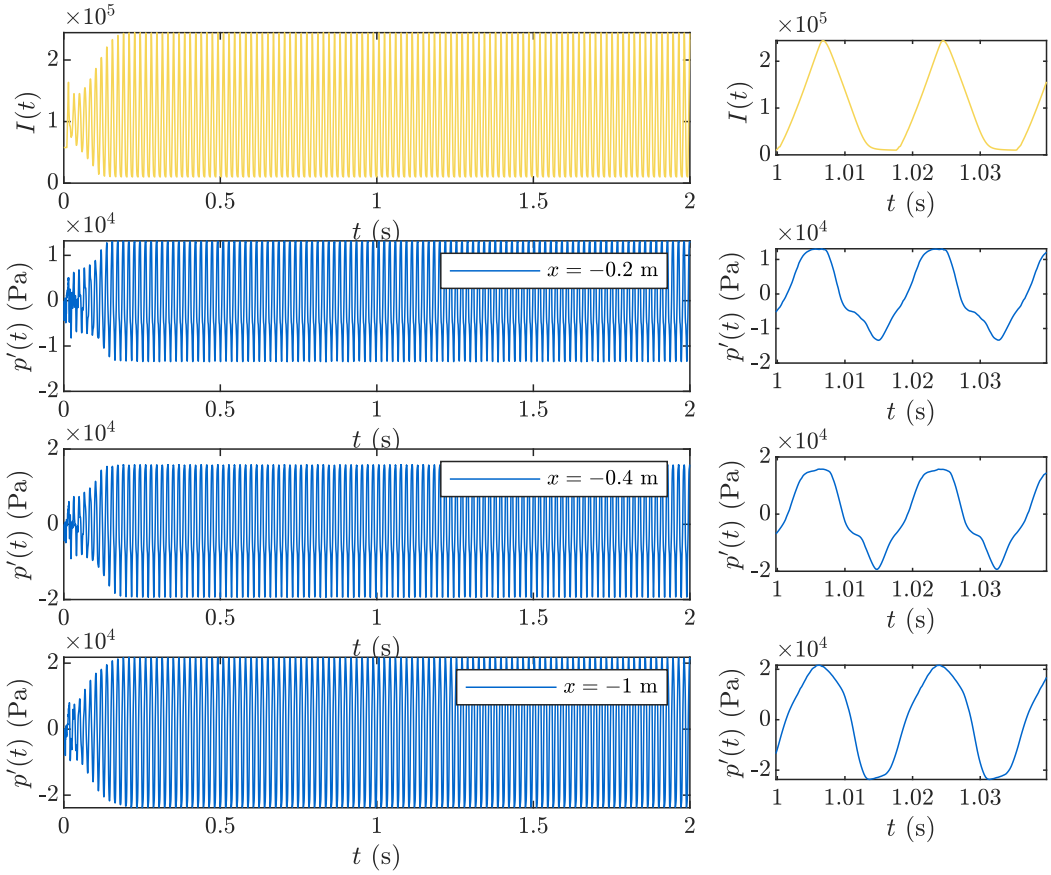
If you want to visualize the flame and the acoustic field in the combustor, uncomment and run the following commands. You may want to change the starting and end points, but try to keep the time difference around $\Delta t = 0.02$ s, or else terminate the animation using the `ctrl+c` command.

```
% t_start = 1.02;
% t_end = t_start + 0.02;
% Num_of_time_steps_in_plot = 400;
% stepsize = 2;
% fn_view_simulation(Sim,stepsize,t_start,t_end,Num_of_time_steps_in_plot)
```

6. Analysing pressure and heat release data

Let's begin by plotting the pressure and heat release measurements. Keep in mind that the pressure measurements obtained from the microphones will only provide us with the fluctuating part $p'(x,t)$ (since they are "differential pressure" transducers), while the PMT measurement will provide only an Intensity $I(t)$.

```
zoom_in_time_steps = 200;
fn_plot_Signals(Signals,Measurement,zoom_in_time_steps)
```



From this plot one can clearly see (again) that the operating point is self-excited. From the $t = 0$, the signals start to grow until they reach a certain amplitude. This amplitude is maintained through the rest of simulation. During self-excitation both the heat release and the pressure signals are **periodic**, but they are not **harmonic***. This is an indication of the presence of nonlinearities in the system. This operating point is of no interest because if we try to force the flame, then we will not know if the response is due to forcing or due to self-excitation.

* A harmonic signal is often referred as a signal oscillating at a single frequency.

7. Stable operating point

Let's change the operating conditions and find a stable point where we can force the flame to compute the FTF. Here we reduce the inlet velocity $\bar{u}_1$:

```
% Change the inlet velocity
u1 = 20;       % Inlet velocity (m/s)
```

Now let's run the simulation. It will take about 3 mins, wait until it says "Finished"!

```
% Run the simulation
[~,~,Signals,Sim] = Run_Simulation(u1,T1,p2,Qbar,Forcing,Measurement);
```

```
Progress: 0%, 5%, 10%, 15%, 20%, 25%, 30%, 35%, 40%, 45%, 50%, 55%, 60%, 65%,
 70%, 75%, 80%, 85%, 90%, 95%,
 Finished
```

In the meantime take a look at the corresponding animation:

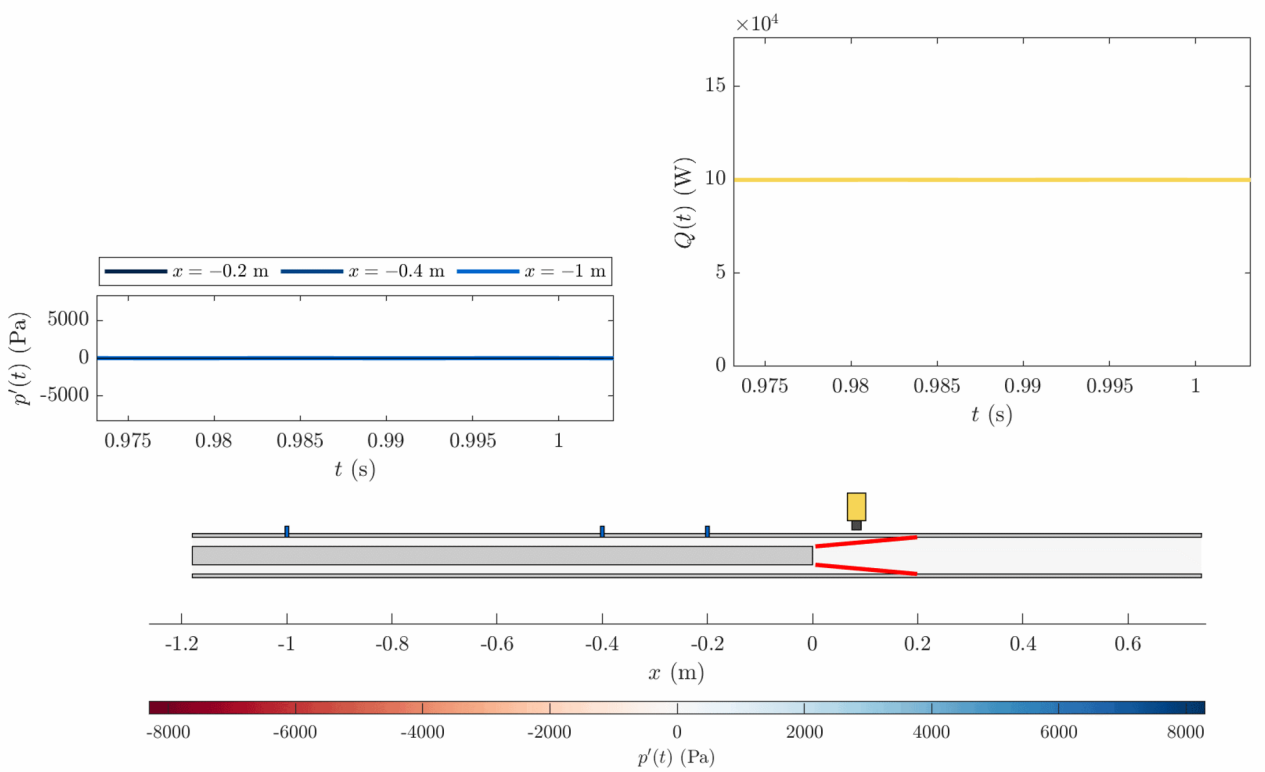


Fig. 4: Simulation of the combustor during in a stable operating point.

Now there are no fluctuations in heat release or pressure. The system is stable!

- Since there are no pressure oscillations the flame stays fixed in its mean position.

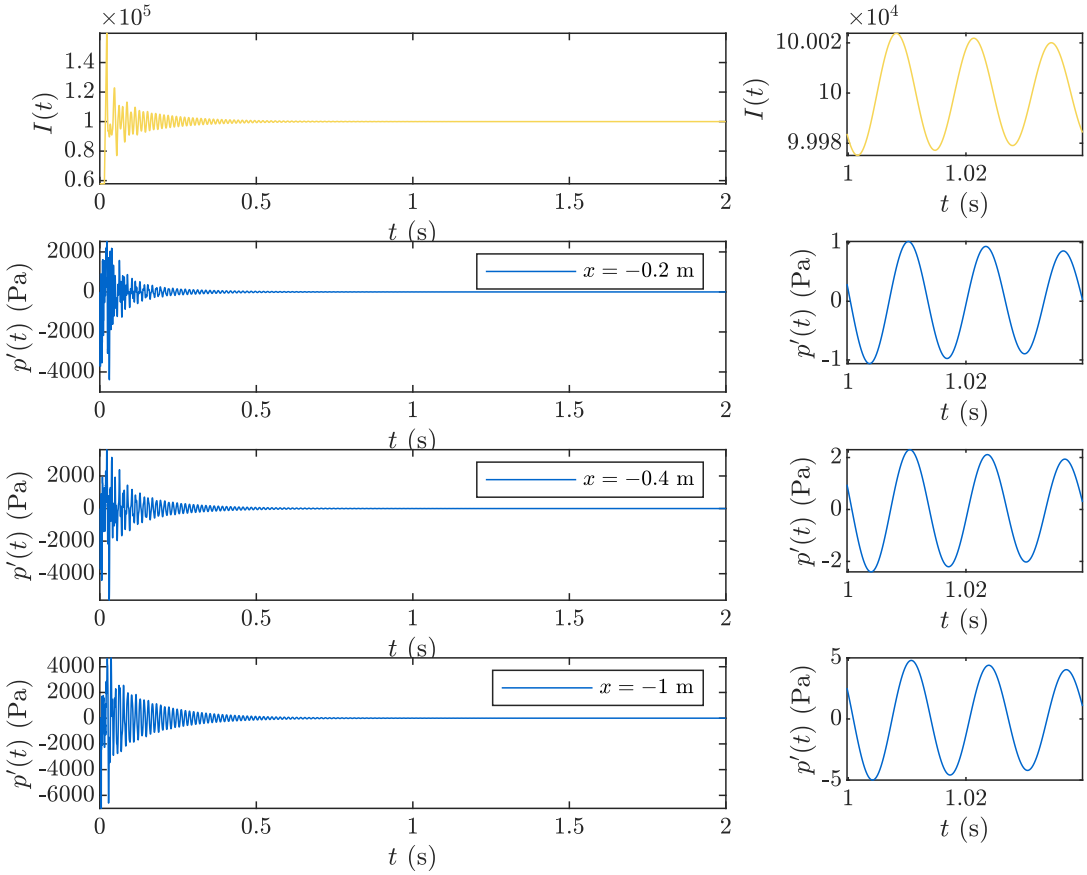
Optional:

If you want to visualize the flame and the acoustic field in the combustor, uncomment and run the following commands. You may want to change the starting and end points, but try to keep the time difference around $\Delta t = 0.02$ s, or else terminate the animation using the `ctrl+c` command.

```
% t_start = 1.02;
% t_end = t_start + 0.02;
% Num_of_time_steps_in_plot = 400;
% stepsize = 2;
% fn_view_simulation(Sim,stepsize,t_start,t_end,Num_of_time_steps_in_plot)
```

Let's plot the signals:

```
zoom_in_time_steps = 200;
fn_plot_Signals(Signals,Measurement,zoom_in_time_steps)
```



After an initial transient the fluctuations die out exponentially fast. After 1 second we can still observe some oscillations in the pressure field and the rate of heat release, but they are very small (less than 0.005% of the mean pressure). This means that the chosen operating point is indeed stable. We will use this point to measure the response of the flame to acoustic forcing.

8. Forcing the flame

Now that we have a stable operating point, let's mount the speaker and force the flame.

```
% Define Forcing Conditions:
Forcing.Speaker = 'ON';
Forcing.Voltage = 10;     % (V)
Forcing.Frequency = 100; % (Hz)
```

Now run the simulation:

```
% Run the simulation
[~,~,Signals,Sim] = Run_Simulation(u1,T1,p2,Qbar,Forcing,Measurement);
```

```
Progress: 0%, 5%, 10%, 15%, 20%, 25%, 30%, 35%, 40%, 45%, 50%, 55%, 60%, 65%,
  70%, 75%, 80%, 85%, 90%, 95%,
 Finished
```

In the meantime observe the corresponding animation:

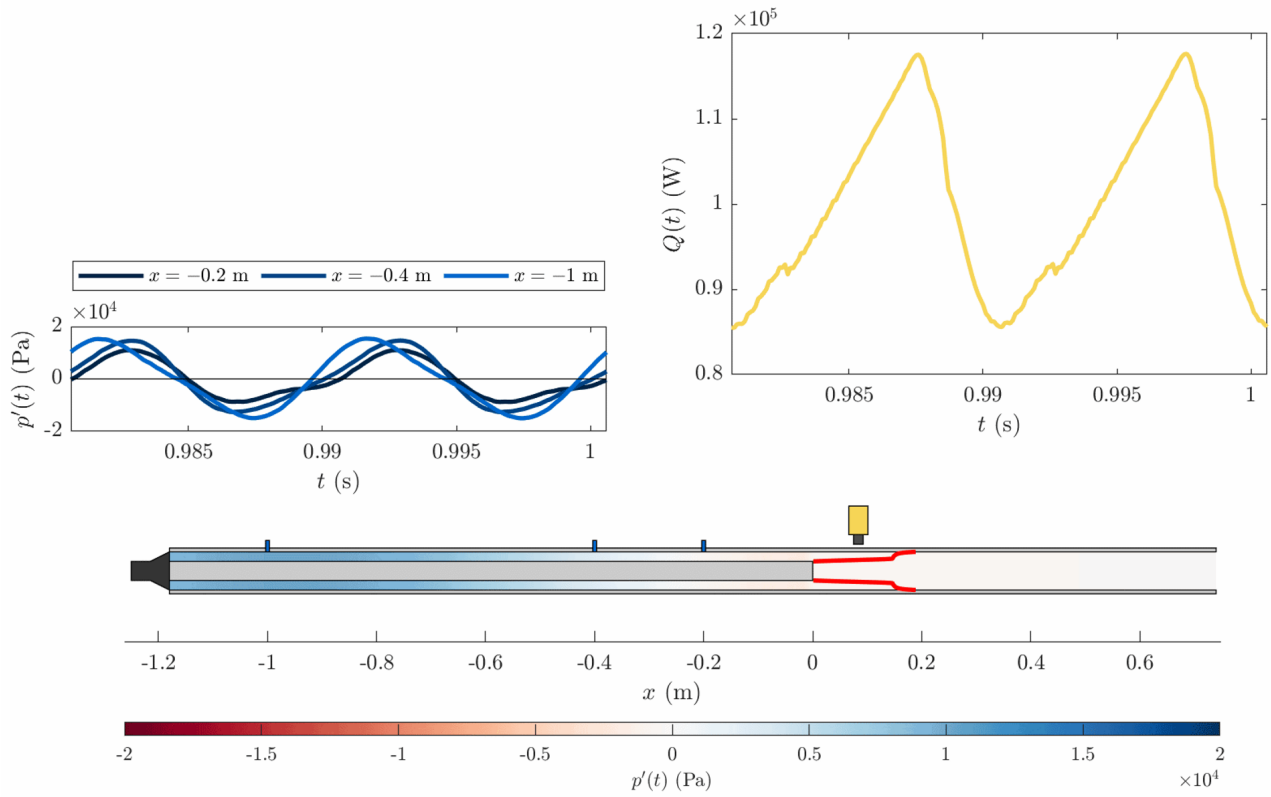


Fig. 5: Forcing the flame at high amplitudes.

From this animation we can see that again, the heat release rate and the pressure signals are periodic, but still not harmonic.

- Can you think of why?

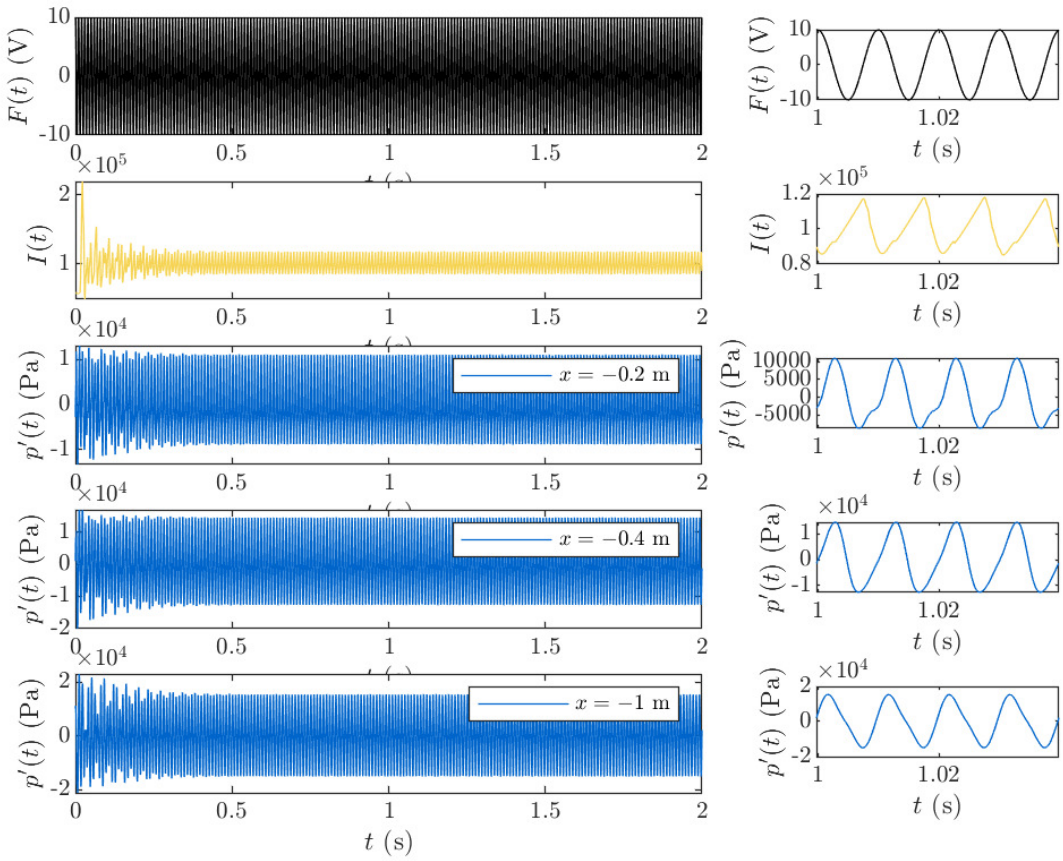
Optional:

If you want to visualize the flame and the acoustic field in the combustor, uncomment and run the following commands. You may want to change the starting and end points, but try to keep the time difference around $\Delta t = 0.02$ s, or else terminate the animation using the `ctrl+c` command.

```
% t_start = 1.02;
% t_end = t_start + 0.02;
% Num_of_time_steps_in_plot = 400;
% stepsize = 2;
% fn_view_simulation(Sim,stepsize,t_start,t_end,Num_of_time_steps_in_plot)
```

Let's plot the signals before proceeding to their discussion:

```
zoom_in_time_steps = 200;
fn_plot_Signals(Signals,Measurement,zoom_in_time_steps)
```



First observe that the forcing signal (black) is sinusoidal, i.e., harmonic. However, the response of the flame and the acoustic field given by the PMT (yellow) and the microphones (blue) is not harmonic. This again showcases the presence of nonlinearities.

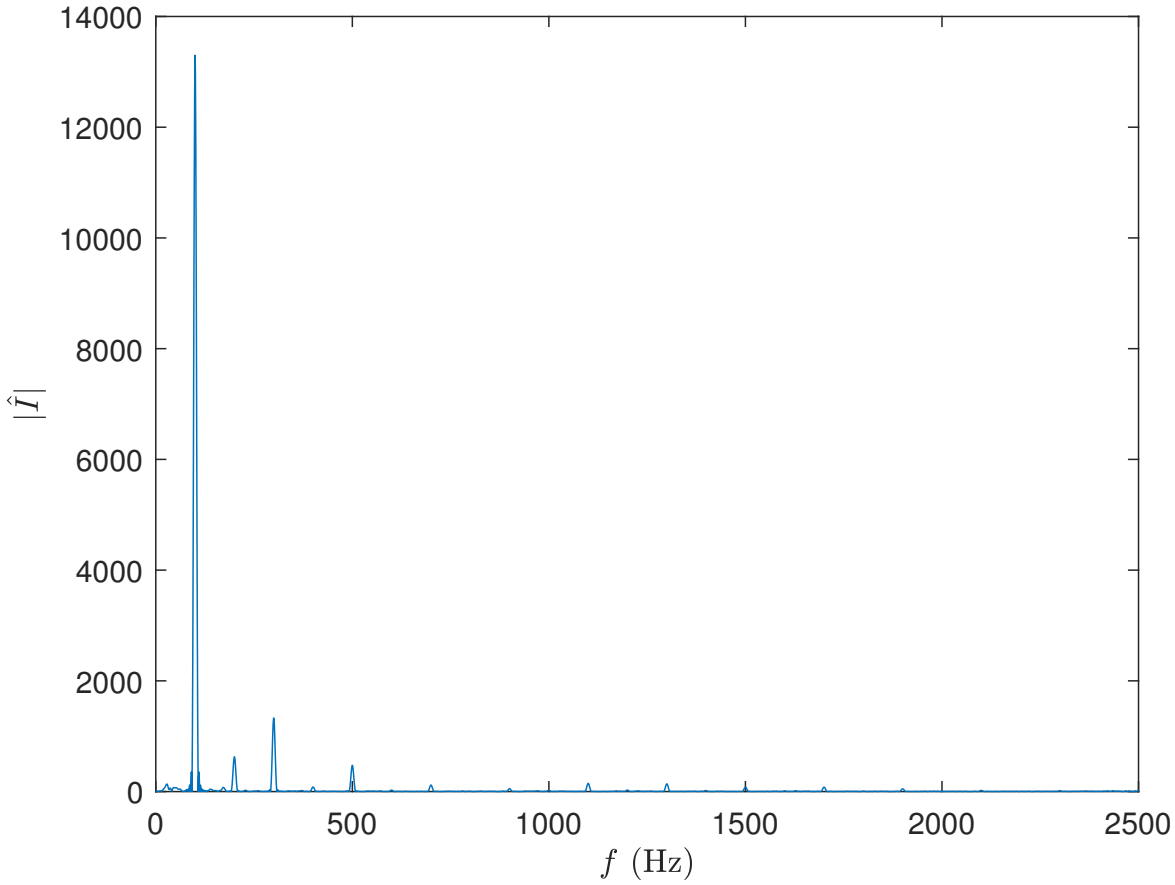
Recall that:

- A **linear** system forced at a frequency ω, responds at the same frequency ω.
- A **nonlinear** system forced at a frequency ω, may respond at the same frequency ω but also at other frequencies.

The latter is our case, since our forcing signal is harmonic but the response contains a much richer spectrum. Let's take a look at the frequency content:

```
% Read the signals:
t    = Signals.t;
sref = Signals.Forcing;
I    = Signals.PMT;
% Remove the transient at start:
sref = sref(t > 1);
I = I(t > 1);
t = t(t > 1);
% Remove the mean value of the PMT:
I = I - mean(I);
% Compute the spectral amplitudes
[f,Ih] = fn_spectral_est(t,I,sref,'welch');
```

```
figure
plot(f,abs(Ih))
xlim([f(1) f(end)])
xlabel('$f$ (Hz)','Interpreter','Latex')
ylabel('$|\hat{I}|$','Interpreter','Latex')
```



The spectrum analysis of the PMT signal reveals that the system is responding very strongly to the forcing frequency $f = 100$ Hz, but it also shows the presence of other strong harmonics in the signal. For instance, you can see that the next prominent peak appears 300 Hz and it has an amplitude about 10% of the dominant.

The question is

- What is causing the system to display nonlinearities?

The answer is simple, the forcing level. For this example, we use a voltage of $V = 10$ V, which is apparently very high. It is natural to ask:

- What is an appropriate forcing level?

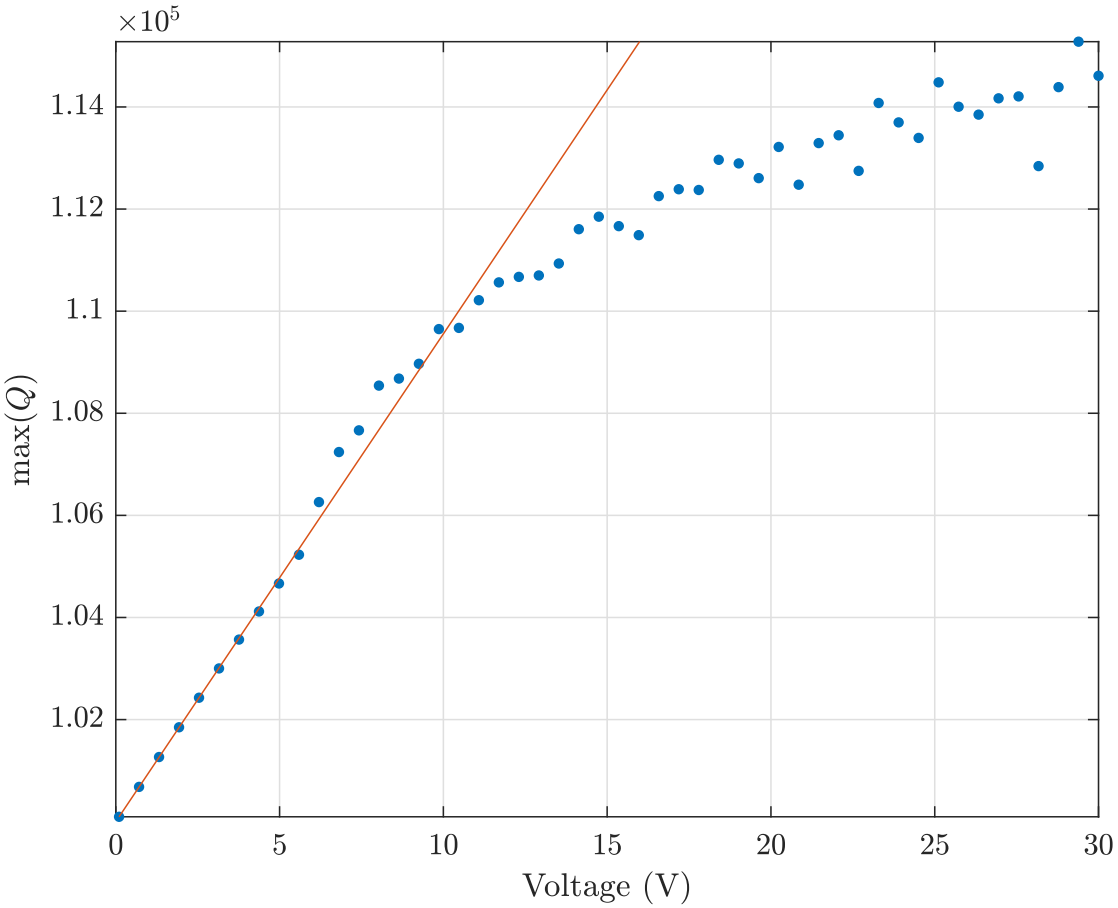
9. Saturation curve

The answer to the last question depends on each system. One way to figure this out is to plot a curve of voltage versus the peak amplitude of the heat release rate. With the following code you can generate such curve, but since we have to force the flame several times it may take a long time. So, you might want to skip it.

```
% N = 10;
% Vs = linspace(0.1,30,N);
% Qmax = zeros(1,N);
% for j = 1:N
%     % Change voltage:
%     Forcing.Voltage = Vs(j);
%     % Reduce spatial discretization for stability of the simulation:
%     spatial_disc_order = 1;
%     % Run simulation
%     [Mean,Geom,Signals,Sim] = Run_Simulation(u1,T1,p2,Qbar,Forcing,Measurement,...
%     spatial_disc_order);
%     t = Signals.t;
%     I = Signals.PMT;
%     % Keep only the signal after the transient, here I will keep only the
%     % signal after 1 second.
%     I = I(t>1);
%     Qmax(j) = max(I);
% end
% figure
% plot(Vs,Qmax,'o')
```

Here you can directly plot the saturation curve

```
fn_plot_saturation_curve
```



Up until $V = 5$ volts the system is behaving linearly, in the sense that an increase in forcing voltage is directly proportional to an increase in the rate of heat release. Around $V = 6$ volts we observe that the curve starts to saturate and behave nonlinearly.

- Why is that?

When we forced at $V = 10$ volts we were clearly in the saturation region. This explains the appearance of harmonics in the signals!

In real experiments it is common to use as a forcing amplitude the quantity $A = |\hat{u}|/\bar{u}$ and it is normally kept to about 5% to ensure that the system is still in the linear regime. Now that we have identified the linear regime, lets compute the Flame Transfer Function.

** The spread in the data above 10 Volts is due to numerical error due to the discretization schemes.

10. Flame Transfer Function

First let's set the forcing level to something more adequate and run the simulation.

```
Forcing.Voltage = 1;      % (V)
% Run the simulation
[Mean,Geom,Signals,Sim] = Run_Simulation(u1,T1,p2,Qbar,Forcing,Measurement);
```

```
Progress: 0%, 5%, 10%, 15%, 20%, 25%, 30%, 35%, 40%, 45%, 50%, 55%, 60%, 65%,
  70%, 75%, 80%, 85%, 90%, 95%,
 Finished
```

In the meantime take a look at the animation:

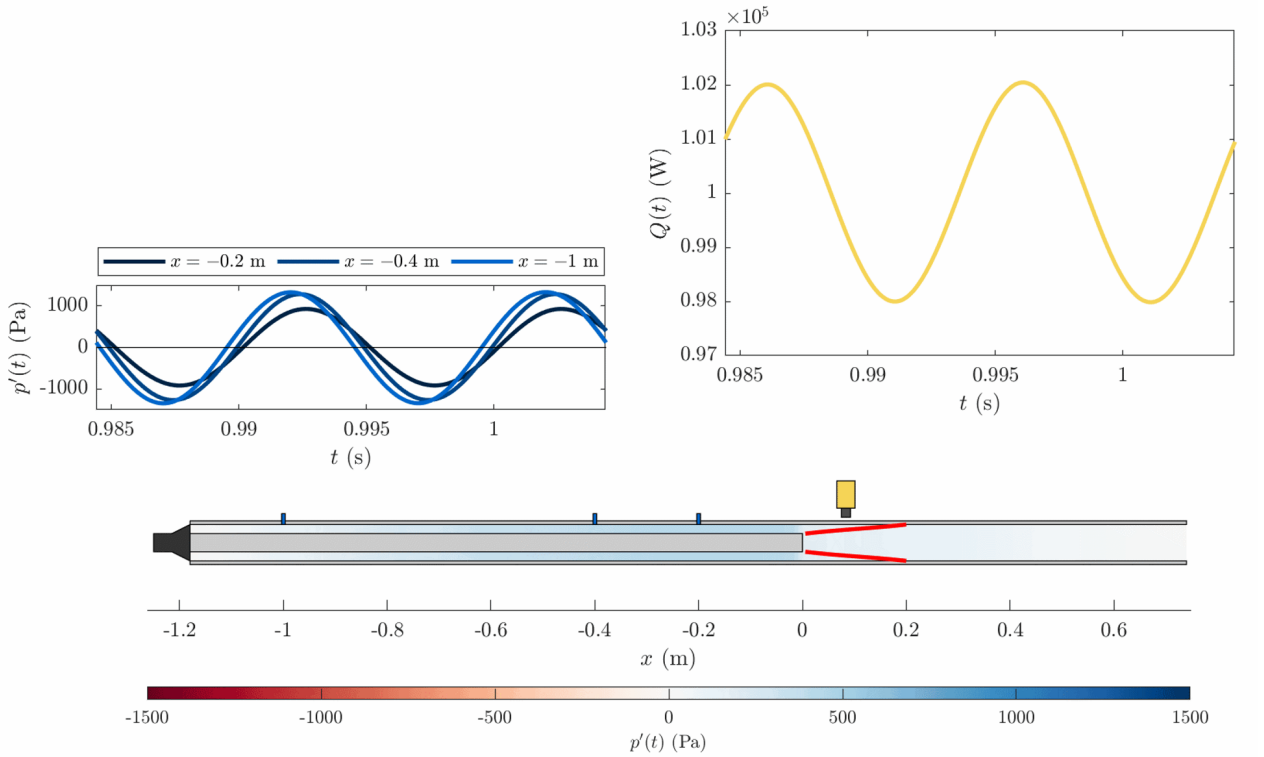


Fig. 6: Forcing the flame with low amplitudes.

This time the flame motion is almost imperceptible. Now both the heat release rate and the pressure signals look harmonic!

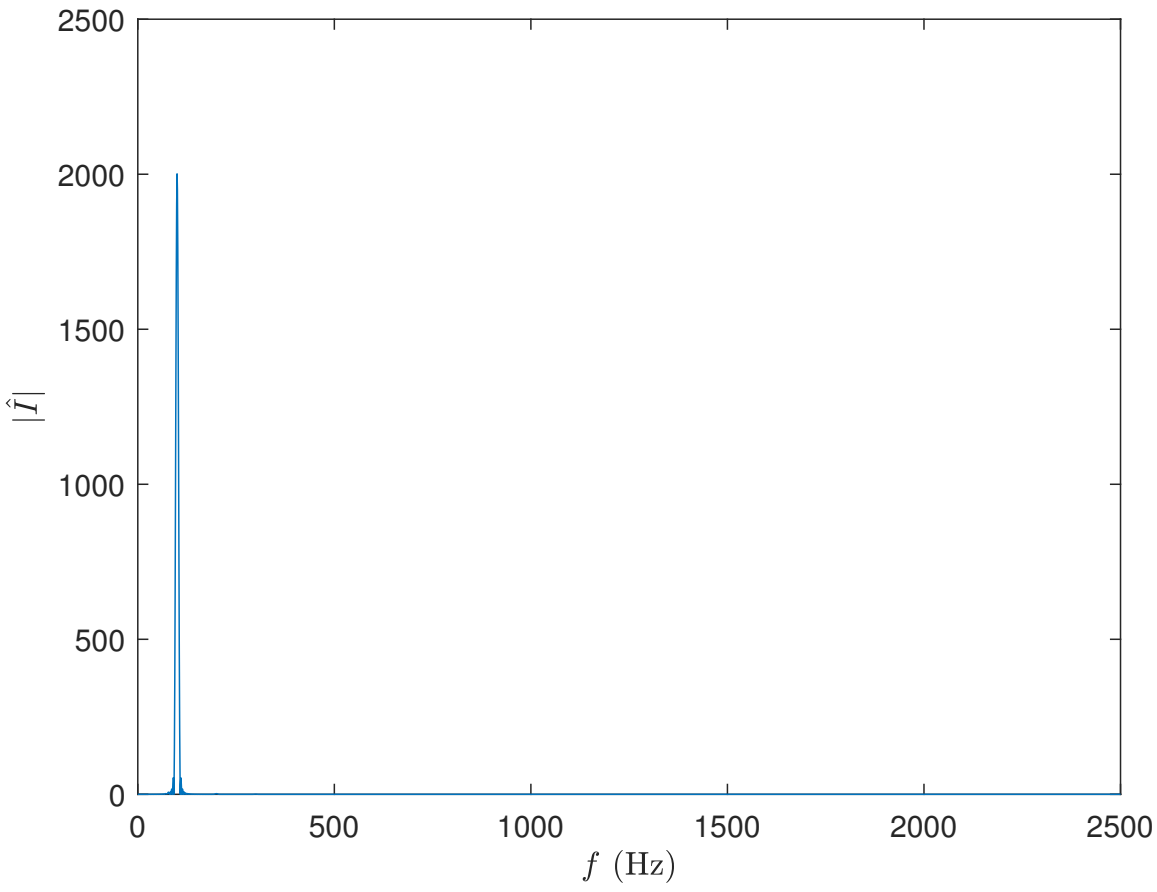
Optional:

If you want to visualize the flame and the acoustic field in the combustor, uncomment and run the following commands. You may want to change the starting and end points, but try to keep the time difference around $\Delta t = 0.02$ s, or else terminate the animation using the `ctrl+c` command.

```
% t_start = 1.02;
% t_end = t_start + 0.02;
% Num_of_time_steps_in_plot = 400;
% stepsize = 2;
% fn_view_simulation(Sim,stepsize,t_start,t_end,Num_of_time_steps_in_plot)
```

Let's confirm that our signals are harmonic by plotting their spectral estimates:

```
% Read the signals:
t    = Signals.t;
sref = Signals.Forcing;
Q    = Signals.PMT;
% Remove the transient at start:
sref = sref(t > 1);
Q = Q(t > 1);
t = t(t > 1);
% Remove the mean value of the PMT:
Q_mean = mean(Q);
Q = Q - Q_mean;
% Compute the spectral amplitudes
[f,Qh] = fn_spectral_est(t,Q,sref,'welch');
figure
plot(f,abs(Qh))
xlim([f(1) f(end)])
xlabel('$f$ (Hz)','Interpreter','Latex')
ylabel('$|\hat{I}|$','Interpreter','Latex')
```



```
% Print the peak value and the peak frequency
[Qh_max,ix] = max(Qh);
freq = f(ix);
fprintf(['f = ',num2str(freq),' Hz \n'])
```

```
f = 100.0023 Hz
```

```
fprintf(['|Qh| = ',num2str(abs(Qh_max)),' W \n'])
```

```
|Qh| = 2001.1922 W
```

We can clearly see that there is a single peak in the spectra corresponding to the forcing frequency $f = 100$ Hz. This confirms that signals are harmonic!

Next, according to Eq. (1) the FTF needs as an input the velocity fluctuations at a **reference point**. This point needs to be chosen carefully, since the interpretation of the results and its use in stability analyses depend on it. Since our flame anchors at $x = 0$, this is a convenient reference point.

In our configuration we have 3 microphone probes which we will use to reconstruct the acoustic field. We will be using the same functions we generated in the MMM tutorial **(make sure they can handle 3 microphones)**, except that we are going to keep the reference point just before the area expansion.

```
% Multi-Microphone-Method:
[F,G,omega] = fn_MMM(Mean,Signals,Measurement);
% Signal Reconstruction:
x_ref = 0;
[~,uh] = fn_acoustic_field(Mean,F,G,omega,x_ref);
```

This should give a forcing level in terms of $A = |\hat{u}|/\bar{u} = 0.177$:

```
u_mean = Mean.u1;
A = abs(uh)/u_mean
```

```
A = 0.1770
```

This number is a bit too high for a real experiment, ideally one would reduce the forcing voltage to keep this **forcing level around 5%**. For our toy model this would imply a forcing voltage of about 0.3 V. In an experiment it is common to adjust the forcing level depending on the frequency. Remember that as we change the frequency the velocity modeshape in the injection pipe changes. This means that at some frequencies the dump plane might be near a **velocity node** (low $|\hat{u}|$) or near a **velocity antinode** (high $|\hat{u}|$). This is independent of the forcing level. So if we are near a velocity antinode and we do not adjust the forcing level, the forcing amplitude might be higher than what we measured at a different frequency.

According to the previous measurement at a frequency of 100 Hz, the fluctuations ($|\hat{u}|$) will become of the order of the mean when $A = 1$ which occurs when $V \to 5.88$ volts. Anything higher than this and the **flow** will **reverse** in parts of the cycle. This is the effect we observed in Fig. 5 in which the flame was flashbacking in part of the cycle.

Having reconstructed the acoustic field, we can easily compute the FTF:

```
FTF = (Qh_max/Q_mean)/(uh/u_mean)
```

```
FTF = 0.0602 - 0.0957i
```

```
Gain = abs(FTF)
```

```
Gain = 0.1131
```

```
phase = angle(FTF)
```

```
phase = -1.0089
```

We have now computed the FTF for a single frequency $f = 100$ Hz. Now it is your turn to compute it to many other frequencies.

Task

- Complete the function `fn_FTF` and loop through several frequencies.

Once you have finished, run the following code. In the spirit mentioned before, let's start by reducing the forcing level:

```
Forcing.Voltage = 0.3;
```

Now set the frequency range and the number of points. I recommend starting with 1 or 2 points until you get it running right. If you choose too many points, consider using a `parfor` loop in the function `fn_FTF`.

```
Opts.fmin = 10;
Opts.fmax = 400;
Opts.N = 4;
```

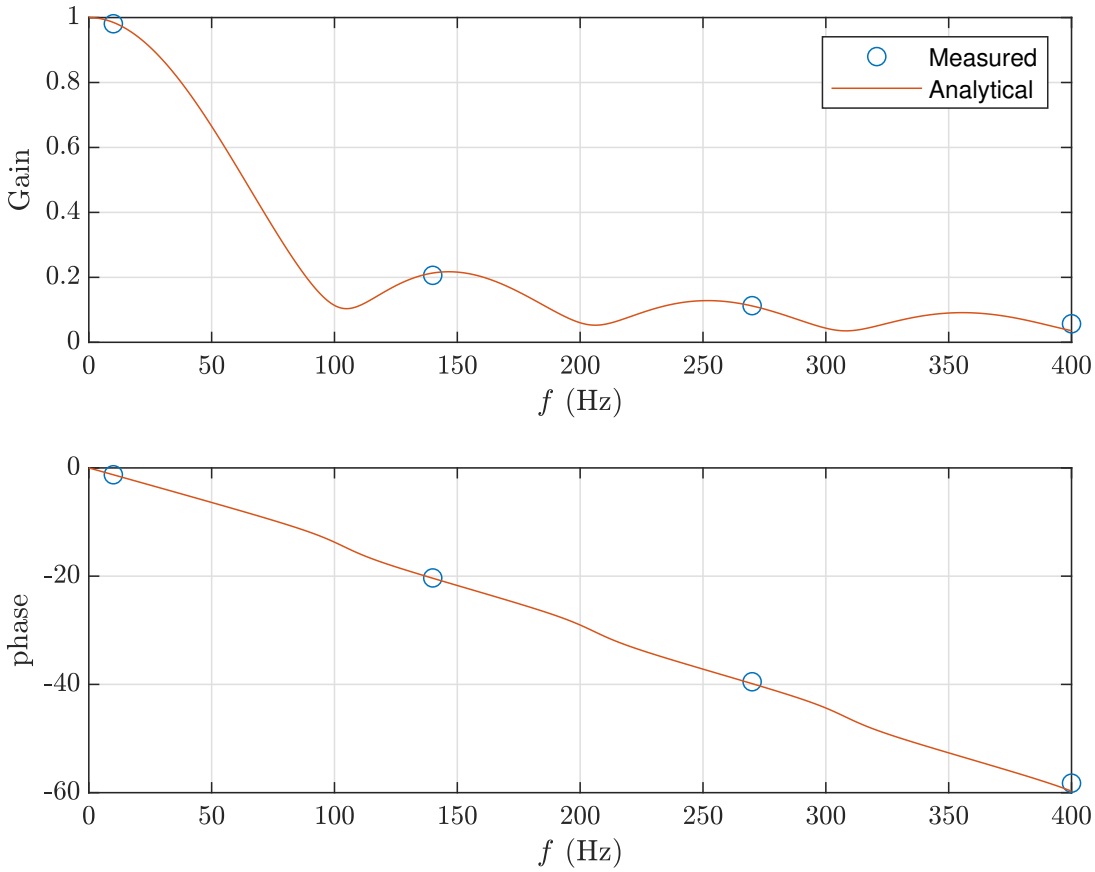
Finally compute the flame transfer function:

```
[FTF,f] = fn_FTF(u1,T1,p2,Qbar,Forcing,Measurement,Opts);
```

```
Progress: 0%, 5%,  10%,  15%,  20%,  25%,  30%,  35%,  40%,  45%,  50%,  55%,  60%,  65%,
  70%,  75%,  80%,  85%,  90%,  95%,
 Finished
f = 10.004 Hz
Forcing Amplitude: 0.04193
Progress: 0%, 5%,  10%,  15%,  20%,  25%,  30%,  35%,  40%,  45%,  50%,  55%,  60%,  65%,
  70%,  75%,  80%,  85%,  90%,  95%,
 Finished
f = 139.9994 Hz
Forcing Amplitude: 0.040517
Progress: 0%, 5%,  10%,  15%,  20%,  25%,  30%,  35%,  40%,  45%,  50%,  55%,  60%,  65%,
  70%,  75%,  80%,  85%,  90%,  95%,
 Finished
f = 270.0043 Hz
Forcing Amplitude: 0.032
Progress: 0%, 5%,  10%,  15%,  20%,  25%,  30%,  35%,  40%,  45%,  50%,  55%,  60%,  65%,
  70%,  75%,  80%,  85%,  90%,  95%,
 Finished
f = 399.9996 Hz
Forcing Amplitude: 0.085668
Progress: 0%, 5%,  10%,  15%,  20%,  25%,  30%,  35%,  40%,  45%,  50%,  55%,  60%,  65%,
  70%,  75%,  80%,  85%,  90%,  95%,
 Finished
f = 399.9996 Hz
Forcing Amplitude: 0.050001
```

Once the computation is finished, plot it and compare it to the analytical model:

```
% Plot the FTF and compare:
fn_plot_FTF(f,FTF,Mean,Geom)
```



If done accurately both curves should lie on top of each other. There is some discrepancy at frequencies above 300 Hz which arise due to the spatial discretization used for the simulation of the flame.

Question to ask:

- What would happen if we increase the forcing level but remain in the linear regime, will the FTF change?

11. Experimental FTF

Now that we have measured the FTF of a kinematic model, let's try it in experimental data, which you can download from this link.

This data can be seen in Fig. 11 in the following paper:

Æsøy, Eirik, et al. "Scaling and prediction of transfer functions in lean premixed H2/CH4-flames." Combustion and Flame 215 (2020): 269-282

We have 2 cases of pure hydrogen flames at:

- $P = 8$ kW and,
- $P = 12$ kW.

Let's load the data for the 8 kW case at a frequency of 200 Hz:

```
WS = load('P8kW\F200.mat')
```

```
WS =
        P1: [1x508800 double]
        P2: [1x508800 double]
        P3: [1x508800 double]
    Params: [1x1 struct]
      Pref: [1x508800 double]
         Q: [1x508800 double]
```

```
WS.Params
```

```
ans =
               fs: 51200
                U: 34.2312
                c: 371.8949
              rho: 1.0248
    Microphonepos: [-0.0445 -0.0695 -0.1295]
            Power: 8
```

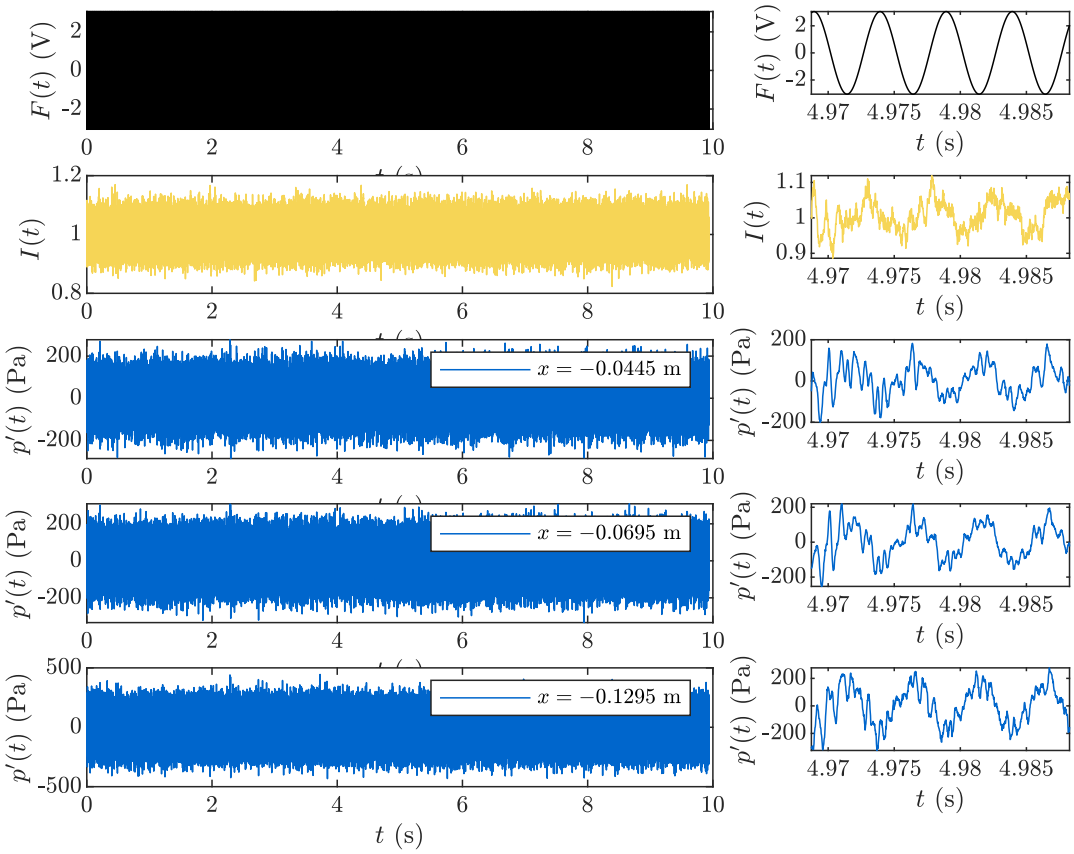
Each file contains the readings of 3 microphones, the PMT and the reference signal. The structure `Params` contains further information such as the forcing frequency, the mean velocity, the speed of sound, and the density of the flow. It also has the positions of the microphones.

Now let's put it in the format we have been using so far:

```
clear Mean Geom Signals Measurement
% Generate time vector:
fs = WS.Params.fs;
t = 0:(1/fs):((length(WS.P1)-1)/fs);
% Update Signals structure
Signals.t = t;
Signals.Forcing = WS.Pref;
Signals.PMT = WS.Q;
Signals.P_mic = [WS.P1;WS.P2;WS.P3];
% Update Measurement structure
Measurement.Fs = WS.Params.fs;
Measurement.Mic_Pos = WS.Params.Microphonepos;
```

Let's first take a look at the signals:

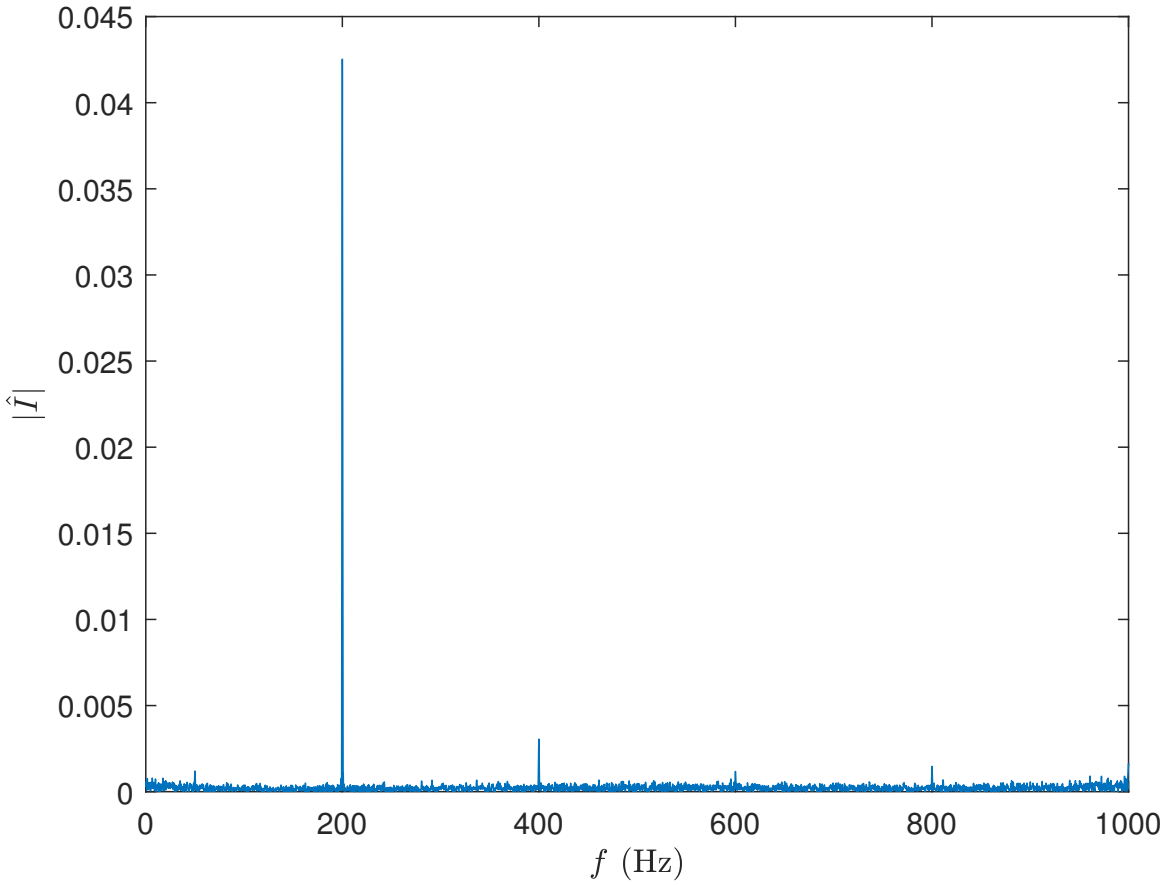
```
Zoom_in_time_steps = 1000;
fn_plot_Signals(Signals,Measurement,Zoom_in_time_steps);
```



Unlike our toy model, these signals contain noise, but fortunately the tools we need to analyse them are the same.

Let's begin by looking at the frequency content of the PMT signal. The signal from the PMT has been normalized such that the mean equals unity.

```
% Read the signals:
t    = Signals.t;
sref = Signals.Forcing;
Q    = Signals.PMT;
% Remove the mean value of the PMT:
Q_mean = mean(Q);
Q = Q - Q_mean;
% Compute the spectral amplitudes
[f,Qh] = fn_spectral_est(t,Q,sref,'welch');
figure
plot(f,abs(Qh))
xlim([f(1) f(end)])
xlabel('$f$ (Hz)','Interpreter','Latex')
ylabel('$|\hat{I}|$','Interpreter','Latex')
xlim([0 1000])
```




```
% Print the peak value and the peak frequency
[Qh_max,ix] = max(Qh);
freq = f(ix);
fprintf(['f = ',num2str(freq),' Hz \n'])
```

```
f = 200 Hz
```

```
fprintf(['|Qh| = ',num2str(abs(Qh_max)),' \n'])
```

```
|Qh| = 0.042526
```

The peak appears at $f = 200$ Hz, there is another small peak at $f = 400$Hz but its amplitude is small, around 7% of the main. It may be tempting to think that the forcing level is too high, and nonlinearities may be present. However, in real geometries it is "easier" to excite the harmonics, than in idealized models. For this reason, a better estimate is to consider the forcing amplitude, but for that, we need to perform the MMM on the injector geometry.

a) MMM in the injector

The first task will be to compute the velocity fluctuations at the dump plane using the pressure signals. The following figure illustrates the injector geometry. It is composed of a bluff body inside a pipe. The reference point for all these measurements will be again $x = 0$.

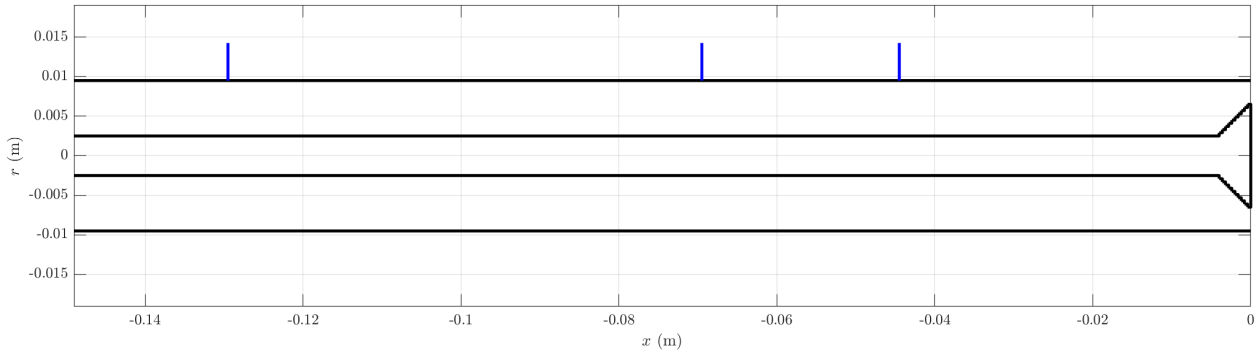


Fig. 7: Injector geometry. The blue lines indicate the microphone positions.

First let's update the geometric and the mean structures accordingly:

```
dp = 0.019; % pipe diameter (mm)
dr = 0.005; % rod diameter (mm)
db = 0.013; % bluff body diameter (mm)
A1 = pi/4 * (dp^2 - dr^2);
A2 = pi/4 * (dp^2 - db^2);
% Update Geom structure
Geom.A1 = A1;
Geom.A2 = A2;
% Update the Mean structure
Mean.c1   = WS.Params.c;
Mean.rho1 = WS.Params.rho;
% Here I will assume that the density and the speed of sound do not change
% with the area contraction:
Mean.c2   = WS.Params.c;
Mean.rho2 = WS.Params.rho;
% But the mean velocity does change:
u2 = WS.Params.U;
u1 = u2*A2/A1;
Mean.u2 = u2;
Mean.u1 = u1;
```

Using the functions developed in the MMM tutorial we will propagate the signal to $x = 0$. For the bluff body we will consider a single area expansion located at the end.

```
% 1) Multi-Microphone-Method:
[F,G,omega] = fn_MMM(Mean,Signals,Measurement);
% 2) Acoustic field Reconstruction:
x0 = 0;
[ph1,uh1] = fn_acoustic_field(Mean,F,G,omega,x0);
% 3) Compute F and G after the area expansion:
[F2,G2] = fn_Area_Change(Mean,Geom,ph1,uh1,omega,x0);
% 4) Compute uh after the area expansion:
[~,uh] = fn_acoustic_field(Mean,F2,G2,omega,x0);
```

Now we can compute the forcing amplitude:

```
u_mean = u2;
A = abs(uh)/u_mean
```

```
A = 0.0468
```

We obtained a value of $A = 0.046$ which indicates that the forcing level is indeed small, around 5%.

b) FTF

Having computed the velocity fluctuations at the reference point we can easily compute the FTF:

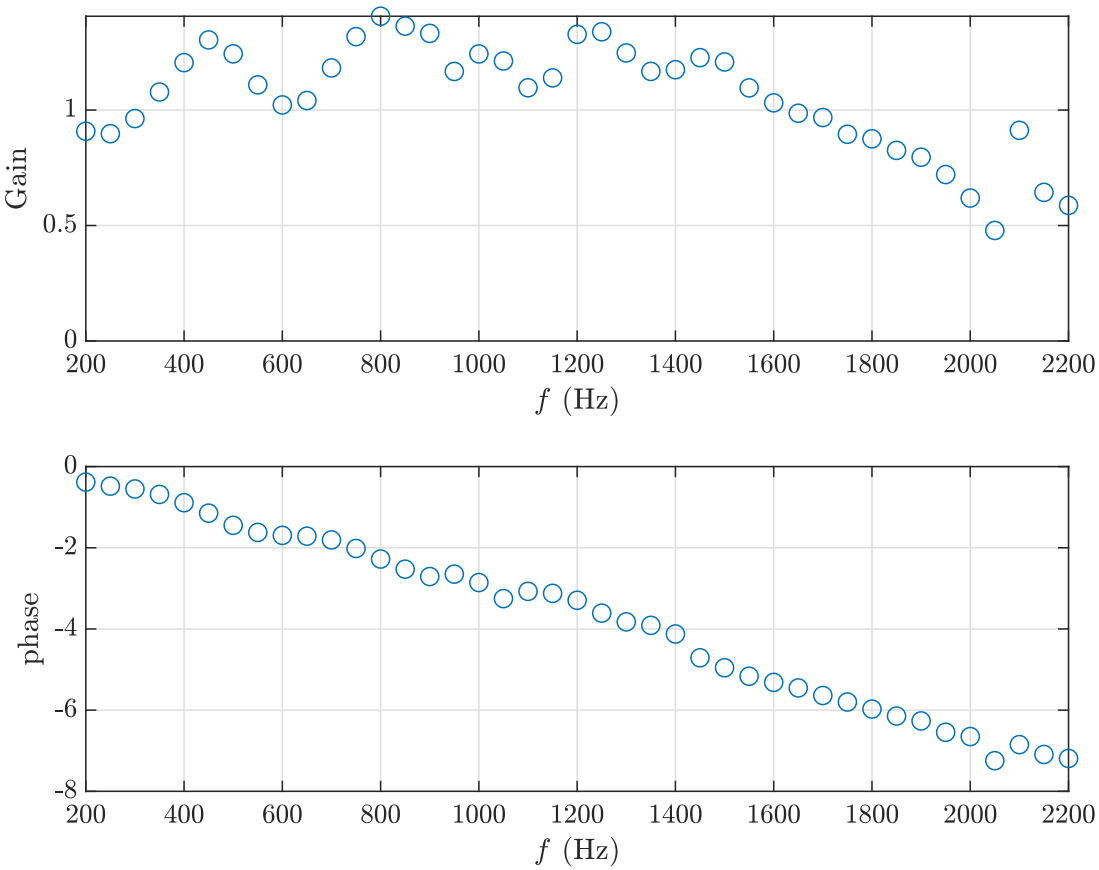
```
FTF = (Qh_max/Q_mean)/(uh/u_mean)
```

```
FTF = 0.8427 - 0.3389i
```

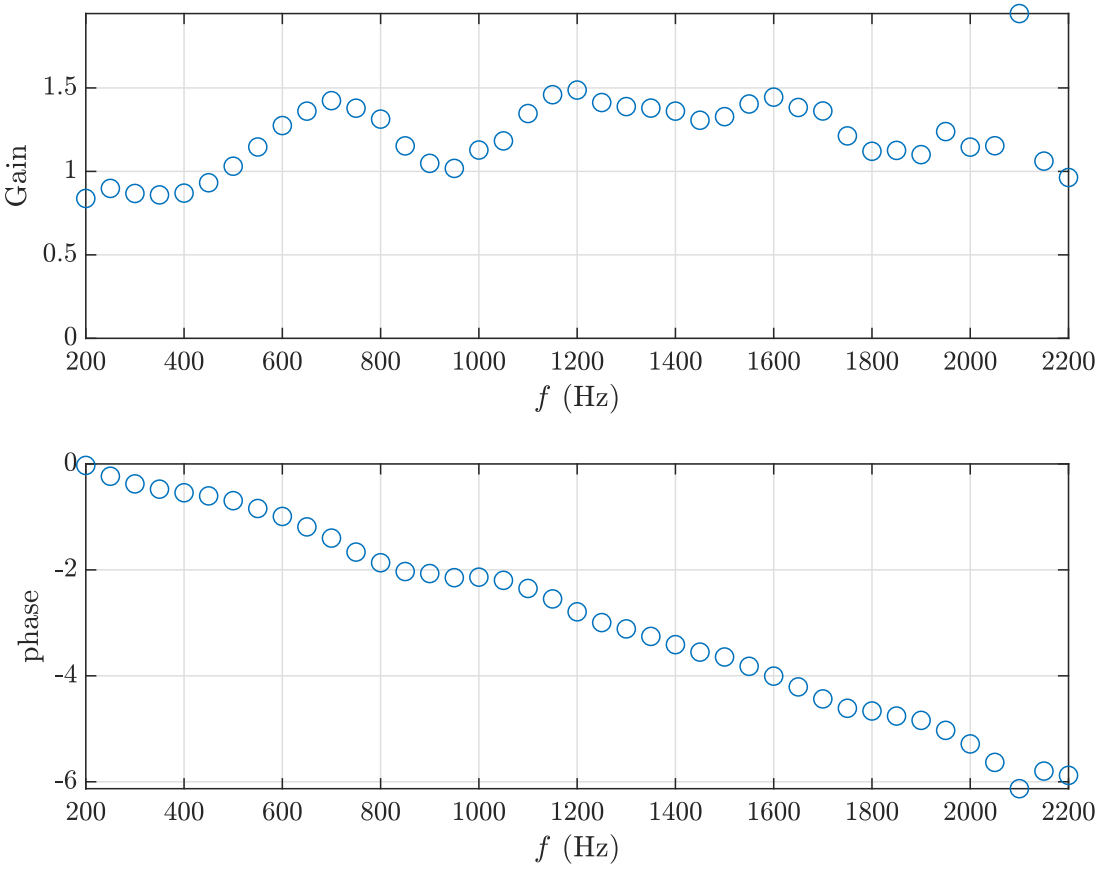
Now, let's repeat the process with all the dataset:

```
[f_8kw,FTF_8kw] = fn_FTF_exp('P8kW');
```

```
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 4).
```




```
[f_12kw,FTF_12kw] = fn_FTF_exp('P12kW');
```



This Flame Transfer functions behave a bit different than the ones in our toy model. In the next tutorial you will look into some of their features.

12. Further Considerations

1. The FTF by definition is a linear function which means that it is only valid when the amplitude of the heat release is not saturated. Its extension to consider the effects of amplitudes is the Flame Describing Function:

$$\mathrm{FDF}\left(A,\omega\right)=\frac{\hat{Q}/\bar{Q}}{\hat{u}/\,\bar{u}}$$

In this case one can observe that as the forcing amplitude increases saturation effects will cause the gain to start reducing.

2. The FTF is defined in the frequency domain, in the time domain the corresponding function is an impulse response $h(t)$ that relates Q' to u' through a convolution integral:

$$\frac{Q'\left(t\right)}{\bar{Q}}=\int_{-\infty}^{\infty}h\left(T\right)\frac{u'\left(t-T\right)}{\bar{u}}\ \mathrm{d}\,T$$

The impulse response of the $n-\tau$ model is given by:

$$h(t) = n\delta(t - \tau)$$

In the next tutorial you will learn more about this.