Electronic Duffing Oscillator Exercise

PhD Course in Advanced Structural Dynamics 16-19 June 2025

Experimental Background

You will test an electronic circuit replicating a Duffing oscillator governed by the nonlinear ordinary differential equation

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) + k_3x^3(t) = f(t), \tag{1}$$

where m, c, k and k_3 are the mass, damping, linear stiffness and nonlinear stiffness of the oscillator, respectively, x is its displacement and f the external forcing applied to it. The circuit realizes a linear oscillator using analog integrators, and analog multipliers are added to implement the cubic nonlinearity[1].

One BNC connector of the circuit is used to impose the forcing signal f. The two other connectors allow for the measurement of signals proportional to the displacement x and velocity $\dot{x}$. Due to hardware limitations, only the former will be recorded.

The oscillator is controlled in real-time thanks to a microcontroller (Arduino Due). An interface circuit is used to match the voltage ranges between the two electronic circuits. They are powered with a USB power supply. The microcontroller is not suited to record the signals directly, which is why a digital oscilloscope (PicoScope 2204A) is used. A schematics of the setup is shown in **Figure 1**.

You are to use various testing approaches to characterize the nonlinear dynamics of the oscillator. The first type of approach uses open-loop swept-sine excitation, while the second one uses a phase-locked loop (PLL). The former is simpler but provides limited insight.

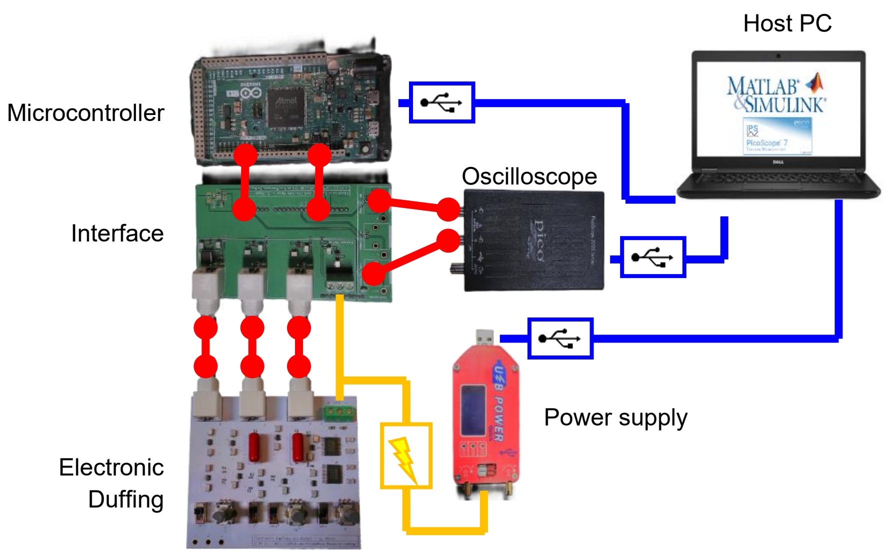


Figure 1 Experimental setup schematics.

[1] See https://github.com/GhislainRaze/Electronic-Duffing for more details.

Group Tasks

1) Measure the response of the oscillator to swept-up and swept-down sine excitations at various forcing amplitudes (Step D).
2) Use a PLL testing approach to measure a few nonlinear frequency responses of the oscillator, ideally at the same forcing amplitudes as the swept-sine tests (Step E).
3) Use a PLL testing approach to extract the phase resonance backbone curve of the oscillator (Step F).
4) From your measurements, provide evidence that the system is behaving nonlinearly.
5) Estimate the frequency response function of the system from the swept-sine tests. Do you think it is an appropriate way to represent the steady-state responses of the oscillator?
6) Superimpose the swept-sine results with nonlinear frequency responses measured with the PLL. Comment on their differences.
7) Superimpose the nonlinear frequency responses and backbone curve in a plot. What are the advantages of measuring the backbone directly?

Carry out the experiment

You will now go through the following (you can start from Step C, Steps A and B are just for info):

A. Check the hardware
B. Check the software
C. Use a sinusoidal excitation to see that the setup works
D. Perform swept-sine tests
E. Perform PLL tests to measure the nonlinear frequency response
F. Perform PLL tests to measure the backbone curve
G. Export the measurements

Make sure to perform step G after each measurement!

A. Check hardware setup

1) Check the connections depicted in **Figure 1** (USB connection between the oscilloscope/power supply and host computer, microcontroller and interface, BNC connection between the interface and electronic Duffing oscillator, probes between the interface and oscilloscope, and power supply to electronic Duffing oscillator and interface). Make sure that the power supply is not turned on yet (no LED should emit light).
2) Launch MATLAB. Plug the Arduino Due board to the host computer via its programming port (**Figure *2***). You should see the following message appear in the MATLAB terminal, confirming that the Arduino Board is correctly connected to the host PC:

```
Arduino Due Programming Port detected.
This device is ready for use with MATLAB Support Package for Arduino
Hardware. Get started with examples and other documentation.
This device is ready for use with Simulink Support Package for Arduino
Hardware. Get started with examples and other documentation.
```

3) Set up the power supply. If the screen displays IN, press the **SW** button once and it should show OUT. Press the **ON** button; the ON led should light up. Check that the displayed output voltage is between 20 and 30 V (if not, adjust it with the CV knob). If the CC led lights up, there is a short-circuit somewhere (or the circuit draws more current than the limit, which can be solved by turning the **CC** knob in the clockwise direction).

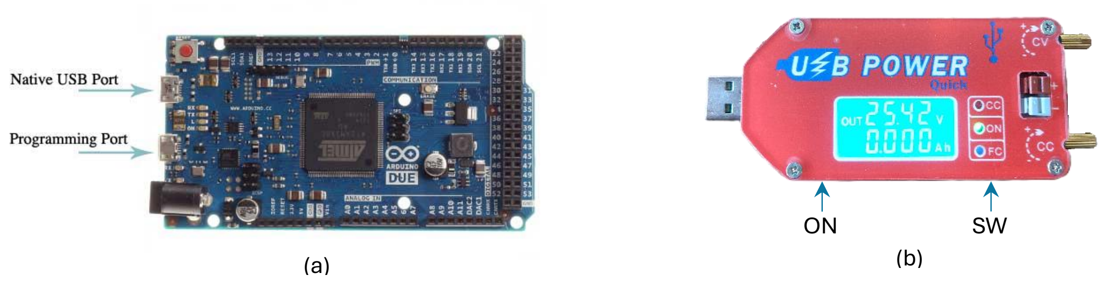


Figure 2 (a) USB ports of the Arduino Due board (source: https://store.arduino.cc/products/arduino-due), (b) correct display of the power supply.

B. Configure the software

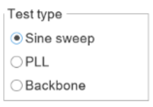
1) Launch PicoScope 7 T&M. In the Connect Device window, select **PicoScope 2204A** (if this choice does not appear, check the connection between the oscilloscope and the host PC).
2) Click A. In the **Vertical** tab, select **Manual**, set the voltage range to **±5 V** and **Coupling mode** to **AC**. In the **Probes** tab, click **10:1 probe**. Repeat the same operations for the B probe, but set the voltage range to **±1 V**.
3) Click Scope. In the **Timebase** tab, select **1 s/div**. In the Sampling tab, select **500 kS**.
4) In MATLAB, run the `DuffingArduinoControlParameters` script.
5) Open the `DuffingArduinoControl.slx` file in Simulink. This file contains a graphical representation of the program to be uploaded to the Arduino Due board. If everything went fine in the previous step, no block should have a red outline (which would mean that some parameters are not defined).

C. Sinusoidal excitation

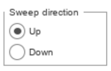
1) In the Simulink file, go to the **HARDWARE** tab and click Monitor & Tune **Monitor & Tune**. Wait for the compiler to build the file and upload it to the Arduino board.
2) Set the forcing amplitude with the text field **f0**, e.g., 0.1, and then click Reset **Reset**.
3) Go to the PicoScope 7 T&M software. Check that signal B (the external forcing of the Duffing oscillator) is a sinusoid, and that signal A (the displacement of the oscillator) is a periodic response with the same period. Note that input B does not have the same amplitude as that specified in **f0** (due to some gains in the circuit), but scales linearly with it.

D. Swept-sine excitation

1) Before every measurement, make sure that the PicoScope software is in the **Running** state.


Test type
● Sine sweep
○ PLL
○ Backbone

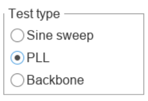
2) Make sure that the radio buttons for **Test type** are set to **Sine sweep** .
3) Set the forcing amplitude with the text field **f0**, e.g., 0.1, and then click **Reset**.


Sweep direction
● Up
○ Down

4) Use the radio buttons **Sweep direction** to choose whether a sweep up or down will be performed.
5) Start the sweep with **Start**.
6) Once the test is complete, save your results (Step G).
7) Change the forcing amplitude or the sweep direction to enrich your test data. It is suggested to test amplitudes from 0.1V to 0.5V with a step of 0.1V, both with swept-up and swept-down sine excitations.

E. PLL – nonlinear frequency response

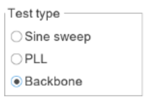
1) Before every measurement, make sure that the PicoScope software is in the **Running** state.


Test type
○ Sine sweep
● PLL
○ Backbone

2) Use the radio buttons to set **Test type** to **PLL** .
3) Set the forcing amplitude with the text field **f0**, e.g., 0.1, and then click **Reset**.
4) Start the sweep with **Start**.
5) Once the test is complete, save your results (Step G).
6) Change the forcing amplitude to enrich your test data. It is suggested to test amplitudes from 0.1V to 0.5V with a step of 0.1V. If the PLL has trouble locking in, select Sine sweep in Test type, click Reset, and then re-select PLL in Test type.

F. PLL – backbone curve

1) Before every measurement, make sure that the PicoScope software is in the **Running** state.


Test type
○ Sine sweep
○ PLL
● Backbone

2) Use the radio buttons to set **Test type** to **Backbone** .
3) Set the initial forcing amplitude to a low value (e.g. 0.05V) with the text field **f0**, and then click **Reset**.
4) Start the backbone measurement with **Start**.
5) Once the backbone test completes, save your results (Step G).

G. Export your measurements

1) Once your test is complete, click **Running**, which should turn to **Stopped**. To save the recorded results, click **Save** . Select an appropriate folder and file name. In **File type**, select **MATLAB file**.

2) The recorded data is saved in the specified folder in terms of frames (“Waveforms”), and there can be up to 64 of them. Each file has a name `[filename]_x.mat`, where `[filename]` is the name selected earlier, and `x` is the frame number. To assemble the measurements into one continuous time series, the `assemblePicoscopeSignals` function can be used in MATLAB by typing (with the appropriate folder name `folder`)

   ```
   >> [t,va,vb] = assemblePicoscopeSignals(folder);
   ```

   `va` and `vb` represent the signals recorded by scopes A (displacement) and B (force), respectively, while `t` represents time. An optional version of this function performs frequency-domain filtering by cutting off the frequency content beyond the angular frequency `w0`. This can be used with

   ```
   >> [t,va,vb,vaf,vbf] = assemblePicoscopeSignals(folder,w0);
   ```

   where `vaf` and `vbf` are filtered versions of `va` and `vb`.
3) Check that everything went fine by loading the generated file, and typing the following commands into the MATLAB terminal

   ```
   >> figure
   >> plot(t,va)
   ```

Postprocessing tips

The frequency response function estimation in Task 5 can be performed easily with off-the-shelf functions. In MATLAB, it can be computed with `tfestimate`. In Python, the `scipy.signal.csd` function can be used to estimate cross-power spectral densities, from which the transfer function can be computed. Namely, for single-input-single-output systems, the H_1 and H_2 transfer function estimators are given by

$$H_1(\omega) = \frac{S_{yx}(\omega)}{S_{xx}(\omega)}, \qquad H_2(\omega) = \frac{S_{yy}(\omega)}{S_{xy}(\omega)},$$

respectively, where $S_{uv}(\omega)$ is the cross-power spectral density between signals u and v, and ω is the angular frequency.

In Task 6, the nonlinear frequency response is usually represented in a frequency vs amplitude plot (much like frequency response functions). In Matlab, the frequency of the signal can be estimated with the `instfreq` command using the force signal. A simple way to estimate the corresponding amplitude of the response is to look at the maximum of the absolute value of the displacement signal over one period centered around the time at which the instantaneous frequency is estimated.