TopEdu

PROJECT 7 INSTRUCTIONS

Why a plucked note changes as it fades

A guitar note changes as it fades. Capture that on your phone and explain what the simple model misses.

START HERE

Set up in ten minutes

This project comes with one short Python file, `07_damped_string.py`. It solves the project's equation with simple methods you can read line by line, checks itself against an answer that is known exactly, and ends with a list of first experiments. The research is what you do after that.

1. **Install Python 3** from python.org (on a Mac it may already be there: type `python3 --version` in Terminal).
2. **Download the starter** from the project's page, or all ten as a zip from topedu.club/topedu/research/starter-code/.
3. **Install the two libraries** in Terminal (Mac) or PowerShell (Windows):

   ```
   python3 -m pip install numpy matplotlib
   ```

   On Windows, type `py` where this guide says `python3`.
4. **Run it:**

   ```
   python3 07_damped_string.py --show
   ```

   The CHECK line must say PASS. `--show` opens the plots; without it they are saved in the `output` folder.

No computer of your own? Google Colab works in a browser for free: open a new notebook, upload the file with the folder icon on the left, and run `!python 07_damped_string.py` in a cell.

The rules that make it research

- **Predict before you run.** Write your guess, and why, in your log first. A wrong prediction you can explain is worth more than a right one you cannot.
- **Keep a log.** Date, what you changed, what you expected, what happened. Your paper is written from it.
- **Check the grid.** Run your key result again with twice the points. If the answer moves, report by how much.
- **Never edit the CHECK to make it pass.** A FAIL after your change is information: undo, confirm PASS, then change one thing at a time.
- **Own every line.** Your mentor explains; you write and run. Name everyone and everything that helped, including this starter.

The full guide, with all ten projects and five numerical ideas the starters use, is at topedu.club/topedu/research/starter-code/.

PROJECT 7 · PHYSICS AND MUSIC

Why a plucked note changes as it fades

A guitar note changes as it fades. Capture that on your phone and explain what the simple model misses.

FILE	MATH	YOU NEED
`07_damped_string.py`	Algebra II, trigonometry, logarithms	A guitar or a home-made string, and a phone

The question

This damping model makes every harmonic fade at one rate. In a real guitar the upper harmonics die sooner. By how much, and how does it depend on the string's material and gauge?

Objectives

Weeks 1 to 3	**Learn to trust the model.** Run 07_damped_string.py until its CHECK line says PASS, and explain every line of it to your mentor. Then reproduce a known answer: Each mode has the exact solution e^(-at/2) times an oscillation at $\sqrt{(\omega^2 - a^2/4)}$. Checkpoint: you reproduce a known answer and report how close you got.
Weeks 4 and 5	**Ask your own question.** Narrow the question above to one sentence that could turn out to be wrong, and write your prediction before you run anything. The first experiments below are a place to start. Checkpoint: a one-page plan with your question and prediction.
Weeks 6 to 9	**Run it, then measure the world.** Fit the damping from a recording, simulate the same note, and measure where the single-rate model departs from the data for each harmonic. Then step away from the screen: record plucked strings on a phone and measure each harmonic's decay on a spectrogram. Checkpoint: one chart comparing your model with reality.
Weeks 10 to 12	**Explain the gap, and share it.** Work out why your model and your measurements disagree, test at least one explanation, and write it up yourself. Say which numbers came from the starter and which from the lab. Keep the honest limit in view: one constant damping term cannot describe a real string. Showing exactly how it fails, and proposing a fix, is the result. Checkpoint: a final paper, your code and a ten-minute talk.

The model in the starter

A string of length L vibrates as u_tt + a u_t = c^2 u_xx, fixed at both ends. The term a u_t is a simple drag. In this model every harmonic fades as e^(-a t / 2), all at the same rate. A real guitar loses its high harmonics sooner, so the sound gets mellower as it fades. Measuring by how much is the research.

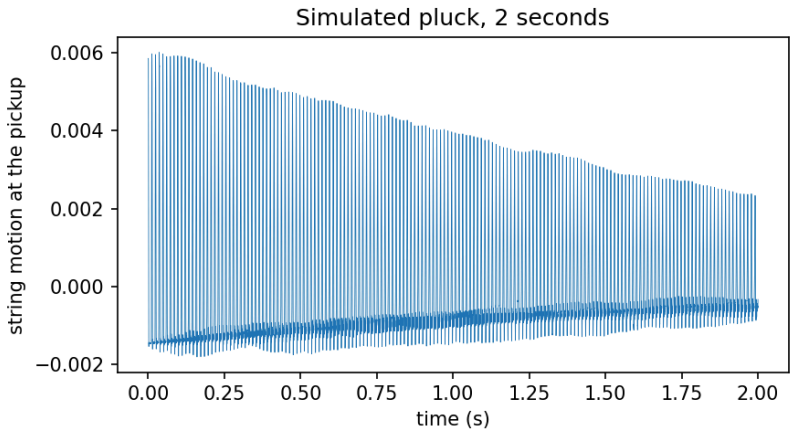
What the starter does

- Stage 1: run this file. It plucks a low E string (82.4 Hz), steps it for four periods, and compares the whole string with the exact answer. Then it measures each harmonic's fade from the simulated sound. The CHECK line should say PASS.
- Stage 2: record real strings, measure each harmonic's fade the same way, and show where one damping rate fails.

What you should see

Run `python3 07_damped_string.py`. This is the output from the run that built this guide. The numbers that depend on your computer's speed may differ; the CHECK line must say PASS.

```
after four periods: largest error along the string 3.04e-06 (pluck size 1.3)
CHECK string against the exact damped answer -> PASS (limit 1% of the pluck)
each harmonic's fade in the model (strength ~ e^(-rate t)); the model says every rate is a/2
= 0.5
  harmonic 1 (  82.4 Hz): rate 0.500 per second
  harmonic 2 ( 164.8 Hz): rate 0.500 per second
  harmonic 3 ( 247.2 Hz): rate 0.500 per second
  harmonic 4 ( 329.6 Hz): rate 0.500 per second
```



Your first experiments

From the YOUR TURN list at the bottom of the file. Write your prediction in your log *before* you run each one.

1. Record one plucked string on a phone (a quiet room, 3 seconds, saved as .wav). Set F to its note and run: python 07_damped_string.py my_pluck.wav
2. Plot rate against harmonic number. One a cannot fit it. Try rate = b0 + b2 * k^2. Does that fit better?
3. Compare a wound and a plain string, or nylon and steel. Which loses its high harmonics fastest?
4. Then fix the model: what term would make high harmonics fade faster? (Hint: a drag on u_xxt.)

The data

Settings in the online lab

Setting	Range	Starts at
Note frequency	40 to 440 Hz	82.4
Damping	0.1 to 8 1/s	1

Checked on the research solver

Solver (Telegraph, 1D)	largest error against the exact solution over 8 frames: 7.05e-4
Known answer	exact damped two-mode solution, amplitude 1.3
Error	0.05% (limit 1.00%): passed

The real world

Record plucked strings on a phone and measure each harmonic's decay on a spectrogram.

The honest limit

One constant damping term cannot describe a real string. Showing exactly how it fails, and proposing a fix, is the result.

How the full project checks itself: Each mode has the exact solution e^(-at/2) times an oscillation at $\sqrt{(\omega^2 - a^2/4)}$. The online lab runs the same project on the research solver (Telegraph, 1D).

REFERENCE

When something goes wrong

You see	Try this
`No module named numpy`	Run the install line again with the same Python you use to run the file: `python3 -m pip install -r requirements.txt`
Numbers become nan or huge	A time step is past its speed limit, usually after you made the grid finer. Each solver works out its step from the grid; check you did not fix it by hand.
`did not settle`	An iterative solver needs more sweeps for a finer grid. Raise its limit, or keep the grid smaller while you explore.
A CHECK line says FAIL after your change	Good: your change moved the model away from the case with a known answer. Undo it, confirm PASS, then change one thing at a time.
No window opens	Add `--show`, or open the PNG files in the `output` folder.
A recording gives strange notes	Record in a quiet room, save as .wav (not .m4a), trim silence at the start, and set the note you expect in the file.

How the starters relate to the online lab

The online lab, for members, runs each project on the research solver (AutoSBPSAT): high-order SBP-SAT methods, the full 3D elasticity for project 9, and frames of a time series. The starters use plain second-order methods instead, so every line is short enough to read. Both are checked against the same kind of exact answers. Use the starter to understand and to explore quickly; use the lab for results you will quote, and say in your paper which one produced each number.

How these starters were checked

Every number in "What you should see" came from running the starter while this guide was built, on Python 3.9.23, numpy 1.24.4, matplotlib 3.5.2. `check_all.py` also feeds each optional data reader (the soil-station CSV, the ruler CSV and two .wav recordings) a made-up file with a known answer, and confirms the answer comes back. They have not yet been used by a student. The twelve-week plan and the time estimates in it are a design, to be revised after the first group: tell your mentor what was unclear.