Programming Workshop 1: From Measurements to an Estimate

Preparation Handout

Signal Processing (ECE-Y523) · University of Patras
Konstantinos Chatzilygeroudis · 10 November 2026
Group 1: 11:00–13:00 Group 2: 13:00–15:00

The Programming Workshop. The Programming Workshop will take place at the ECE Department's Computing Center (KYPES). At the start of the session, we will assign you to teams of four or five students. Each team receives its own measurement problem; no two teams have the same one. In two hours your team works through the problem in a Jupyter notebook, from the raw measurements to an estimate with an uncertainty. The notebook is submitted at the end of the session and graded automatically.

What this handout is for. It tells you what kind of problem to expect, what the Programming Workshop will ask you to do, how the notebook works, and which lecture topics to revise. The problem itself, and the precise tasks, are given on the day.

1 The kind of problem

Several sensors (or channels of one instrument) record at the same time while some known input changes: a time, a temperature, a position, an excitation signal. A physical model, linear in a few unknown parameters, relates the inputs to what each channel should read. The measurements are noisy, the noise of different channels is correlated and not equally large, and the parameters are hard to separate from the data alone. You also receive recordings of the noise alone, and prior information about the parameters, such as nominal or datasheet values with their uncertainty. The illustrated overview at the end (Section 6) introduces the physical systems you may encounter.

2 What you will do

The work is divided into seven stages. Each is worth 10 points, 70 in total.

	Stage	Goal	Lectures
1	Model	write the model of the whole data set in matrix form	1, 3
2	Least squares	fit the parameters and look at what is left	1, 2
3	Conditioning	find out how reliable that fit is, and stabilise it	2
4	Noise	learn the noise from the noise-only recordings and use it to weight the data	3, 4, 5
5	Likelihood	quantify the uncertainty under a Gaussian noise model	5
6	Prior	combine the data with the prior information	6
7	Final results	compute final estimates, uncertainties and physical quantities	1–6

In each stage you implement one or more short Python functions whose names, inputs and outputs are given in the notebook, and you look at what they produce on your team's data.

3 The notebook

Cells you complete. A cell to complete contains a function whose body you have to write between the lines `### TODO: ADD YOUR CODE HERE` and `### END of TODO`. Only these regions are graded.

Test cells. Cells starting with `### TEST CELL - DO NOT EDIT` run a few checks on a small example, so you can see whether your function behaves as intended. Further checks on your team's data are run during grading, so passing the visible checks is necessary but not sufficient.

Grading.

- The stages are graded independently: a mistake in one stage does not cost points in another. Only the final results of Stage 7 depend on all the earlier stages working.
- Do not delete, split, move or rename cells, and do not edit test cells. You may add cells for your own experiments; they are not graded.
- The submitted notebook is run from the top in a fresh session. Before submitting, use **Kernel → Restart Kernel and Run All Cells** and check that every cell runs.

What you may use

The aim is to implement the methods yourselves.

- **Allowed:** NumPy array operations (`@`, `.T`, `np.sum`, `np.mean`, `np.sqrt`, ...) and the `np.linalg` functions `solve`, `inv`, `cholesky`, `qr`, `svd`, `eigh` and `norm`. You may call your own functions from earlier stages.
- **Not allowed:** `np.linalg.lstsq`, `np.linalg.pinv`, `np.linalg.cond`, `np.linalg.matrix_rank`, `np.linalg.tensorsolve`, `np.linalg.tensorinv`, `np.cov`, `np.corrcoef`, `np.polyfit`, `np.polynomial`, and any other library (SciPy, scikit-learn, statsmodels, ...).

The grader checks this in hidden tests. The visible tests check numerical results; passing them does not confirm that you used only permitted tools. A function that uses a forbidden helper, directly or through another function of yours, receives 0 points.

4 How to prepare

JupyterLab. Work through the JupyterLab guide at `https://github.com/upatras-lar/jupyter-tutorial` (about 45 minutes): running cells, the order in which cells run, restarting the kernel, reading error messages.

Python and NumPy. Be comfortable with functions and `return`, NumPy arrays and their shapes, indexing and slicing rows and blocks of a matrix, `reshape`, matrix products with `@`, and the `np.linalg` functions listed above. The Python fundamentals notebook of the practice sessions covers the basics.

Lecture topics to revise.

Lecture	Topics
1	linear models, design matrix, residuals, least squares
2	normal equations, QR, SVD and singular values, condition number, ill-conditioning, regularisation
3	multichannel signals, stacking and vectorisation, Kronecker structure
4	random vectors, covariance and its estimation, Cholesky factorisation, whitening
5	likelihood and log-likelihood, maximum likelihood, estimating the noise variance, weighted least squares, Fisher information and error bars
6	priors, MAP estimation, precision matrices, regularisation as prior information

5 On the day

Teams are formed at the start of the session. Read your problem together first: everyone should understand the model before the work is split. The stages have fixed function names, inputs and outputs, so team members can work on different stages at the same time and then bring their code into one team notebook. Allow 10–15 minutes for team allocation, login and notebook setup, leaving 105–110 minutes for the work and submission. Aim to agree on the model within the next 15 minutes, then divide the implementation into four parallel tasks: Stages 2–3; Stage 4; Stage 5; and Stage 6 with the derived quantity of Stage 7. The Stage 5 owner also finishes Stage 1 in a team of four; in a team of five, the fifth member takes Stage 1 and coordinates integration. The small public examples let you check functions before the full team-data calculation is ready. Start combining code by minute 80 of the session. Reserve 20 minutes for integration and debugging, 10 for the final results and interpretation, and the last 10 for restarting, running, saving and submitting the notebook.

At the end of the session

One notebook per team is submitted at the end of the two hours. It must run from top to bottom after a restart. In the cell containing `members`, enter every team member's student ID (AM) as a Python list of integers, without quotation marks. Example for four members:

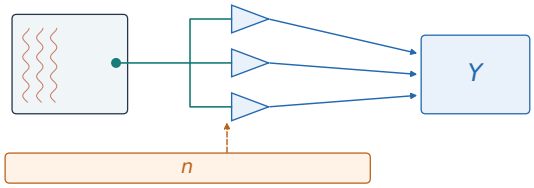
```
members = [1234567, 1234568, 1234569, 1234570]
```

Replace the example numbers with your actual IDs. For a five-person team, add the fifth ID to the same list.

6 Possible physical systems

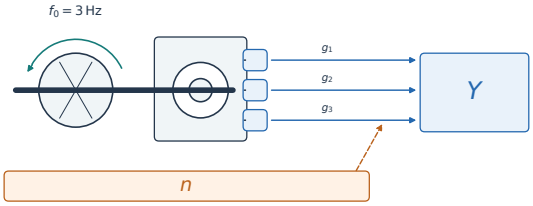
Problems come from the eight families below. Each drawing shows one representative arrangement; the text introduces both possible applications. Your team receives its specific problem on the day.

1. Sensor calibration



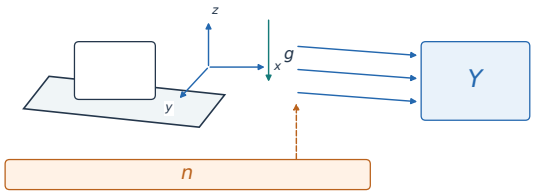
A thermocouple amplifier or a weighing platform is calibrated against known temperatures or loads. Several channels share a reference or a supply.

2. Mechanical and acoustic tones



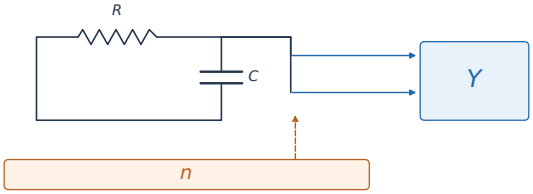
Accelerometers measure a rotating shaft, or microphones record two engine tones. A short record makes amplitudes and offsets difficult to separate.

3. Inertial sensors



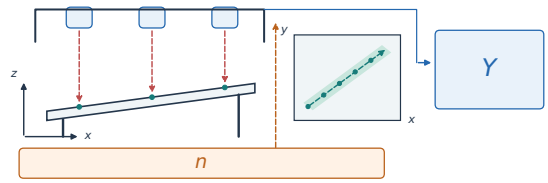
A three-axis accelerometer or gyroscope is placed in known poses or rotations. Estimate each axis's bias and scale factor despite shared electronics.

4. Electrical transients



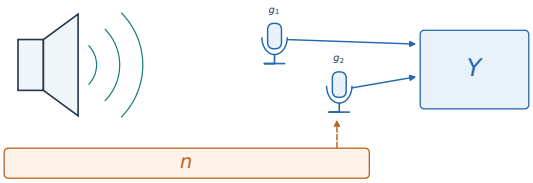
Observe an RC discharge or battery-voltage relaxation. Several voltage channels share a ground or a multiplexer while the voltage evolves over time.

5. Surface geometry



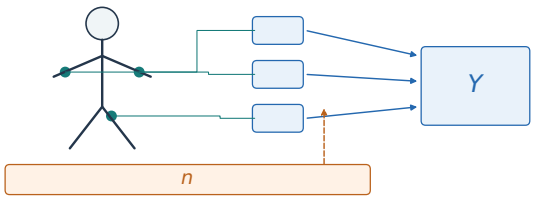
Laser sensors scan a workbench or a solar panel. Infer surface slopes from measured heights; sensor offsets and a restricted scan path complicate the fit.

6. System impulse responses



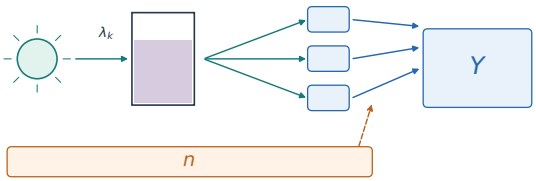
A test signal passes through a loudspeaker–room path or a cable with reflections. Two receivers measure a common short response with different known gains.

7. Mains interference



ECG leads or a stereo audio line pick up 50 Hz hum and harmonics. Estimate the interference from a short quiet interval in several correlated channels.

8. Spectral mixtures



Visible-light measurements of dyes or infrared measurements of gases contain overlapping absorption bands. Estimate concentrations while accounting for a baseline and shared detector noise.