Sensors, Data, and Models in Engineering Systems

Mario Bergés and Claude

2026-08-27

Table of contents

Welcome

This is the course website and living reference for **12-778: Sensors, Data, and Models in Engineering Systems**, offered by the Department of Civil and Environmental Engineering at Carnegie Mellon University in Fall 2026.

The course provides end-to-end foundations for developing reliable data-driven solutions to engineering problems: from measurement physics and instrumentation, through data management and representation, to statistical modeling and inference. See the Syllabus for details and the Schedule for the lecture calendar.

A note on how this book is written

The lecture notes collected here are drafted with the assistance of large language models and curated by the instructor. This is a deliberate experiment, and it is also why a portion of your grade rewards **written knowledge contributions**: if you find an error, an omission, or something that could simply be explained better, submit a correction. Details are in the Syllabus.

Part I

Logistics

1 Syllabus

12-778 | Sec. A | Fall 2026

TR 2:00PM - 3:20 PM ET Location: PH A7F

Instructors: Mario Bergés, PH 123L, Phone: x8-4572 **Office Hours:** Tuesdays 1:00 PM - 2:00 PM ET in PH 123G and by appointment.

Teaching Assistant: Avi Dube (avid@andrew.cmu.edu) **Office Hours:** Tuesdays 1:00 PM - 2:00 PM ET in 118G and by appointment.

Textbooks (optional):

1. Though no official textbooks are required, the instructor will draw from multiple sources and post the relevant references on Canvas.

Prerequisites: Probability & Statistics, Linear Algebra

1.1 Official Course Description

AI and machine learning are rapidly becoming standard tools for civil and environmental engineering, but their performance depends on how data are measured, stored, and connected to the physical systems they represent. This course provides the end-to-end foundations needed to develop reliable data-driven solutions for engineering problems (from sensing and digitization to data management and modeling) serving as a bridge into graduate machine learning. The course begins with measurement physics and instrumentation, studying how physical processes are transduced into electrical signals, conditioned and filtered, and converted to digital information. Topics include common sensor modalities, interface circuits, DC/AC circuit analysis, amplification, filtering, frequency-domain analysis, and analog-to-digital conversion. The course then develops the "data layer" required for scalable deployments, including uncertainty and measurement error, compression and representation, metadata, and database design. Finally, students learn modeling and inference concepts that prepare them for ML across learning paradigms, including supervised learning foundations (regression, regularization, validation, and generalization) and unsupervised methods for structure discovery and monitoring (dimensionality reduction, clustering, and anomaly detection). A final project integrates these components to design a sensing-to-modeling pipeline for a civil or environmental engineering application, emphasizing domain knowledge and physical consistency throughout the designed solution.

2 Grading

Component	Weight
Assignments	20%
Mid-term Exam	10%
Progress Update Report	20%
Final Exam	20%
Final Project Demo/Poster	20%
Knowledge Contributions	10%

2.1 Assignments

A total of four assignments are scheduled. The topics covered in each assignment will closely follow the concepts covered in class. Beyond exercising these concepts, the assignments are designed to progressively build the practical skills you will need for the final project: setting up and using sensing hardware, programming microcontrollers, designing and populating databases, and training models.

All assignments are to be solved individually. Discussions and conversations with other students regarding the problem sets are encouraged. However, the final solutions along with the reasoning behind them need to come from you and be clearly explained in the submitted documents.

Each assignment will be worth 5% and is due at the beginning of class on the date that is indicated in each assignment. Assignments that are submitted before this deadline can receive 100% of the available credit. There is a 3 day grace period for late submissions, with the following available credit in each of those three days: 1 Day Late: 90%, 2 Days late: 70%, 3 Days late: 40%. After this time, assignments will not be graded. Of course, if you anticipate not being able to meet this schedule due to a major problem, please talk to the instructor as soon as possible.

2.2 Written Knowledge Contributions

A small portion (10%) of the grade for the course will be based on the individual contributions each of you make to the body of knowledge we are collectively capturing on a website/book for the course. The main way you will gain points here is by suggesting corrections for errors, omissions or other possible improvements to the content. I'm running an experiment to test whether pre-trained LLMs from frontier labs can generate the content for the lecture notes. This is the third time I run this experiment, but it is still far from perfect. I expect there will be multiple opportunities to spot errors. The other way to contribute and earn these points is by submitting your own content: terms, definitions, methods, summaries, worked-out exercises, diagrams, and other contributions are fair game for this purpose. The website is hosted on a private Gitlab instance, and will require that you log in with an account in order to edit it. This will give me the opportunity to evaluate your contributions by simply following the commit log of the repository.

2.3 Project Update

A fifth (20%) of the grade for the course will be based on a progress update that will take place during the second half of the course. This update will consist of: (1) a written 2-page report, authored by all team members; and (2) a 5 minute individual meeting with the instructor to discuss the project, its goals and the plan forward. The written progress report will confer 10% of the final grade, while the individual meeting discussion will be used to provide the remaining 10%.

2.4 Final Project Demo/Poster

The final project culminates in an in-person demo/poster session held on the last day of classes (Friday, December 4). It is worth a fifth of the total grade (20%) and, in some ways, is the most important assignment. There is no written report: the poster and the accompanying demonstration are the deliverables. We recommend a 36”×48” poster (the standard size for academic conference posters). Use your own words when preparing them and avoid plagiarism of any kind. On the poster, you will describe, in simple terms, the motivation and specific objectives of your project, the design choices made for the hardware and/or software prototype that you put together, the experiments you performed to validate whether or not your solution satisfies the objectives, and a discussion about the results and limitations. You should make all code and datasets available as part of your submission, along with a digital copy of the poster, by the end of the day on December 4.

The rubric that will be used for grading the poster and demo is as follows:

- Formatting and organization of the poster (15%)
- Grammar, clarity and accuracy of ideas (15%)
- Display of mastery of concepts covered in class (35%)
- Creativity expressed in the solution (10%)
- Depth of discussion related to the pros/cons of the implemented solution (15%)

- Overall assessment of the project's idea and execution (10%)

3 Course Policies

Though there are definitely reasons to like anarchism, I still prefer democracy, so here are the rules of the game as they are now (and subject to change if enough of you request that I do).

3.1 Collaboration

Collaboration is expected within the limits of discussing concepts and problems. However, each student must produce his/her own solution to the problems. Copying from another student's assignment is clearly plagiarism. Using information directly from websites, books, papers and other literary sources without appropriate attribution is also plagiarism. Assignments submitted for this class will be reviewed by the instructor and TA and may be scanned through web-based academic integrity software. Occurrences of cheating or plagiarism will be handled according to the university policy on Academic Integrity, https://www.cmu.edu/policies/documents/Academic%20Integrity.htm. Students are expected to have read this policy and conform to the highest standards of academic integrity. For incidents of academic misconduct, the University Academic Disciplinary Actions Policy, found at https://www.cmu.edu/student-affairs/theword/acad_standards/creative/disciplinary.html, will be followed.

3.2 Use of AI

Collaboration is no longer limited to the human-human pair, so I need to address collaboration with AI systems separately.

You are welcome to use generative AI systems in this course. Just like having a smart and dedicated classmate to study and work together with, AI systems can enhance your learning, cut down the time you need to take to deliver assignments, and become more productive overall. However, your ethical responsibilities as a student remain the same. Crucially, you want to ensure that you leave this class with independent command of the concepts we are covering in class. If you want to be attractive in the job market, you clearly need to be able to deploy this newly acquired knowledge more effectively than the AI system alone (or your smart and dedicated classmate, for that matter).

You must follow CMU's academic integrity policy. Note that this policy applies to all uncited or improperly cited use of content, whether that work is created by human beings alone or in collaboration with a generative AI. If you use a generative AI tool to develop content for an assignment, you are required to cite the tool's contribution to your work. In practice, cutting and pasting content from any source without citation is plagiarism. Likewise, paraphrasing content from a generative AI without citation is plagiarism. Similarly, using any generative AI tool without appropriate acknowledgement will be treated as plagiarism.

3.3 Class Participation

Students are expected to be in class on time and participate in class discussions. If you cannot make class, please inform your instructors and group members ahead of time. In class, students are expected to be courteous and respectful of the views and needs of other students and instructors.

3.4 Students with disabilities

Students requesting classroom accommodation must first register with the Dean of Students Office. The Dean of Students Office will provide documentation to the student who must then provide this documentation to the Instructor when requesting accommodation.

3.5 Posting of course materials

All the material used in the course (syllabus, readings, problem sets, reports) is intended for use in the class only. No unauthorized posting, publication or redistribution is expected. Uploading course materials to Course Hero or other web sites is not an authorized use of the course material.

4 Course Outline

Here is a more detailed overview of how the lectures will unfold over the three periods. Most of this is subject to change. The list of "learning objectives" should be understood as responding to the phrase *By the end of the lecture, students should be able to…*

4.1 First third: Sensors

Lecture #1: Course Introduction

Learning Objectives:

- Recognize the relative importance of the assessed components of the course (final project, assignments and written contributions)
- Get to know other students in the course and their motivation for joining it
- Understand the goals of the course and its structure

References:
No references for this week.

Lecture #2: Physical Principles of Sensing

Learning Objectives:

- Describe the measurement chain that links a physical quantity of interest (the measurand) to a recorded value: transduction, conditioning, digitization, and storage
- Classify common transduction principles (resistive, capacitive, inductive, piezoelectric, thermoelectric, optical) and give an example sensor for each
- Interpret the static characteristics reported in a sensor datasheet: sensitivity, range, resolution, linearity, hysteresis, and drift
- Select an appropriate sensing modality for common measurands in civil and environmental engineering (strain, displacement, acceleration, temperature, humidity, flow, water quality)
- Explain how the physics of a transducer constrains the quality and interpretation of

the data it produces

References:

- Fraden, Jacob. *Handbook of Modern Sensors: Physics, Designs, and Applications*, 5th ed. Springer, 2016. (Chs. 1–4)
- Morris, Alan S., and Reza Langari. *Measurement and Instrumentation: Theory and Application*, 2nd ed. Academic Press, 2015. (Chs. 1–2)
- Figliola, Richard S., and Donald E. Beasley. *Theory and Design for Mechanical Measurements*, 7th ed. Wiley, 2019. (Chs. 1–2)

i Lecture #3: Electrical Circuits

Learning Objectives:

- Apply Ohm's law and Kirchhoff's voltage and current laws to analyze simple resistive networks
- Analyze voltage dividers and Wheatstone bridge circuits, and compute the output voltage of a bridge with one or more active sensing elements (e.g., strain gauges)
- Use complex impedance and phasors to analyze simple AC circuits containing resistors, capacitors, and inductors
- Derive the time- and frequency-domain behavior of first-order RC circuits
- Reduce a circuit to its Thévenin equivalent and explain why source and input impedances matter when connecting sensors to instruments

References:

- Alexander, Charles K., and Matthew N. O. Sadiku. *Fundamentals of Electric Circuits*, 7th ed. McGraw-Hill, 2020. (Chs. 2–4, 9)
- Horowitz, Paul, and Winfield Hill. *The Art of Electronics*, 3rd ed. Cambridge University Press, 2015. (Ch. 1)

i Lecture #4: Fundamentals of Data Acquisition

Learning Objectives:

- Identify the components of a data acquisition (DAQ) pipeline and the role each plays: sensor, signal conditioning, multiplexer, sample-and-hold, analog-to-digital converter
- Explain how an analog-to-digital converter maps a continuous voltage to a discrete code, and compute the resolution implied by a bit depth and full-scale range
- Quantify quantization error and its contribution to the signal-to-noise ratio of a

measurement
- Distinguish single-ended from differential measurements and identify when each is appropriate
- Specify DAQ requirements (bit depth, input range, sampling rate, channel count) for a given sensing application
- Describe how networked sensing architectures (IoT devices, wireless sensor networks) extend the single-instrument DAQ pipeline, and the constraints (power, bandwidth, synchronization) they introduce

References:

- Taylor, H. Rosemary. *Data Acquisition for Sensor Systems.* Springer, 1997.
- Fraden, Jacob. *Handbook of Modern Sensors*, 5th ed. Springer, 2016. (Ch. 5)

Lecture #5: Signal Conditioning and Sampling

Learning Objectives:

- Explain the purposes of signal conditioning: amplification, level shifting, isolation, and filtering
- Analyze basic operational amplifier configurations (inverting, non-inverting, differential/instrumentation amplifiers) under the ideal op-amp assumptions
- State the Nyquist-Shannon sampling theorem and predict the aliased frequency of an undersampled tone
- Explain why anti-aliasing filters must be analog and placed before the ADC, and design a first-order low-pass filter for that purpose
- Choose a sampling rate for a given application, balancing bandwidth, storage, and aliasing concerns

References:

- Horowitz, Paul, and Winfield Hill. *The Art of Electronics*, 3rd ed. Cambridge University Press, 2015. (Ch. 4)
- Lyons, Richard G. *Understanding Digital Signal Processing*, 3rd ed. Prentice Hall, 2010. (Chs. 1–2)

Lecture #6: Frequency Domain Analysis

Learning Objectives:

- Explain, conceptually, how the Fourier series and Fourier transform decompose a signal into sinusoidal components

- Compute and interpret the discrete Fourier transform (DFT) of a sampled signal, relating bin indices to physical frequencies
- Recognize spectral leakage and explain how windowing mitigates it
- Estimate and interpret the power spectral density of a stationary signal
- Explain how a linear time-invariant system is characterized by its frequency response, and use transfer functions to reason about the dynamics of sensors and filters
- Apply the FFT to real measurement data (e.g., identify the dominant modal frequency of a vibrating structure from accelerometer data)

References:

- Oppenheim, Alan V., and Alan S. Willsky. *Signals and Systems*, 2nd ed. Prentice Hall, 1996. (Chs. 3–5)
- Lyons, Richard G. *Understanding Digital Signal Processing*, 3rd ed. Prentice Hall, 2010. (Chs. 3–4)

i Lecture #7: Measurement Error and Uncertainty

Learning Objectives:

- Distinguish systematic from random errors, and accuracy from precision, in a measurement system
- Identify common noise sources in instrumentation (thermal noise, interference, quantization) and strategies to mitigate them
- Explain the role of calibration in reducing systematic error
- Quantify measurement uncertainty from repeated observations (Type A) and from specifications or prior knowledge (Type B)
- Propagate uncertainty through a functional relationship using first-order (Taylor series) methods, and explain why propagation for linear Gaussian models is exact while non-linear models require linearization or Monte Carlo approaches
- Combine independent uncertainty components and report a measurement result with a defensible uncertainty statement

References:

- Taylor, John R. *An Introduction to Error Analysis*, 2nd ed. University Science Books, 1997.
- Holman, Jack P. *Experimental Methods for Engineers*, 8th ed. McGraw-Hill, 2011. (Ch. 3)
- JCGM 100:2008. *Evaluation of Measurement Data — Guide to the Expression of Uncertainty in Measurement (GUM)*. https://www.bipm.org/en/committees/jc/jcgm/publications

4.2 Second third: Data

ℹ Lecture #8: Time-Series Data: Representation and Processing

Learning Objectives:

- Represent time-series data unambiguously, handling timestamps, time zones, and irregular sampling
- Decompose a time series into trend, seasonal, and residual components to reveal patterns in sensor data
- Resample, interpolate, and align time series recorded at different rates or with gaps, using regression to estimate unobserved values
- Detect and handle missing values and outliers in sensor data (e.g., using Chauvenet's criterion), and articulate the assumptions each strategy makes
- Explain what virtual sensors are and how they derive measurements from other sensors or models
- Compute rolling statistics and apply simple smoothing filters to noisy measurements
- Compare storage formats for time-series data (CSV, Parquet, purpose-built time-series databases) in terms of size, speed, and interoperability
- Use `pandas` to carry out all of the above on real sensor data

References:

- McKinney, Wes. *Python for Data Analysis*, 3rd ed. O'Reilly, 2022. (Ch. 11) Open edition: https://wesmckinney.com/book/
- Hyndman, Rob J., and George Athanasopoulos. *Forecasting: Principles and Practice*, 3rd ed. OTexts, 2021. (Ch. 3) Free at https://otexts.com/fpp3/

ℹ Lecture #9: Relational Databases and the Entity-Relationship Model

Learning Objectives:

- Define sets and apply the basic set operations (union, intersection, difference, complement, Cartesian product) and their properties, and explain how set theory underpins the relational data model
- Identify entities, attributes, relationships, and cardinalities — including weak entity sets, roles, and subclasses — in a description of an engineering system
- Construct an entity-relationship diagram (ERD) for a sensing deployment (e.g., buildings, floors, sensors, measurements)
- Relate the components of the relational model (relations, attributes, tuples, types) to their table counterparts (tables, columns, rows, domains), and distinguish a schema from an instance

- Apply and compose the core relational algebra operators (select, project, join, union, difference, rename) to express queries
- Translate an ERD into a relational schema with appropriate primary and foreign keys

References:

- Garcia-Molina, Hector, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems: The Complete Book*, 2nd ed. Prentice Hall, 2009. (Chs. 2, 4)

Lecture #10: The Structured Query Language (SQL)

Learning Objectives:

- Distinguish the Data Definition Language (DDL) and Data Manipulation Language (DML) families of SQL statements, and create tables, insert, update, and delete records with them
- Write `SELECT` queries with filtering, ordering, aggregation, and `GROUP BY`
- Combine data from multiple tables using joins, and explain the difference between inner and outer joins
- Use subqueries and nested `SELECT` statements to answer multi-step questions
- Translate between relational algebra expressions and their equivalent SQL queries
- Set up and issue queries against a local DBMS (e.g., SQLite or MySQL), including from Python (`sqlite3`, SQLAlchemy) with results loaded into `pandas`

References:

- Garcia-Molina, Hector, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems: The Complete Book*, 2nd ed. Prentice Hall, 2009. (Ch. 6)

Lecture #11: Database Design Principles

Learning Objectives:

- Identify insertion, update, and deletion anomalies in a poorly designed schema
- Derive functional dependencies from the properties of the data, and compute closures and keys of relations
- Normalize a schema through the normal forms, decomposing a relation up to Boyce-Codd Normal Form (BCNF) when possible
- Explain what transactions and the ACID properties guarantee, and why they matter for concurrent data collection
- Describe how indexes speed up queries and what they cost, and weigh normalization

against denormalization for high-volume time-series workloads

References:

- Garcia-Molina, Hector, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems: The Complete Book*, 2nd ed. Prentice Hall, 2009. (Ch. 3)

Lecture #12: Data Representation and Compression

Learning Objectives:

- Explain how integers, floating-point numbers, and ASCII text are represented in binary, and predict the size of a simple file from its content
- Define information entropy and use it to bound the compressibility of a data source
- Apply Huffman coding to an arbitrary string, and describe other common lossless techniques (run-length, dictionary-based) and where each performs well
- Contrast lossless and lossy compression, and explain how frequency-domain (transform) representations enable lossy schemes such as JPEG
- Apply delta and delta-of-delta encoding ideas to time-series data and estimate the achievable savings
- Debate whether large language models are, in a meaningful sense, compression algorithms

References:

- Sayood, Khalid. *Introduction to Data Compression*, 5th ed. Morgan Kaufmann, 2017. (Chs. 1–3)

Lecture #13: Other Database Models: Graphs, Objects, Key-Value Stores

Learning Objectives:

- Recognize the workloads for which the relational model becomes awkward, and the alternatives that emerged in response
- Describe the key-value, document, columnar, object, and graph data models, with an example system for each
- Explain, at a high level, the trade-offs distributed data stores make between consistency, availability, and partition tolerance
- Describe what purpose-built time-series databases optimize for
- Select an appropriate storage model for a given sensing deployment and justify the choice

References:

- Silberschatz, Abraham, Henry F. Korth, and S. Sudarshan. *Database System Concepts*, 7th ed. McGraw-Hill, 2019. (Ch. 8)
- Robinson, Ian, Jim Webber, and Emil Eifrem. *Graph Databases*, 2nd ed. O'Reilly, 2015. (Chs. 1–2)

ℹ Lecture #14: Metadata and the Resource Description Framework

Learning Objectives:

- Explain why metadata, not sensor data itself, is often the bottleneck for scaling data-driven applications across buildings and infrastructure systems
- Represent facts as subject-predicate-object triples using the Resource Description Framework (RDF)
- Explain the role of ontologies and controlled vocabularies in making metadata machine-actionable
- Write simple SPARQL queries against an RDF graph
- Use a domain schema such as Brick to describe the sensors, equipment, and relationships in a real building

References:

- Balaji, Bharathan, et al. "Brick: Towards a Unified Metadata Schema for Buildings." *Proceedings of ACM BuildSys 2016*. https://doi.org/10.1145/2993422.2993577
- Allemang, Dean, and James Hendler. *Semantic Web for the Working Ontologist*, 2nd ed. Morgan Kaufmann, 2011. (Chs. 1–3)

4.3 Last third: Models

ℹ Lecture #15: Principles of Statistical Learning

Learning Objectives:

- Distinguish supervised from unsupervised learning, and regression from classification, and give a civil/environmental engineering example of each
- Formulate learning as minimizing expected loss, and distinguish training error from generalization error
- Explain the bias-variance trade-off and how model flexibility drives it
- Diagnose overfitting and underfitting from training and validation performance
- Design a defensible model assessment protocol using train/validation/test splits and cross-validation, including the pitfalls of temporal data

References:

- James, Gareth, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. *An Introduction to Statistical Learning with Applications in Python.* Springer, 2023. (Ch. 2) Free at https://www.statlearning.com

ℹ Lecture #16: Linear Models for Regression

Learning Objectives:

- Formulate the ordinary least squares problem and derive its closed-form solution
- Interpret estimated coefficients, standard errors, and diagnostic plots
- Connect least squares to maximum likelihood estimation under Gaussian noise
- Apply ridge and lasso regularization, and explain how each controls model complexity
- Engineer features (transformations, interactions, encodings of categorical variables) to improve a linear model
- Fit and validate a regression model on real sensor data (e.g., predicting building energy use from weather)

References:

- James, Gareth, et al. *An Introduction to Statistical Learning with Applications in Python.* Springer, 2023. (Chs. 3, 6)

ℹ Lecture #17: Linear Models for Classification

Learning Objectives:

- Explain why least squares is ill-suited to classification and formulate logistic regression instead
- Interpret logistic regression coefficients in terms of log-odds and predicted probabilities
- Visualize and reason about linear decision boundaries
- Evaluate classifiers using confusion matrices, precision, recall, and ROC curves, and choose metrics appropriate to the application
- Recognize the effects of class imbalance and strategies to address it
- Apply classification to an engineering monitoring problem (e.g., damage or fault detection)

References:

- James, Gareth, et al. *An Introduction to Statistical Learning with Applications in Python.* Springer, 2023. (Ch. 4)

Lecture #18: Beyond Linearity

Learning Objectives:

- Extend linear models with polynomial terms, step functions, and splines
- Explain how basis expansions retain the machinery of linear models while capturing nonlinear relationships
- Describe k-nearest neighbors and the effect of k on the bias-variance trade-off
- Recognize the curse of dimensionality and its consequences for local methods
- Select an appropriate level of model flexibility for a given dataset using validation evidence

References:

- James, Gareth, et al. *An Introduction to Statistical Learning with Applications in Python*. Springer, 2023. (Ch. 7)

Lecture #19: Tree-Based Methods

Learning Objectives:

- Explain how decision trees are grown, and why single trees tend to overfit
- Describe how bagging and random forests reduce variance through ensembling
- Describe boosting and contrast it with bagging
- Interpret variable importance measures and partial dependence, and state their limitations
- Apply tree ensembles to tabular engineering data and compare their performance against linear baselines

References:

- James, Gareth, et al. *An Introduction to Statistical Learning with Applications in Python*. Springer, 2023. (Ch. 8)

Lecture #20: Unsupervised Learning

Learning Objectives:

- Formulate principal component analysis as variance maximization and use it for dimensionality reduction and visualization
- Apply and compare clustering algorithms (k-means, hierarchical, density-based) and assess cluster quality

- Frame anomaly detection as an unsupervised problem and apply distance-, density-, and reconstruction-based detectors
- Connect these tools to monitoring applications: novelty detection in structural health monitoring, load pattern discovery in energy data
- Articulate why validating unsupervised methods is fundamentally harder than validating supervised ones

References:

- James, Gareth, et al. *An Introduction to Statistical Learning with Applications in Python*. Springer, 2023. (Ch. 12)

Lecture #21: Neural Networks and Sequential Models

Learning Objectives:

- Describe the architecture of a multilayer perceptron: layers, weights, activation functions
- Explain, conceptually, how networks are trained via gradient descent and backpropagation
- Recognize the practical levers of training: learning rate, batch size, early stopping, and regularization
- Explain how recurrent architectures and autoregressive formulations model sequential dependence in time-series data
- Identify when deep learning is (and is not) warranted for engineering problems, considering data volume, interpretability, and physical consistency

References:

- James, Gareth, et al. *An Introduction to Statistical Learning with Applications in Python*. Springer, 2023. (Ch. 10)
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. (Chs. 6, 10) Free at https://www.deeplearningbook.org

4.4 Coda: Sensors, Data, and Models

ℹ Lecture #22: Putting It All Together

Learning Objectives:

- Trace a complete pipeline from physical phenomenon to actionable prediction for a real case study, identifying every design decision along the way
- Explain how choices made at the sensing and data layers (sampling rate, resolution, storage, metadata) constrain what models can later achieve
- Plan a monitoring campaign end-to-end, drawing on lessons learned from computer systems research about instrumentation, logging, and measurement at scale
- Critique an end-to-end system design for physical consistency, scalability, and maintainability
- Map the concepts from each third of the course onto your own final project, and identify the weakest link in your pipeline

References:

- Selected case studies posted on Canvas.

5 Schedule

The schedule is subject to change. Here's the most updated version:

Week	Date	Lecture	Notes
1	Aug 25 (Tue)	**Introduction**	
	First Third	**SENSORS**	
1	Aug 27 (Thu)	**Physical Principles of Sensing**	
2	Sep 1 (Tue)	**Electrical Circuits**	
2	Sep 3 (Thu)	**Fundamentals of Data Acquisition**	
3	Sep 8 (Tue)	**Signal Conditioning and Sampling**	
3	Sep 10 (Thu)	**Frequency Domain Analysis**	
4	Sep 15 (Tue)	**Measurement Error and Uncertainty**	
	Second Third	**DATA**	
4	Sep 17 (Thu)	**Time-Series Data: Representation and Processing**	
5	Sep 22 (Tue)	**Relational Databases and the Entity-Relationship Model**	
5	Sep 24 (Thu)	**The Structured Query Language (SQL)**	
6	Sep 29 (Tue)	**Database Design Principles**	
6	Oct 1 (Thu)	**Data Representation and Compression**	
7	Oct 6 (Tue)	**Other Database Models: Graphs, Objects, Key-Value Stores**	

Week	Date	Lecture	Notes
7	Oct 8 (Thu)	**Metadata and the Resource Description Framework**	
8	Oct 13 (Tue)	**FALL BREAK**	No Class
8	Oct 15 (Thu)	**FALL BREAK**	No Class
9	Oct 20 (Tue)	**MID-TERM EXAM**	Covers Sensors and Data
	Last Third	**MODELS**	
9	Oct 22 (Thu)	**Principles of Statistical Learning**	
10	Oct 27 (Tue)	**Linear Models for Regression**	
10	Oct 29 (Thu)	**Linear Models for Classification**	
11	Nov 3 (Tue)	**DEMOCRACY DAY**	No Class
11	Nov 5 (Thu)	**Beyond Linearity**	
12	Nov 10 (Tue)	**Tree-Based Methods**	
12	Nov 12 (Thu)	**Unsupervised Learning**	
13	Nov 17 (Tue)	**Neural Networks and Sequential Models**	
	Coda	**SENSORS, DATA, AND MODELS**	
13	Nov 19 (Thu)	**Putting It All Together**	Sensors, Data and Models
14	Nov 24 (Tue)	**Project Feedback / Guidance Session** (Zoom)	
14	Nov 26 (Thu)	**THANKSGIVING**	No Class
15	Dec 1 (Tue)	**Project Feedback / Guidance Session**	
15	Dec 3 (Thu)	**Project Feedback / Guidance Session**	

Additional Dates:

- **Friday, December 4:** In-person final project demo/poster session (last day of classes). Digital copy of the poster, code, and datasets due by end of day.

Part II

Lectures

6 Lecture 02: Physical Principles of Sensing

6.1 Lecture Overview

Learning Objectives

By the end of this lecture, students should be able to:

- Describe the measurement chain that links a physical quantity of interest (the measurand) to a recorded value: transduction, conditioning, digitization, and storage
- Classify common transduction principles (resistive, capacitive, inductive, piezoelectric, thermoelectric, optical) and give an example sensor for each
- Interpret the static characteristics reported in a sensor datasheet: sensitivity, range, resolution, linearity, hysteresis, and drift
- Select an appropriate sensing modality for common measurands in civil and environmental engineering (strain, displacement, acceleration, temperature, humidity, flow, water quality)
- Explain how the physics of a transducer constrains the quality and interpretation of the data it produces

Topics Covered

- Measurements and instruments: from rulers and scales to digital instruments
- The measurement chain and the anatomy of a sensor
- Transduction principles and the physics inside common sensors
- Static characteristics on a datasheet
- Matching sensors to CEE measurands
- How transducer physics shapes the data downstream

6.2 Measurements and Instruments

Every claim in engineering rests on a measurement: a number, attached to a physical quantity, by some procedure. The deflection limit you checked in a design code, the concrete strength you accepted from the lab, the "the building used 12% less energy" line in a report — each is only as trustworthy as the procedure that produced the number. And yet, when data arrive in a spreadsheet, we tend to treat them as facts rather than as the output of a device somebody designed. This course is about closing that gap, and this first lecture starts with the most basic question: what is a measurement, and what is the thing that makes one?

A *measurement* is the assignment of a value to a quantity by comparing it against a reference. When you say a beam is 3.2 m long, you are saying it matches 3.2 copies of a reference length. Nothing in that statement is automatic: someone had to decide what the reference is, build an object that embodies it, and define a comparison procedure whose result you can trust. The device that carries out that comparison is an *instrument*.

6.2.1 Everyday Instruments

The instruments you already own show the whole story. A ruler looks like a stick with marks on it, but consider the decisions behind it: a range (30 cm, because that fits a desk), a resolution (millimeter marks, because your eye cannot reliably split them), a material chosen so that the marks do not move (steel or a stable plastic, not rubber), and a manufacturing step in which the marks were printed against a more accurate reference, that is, a *calibration*. Consider also how it can be wrong: the marks were printed with some tolerance, the plastic expands on a hot day, the end wears down after years of use so the zero is no longer at zero. Every one of those failure modes has a name on a datasheet — accuracy, drift, offset — and we will meet them formally in Section 6.5. The ruler has them too; it simply does not come with a datasheet.

A bathroom scale is a richer example because you cannot see the reference. Inside is a spring (in old ones) or a strain-sensing element (in new ones) whose deformation under your weight is compared, through a mechanism or a circuit, against the behavior the designer measured in the factory with known masses. That comparison is stored in the scale — as the shape of a dial, or as a few numbers in a chip — and it is only valid as long as the spring behaves as it did in the factory. Leave the scale in a damp bathroom for a decade and the stored comparison quietly stops being true. This is why serious instruments get *re*-calibrated on a schedule, and why "the sensor said so" is never the end of an argument.

Active Learning: An Instrument You Own

Pick either a ruler or a bathroom scale and write down, in a few lines each: what it was designed to measure and over what range; what its resolution is and what limits it; how it was calibrated (what was the reference?); and at least three distinct ways it could give you a wrong number. Keep your list — by the end of Section 6.5 you should be able to name every item on it with a datasheet term.

6.2.2 Modern Instruments

A digital thermometer does the same job as a mercury one, but open it up and you find a very different object: a tiny element whose electrical resistance depends on temperature, a circuit that turns that resistance into a voltage, a chip that turns the voltage into a number, a display, a battery, and maybe a radio. Only one of those parts actually responds to temperature. The rest exist to make the response usable — to amplify it, to digitize it, to show it, to send it — and every one of them can add error of its own.

That is the picture of a modern instrument worth keeping in mind for the entire course: **a transducer surrounded by supporting components**. The transducer is where physics happens; the supporting components are where engineering happens; the data you eventually analyze are the product of both. When the course description says a model's performance "depends on how data are measured," this is what it means concretely. If you want to trust a model built on sensor data, you have to understand what the transducer could and could not see, and what the chain around it did to the signal on the way to storage. The next section names the links of that chain; the rest of the lecture is about the transducer at its head.

6.3 The Measurement Chain

Suppose a temperature reading in your building database is wrong. Where is the fault? It could be in the probe itself, in the wiring that connects it, in the amplifier, in the analog-to-digital converter, in the logger that timestamps it, or in the database that stores it. Unless you can name those stages, you cannot even begin to debug, and you certainly cannot say which errors a downstream model should worry about. The *measurement chain* is that list of stages (Figure 6.1), and it doubles as the map of the SENSORS third of this course: this lecture covers the first block, Lectures 3–5 cover the middle, and the DATA third picks up where the chain ends.

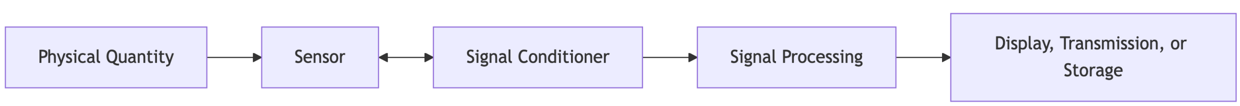


Figure 6.1: The measurement chain of a typical instrument. The double arrow between sensor and signal conditioner reflects that many sensors need an excitation signal supplied to them.

6.3.1 What Is a Sensor?

A thermistor does not output "temperature." It outputs a resistance, which something else must read. That small observation forces us to be precise about words. Following Fraden (2016), we will say:

> A **sensor** is a device that receives a stimulus and responds with an electrical signal.

The *stimulus* (or *measurand*, when we want to stress that we intend to quantify it) is the physical quantity being sensed: temperature, strain, light, current. The response is electrical — a voltage, a current, a charge, a resistance — because electrical signals are what circuits, converters, and computers can handle. A sensor is therefore a translator from the physical world into the electrical one.

A **transducer** is the more general device: it converts energy from one form into another, in *any* direction. A loudspeaker is a transducer (electrical → acoustic); so is an electric motor (electrical → mechanical); neither is a sensor. Transducers that go the other way, into electrical energy, are sensors. So every sensor is a transducer, but not every transducer is a sensor — and in engineering practice you will hear the two used interchangeably, which is fine as long as you know what the speaker means. Transducers that go from electrical to physical are usually called *actuators*, and you will meet plenty of them in control systems.

One more subtlety matters for us. Many sensors are really a *chain* of transducers ending in one that produces electricity. A fiber-optic pressure sensor turns pressure into strain in a fiber, strain into a change of refractive index, refractive index into a change in light intensity, and only then light into current at a photodiode. Fraden calls the last stage a *direct sensor* and the whole assembly a *complex sensor*. We will meet the most important example of this pattern, the accelerometer, at the end of Section 6.4.

6.3.2 Classifying Sensors

A catalog contains thousands of sensors. Without a few organizing axes there is no way to reason about which family suits a job, so here are the three that Fraden proposes and that we will use throughout (Fraden 2016, Ch. 1):

1. **By energy source: passive or active.** A *passive* sensor generates its electrical signal directly from the stimulus, with no external power: a thermocouple, a photodiode, a piezoelectric crystal. An *active* sensor needs an *excitation* signal, which the stimulus then modulates: a thermistor is just a resistor until you pass a current through it, and a strain gauge likewise. Active sensors are sometimes called *parametric* because a parameter of theirs (resistance, capacitance) is what changes. This axis matters practically: passive sensors can run without a supply, active ones need a circuit that provides one, and (as we will see in Section 6.7) that excitation can itself disturb the measurement. The double arrow in Figure 6.1 is that excitation.
2. **By reference: absolute or relative.** An *absolute* sensor responds relative to a fixed physical scale independent of conditions — a thermistor's resistance is a function of absolute temperature. A *relative* sensor responds to the difference from some baseline: a thermocouple produces a voltage that depends on the temperature *difference* between its

two junctions, so it cannot tell you a temperature without a known reference. Gauge versus absolute pressure sensors are the other classic pair.

3. **By characteristics.** Fraden's remaining tables list the ways a sensor can be described: by its specifications (the datasheet numbers of Section 6.5), by material, by *detection means* (electric, magnetic, thermal, mechanical, radiation, …), by *conversion phenomenon* (thermoelectric, photoelectric, electromagnetic, …), by field of application, and by stimulus. The conversion phenomenon is what we will call the *transduction principle* in Section 6.4 — arguably the single most informative label a sensor can carry.

6.3.3 Three Sensors Up Close

Definitions stick only once you have applied them to devices you could actually buy, so here are the three we will keep returning to. Each is cheap, common in student projects, and comes with a real datasheet you should open now (links in Table 6.1).

Accelerometer: Analog Devices ADXL335. A three-axis accelerometer on a 4 mm × 4 mm chip. Inside is a polysilicon proof mass suspended on springs above a silicon wafer; its motion unbalances a *differential capacitor* formed between plates on the mass and fixed plates around it, and on-chip electronics turn that imbalance into three voltages, one per axis. It needs a supply (1.8–3.6 V) — an *active* sensor — and it responds to acceleration relative to free fall, so it reads 1 g sitting on the table: an *absolute* sensor. Its conversion phenomenon is capacitive, its stimulus mechanical. At 3 V it produces about 300 mV per g over a ± 3 g range.

Photoresistor: GL5528 CdS cell. A disc of cadmium sulfide whose resistance falls as light falls on it: from more than 1 MΩ in darkness to roughly 8–20 kΩ at 10 lux. It generates nothing by itself — you must pass a current through it — so it is *active*; it responds to light level referenced to darkness, so *absolute*. Its detection means is optical, but its conversion phenomenon is photoconductive: the output is a resistance, which is why we will file it under *resistive* in Section 6.4. It is slow (rise and fall times of tens of milliseconds) and its resistance also depends on temperature, two facts that will matter later.

Split-core current transformer: CR Magnetics CR3110-3000. A hinged iron ring that clamps around a live conductor without cutting it. The alternating current in the conductor creates an alternating magnetic field in the ring, which induces a current in a coil of about 3000 turns wound on it; the induced current is the primary current divided by the turns ratio. The energy comes from the conductor being measured, so the CT is *passive*; it reports current referenced to zero, so *absolute*. Its conversion phenomenon is electromagnetic induction — *inductive* in our scheme — and it only works for AC (20 Hz to 1 kHz on this part), which is the kind of limitation the physics imposes and a datasheet quietly records.

Table 6.1: The three sensors used as running examples, classified along the axes of Section 6.3.2.

Sensor	Part	Stimulus → output	Energy	Reference	Principle	Datasheet
Accelerometer	ADXL335	acceleration → voltage	active	absolute	capacitive	Analog Devices
Photoresistor	GL5528	illuminance → resistance	active	absolute	resistive (photoconductive)	GL55 series sheet
Current transformer	CR3110-3000	AC current → current	passive	absolute	inductive	CR Magnetics CR3100 series

6.3.4 The Rest of the Chain

A raw sensor signal is usually the wrong size and the wrong shape: millivolts when the converter wants volts, a resistance when the converter wants a voltage, a clean signal riding on noise picked up by a meter of cable. None of it can go straight into a display or a database. The rest of the chain exists to fix that, and each link gets a lecture of its own; here we only fix what each is responsible for.

The **signal conditioner** turns the sensor's output into a clean voltage of convenient size: it supplies excitation to active sensors, converts resistance or current to voltage, amplifies, filters, and isolates (Lectures 3 and 5). **Signal processing** is everything done to that voltage before it becomes information: in a digital instrument, that means sampling and analog-to-digital conversion (Lecture 4), followed by whatever arithmetic runs on the resulting numbers — from a moving average to a full frequency analysis (Lecture 6). Finally, the result is **displayed, transmitted, or stored**, and the moment it is stored it becomes *data* — the subject of the second third of the course.

Figure 6.2 and Figure 6.3 show the two classic ways the chain is realized. In an analog instrument everything stays a continuous voltage until a needle or a chart recorder shows it; in a digital one an analog-to-digital converter sits in the middle and a computer does the rest. Nearly everything you will build is the second kind, but the first is worth knowing because the analog front end — input circuit, preamplifier, filters — is still there in front of every converter.

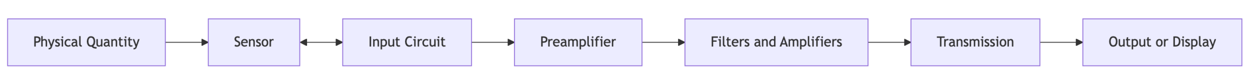


Figure 6.2: A typical analog instrument.

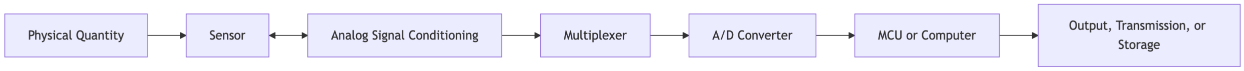


Figure 6.3: A typical digital instrument.

6.4 Transduction Principles: The Physics Inside

The three sensors we just met look nothing alike: a silicon chip, a blob of cadmium sulfide, a hinged iron ring. Yet each is a single physical effect that turns its stimulus into an electrical quantity, and knowing *which* effect predicts a great deal — how fast the sensor is, whether it needs power, how it will react to temperature, what circuit it needs — before you read a single line of its datasheet. Six effects cover most of what a civil or environmental engineer will ever deploy. For each we ask the same four things: what problem it solves best, what relation governs it, which electrical quantity changes, and where it shows up in our field.

6.4.1 Resistive

The cheapest way to sense strain or temperature is to let the stimulus change a resistance, because a resistance is the easiest electrical quantity to read with a simple circuit (Lecture 3 will show you exactly how). The resistance of a conductor of resistivity ρ, length l, and cross-sectional area a is

$$R = \rho \, \frac{l}{a}. \tag{6.1}$$

Anything that changes l, a, or ρ changes R. Stretch the conductor and l grows while a shrinks: that is the **strain gauge**, the workhorse of structural monitoring. How much R changes per unit strain is the *gauge factor* $\mathrm{GF} = (\Delta R/R)/\varepsilon$. A quick way to estimate it is to assume the conductor keeps its volume $v = l\,a$ as it stretches, so $R = \rho\, l^2/v$ and $\mathrm{d}R/\mathrm{d}l = 2\rho\, l/v$, which gives $\mathrm{GF} = 2$ exactly. The honest version is $\mathrm{GF} = 1 + 2\nu + (\Delta\rho/\rho)/\varepsilon$, where ν is Poisson's ratio: the geometric part alone is 1.6 for $\nu = 0.3$, the constant-volume shortcut corresponds to $\nu = 0.5$, and the last term — the *piezoresistive* effect, the change of resistivity itself with strain — supplies the rest. Metal foil gauges land near 2; semiconductor gauges exploit a large piezoresistive term and reach 100 or more.

Heat the conductor instead and ρ changes. For metals the relation is close to linear over useful ranges,

$$R = R_0 \, [1 + \alpha \, (T - T_0)], \tag{6.2}$$

which is the **resistance temperature detector** (RTD); a platinum Pt100 has $R_0 = 100\ \Omega$ at $0\,^\circ$C and $\alpha \approx 0.00385\ /^\circ$C, so it reads $138.5\ \Omega$ at $100\,^\circ$C. **Thermistors** are the semiconductor cousin: far more sensitive, strongly nonlinear, and cheaper, which is why they are in your thermostat while the RTD is in the calibration lab. And the **photoresistor** belongs here too: light frees charge carriers in the CdS, ρ drops, and the readout is a resistance — an optical stimulus with a resistive output.

6.4.2 Capacitive

A capacitor is two conductors separated by an insulator, and its capacitance depends only on geometry and the material between the plates. For parallel plates of overlap area A separated by a gap d,

$$Q = CV, \qquad C = \varepsilon_0 \, \varepsilon_r \, \frac{A}{d}, \tag{6.3}$$

with $\varepsilon_0 = 8.854 \times 10^{-12}$ F/m the permittivity of free space and ε_r the relative permittivity of the dielectric. Change d, A, or ε_r and C changes. The beauty of this principle is that, once charged, an ideal capacitor draws no current: capacitive sensors can run for years on a coin cell, which is why MEMS accelerometers and humidity sensors are everywhere.

The ADXL335 moves the gap. Its proof mass carries a comb of thin fingers that interleave with fixed fingers; acceleration deflects the mass, one gap shrinks while its mirror image grows, and the *difference* between the two capacitances is proportional to the deflection. A typical humidity sensor instead changes ε_r: a polymer film between the plates absorbs water vapor, and water's high permittivity raises C. Figure 6.4 shows the single-gap relation for finger dimensions of the order used in MEMS parts — and shows something the datasheet section will formalize: $C(d)$ is a hyperbola, not a line, so the sensor is only linear for small excursions about its rest gap. (The differential arrangement in the ADXL335 largely cancels this, which is one reason it is built that way.)

```
eps0 = 8.8541878128e-12
A = 125e-6 * 3e-6              # finger overlap area, m^2
d0 = 1.3e-6                    # rest gap, m
d = np.linspace(0.6e-6, 2.0e-6, 300)
C = eps0 * A / d
C0 = eps0 * A / d0
slope = -C0 / d0               # dC/dd at d0
fig, ax = plt.subplots(figsize=(6, 3.6))
ax.plot(d * 1e6, C * 1e15, label=r"$C = \varepsilon_0 A / d$")
ax.plot(d * 1e6, (C0 + slope * (d - d0)) * 1e15, "--", label="tangent at rest gap")
ax.plot([d0 * 1e6], [C0 * 1e15], "ko")
ax.annotate(f"rest gap: C  = {C0*1e15:.2f} fF", (d0 * 1e6, C0 * 1e15), xytext=(1.45, 4.2),
            arrowprops=dict(arrowstyle="->"))
```

```
ax.set_xlabel("gap $d$ (µm)"); ax.set_ylabel("capacitance (fF)"); ax.legend()
plt.show()
```

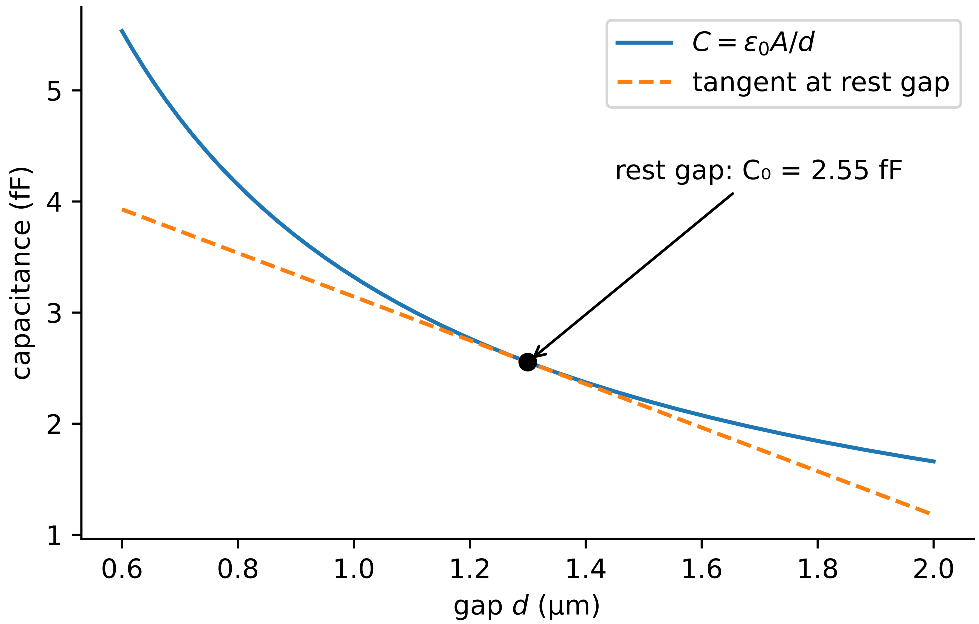


Figure 6.4: Capacitance of a single MEMS-scale finger pair (overlap 125 µm × 3 µm, rest gap 1.3 µm) as the gap changes. The tangent at the rest gap is the local sensitivity; the departure from it is the nonlinearity a single-ended capacitive sensor would show.

6.4.3 Inductive

Reading the current in a live conductor without cutting it sounds impossible until magnetic coupling does the work for you. A current I_p in a wire creates a magnetic field around it; a coil of N turns wound on an iron ring around the wire links that field; and Faraday's law of induction says a *changing* field induces a current in the coil. For an ideal transformer the induced secondary current is

$$I_s = \frac{I_p}{N}, \tag{6.4}$$

so a 3000-turn CT turns 75 A into 25 mA. Pass that through a *burden* resistor R_b and you have a voltage, $V = I_s R_b$, that a converter can read. Real CTs deviate slightly from Equation 6.4: the CR3110-3000's sheet quotes an *effective* turns ratio of 3100 (including losses), a secondary resistance of 440 Ω, and a saturation voltage; it also tells you to keep V below 80% of the available voltage headroom for good linearity, so a 100 Ω burden gives 2.42 V at full current, comfortably inside the linear region.

The physics also sets the limits. Induction needs a *changing* field, so a CT cannot see direct current at all, and this one is rated from 20 Hz to 1 kHz. The same coupling principle, applied to a core that moves inside a pair of coils, gives the **LVDT** (linear variable differential transformer), the standard for precise, wear-free displacement measurement in structural testing, and **eddy-current** probes for non-contact proximity.

6.4.4 Piezoelectric

Measuring an impact or a vibration at kilohertz rates needs a sensor that generates its own signal, instantly, with no bridge and no supply. Certain crystals do exactly that (Figure 6.5). In quartz, and in ceramics whose internal dipoles have been permanently aligned, mechanical stress displaces charge within the lattice so that a charge Q appears across the faces, proportional to the applied force F,

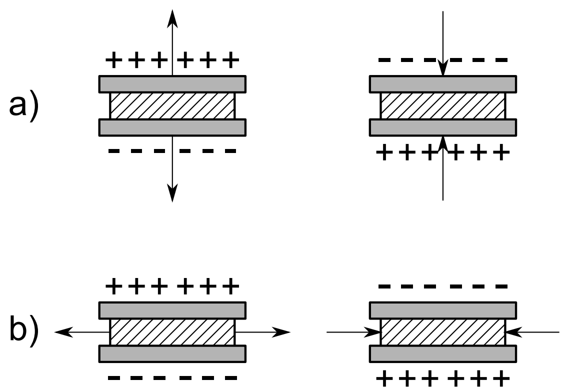
$$Q = d\,F, \qquad V = \frac{Q}{C} = \frac{d\,F}{C}, \tag{6.5}$$

where d is the piezoelectric coefficient (about 2.3 pC/N for quartz) and C the capacitance of the crystal and its wiring. Ten newtons on a quartz element with 100 pF of total capacitance gives 23 pC and 0.23 V — small, but enormous compared with the nothing that a resistive element would produce without excitation.

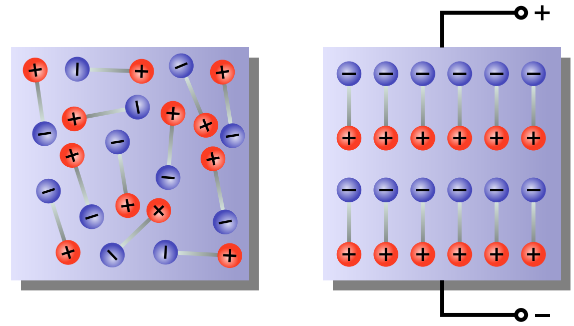
Two consequences follow directly from Equation 6.5. Because the crystal produces charge only while stress *changes*, the charge leaks away through any finite resistance, and a piezoelectric sensor cannot hold a static reading: it is blind to DC and to very slow signals. And because the effect is a lattice property with no moving parts to speak of, it is extremely fast and stiff, with resonances in the tens of kilohertz. That combination — self-generating, fast, no DC — is the signature of the piezoelectric accelerometers used for experimental modal analysis of structures. The Wikipedia article on piezoelectricity goes further into the crystal physics.

6.4.5 Thermoelectric

Some temperatures — inside a furnace, on rebar during a fire test — are beyond what a thermistor survives. A junction of two dissimilar metals produces a voltage that depends on the temperature *difference* between it and a second, reference junction: the Seebeck effect. The **thermocouple** built on it is passive, rugged, cheap, and covers ranges of well over a thousand degrees; the price is a small signal (tens of microvolts per degree), a nonlinear relation, and the fact that it only ever measures a difference, so the reference junction's temperature must be known — the *cold-junction compensation* that every thermocouple readout performs, usually with a thermistor. This is the relative sensor of Section 6.3.2.



(a) An X-cut piezoelectric plate: stretching or compressing it along either axis separates charge of opposite sign on the two faces, and the sign reverses with the direction of the force. (CLI, Wikimedia Commons, CC0.)



(b) Poling a ceramic: dipoles that are randomly oriented as manufactured (left) are aligned by a strong electric field above the Curie temperature and stay aligned once cooled (right), giving the ceramic a net piezoelectric response. (wdwd, Wikimedia Commons, CC BY-SA 3.0.)

Figure 6.5: The direct piezoelectric effect in a crystal plate, and the poling that creates it in a ceramic.

6.4.6 Optical

Copper wires pick up electrical noise and corrode, which matters when you instrument a dam or a bridge deck for decades. Optical sensors modulate light instead — its intensity, phase, wavelength, or time of flight — and light does not care about electromagnetic interference or moisture. The **fiber Bragg grating** (Wikipedia) is the structural engineer's example: a periodic pattern written into an optical fiber reflects one specific wavelength, and stretching the fiber changes the pattern's period and therefore the reflected color, so strain and temperature are read as a wavelength shift. Dozens of gratings can sit on one fiber. **LiDAR** measures distance by time of flight, **turbidity meters** measure water quality by scattered intensity, and the humble photoresistor measures light level — though, as noted, its readout is resistive.

6.4.7 One Measurand, Three Principles: The Accelerometer

The accelerometer breaks our neat one-principle-per-sensor picture, and the way it breaks it is the most useful idea in this section. You can buy piezoelectric accelerometers, piezoresistive ones, and capacitive ones. How can three different physical effects all measure the same thing?

The answer is that none of them measures acceleration directly. Every accelerometer is a *complex sensor* in Fraden's sense: a mechanical front end followed by a displacement transducer.

The front end is a *seismic mass* m attached to the case through a spring of stiffness k and a damper c (Figure 6.6). When the case accelerates with $\ddot{x}$, the mass lags behind, and its displacement z relative to the case obeys

$$m\,\ddot{z} + c\,\dot{z} + k\,z = -m\,\ddot{x}. \tag{6.6}$$

For slow enough motion the inertia and damping terms are negligible and $z \approx -(m/k)\,\ddot{x}$: the relative displacement is simply proportional to the acceleration. The job of the second stage is to read z, and any displacement transducer will do:

- a **piezoelectric** element, squeezed by the mass, producing charge (fast, high resonance, no DC — the modal-analysis instrument);
- a **piezoresistive** element, a semiconductor strain gauge on the flexure holding the mass (responds down to DC, good for slowly varying loads);
- a **capacitive** element, the mass carrying one set of plates (the ADXL335 and every MEMS accelerometer in a phone: down to DC, low power, tiny).

So the datasheet difference between a laboratory piezoelectric accelerometer for modal testing and an ADXL335 is not in the front end but in the readout — and, as Section 6.7 will show, the front end's own dynamics in Equation 6.6 set the usable frequency range of all three.

```
fig, ax = plt.subplots(figsize=(5.2, 3.4)); ax.set_aspect("equal"); ax.axis("off")
ax.plot([0, 0, 6, 6], [4, 0, 0, 4], color="k", lw=2)                        # case
ax.fill_between([-.5, 6.5], -.8, 0, color="0.8", hatch="///", edgecolor="0.5", lw=0)
ax.text(3, -1.75, "structure surface (moves with x)", ha="center", fontsize=8)
ys = np.linspace(0, 2.2, 12)                                                 # spring
xs = 1.6 + 0.25 * np.where(np.arange(12) % 2 == 0, 1, -1); xs[0] = xs[-1] = 1.6
ax.plot(xs, ys, color="k"); ax.text(0.7, 1.1, "k", fontsize=12)
ax.plot([4.4, 4.4], [0, 0.9], color="k"); ax.plot([4.1, 4.7], [0.9, 0.9], color="k")   # dam
ax.plot([4.05, 4.05, 4.75, 4.75], [1.5, 0.75, 0.75, 1.5], color="k"); ax.plot([4.4, 4.4], [1
ax.text(5.05, 1.1, "c", fontsize=12)
ax.add_patch(plt.Rectangle((1.2, 2.2), 3.6, 1.0, facecolor="#9ecae1", edgecolor="k"))   # mas
ax.text(3, 2.7, "seismic mass  m", ha="center", va="center", fontsize=10)
ax.add_patch(plt.Rectangle((2.75, 0.2), 0.5, 2.0, facecolor="#fdd0a2", edgecolor="k"))  # tra
ax.annotate("displacement transducer", (3.0, 0.25), xytext=(3.0, -1.2), ha="center", fontsiz
ax.annotate("", (6.6, 1.2), (6.6, 0.0), arrowprops=dict(arrowstyle="<->")); ax.text(6.75, 0.
ax.annotate("", (5.6, 3.4), (5.6, 2.2), arrowprops=dict(arrowstyle="<->")); ax.text(5.75, 2.
ax.set_xlim(-0.6, 8.2); ax.set_ylim(-2.0, 3.8)
plt.show()
```

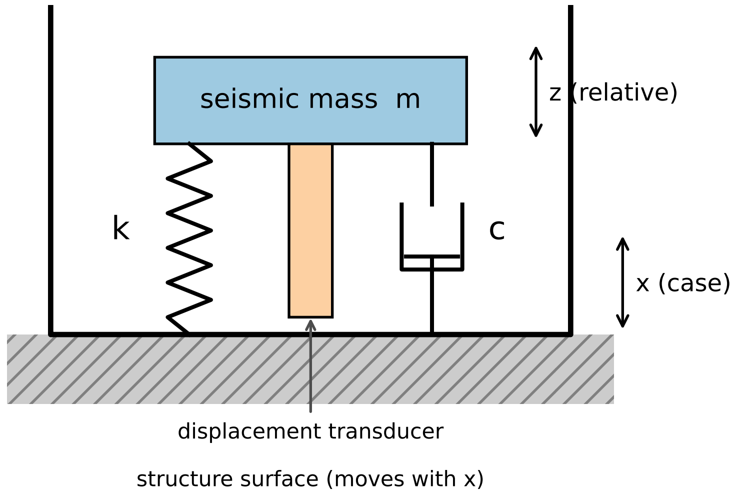


Figure 6.6: The seismic-mass model behind every accelerometer: a mass on a spring and damper inside a case that moves with the structure. The displacement transducer (piezoelectric, piezoresistive, or capacitive) reads the relative displacement z.

Table 6.2: The six transduction principles at a glance.

Principle	Governing relation	What changes	Passive / active	Canonical CEE sensor	Typical measurand
Resistive	$R = \rho\, l/a$; $R = R_0[1 + \alpha\Delta T]$	resistance	active	foil strain gauge; RTD/thermistor; photoresistor	strain; temperature; light
Capacitive	$C = \varepsilon_0\varepsilon_r A/d$	capacitance	active	MEMS accelerometer; polymer humidity sensor	acceleration; humidity; level
Inductive	$I_s = I_p/N$ (Faraday)	current / inductance	passive (CT), active (LVDT)	split-core CT; LVDT	AC current; displacement
Piezoelectric	$Q = d\, F$	charge	passive	piezo accelerometer; dynamic force sensor	vibration; impact; dynamic pressure

Principle	Governing relation	What changes	Passive / active	Canonical CEE sensor	Typical measurand
Thermoelectric	Seebeck voltage $\propto \Delta T$	voltage	passive	thermocouple	high / wide-range temperature
Optical	e.g. Bragg wavelength shift	light intensity / wavelength / time of flight	active (needs a source)	fiber Bragg grating; LiDAR; turbidity meter	strain; distance; water quality

Active Learning: Classify Six Sensors

Six sensors, four of which you have only seen mentioned so far — here is what each one is, in a line:

- **Thermocouple** — two wires of different metals welded together at the tip; the voltage between the free ends depends on the tip temperature (background; NIST reference tables).
- **Pt100 RTD** — a thin platinum resistor trimmed to exactly 100 Ω at 0 °C (background; Vishay PTS datasheet).
- **LVDT** — a movable iron core sliding inside a primary coil and two secondary coils; the core's position sets how the AC excitation couples into each secondary (how it works; background).
- **DHT11** — a hobbyist temperature-and-humidity module: a capacitive humidity element and a thermistor read by a tiny microcontroller that outputs digital values (datasheet).
- **Laboratory piezoelectric accelerometer** — e.g. the PCB 352C33: a ceramic shear element under a seismic mass, with built-in electronics that convert charge to voltage (spec sheet).
- **Fiber Bragg grating strain sensor** — e.g. the Luna os3100: an FBG held in tension in a steel carrier that is welded or glued to the structure (product page).

Now, *before* opening the links, classify each along both axes — device kind (passive/active, absolute/relative, from Section 6.3.2) and transduction principle (from Table 6.2). Then open each datasheet and check your answers against what it says about excitation and output. One of the six is a trap: its transducer is passive, but the electronics packaged with it need a supply — decide how you want to classify that.

Concept Check: Which one needs an excitation source?

Recall the two temperature sensors of Section 6.4: a *thermocouple* is a junction of two dissimilar metals that produces a Seebeck voltage, and an *RTD* is a platinum resistor whose resistance follows Equation 6.2. Both measure temperature. Which of them needs an excitation source to produce an output, and what does that tell you about the physics each relies on?

Answer

The RTD. Its output is a *resistance*, and a resistance is not a signal until you pass a current through it — it is an active (parametric) sensor built on Equation 6.2. The thermocouple *generates* a voltage from the Seebeck effect with no supply at all — it is passive. The general rule: sensors whose governing relation ends in a resistance, capacitance, or inductance need excitation; those whose relation ends in a charge or a voltage do not.

6.5 Reading a Datasheet: Static Characteristics

Open the ADXL335 and CR3110-3000 sheets you found in Section 6.3.3. Between them they use every term in this section's learning objective — sensitivity, range, resolution, nonlinearity, hysteresis, drift — plus accuracy and repeatability, and they use them in slightly different ways, because each manufacturer decides how to present its own numbers. Choosing the CT for a submetering job, or the accelerometer for a footbridge, is impossible without knowing what those numbers promise and, just as important, what they leave out. The key to reading any datasheet is that all of its numbers describe *one object*.

6.5.1 The Transfer Function

That object is the **transfer function**: the relation between the stimulus s and the sensor's electrical response S,

$$S = f(s), \tag{6.7}$$

in Fraden's notation (Fraden 2016, Ch. 2). For an ideal sensor, f is a known curve — a formula from the physics, a calibration table, or a fitted polynomial — and if you know f you can invert it to recover the stimulus from what you measured: $s = f^{-1}(S)$. In practice we usually want f to be a straight line,

$$S = S_0 + B\,(s - s_0), \tag{6.8}$$

because a line has only two parameters, needs only two calibration points, and inverts trivially. Most of the datasheet is then either a property of that line or a bound on how far the real sensor strays from it. Figure 6.7 puts every term of this section on one picture; the subsections that follow just walk around it.

```
s = np.linspace(0, 10, 400)
B, S0 = 10.0, 0.0                                           # ideal: 10 mV per mm
ideal = S0 + B * s
bend = 22.0 * np.sin(np.pi * s / 10)                        # nonlinearity, exaggerated to be legible
loop = 9.0 * np.sin(np.pi * s / 10)                         # half-width of the hysteresis loop, exagg
real_up = ideal + bend - loop                               # loading sweep
real_dn = ideal + bend + loop                               # unloading sweep
drift = ideal + 16.0
fig, ax = plt.subplots(figsize=(6.8, 4.8))
ax.plot(s, ideal, "k--", lw=1.2, label="ideal line (slope = sensitivity $B$)")
ax.plot(s, real_up, color="C0", label="real curve, loading")
ax.plot(s, real_dn, color="C0", ls=":", label="real curve, unloading")
ax.plot(s, drift, color="C3", lw=1, alpha=0.7, label="same sensor after drift")
# nonlinearity: vertical gap between the ideal line and the mid-loop curve at s = 7
i7 = 280
ax.annotate("", (7, ideal[i7]), (7, ideal[i7] + bend[i7]), arrowprops=dict(arrowstyle="<->",
ax.text(7.2, ideal[i7] + bend[i7] / 2, "  nonlinearity", color="C2", fontsize=9, va="center"
# hysteresis: horizontal gap between loading and unloading at S = 35 mV
i_up = np.argmin(abs(real_up - 40)); i_dn = np.argmin(abs(real_dn - 40))
ax.annotate("", (s[i_up], 40), (s[i_dn], 40), arrowprops=dict(arrowstyle="<->", color="C1"))
ax.text((s[i_up] + s[i_dn]) / 2, 31, "h  hysteresis", color="C1", ha="center", fontsize=9)
# range
ax.axvline(0, color="0.6", lw=0.8); ax.axvline(10, color="0.6", lw=0.8)
ax.annotate("", (0, 136), (10, 136), arrowprops=dict(arrowstyle="<->", color="0.4"))
ax.text(5, 138, "range (span, full scale)", ha="center", fontsize=9, color="0.3")
# resolution: staircase near the origin
ss = np.linspace(0, 1.2, 200); ax.step(ss, np.floor(B * ss / 2) * 2, where="post", color="C4
ax.text(1.35, 3, "resolution\n(smallest step)", color="C4", fontsize=8, va="center")
ax.annotate("drift", (8.5, drift[340]), xytext=(6.2, 124), color="C3", fontsize=9, arrowprops
ax.set_xlabel("stimulus $s$ (mm)"); ax.set_ylabel("output $S$ (mV)"); ax.set_xlim(-0.3, 10.8
ax.legend(loc="lower right", fontsize=8)
plt.show()
```

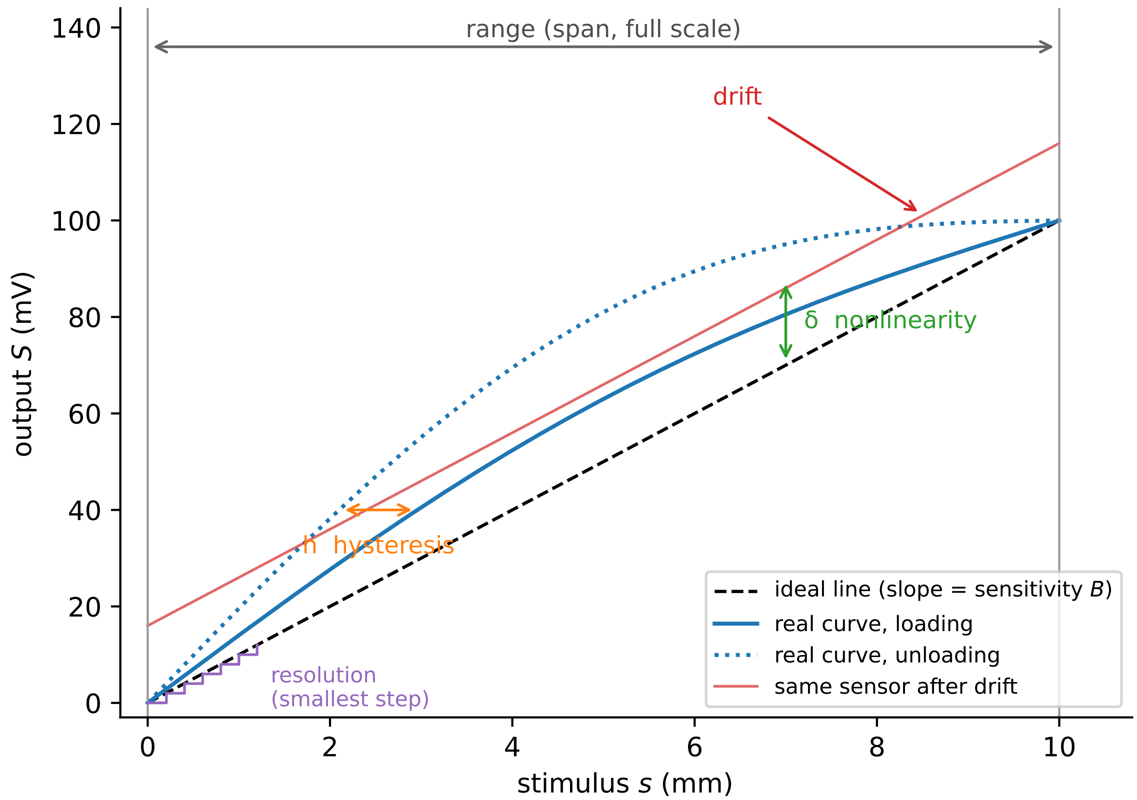


Figure 6.7: One transfer function carrying every static characteristic (departures exaggerated for legibility): the ideal line and its slope (sensitivity) over the guaranteed span (range); a real curve that departs from the line (nonlinearity δ), that differs between loading and unloading (hysteresis h), that is only resolved in finite steps (resolution), and that shifts as a whole with time or temperature (drift).

6.5.2 Sensitivity, Range, and Resolution

Before you buy a sensor you must answer three questions: how much output do I get per unit of input, over what span is that guaranteed, and how small a change can I actually see?

Sensitivity is the slope B of the transfer function — output per unit stimulus. The ADXL335 quotes 300 mV/g (typical) at a 3 V supply, with a guaranteed window of 270–330 mV/g, i.e. ±10% part-to-part. Read that tolerance twice: unless you calibrate each device, a 1 g reading could be off by 0.1 g from sensitivity alone, which as we will see dwarfs every other error on the sheet. The CT expresses the same idea as a ratio: 1 A of primary current gives 1/3100 A in the secondary, so with a 100 Ω burden the sensitivity is 32 mV/A. For nonlinear sensors the sensitivity is the local slope $\mathrm{d}S/\mathrm{d}s$ and varies with the stimulus — the photoresistor's resistance-versus-lux curve is one, which is why its sheet gives a *gamma* (the log-log slope between 10 and 100 lux) rather than a single number.

Range (span, or *input full scale*) is the stimulus interval over which the datasheet's promises hold. The ADXL335 guarantees ±3 g; the CT senses up to 75 A linearly and tolerates four times that briefly. The corresponding output interval is the *full-scale output* (FSO): ±3 g × 300 mV/g = ±0.9 V about the 1.5 V zero-g level, i.e. 0.6–2.4 V, safely inside the 0–3 V supply rails. Push past the range and most sensors *saturate* — the output stops responding — which is what the CT's "maximum voltage" specification is warning about.

Resolution is the smallest change in stimulus that produces a distinguishable change in output. For a digital sensor it is set by the number of bits; for an analog one it is usually set by noise. The ADXL335 does not quote a resolution directly; it quotes a *noise density*, 150 µg/√Hz on the X and Y axes, from which the resolution follows once you decide the bandwidth: over 100 Hz, $150\ \mu\text{g} \times \sqrt{1.6 \times 100\ \text{Hz}} \approx 1.9$ mg rms (the factor 1.6 accounts for the filter's shape). Lecture 4 will revisit this from the converter's side.

6.5.3 Accuracy and Repeatability

A sensor that reports six decimal places can still be wrong by a full percent, and a sensor that is always wrong by the same amount is more useful than one that is right on average but scatters. These two sentences are the whole content of the most-confused pair of terms on any datasheet.

Accuracy (Fraden points out it really means *inaccuracy*) is the largest deviation between the value a sensor reports and the true value of the stimulus, including *every* contributing error: calibration, nonlinearity, hysteresis, drift, part-to-part variation. It is quoted in the units of the stimulus (±0.5 °C), as a percentage of full scale (±2% FS), or as a percentage of the reading. Many sheets — the ADXL335's among them — never quote a single accuracy number at all; they quote the components and leave the combination to you.

Repeatability is the spread of outputs for repeated, identical stimuli under identical conditions — the width of the cloud, not its distance from the truth. It is a component of accuracy but is often quoted separately because it is the one error you cannot calibrate away: a systematic offset can be measured and subtracted, random scatter cannot. And **resolution**, from the previous subsection, is a third, independent quantity: the granularity of the output says nothing about whether the value is right. A digital scale reading to 0.01 g can be repeatable to 0.01 g and inaccurate by 5 g if it was calibrated with the wrong reference weight.

Concept Check: Which sensor do you choose?

Three displacement sensors are on offer for monitoring the slow settlement of a foundation over a year. Sensor A resolves 1 µm but its readings scatter by ±50 µm when the same displacement is repeated. Sensor B resolves only 20 µm but repeats to ±20 µm and is accurate to ±30 µm. Sensor C resolves 5 µm, repeats to ±5 µm, and is accurate to ±200 µm because its factory calibration is poor. Which do you choose, and why does the one with the most decimal places not win?

Answer

Sensor C, provided you calibrate it yourself once against a reference — its poor accuracy is a *systematic* error, which one calibration removes, leaving ±5 µm repeatability. Sensor A's fine resolution is useless because its ±50 µm scatter is random and cannot be removed. Sensor B is the safe choice if you cannot calibrate. Resolution tells you the granularity of the number, repeatability tells you how much of it is noise, accuracy tells you how far it may be from the truth — and only the middle one is beyond your control.

6.5.4 Linearity, Hysteresis, and Drift

The catalog assumes the ideal line of Equation 6.8, but your bridge sensor returns different readings on the way up and on the way down, and different readings in July than in January. Three numbers on the datasheet bound those departures.

Nonlinearity is the maximum deviation of the real transfer function from the straight line used to approximate it, usually quoted as a percentage of full scale. The ADXL335 claims ±0.3% FS, which on a 6 g span is 18 mg. Beware that the number depends on *which* line: through the end points, the best-fit line, or a line tangent at a calibration point can give quite different nonlinearity figures for the same sensor, and a manufacturer will naturally publish the smallest. If the sheet does not say which line, the number is not fully meaningful.

Hysteresis is the difference in output at the same stimulus depending on the direction from which it was approached — the loop between the loading and unloading curves in Figure 6.7. Friction, mechanical play, and material memory cause it; it is common in anything with a moving part or a polymer in it, and it is usually quoted as a percentage of full scale.

Drift is the slow change of the transfer function with time or with an environmental variable, most often temperature. Short-term drift over minutes to days is effectively a form of repeatability error; long-term drift over months is aging and is usually one-directional. Temperature drift is quoted per degree: the ADXL335 lists a zero-g offset drift of ±1 mg/°C and a sensitivity drift of ±0.01%/°C. Over a 25 °C swing, the offset alone moves 25 mg — more than the nonlinearity.

Putting these together into a statement about one reading is the business of Lecture 7, but you can already see the shape of it: at 1 g on an uncalibrated ADXL335, the sensitivity tolerance contributes up to 100 mg, the nonlinearity 18 mg, and a 25 °C temperature excursion 25 mg. The lesson generalizes far beyond this part: **the biggest error term is usually the one that calibration would remove**, which is why instruments get calibrated.

Figure 6.8 does the datasheet's job in reverse: given a calibration sweep of a fictional displacement sensor — up and down, repeated — it extracts the sensitivity, the nonlinearity, the hysteresis, and the repeatability, exactly as a manufacturer's test bench would.

```
rng = np.random.default_rng(12778)
s_pts = np.linspace(0, 10, 11)
B_true, hyst_true, noise = 10.0, 1.0, 0.3         # mV/mm, mm, mV (departures exaggerated fo
def true_curve(s, direction):                      # direction: +1 loading, -1 unloading
    return B_true * s + 7.0 * np.sin(np.pi * s / 10) - direction * B_true * hyst_true / 2 * 
runs_up = np.array([true_curve(s_pts, +1) + rng.normal(0, noise, s_pts.size) for _ in range(
runs_dn = np.array([true_curve(s_pts, -1) + rng.normal(0, noise, s_pts.size) for _ in range(
S_all = np.concatenate([runs_up, runs_dn]); s_all = np.tile(s_pts, 6)
B_fit, A_fit = np.polyfit(s_all, S_all.ravel(), 1)
fs_out = B_fit * 10
nonlin = np.max(np.abs(S_all.mean(axis=0) - (A_fit + B_fit * s_pts))) / fs_out * 100        
hyst_mV = np.max(np.abs(runs_dn.mean(axis=0) - runs_up.mean(axis=0)))
repeat = max(runs_up.std(axis=0, ddof=1).max(), runs_dn.std(axis=0, ddof=1).max())
fig, ax = plt.subplots(figsize=(6.2, 4))
for r in runs_up: ax.plot(s_pts, r, "o", color="C0", ms=3, alpha=0.6)
for r in runs_dn: ax.plot(s_pts, r, "s", color="C1", ms=3, alpha=0.6)
ax.plot(s_pts, runs_up.mean(axis=0), color="C0", label="loading (mean of 3)")
ax.plot(s_pts, runs_dn.mean(axis=0), color="C1", label="unloading (mean of 3)")
ax.plot(s_pts, A_fit + B_fit * s_pts, "k--", label=f"best fit: {B_fit:.2f} mV/mm")
ax.set_xlabel("displacement (mm)"); ax.set_ylabel("output (mV)"); ax.legend()
ax.set_title(f"nonlinearity {nonlin:.1f}% FSO · hysteresis {hyst_mV/B_fit:.2f} mm · repeatab
plt.show()
```

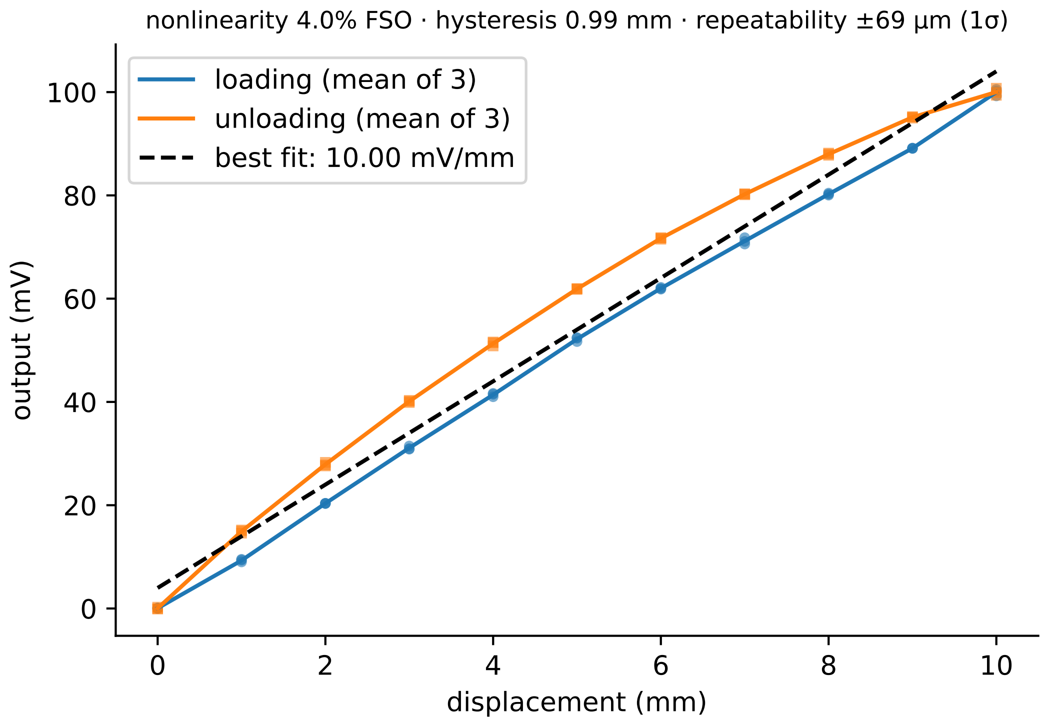


Figure 6.8: A synthetic calibration record for a displacement sensor (nominal 10 mV/mm, 0–10 mm; departures exaggerated for legibility): three loading/unloading cycles with noise. The best-fit line gives the sensitivity; the largest departure from it the nonlinearity; the loading–unloading gap the hysteresis; the scatter of repeated points the repeatability.

Active Learning: One Sheet, Three Jobs

Take the ADXL335 datasheet and decide, with numbers pulled from the sheet, whether the part can serve each of these three jobs: (a) detecting the ~2 Hz sway of a pedestrian bridge with ±10 mg amplitude; (b) measuring the tilt of a retaining wall to 0.1° (hint: tilt is a change in the static 1 g reading — how many mg is 0.1°?); (c) logging truck-induced vibration on a highway overpass at up to 500 Hz. For each, name the single characteristic that decides the answer. Then do the same for the CR3110-3000 with: (d) monitoring a 20 A single-phase circuit; (e) a 200 A service entrance; (f) a DC solar array.

6.6 Matching Sensors to CEE Measurands

Your final project will require picking real sensors for real measurands, and the catalog offers a dozen ways to measure any of them. But the previous three sections already gave you the filters. Sensor selection is a short sequence of questions, each of which points back to something you now know:

1. **What quantity, exactly?** Strain or displacement? Acceleration or tilt? Current or power? (Section 6.3 — name the measurand before the sensor.)
2. **Static or dynamic?** Do you need DC response (settlement, tilt, temperature) or only changes (vibration, impact)? This alone rules piezoelectric sensors in or out (Section 6.4).
3. **What range and resolution?** Full scale with margin, and the smallest change that matters — read against the sheet's sensitivity, span, and noise (Section 6.5).
4. **What environment?** Temperature swings (drift), moisture and corrosion (optical wins), electromagnetic interference (optical again), access for recalibration.
5. **What installation, power, and budget?** Passive or active, wired or wireless, how many channels, and what data-acquisition system it must talk to (Lecture 4). ### Modalities by Application Domain

Project teams think in terms of a system to instrument rather than a list of measurands, so Table 6.3 is organized the way you will use it. Each cell names the two or three modalities that dominate practice, by the principle we have studied, and why they won.

Table 6.3: Dominant sensing modalities by application domain.

Domain	Measurand	Dominant modalities (principle)	Why they won
Structural health monitoring	strain	foil strain gauge (resistive); fiber Bragg grating (optical)	gauges are cheap and well understood; FBGs multiplex on one fiber and ignore EMI for long-term deployments
	displacement	LVDT (inductive); laser/optical; GNSS at bridge scale	LVDT is wear-free and precise; optical for non-contact
	acceleration	MEMS capacitive (e.g. ADXL335) for ambient vibration; piezoelectric for modal testing	MEMS: DC-capable, cheap, low power; piezo: high bandwidth, self-generating

Domain	Measurand	Dominant modalities (principle)	Why they won
Building energy	temperature	thermistor / RTD (resistive); thermocouple for extremes	thermistors are cheap and sensitive; RTDs stable; thermocouples rugged
	humidity	capacitive polymer sensor	low power, adequate accuracy, integrates with temperature on one chip
	electrical current / power	split-core CT (inductive); Hall-effect for DC	CTs clamp on without rewiring; Hall sensors see DC
Transportation	vehicle presence / count	inductive loop; piezoelectric strip; camera / LiDAR (optical)	loops are embedded and durable; piezo strips give axle timing; optical adds classification
	pavement / bridge response	strain gauge, accelerometer (as above)	—
Water (brief)	flow	electromagnetic, ultrasonic, differential pressure	no moving parts (EM, ultrasonic) or simplicity (ΔP)
	water quality	electrochemical probes (pH, DO); optical turbidity	each parameter has its own dedicated chemistry or optics

Exercise: A Pedestrian Bridge Campaign

A 60 m steel pedestrian bridge is to be monitored for one year to characterize pedestrian-induced vibration (expected 1–3 Hz, up to 0.1 g), thermal expansion at the bearings (±15 mm over the year), and strain at two welded connections. The budget is modest, mains power is available at one abutment only, and the deck is exposed to weather. Specify a sensor for each of the three measurands and justify each choice against the five selection questions.

One defensible answer

Vibration: a MEMS capacitive accelerometer (ADXL335-class or a digital successor): the 1–3 Hz, 0.1 g signal is well inside its range and bandwidth, it responds down to DC so the mounting orientation can be checked, and it runs on milliwatts — decisive given power is only available at one end. A piezoelectric accelerometer would need a powered charge amplifier and offers bandwidth you do not need. *Bearing displacement:* an LVDT or a draw-wire potentiometer: ±15 mm is a large, slow, DC displacement, which rules out anything piezoelectric and favors a robust contact sensor; both need excitation, so run them from the powered abutment. *Strain:* foil strain gauges in a bridge configuration are the budget answer, with temperature-compensating dummy gauges since the deck is exposed; if the budget stretched and EMI or long cable runs were a concern, FBGs on a single fiber would be the premium choice.

6.7 How Transducer Physics Shapes Your Data

An anomaly detector once flagged a "sudden" temperature spike in a plant room that never happened. The probe had lagged behind a real, slow change and then caught up; the model, trained on the recorded values, learned the sensor rather than the room. Nothing in the data said so. This section is about the ways the physics that makes a transducer work also limits it — its own dynamics, its interaction with the thing measured, and its response to stimuli you never meant to measure — and about why those limits reappear downstream as artifacts a model will cheerfully fit.

6.7.1 Dynamics and Bandwidth

Everything in Section 6.5 quietly assumed the stimulus held still — that is why those characteristics are called *static*. Drop a thermometer into hot water and it does not read the true temperature instantly, because heat must flow into it first. The accelerometer's seismic mass, likewise, cannot follow a vibration faster than its own spring lets it. When the stimulus varies in time, the sensor's response generally does not follow with perfect fidelity, and the difference is the *dynamic error*.

Fraden classifies sensors by the order of the differential equation that links output to input. A **zero-order** sensor responds instantly (a potentiometer, near enough). A **first-order** sensor has one energy store — the heat capacity of a thermometer — and obeys

$$\tau \frac{\mathrm{d}y}{\mathrm{d}t} + y = x, \tag{6.9}$$

where x is the true stimulus, y the reading, and τ the *time constant*. After a step change in x, the reading covers 63.2% of the change at $t = \tau$ and 95% at $t = 3\tau$; equivalently, the sensor attenuates sinusoidal stimuli above a cutoff frequency $f_c \approx 0.159/\tau$. A DHT11 humidity module quotes a 6 s response time in moving air; a sheathed thermocouple in still air can have τ of tens of seconds. Figure 6.9 is the plant-room story from the opening of this section, simulated: the same first-order probe following a step and a slow ramp, and a naive detector misreading the lag.

```
from scipy.integrate import solve_ivp
tau = 30.0
t = np.linspace(0, 600, 1201)
x_step = np.where(t >= 60, 1.0, 0.0)
x_ramp = np.clip((t - 60) * (2.0 / 300), 0, 2.0)
def respond(x_fun):
    sol = solve_ivp(lambda tt, y: (np.interp(tt, t, x_fun) - y) / tau, [0, 600], [0.0], t_eva
    return sol.y[0]
y_step, y_ramp = respond(x_step), respond(x_ramp)
lag_ss = tau * (2.0 / 300)
fig, axs = plt.subplots(3, 1, figsize=(6.4, 6.2), sharex=True)
axs[0].plot(t, x_step, "k", label="true (step)"); axs[0].plot(t, y_step, "C0", label="reading
axs[0].axvline(60 + tau, color="0.7", ls=":"); axs[0].text(60 + tau + 3, 0.2, "t =  : 63%", f
axs[0].set_ylabel("°C above start"); axs[0].legend(loc="lower right", fontsize=8)
axs[1].plot(t, x_ramp, "k", label="true (ramp)"); axs[1].plot(t, y_ramp, "C0", label="reading
axs[1].annotate("", (200, x_ramp[400]), (200, y_ramp[400]), arrowprops=dict(arrowstyle="<->"
axs[1].text(212, y_ramp[400] - 0.12, f"lag =  ·slope = {lag_ss:.2f} °C", color="C3", fontsize
axs[1].set_ylabel("°C above start"); axs[1].legend(loc="lower right", fontsize=8)
rate_true = np.gradient(x_ramp, t) * 60; rate_read = np.gradient(y_ramp, t) * 60
thr = 0.2   # °C per minute
axs[2].fill_between(t, 0, 1, where=rate_true > thr, color="k", alpha=0.15, label="true rate 
axs[2].fill_between(t, 0, 1, where=rate_read > thr, color="C3", alpha=0.35, label="detector 
axs[2].set_yticks([]); axs[2].set_xlabel("time (s)"); axs[2].legend(loc="upper right", fonts
plt.tight_layout(); plt.show()
```

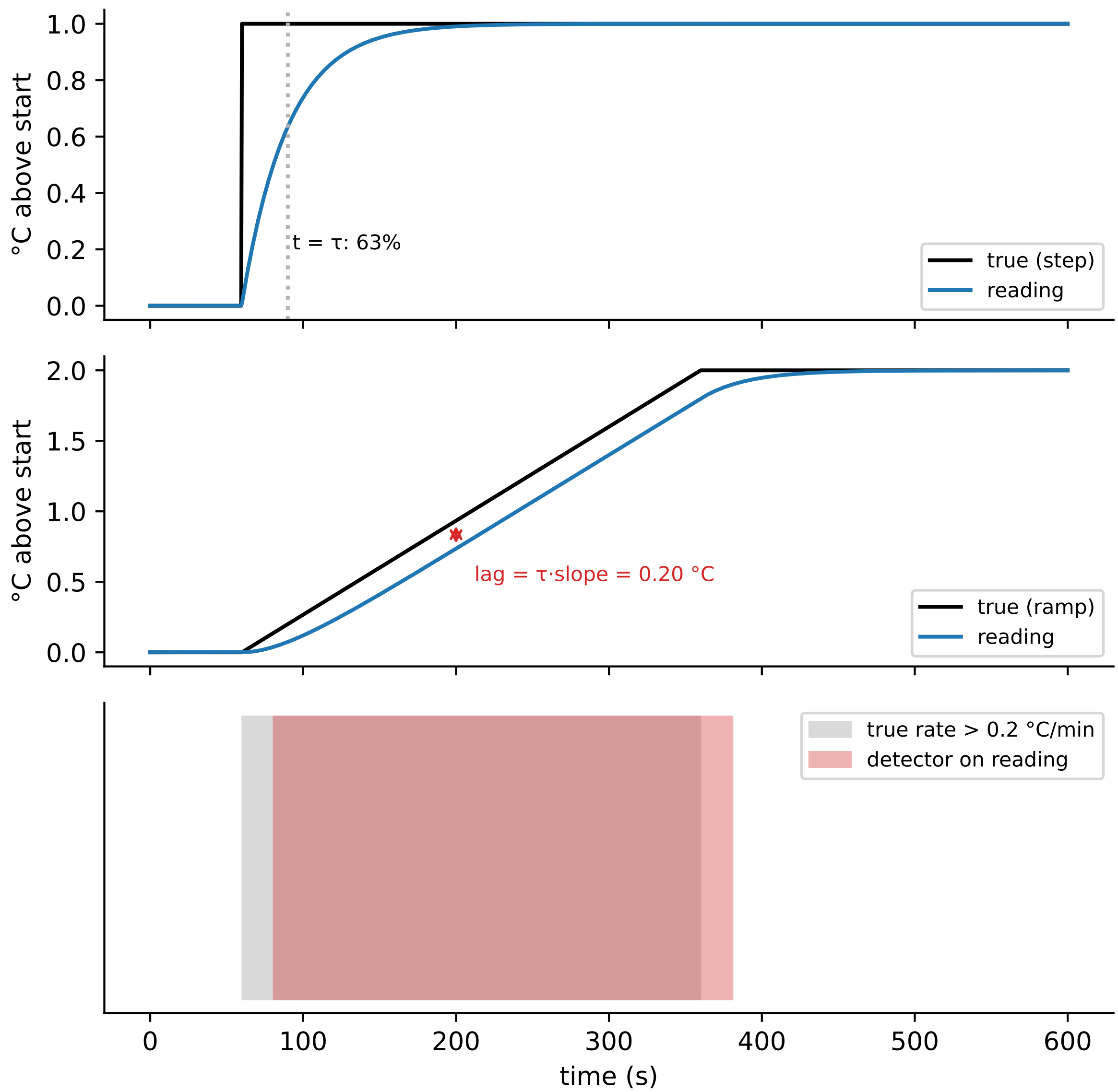


Figure 6.9: A first-order temperature probe ($\tau = 30$ s) following a step and a 2 °C ramp over 5 minutes. During the ramp the reading lags the truth by a constant $\tau\times$ slope $= 0.2$ °C; after the ramp ends it keeps rising for about 3τ. A detector thresholding the reading's rate of change (bottom) turns on and off about $\tau \ln 2 \approx 21$ s late — it is detecting the sensor, not the room.

A **second-order** sensor has two energy stores — the mass and the spring of Equation 6.6 — and adds a *resonance*: a frequency $f_n = \frac{1}{2\pi}\sqrt{k/m}$ near which the response is amplified rather than attenuated, by roughly $1/(2\zeta)$ for damping ratio ζ. Below about 20% of f_n the response is flat to within a few percent — that is the usable band in which $z \propto \ddot{x}$ holds — and above f_n it rolls off steeply. Figure 6.10 shows the shape; the derivation belongs to Lecture 6, but the consequence is immediate. A laboratory piezoelectric accelerometer for modal testing typically resonates above 30 kHz, giving a flat band to several kilohertz; a MEMS part resonates in the low kilohertz, and the ADXL335 additionally lets you *choose* its bandwidth with an external capacitor, $f = 1/(2\pi \cdot 32\ \mathrm{k}\Omega \cdot C)$ — 1.6 kHz with 3 nF, 500 Hz with 10 nF, and 0.5 Hz with 10 µF. Set 0.5 Hz because you only care about tilt, and the bridge vibration in your data simply vanishes — not because it stopped, but because you told the sensor not to see it. And on the other edge: the piezoelectric sensor's blindness to DC (Section 6.4) is a *lower* cutoff of the same band.

```
r = np.logspace(-2, 1, 500)                       # f / f_n
fig, ax = plt.subplots(figsize=(6.2, 3.8))
for zeta in (0.05, 0.3, 0.7):
    H = 1 / np.sqrt((1 - r**2) ** 2 + (2 * zeta * r) ** 2)
    ax.loglog(r, H, label=f"  = {zeta}")
ax.axvspan(1e-2, 0.2, color="C2", alpha=0.12); ax.text(0.03, 0.55, "usable band\n(flat withi
ax.axvline(1, color="0.6", ls=":"); ax.text(1.05, 0.25, "resonance $f_n$", fontsize=8, color
ax.set_xlabel("frequency / resonant frequency"); ax.set_ylabel("normalized response"); ax.le
plt.show()
```

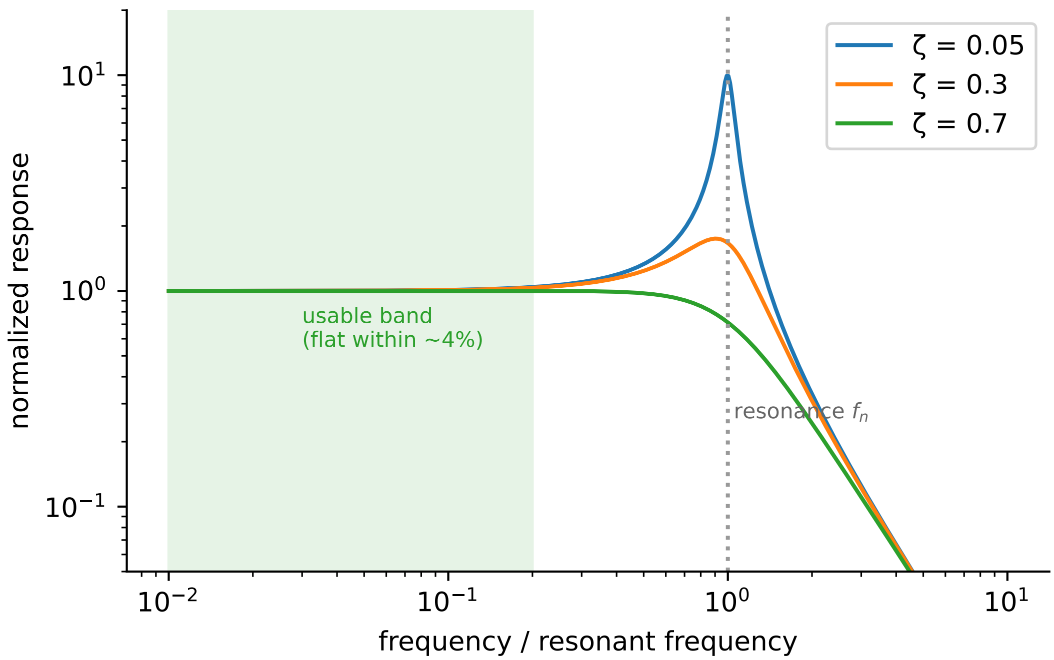


Figure 6.10: Magnitude of the seismic-mass response (relative displacement per unit base acceleration, normalized to its low-frequency value) for three damping ratios. The flat region is the usable band; the resonance amplifies by about $1/(2\zeta)$; above it the sensor rolls off. Presented, not derived — see Lecture 6.

6.7.2 Loading and Cross-Sensitivity

A heavy accelerometer bolted to a light beam changes how the beam vibrates. A flow meter inserted in a pipe changes the flow it is there to measure. A thermocouple wicks heat away from the point whose temperature it reports. Attaching a sensor always changes the system, an effect called **loading**, and the honest question is never "does it?" but "by how much?" — a 10 g MEMS board on a bridge girder, not at all; a 100 g piezoelectric accelerometer on a laboratory-scale model, possibly a great deal. Active sensors add a loading of their own: the excitation current that a thermistor needs also *heats* it, so a thermistor in still air reads its own self-heating on top of the air temperature, which is why datasheets specify a maximum excitation current in air separately from that in water.

Most transducers also answer to more than one stimulus, which is called **cross-sensitivity**. The strain gauge's resistance follows Equation 6.1 *and* Equation 6.2 at the same time, so a gauge on a steel beam in the sun reads thermal expansion and the resistivity change of its own foil as if they were strain — the reason strain-gauge installations carry temperature-compensating dummy gauges. The GL5528's resistance depends on temperature as well as light. The ADXL335's sheet quotes a ±1% *cross-axis sensitivity* (a Y acceleration leaking into the X output) and the temperature drifts we met in Section 6.5. And the current transformer's cross-sensitivity is a blind spot rather than a leak: it cannot see DC, and it attenuates harmonics above 1 kHz, so a distorted waveform is reported as smoother and smaller than it is. The honest transfer function is therefore

$$S = f(s, T, \ldots), \tag{6.10}$$

a function of the stimulus *and* of the environment, and a datasheet is largely a list of the partial derivatives with respect to the variables you did not intend to measure.

6.7.3 Consequences for Data and Models

None of these effects announce themselves in a CSV file. A first-order lag looks like a slow event; a resonance looks like a strong signal at f_n that is really the sensor ringing; a cross-sensitivity looks like a genuine correlation between temperature and strain that a regression model will happily learn and a physicist will immediately distrust. And none of it can be undone after the fact without knowing the physics: you can deconvolve a lag only if you know τ, and you can separate thermal from mechanical strain only if you know the gauge's temperature coefficient and its temperature.

This is the reason the course insists on understanding the chain *before* modeling. Lecture 7 will turn the datasheet bounds of Section 6.5 into a formal uncertainty on each reading; the DATA third will show how metadata — which sensor, which bandwidth setting, which calibration — is what lets a dataset be interpreted at all; and the MODELS third will repeatedly ask whether a pattern a model found is in the world or in the instrument. The three lenses of this lecture — the chain, the principle, the characteristics — are how you tell.

Concept Check: Which accelerometer for a tilting wall?

Two accelerometers have identical sensitivity and range. One is piezoelectric, the other capacitive MEMS. Which do you mount on a retaining wall that is suspected of tilting slowly over months, and why?

Answer

The capacitive MEMS device. A slow tilt is a change in the *static* component of gravity along the sensor's axis — a DC signal — and a piezoelectric sensor cannot hold a static reading (Equation 6.5: charge appears only while stress changes, and leaks away). The MEMS part responds down to DC, so 0.1° of tilt shows up as a persistent shift of about 1.7 mg in the reading along a horizontal axis. The two sheets' identical sensitivity and range say nothing about this; the transduction principle says everything.

6.8 Putting It All Together

An instrument is a transducer surrounded by a chain of supporting links. The transducer works by one of a handful of physical effects, and the effect predicts the device's behavior — speed, power, DC response, environmental sensitivity — before you open its datasheet. The datasheet then describes the transfer function that the effect produces and bounds how far a real device strays from it. And those same physics set what the data downstream can and cannot show, through dynamics, loading, and cross-sensitivity that no later processing can fully remove. The next lectures follow the chain link by link — circuits, data acquisition, signal conditioning, frequency-domain analysis, uncertainty — and the DATA and MODELS thirds inherit exactly what the physics let through, no more.

6.9 Exercises

1. Pick an everyday instrument other than the ruler or scale used above, and describe how it was designed, how it could be calibrated, and at least three ways it could drift or fail.

Solution sketch

A tire pressure gauge: designed around a range of 0–60 psi (or 0–4 bar) and a resolution of 1 psi; a piston against a calibrated spring, or a strain-based element in digital models; calibrated at the factory against a reference pressure standard (a deadweight tester). Failure modes: spring fatigue (sensitivity drift), dirt in the piston (hysteresis and dead band), temperature (the spring's stiffness and the gas law both change), and a leaking seal (an offset that grows during the reading). Each maps onto a term in Section 6.5.

2. For the CR3110-3000 current transformer, draw its measurement chain from the current in the conductor to a number in a database, classify it along the axes of Section 6.3.2, and name its transduction principle.

Solution

Chain: alternating current in the conductor → alternating magnetic flux in the iron ring → induced current in the 3000-turn secondary (the sensor; inductive principle, Faraday's law) → burden resistor converting current to voltage (signal conditioning) → anti-aliasing filter and analog-to-digital converter (signal processing, Lectures 4–5) → microcontroller computing an rms value and timestamping it → database row. Classification: *passive* (the energy comes from the measured circuit; no excitation), *absolute* (referenced to zero current), stimulus electric/magnetic, conversion phenomenon electromagnetic induction.

3. From the ADXL335 datasheet, extract sensitivity, range, resolution, accuracy, repeatability, nonlinearity, hysteresis, and drift, noting which are not given, and estimate the worst-case error on a 1 g reading (a) straight out of the box and (b) after a one-point gain calibration, over a 25 °C temperature swing.

Solution

Sensitivity 300 mV/g typical, 270–330 mV/g guaranteed (±10%) at 3 V; range ±3 g minimum; resolution not quoted directly — noise density 150 µg/$\sqrt{\text{Hz}}$ on X/Y gives 1.9 mg rms over a 100 Hz bandwidth; accuracy not quoted as a single number; repeatability not quoted; nonlinearity ±0.3% FS = 18 mg on a 6 g span; hysteresis not quoted (the sheet mentions temperature hysteresis only qualitatively); drift: zero-g offset ±1 mg/°C, sensitivity ±0.01%/°C. Worst case at 1 g: (a) sensitivity tolerance 100 mg + nonlinearity 18 mg + offset drift 25 mg + sensitivity drift 0.25% × 1 g = 2.5 mg 146 mg; (b) with the ±10% gain error calibrated out, 46 mg, dominated by the temperature drift of the offset. The calibration removes two thirds of the error; the remaining term is the one that argues for a temperature sensor next to the accelerometer.

4. For a monitoring scenario of your choosing in one application domain of Table 6.3, specify a sensor for each of three measurands and justify each choice against the five selection questions of Section 6.6.

Solution sketch

Follow the pattern of the pedestrian-bridge exercise: for each measurand, state the quantity precisely, decide static vs. dynamic (which eliminates or admits piezoelectric devices), read range and resolution off a candidate datasheet, name the environmental threat (temperature, moisture, EMI) and the modality that handles it, and close with power and budget. A building-energy example: room temperature (thermistor; static; 10–35 °C to 0.1 °C; indoor; battery), supply-air humidity (capacitive polymer; slow; 20–90% RH to 3%; condensation risk — pick a sealed part), and circuit current (split-core CT; AC only; 0–75 A; clamp-on with no rewiring; passive).

5. A first-order temperature sensor with $\tau = 30$ s records a room whose temperature ramps by 2 °C over 5 minutes and then holds. Sketch the recorded signal against the true one, state the lag during the ramp, and explain what a detector that flags "rate of change above 0.2 °C/min" would report.

Solution

During the ramp the reading approaches a constant lag of $\tau \times$ slope $= 30\text{ s} \times (2\,^\circ\text{C}/300\text{ s}) = 0.2\,^\circ\text{C}$ behind the truth (reached after a transient of about $3\tau = 90$ s from the start of the ramp); after the ramp stops, the reading keeps rising and closes the gap over another $\approx 3\tau$. The true rate of change is $0.4\,^\circ\text{C/min}$ throughout the ramp and zero afterward. The detector on the *reading* turns on late — the recorded rate rises from zero with time constant τ and crosses 0.2 °C/min at $t = \tau \ln 2 \approx 21$ s after the ramp begins — and turns off late by the same $\tau \ln 2$, since the recorded rate decays with the same time constant after the ramp ends. It reports an event that is shifted relative to the real one, and one that cannot be corrected without knowing τ — exactly the artifact in Figure 6.9.

6.10 Additional Resources

- Datasheets used in this lecture: ADXL335 (Analog Devices), CR3100 series current transformers (CR Magnetics), GL55 series photoresistors.
- Wikipedia entries on the Seebeck effect, piezoelectricity, and fiber Bragg gratings for the physics we only sketched.

6.11 References

- Fraden (2016) (Chs. 1–4)
- Morris and Langari (2015) (Chs. 1–2)

- Figliola and Beasley (2019) (Chs. 1–2)

References

Figliola, Richard S., and Donald E. Beasley. 2019. *Theory and Design for Mechanical Measurements*. 7th ed. Wiley.

Fraden, Jacob. 2016. *Handbook of Modern Sensors: Physics, Designs, and Applications*. 5th ed. Springer.

Morris, Alan S., and Reza Langari. 2015. *Measurement and Instrumentation: Theory and Application*. 2nd ed. Academic Press.