UNIVERSAL KNOWLEDGE LIBRARY

Book 00 - Canon Edition

A self-contained architecture for a book series that teaches everything about everything

Design objective: the series itself becomes the reference for people and AI.

This edition changes the mission of the library. External books, websites, or courses may still be cited for provenance or comparison, but they are no longer allowed to be required for comprehension. Every future volume in the series must teach from first principles, define its own vocabulary, derive its own core equations and algorithms, provide its own examples and exercises, and connect its own frontier chapters to the rest of the canon through internal cross-references.

Series principle: explain internally, verify externally, depend on nothing outside the series for learning.

Contents

- 1. What changes in the Canon Edition
- 2. What 'self-contained' means in this series
- 3. Non-negotiable rules for every volume
- 4. The architecture of a zero-to-frontier canon
- 5. The standard structure of a volume and a chapter
- 6. Writing rules for both people and AI
- 7. Proof, derivation, experiment, design, and frontier standards
- 8. Contributor ladders: how readers become builders and researchers
- 9. Governance, versioning, and correction policy
- 10. Initial map of the full canon
- 11. Two sample self-contained micro-lessons
- Appendix A. Internal citation and notation standard
- Appendix B. Machine-readable canon specification

1. What changes in the Canon Edition

The series is no longer treated as a curated path that sends readers outward. It is treated as the primary learning instrument. That means each book must be able to stand on its own while also fitting into a larger network of volumes. A reader should be able to start from zero, follow the internal prerequisite map, and keep going until they are capable of original work.

In practical terms, the series now has four obligations. First, it must teach fundamentals from first principles. Second, it must preserve rigor by distinguishing definitions, derivations, theorems, empirical rules, engineering heuristics, and open problems. Third, it must teach craft as well as theory: modeling, computation, experimentation, debugging, design tradeoffs, failure analysis, and communication. Fourth, it must be navigable not only by human readers but also by AI systems that need consistent terminology, chunkable sections, explicit assumptions, and machine-readable structure.

The result is a different kind of library: a canonical internal universe of knowledge. External material can still appear in notes or provenance sections, but no chapter is allowed to depend on an outside explanation in order to be understood.

2. What 'self-contained' means in this series

Principle	Meaning inside the series

No hidden prerequisites	Any concept used in a serious way must either be taught earlier in the same book or linked to a precise internal prerequisite section elsewhere in the series.
Definitions before use	Terminology, symbols, and units must be introduced explicitly before a derivation, proof, design workflow, or code sample depends on them.
Worked transitions	The books must not jump from statement to result without showing the intermediate reasoning, derivation, construction steps, or debugging path.
Theory plus practice	Every core theoretical chapter must connect to examples, exercises, experiments, implementations, and failure modes.
Stable internal references	The series must use a stable citation scheme so people and AI can point to the exact definition, theorem, circuit, algorithm, lab, or design rule.
Frontier honesty	A chapter must say what is established, what is approximate, what is controversial, and what remains open.

Self-contained does not mean isolated from history or evidence. It means explanatory independence. The reader should never have to leave the series to get the core understanding. If outside sources appear, they are there to document provenance, compare formulations, or invite broader reading, not to replace missing teaching.

3. Non-negotiable rules for every volume

Rule 1. Start from zero without talking down to the reader.

Each volume opens with a foundations part that assumes curiosity, not prior mastery. The tone should be respectful, ambitious, and exact.

Rule 2. Build a visible ladder from intuition to formalism to mastery.

Readers need intuition, then notation, then derivation, then example, then independent use. Chapters should make that ladder visible.

Rule 3. Label the status of knowledge.

Statements must be tagged in context as definition, theorem, derivation, law, approximation, empirical observation, engineering rule of thumb, implementation choice, or open problem.

Rule 4. Never hide the assumptions.

Every model, theorem, simulation, code example, and design calculation must name its domain of validity and its simplifying assumptions.

Rule 5. Teach error, failure, and misuse.

A learner becomes trustworthy by knowing not only what works but how it breaks: edge cases, numerical instability, false analogies, safety hazards, and measurement traps.

Rule 6. Include multiple representations.

Core ideas should be restated in words, equations, diagrams, tables, algorithms, and examples where appropriate, so different reasoning styles can meet the same concept.

Rule 7. Close the loop with exercises and solutions.

Every chapter needs exercises of increasing difficulty, plus selected solutions or solution outlines. The series must not ask for mastery without giving the reader a place to test it.

Rule 8. End at the frontier, not at the textbook boundary.

The last part of each volume must connect the reader to active questions, unresolved tensions, and concrete directions for contribution.

Editorial test: if a careful beginner reaches a chapter and asks 'where do I learn the missing step?', the answer should be an internal section number, not an external website.

4. The architecture of a zero-to-frontier canon

To teach everything about everything, the series has to behave like a living canon rather than a stack of disconnected books. The canon needs layers. Lower layers establish universal tools that recur across many fields. Middle layers specialize into the sciences, engineering, computation, life sciences, human systems, and creative domains. Upper layers synthesize disciplines and carry the reader toward research and institution building.

4.1 The eight layers

- **Layer A. Language, logic, and epistemic tools** Reading, writing, rhetoric, logic, set-based reasoning, probability of belief, evidence, argument structure, and how knowledge claims are justified.
- **Layer B. Mathematics and formal structure** Arithmetic through calculus, linear algebra, differential equations, probability, statistics, optimization, discrete mathematics, information theory, and numerical methods.

- **Layer C. The physical world** Mechanics, thermodynamics, electromagnetism, optics, quantum theory, statistical physics, chemistry, materials, Earth systems, and energy.
- **Layer D. Life and cognition** Biology, genetics, physiology, neuroscience, biotechnology, bioinformatics, medicine-aware systems thinking, and cognition.
- **Layer E. Computation and intelligence** Algorithms, data structures, systems, programming languages, software engineering, machine learning, LLMs, information retrieval, and quantum information.
- **Layer F. Built artifacts and engineered systems** Circuits, electronics, RF and microwave engineering, semiconductors, fabrication, embedded systems, control, mechanical systems, fluid systems, robotics, and vehicles.
- **Layer G. Human coordination systems** Economics, law, governance, organizations, ethics, education, history, communication, and design.
- **Layer H. Synthesis and frontier** Systems engineering, safety, research methods, metrology, grand challenges, open problems, and how new knowledge is responsibly created and maintained.

4.2 How the volumes should interlock

- Every volume names its prerequisite sections, not just prerequisite subjects.
- Every theorem-heavy chapter is paired with at least one modeling or implementation chapter.
- Every engineering volume inherits mathematics and physics definitions from stable internal sections, so notation stays consistent.
- Every frontier chapter links back to the exact classical ideas that limit or enable it.
- Every book includes a contributor ladder showing how to move from learner to builder to researcher to steward.

This architecture is what allows the library to grow without turning into a maze. New volumes can be added, but they must plug into the existing ladder of concepts and the stable internal reference system.

5. The standard structure of a volume and a chapter

A book that aims to be the source must not be organized like a loose survey. It needs a repeatable internal architecture so readers know how to enter, progress, review, and return.

5.1 Standard volume architecture

Part	Purpose	Typical contents
I	Foundations	Core vocabulary, units, notation, naive intuition, and the minimum mathematical or

		conceptual tools.
II	Core theory	Definitions, laws, derivations, theorems, conceptual structures, and representative models.
III	Practice	Worked examples, code, design workflows, labs, measurements, simulation habits, and debugging.
IV	Synthesis	Cross-domain patterns, historical development, tradeoffs, common misconceptions, and failure cases.
V	Frontier	Current tensions, unresolved questions, competing models, and routes to contribution.
VI	Reference	Glossary, symbol table, theorem index, data tables, formula sheets, problem sets, and solutions.

5.2 Standard chapter architecture

1. Opening map: what this chapter teaches, why it matters, what it depends on, and what later chapters will use from it.
2. Prerequisite capsule: a compressed restatement of required earlier concepts with internal references for deeper review.
3. Concept and notation block: symbols, units, conventions, and domain assumptions.
4. Main exposition: intuition, formal definition, derivation or construction, and conceptual checkpoints.
5. Worked examples: at least one hand-worked example and, where relevant, a computational or laboratory example.
6. Failure modes: common misconceptions, violated assumptions, numerical pathologies, or unsafe interpretations.
7. Exercises in tiers: recall, derivation, modeling, implementation, design, critique, and frontier extension.

8. Internal links forward and backward: exact section pointers to prerequisite refreshers and next uses.

6. Writing rules for both people and AI

A canonical series should be readable by a human in a chair and navigable by an AI in a retrieval system. That requires clear prose plus explicit structure. The goal is not to flatten the writing into metadata. The goal is to make the structure of knowledge visible.

Requirement	Why it helps people	Why it helps AI
Stable section IDs	Readers can return to exact places later.	Retrieval and citation become unambiguous.
Explicit assumptions	Scope and limits stay visible.	Models can be filtered by domain of validity.
Symbol tables	Less cognitive friction from notation changes.	Parsers can attach equations to named variables.
Short semantic blocks	Dense topics become easier to digest.	Chunking and retrieval improve without losing meaning.
Truth-status labels	Readers see what is known versus tentative.	AI can separate consensus from speculation.
Worked examples	Abstract ideas become usable.	Systems can ground concepts in input-output patterns.
Glossaries and aliases	Different phrasings resolve to one concept.	Synonym handling and search improve.

6.1 Required metadata for each section

- Section ID and title
- One-sentence claim about what the section accomplishes
- Prerequisites by internal section ID
- Definitions introduced
- Symbols and units introduced
- Assumptions and simplifications
- Primary equations, algorithms, or constructions
- Failure modes or warnings
- Exercises and solutions linkage
- Forward references to later use

This metadata should exist in a way that is visible enough for readers and structured enough for AI ingestion. It does not need to overwhelm the prose. It can live in compact chapter cards, margin boxes, or a machine-readable companion manifest.

7. Proof, derivation, experiment, design, and frontier standards

7.1 Proof and derivation

Formal disciplines need formal accountability. When a volume states a theorem, the theorem should be isolated clearly from its intuition and its proof. When a volume states an engineering formula, the formula should be connected to its derivation or at least to the governing principles and approximations that justify it.

- Definitions must be distinguished typographically from explanatory prose.
- Theorems, lemmas, propositions, and corollaries should have stable identifiers within the volume.
- Derivations should show the chain of transformations, not only the start and the final result.
- Approximate steps must be marked as approximate, with the reason for the approximation named.
- Dimensional consistency and unit checks should appear wherever they are natural.

7.2 Experiment and simulation

A self-contained canon should teach how knowledge is tested. Simulation is not a substitute for reality, and experiment is not magic. Each scientific and engineering volume should therefore include a chapter on measurement, uncertainty, calibration, artifacts, and the difference between a model result and a physical observation.

- Lab procedures should name apparatus, variables, noise sources, and expected signatures.
- Simulation chapters should name numerical methods, stability constraints, and common discretization errors.
- Data-analysis chapters should show the path from raw measurements to interpreted conclusions.
- Safety-sensitive domains should include explicit hazard notes and misuse warnings.

7.3 Design and implementation

Design chapters are where knowledge becomes action. They should teach tradeoff analysis, requirement flowdown, interfaces, constraint management, verification, and failure recovery. That is true whether the design target is a circuit, a biochemical workflow, a control system, a neural network, or a legal institution.

- Requirements must be made explicit before architecture selection.

- Tradeoffs should be shown as choices under constraints, not merely as opinions.
- Validation and verification should be separated: are we building the right thing, and are we building it correctly?
- Failure analysis should not be optional. A good design chapter teaches what breaks first.

7.4 Frontier chapters

Frontier element	What it must include	What it must avoid
State of the field	Current competing views, benchmark problems, bottlenecks, and why the area matters.	Marketing language that hides unresolved issues.
Open problems	Questions that are specific enough to investigate and bounded enough to act on.	Vague gestures that cannot guide work.
Research methods	What counts as evidence, progress, replication, and failure in this subfield.	Pretending all fields validate knowledge in the same way.
Contribution ladder	Concrete ways a newcomer can reproduce, extend, critique, or integrate results.	Mystifying research as inaccessible art.

8. Contributor ladders: how readers become builders and researchers

The purpose of the canon is not only to inform. It is to produce capability. Each volume should therefore end with a contributor ladder that tells the reader how to move from passive understanding to active contribution.

Stage	Reader can do	Typical outputs
1. Comprehender	State concepts accurately, explain symbols, and solve guided exercises.	Notes, summaries, checked calculations.
2. Practitioner	Solve standard problems independently and implement canonical workflows.	Working code, lab notebooks, circuit builds, simulations.

3. Analyst	Compare methods, critique assumptions, and diagnose failure modes.	Benchmark reports, model comparisons, debug writeups.
4. Builder	Compose multiple ideas into a new design or synthesis.	Prototype systems, integrated essays, designs, experiments.
5. Researcher	Extend, test, or challenge the state of the art responsibly.	Replications, novel methods, ablation studies, formal results.
6. Steward	Teach, curate, refine notation, and improve the canon itself.	Text revisions, errata, new chapters, pedagogical tools.

9. Governance, versioning, and correction policy

A canon that aims to teach everything cannot remain static. It needs a disciplined update process so that revisions improve the library without breaking its internal references or lowering its standards.

- Every volume carries an edition number and revision date.
- Errata are recorded against stable section IDs, not vague page references alone.
- Major notation changes require migration notes so older citations remain intelligible.
- Frontier chapters are reviewed more frequently than settled foundational chapters.
- Safety-relevant and ethics-relevant corrections are prioritized.
- Exercises, datasets, code fragments, and lab procedures are versioned with the surrounding text.

The rule is continuity without drift. The series may evolve, but it must remain citable across time. That is especially important if both people and AI systems will use it as a canonical reference.

10. Initial map of the full canon

Below is a first-pass map of the canon. It is intentionally broad. The point is not to claim that the map is final. The point is to make the scope explicit so the library can grow coherently instead of by accumulation.

ID	Volume title	Domain	One-line scope
00	Canon Edition	Series Constitution	Defines the mission, structure, citation rules, editorial law, and expansion logic of the whole library.
01	Knowledge, Logic, and Learning	Foundations	Logic, argument, evidence, reasoning, cognition of learning, note-making,

			reading, and how knowledge is built.
02	Mathematics I	Foundations	Arithmetic, algebra, functions, graphs, and mathematical habits of mind.
03	Mathematics II	Foundations	Geometry, trigonometry, limits, differential and integral calculus.
04	Mathematics III	Foundations	Linear algebra, vector spaces, matrices, eigenanalysis, and geometric transformations.
05	Mathematics IV	Foundations	Differential equations, complex variables, transforms, and asymptotic thinking.
06	Mathematics V	Foundations	Probability, statistics, stochastic processes, and inference.
07	Mathematics VI	Foundations	Optimization, information theory, discrete mathematics, logic, and computability.
08	Scientific Computing and Modeling	Foundations	Numerical methods, simulation patterns, error analysis, and computational experiments.
09	Physics I	Physical World	Mechanics, conservation laws, oscillations, waves, and thermodynamics.
10	Physics II	Physical World	Electricity, magnetism, Maxwell structure, optics, and electromagnetic waves.
11	Physics III	Physical World	Special relativity, quantum mechanics, measurement, and operator methods.
12	Physics IV	Physical World	Statistical physics, condensed matter, plasma, and many-body viewpoints.
13	Chemistry I	Physical World	Atomic structure, bonding, stoichiometry,

			thermochemistry, and equilibria.
14	Chemistry II	Physical World	Organic chemistry, reaction mechanisms, spectroscopy, and synthesis logic.
15	Chemistry III	Physical World	Physical chemistry, kinetics, transport, electrochemistry, and interfaces.
16	Earth and Planetary Systems	Physical World	Geology, atmosphere, climate, oceans, planetary dynamics, and cycles.
17	Astronomy and Cosmology	Physical World	Stars, galaxies, gravitation at scale, cosmological models, and observational methods.
18	Energy Systems	Physical World	Power generation, storage, distribution, conversion, and energy economics.
19	Biology I	Life	Cells, molecular biology, genetics, inheritance, and evolution.
20	Biology II	Life	Physiology, development, ecology, and multiscale life processes.
21	Biochemistry and Chemical Biology	Life	Proteins, enzymes, metabolism, signaling, and molecular intervention.
22	Biotechnology and Synthetic Biology	Life	Bioprocesses, gene editing concepts, pathway engineering, and design of living systems.
23	Bioinformatics and Computational Biology	Life	Sequence analysis, structural biology basics, omics data, and biological modeling.
24	Neuroscience and Cognitive Systems	Life	Neurons, circuits, perception, memory, decision-making, and embodied cognition.
25	Health, Medicine, and Public Health Systems	Life	Medical reasoning basics, epidemiology, population health, safety, and clinical

			system logic.
26	Computer Science I	Computation	Programming, algorithms, data structures, and computational problem solving.
27	Computer Science II	Computation	Operating systems, networks, databases, and distributed systems.
28	Software Engineering	Computation	Requirements, architecture, testing, versioning, maintenance, and team practice.
29	Programming Languages and Toolchains	Computation	Python, C, C++, C#, compilers, build systems, and language design ideas.
30	Data, Retrieval, and Knowledge Systems	Computation	Databases, search, embeddings, retrieval, RAG, and knowledge organization.
31	AI I: Machine Learning	Computation	Supervised learning, unsupervised learning, optimization, generalization, and evaluation.
32	AI II: Deep Learning and Foundation Models	Computation	Neural networks, transformers, LLMs, multimodality, alignment, and tooling.
33	Quantum Information and Quantum Computing	Computation	Qubits, circuits, algorithms, error correction concepts, and physical implementations.
34	Human-Computer Interaction	Computation	Interfaces, usability, cognition-aware design, and interactive systems.
35	Circuits and Electronics	Hardware	Charge, current, voltage, passive networks, active devices, analog and digital electronics.
36	Measurement and Instrumentation	Hardware	Sensors, signal chains, metrology, calibration, and data acquisition.
37	RF, Microwave, and	Hardware	Transmission lines,

	Antennas		scattering, matching, waveguides, antennas, and EMC.
38	Semiconductors and Devices	Hardware	Band theory, PN structures, transistors, device physics, and process constraints.
39	Fabrication, Lithography, and Packaging	Hardware	Wafer processes, lithography, interconnects, thermal issues, and packaging.
40	Digital Design, HDL, FPGA, and VLSI	Hardware	Logic design, timing, Verilog/VHDL concepts, FPGA workflows, ASIC thinking, and verification.
41	Embedded Systems and Microcontrollers	Hardware	Real-time systems, buses, peripherals, firmware, interrupts, and low-power design.
42	Photonics, Lasers, and Optical Systems	Hardware	Optics in devices, laser principles, photonic components, and system design.
43	Statics, Dynamics, and Solid Mechanics	Engineering	Forces, motion, stress, strain, vibration, and structural response.
44	Fluid Mechanics and Transport	Engineering	Continuum viewpoint, conservation equations, laminar and turbulent flow, and transport.
45	Aerodynamics and CFD	Engineering	Airfoils, compressibility, boundary layers, stability, and computational fluid dynamics.
46	Heat Transfer and Thermal Systems	Engineering	Conduction, convection, radiation, exchangers, and thermal design.
47	Chemical Engineering Operations	Engineering	Reactors, separations, process control, scale-up, and plant thinking.
48	Propulsion, Combustion, and MHD	Engineering	Engines, nozzles, propellants, plasmas, magnetohydrodynamics, and advanced propulsion

			ideas.
49	Mechanical Design and Manufacturing	Engineering	Mechanisms, tolerances, materials selection, processes, and manufacturability.
50	Control Systems	Engineering	Feedback, stability, estimation, observers, and control design.
51	Robotics	Engineering	Kinematics, planning, actuation, sensing, autonomy, and robot system integration.
52	Vehicles, Drones, and Autonomous Systems	Engineering	Aerial, ground, and marine platforms, guidance, navigation, and mission architectures.
53	Materials Science and Engineering	Engineering	Structure-property-processing relationships across metals, ceramics, polymers, and composites.
54	Civil, Structural, and Infrastructure Systems	Engineering	Structures, water systems, transportation, resilience, and built environments.
55	Economics and Decision Systems	Human Systems	Micro, macro, incentives, markets, institutions, and formal decision analysis.
56	Law, Governance, and Institutions	Human Systems	Legal reasoning, constitutions, regulation, administrative systems, and governance design.
57	History of Civilizations and Technology	Human Systems	Historical processes, comparative civilizations, industrialization, and technological change.
58	Philosophy, Ethics, and Epistemology	Human Systems	Metaphysics, ethics, philosophy of science, logic of inquiry, and value conflict.
59	Language, Writing, and Communication	Human Systems	Grammar-aware clarity, rhetoric, technical writing, argument, and translation of ideas.

60	Education and Learning Science	Human Systems	Pedagogy, curriculum design, memory, transfer, and assessment.
61	Art, Design, and Creative Systems	Human Systems	Visual design, music, narrative, aesthetics, and creative process.
62	Business, Operations, and Entrepreneurship	Human Systems	Operations, strategy, supply chains, finance basics, productization, and organizational growth.
63	Systems Engineering, Safety, and Research Practice	Synthesis	Integration, verification, hazard analysis, experimental design, measurement culture, and open problems.

11. Two sample self-contained micro-lessons

The complaint that triggered this edition was simple: a true canon should teach directly, not send the reader away. The two micro-lessons below are tiny demonstrations of the style change. They are not full chapters. They show the pattern each full chapter should follow.

11.1 Micro-lesson A: the derivative from first principles

Question. How do we describe the instantaneous rate at which a quantity changes?

Start with a function y = f(x). Over a finite interval from x to x + h, the average rate of change is the slope of a secant line:

```
(f(x + h) - f(x)) / h
```

If h is large, this is only an average. To capture the instantaneous rate, we ask what happens as h becomes arbitrarily small. If the limit exists, the derivative is defined by:

```
f'(x) = lim_{h -> 0} (f(x + h) - f(x)) / h
```

Worked example. Let f(x) = x^2. Then:

- `f(x + h) = (x + h)^2 = x^2 + 2xh + h^2`
- `f(x + h) - f(x) = 2xh + h^2`
- `(f(x + h) - f(x)) / h = 2x + h`
- `lim_{h -> 0} (2x + h) = 2x`

So the derivative of x^2 is 2x. Interpretation matters: if x is time and f(x) is position, then the derivative has units of position per time. A derivative is not only a symbol-pushing trick. It is a physically and geometrically meaningful rate.

- Assumptions: the limit exists at the point of interest.

- Common failure mode: mistaking average rate of change over an interval for the instantaneous rate at a point.
- Internal links a full chapter would include: limits, continuity, tangent interpretation, chain rule, optimization, differential equations.

11.2 Micro-lesson B: the RC discharge circuit

Question. How does voltage across a capacitor change when the capacitor discharges through a resistor?

Definitions. For a capacitor, charge q and voltage v are related by q = C v. Current is the rate of charge flow, so i = dq/dt = C dv/dt. For a resistor, Ohm's law gives v_R = iR. In a simple resistor-capacitor loop during discharge, Kirchhoff's voltage law says the resistor voltage plus the capacitor voltage sums to zero around the loop.

```
R i + v = 0
```

Substitute i = C dv/dt:

```
R C dv/dt + v = 0
```

Rearrange:

```
dv/dt = -(1 / (R C)) v
```

This differential equation has the solution:

```
v(t) = V_0 e^{-t / (R C)}
```

The product tau = R C is called the time constant. After one time constant, the voltage has fallen to about 36.8 percent of its initial value. Again, interpretation matters. The equation is telling us that the rate of decrease is proportional to the remaining voltage.

- Assumptions: ideal resistor, ideal capacitor, lumped-element model, no external source during discharge.
- Common failure mode: forgetting sign conventions or mixing capacitor current direction with passive sign convention.
- Internal links a full chapter would include: Kirchhoff laws, first-order differential equations, transient response, frequency domain behavior, measurement with oscilloscopes.

12. What the next rewrite must do

The existing library package can now be judged against this canon standard. Any volume that mainly surveys or points outward should be rewritten so that its own chapters carry the teaching burden internally. The first rewrites should be the foundation volumes, because all later books depend on them: knowledge and logic, mathematics, scientific computing, physics, chemistry, programming, and circuits.

- Rewrite the opening chapters of each technical volume to include complete prerequisite capsules.
- Replace 'resources' sections with internal continuation maps and optional provenance notes.
- Add explicit exercise ladders and selected solutions.
- Introduce theorem, law, derivation, algorithm, and warning labels consistently across volumes.
- Generate machine-readable manifests for every chapter so AI systems can cite the series precisely.

Appendix A. Internal citation and notation standard

A canon becomes usable at scale only when its internal references are stable. The following scheme is recommended:

- Volume references use Vxx, where xx is a two-digit or two-character volume ID. Example: V35.
- Chapter references use Vxx.Cy. Example: V35.C4.
- Section references use Vxx.Cy.Sz. Example: V35.C4.S2.
- Definitions use D, theorems use T, laws use L, algorithms use A, figures use F, tables use TB, examples use X, labs use LB, and problems use P.
- A complete citation might look like V35.C4.S2.D3 or V03.C6.S1.T2.

Notation rules. Symbol meaning should be stable whenever possible across the series. If a volume reuses a common symbol in a domain-specific way, it should say so explicitly in the symbol table.

Label	Meaning	Example
D	Definition	V11.C2.S1.D1 = definition of a Hilbert space
T	Theorem	V03.C5.S4.T2 = fundamental theorem statement
L	Law/Principle	V09.C3.S2.L1 = conservation law
A	Algorithm	V31.C7.S3.A1 = gradient descent procedure
X	Worked example	V35.C2.S5.X3 = RC transient worked example
P	Problem	V07.C8.S2.P12 = optimization exercise 12

Appendix B. Machine-readable canon specification

For AI use, every volume should also expose a compact structured manifest. The companion JSON file delivered with this edition is a starter version. It records the series title, editorial laws, section metadata requirements, truth-status tags, and the initial volume map.

- Humans should be able to read the prose without seeing raw JSON.
- AI systems should be able to ingest the JSON without scraping page layout.
- The prose and the JSON should describe the same structure, so the series remains coherent across reading modes.