# Rust Mastery Syllabus for Optimization, Heuristics, Simulation, and ML

A 28-Week Mission Plan

August 16, 2025

## How to Use This Syllabus

This roadmap is structured for a baseline of **6–8 hours/week** (aggressive: **10–12 hours/week**). Each week includes concrete deliverables with ☐ checkboxes. A Gantt-style overview visually maps the 28-week plan.

### Primary Learning Objectives

- Become fluent in idiomatic Rust (ownership, borrowing, traits, lifetimes).
- Build numerical computing skills (linear algebra, statistics, numerical methods).
- Implement optimization heuristics and mathematical programming models.
- Create discrete-event and agent-based simulations with stochastic inputs.
- Integrate machine learning (classical and deep learning) into optimization/simulation pipelines.
- Scale with parallelism, async, and HPC techniques; profile and optimize.

## Weekly Syllabus with Deliverables

| Week | Topics | Activities  Deliverables (check when complete) |
|---|---|---|
| 1 | Intro to Rust; Ownership & Borrowing | ☐ Read Chapters 1–4 of *The Rust Programming Language* (The Book).<br>☐ Install Rustup, Rust toolchain, and verify `cargo`.<br>☐ Exercism: Hello World, Variables, Ownership (Rust Track).<br>☐ Mini: CLI echo tool with file input. |
| 2 | Structs, Enums, Pattern Matching | ☐ Read Ch. 5–6 of The Book.<br>☐ Build a `Vector2D` with AddMul ops and tests.<br>☐ Rust By Example: Structs  Enums (RBE). |

| Week | Topics | Activities Deliverables (check when complete) |
|---|---|---|
| 3 | Traits, Generics, Error Handling | ☐ Read Ch. 8–10 (collections, generics, traits).<br>☐ Implement CLI calculator via traits; robust `Result`, `Option`, and `?`.<br>☐ Add unit tests (`cargo test`). |
| 4 | Iterators, Modules, Crates | ☐ Read Ch. 13–15.<br>☐ CSV parser using `csv` crate; parse to records.<br>☐ Add integration tests and benchmarking scaffold (`criterion`). |
| 5 | ndarray Basics | ☐ Read ndarray docs (arrays, slicing, broadcasting).<br>☐ Implement dot product and matrix multiply; compare naive vs. ndarray.<br>☐ Export results to CSV. |
| 6 | nalgebra & ndarray-linalg | ☐ Solve $Ax = b$ by Gaussian elimination.<br>☐ Compute eigenvalues & eigenvectors; validate on symmetric matrix.<br>☐ Compare ndarray vs. nalgebra ergonomics and speed. |
| 7 | Numerical Methods | ☐ Newton–Raphson root finding with adaptive tolerance.<br>☐ Monte Carlo $\pi$ with `rand`; CI on estimate.<br>☐ Document numeric stability concerns. |
| 8 | Statistics | ☐ Use `statrs` for Normal, Poisson, and sampling.<br>☐ Bootstrap resampling for mean CI.<br>☐ Produce summary plots (CSV + external plotting). |
| 9 | Simulated Annealing & Hill Climbing | ☐ Implement SA & HC for multimodal functions.<br>☐ Experiment with cooling schedules; log convergence.<br>☐ Compare to baseline random search. |
| 10 | Genetic Algorithms (TSP) | ☐ GA for TSP (10–20 nodes): selection, crossover, mutation.<br>☐ Parallelize fitness where possible.<br>☐ Export tour length over generations. |
| 11 | LPMILP with `good_lp` | ☐ Model a facility location or diet problem.<br>☐ Evaluate solver backends & sensitivity analysis. |

| Week | Topics | Activities Deliverables (check when complete) |
|---|---|---|
| | | ☐ Validate with small integer instances. |
| 12 | `argmin` for Continuous Opt. | ☐ Optimize Rosenbrock and Rastrigin.<br>☐ Add simple bound constraints; stopping criteria.<br>☐ Benchmark vs. your SA implementation. |
| 13 | RNG Deep Dive | ☐ Custom seeded RNGs and distributions with `rand`.<br>☐ Reproducibility harness for simulations.<br>☐ Validate streams with simple tests. |
| 14 | Queueing (MM1) | ☐ Discrete-event sim of MM1; collect wait times.<br>☐ Compare empirical to theoretical $L$, $W$, $L_q$, $W_q$.<br>☐ Sensitivity to arrivalservice rates. |
| 15 | Agent-Based Epidemic | ☐ Agent model with SIR dynamics; stochastic transitions.<br>☐ Record time series; export CSV for plots.<br>☐ Evaluate $R_0$ scenarios. |
| 16 | Domain Simulation (Maintenance) | ☐ Airline maintenance scheduling sim with stochastic failures.<br>☐ Resource constraints and downtime metrics.<br>☐ Produce readiness KPIs. |
| 17 | ML with `linfa` | ☐ K-means clustering; evaluate inertia & silhouette.<br>☐ Logistic regression for classification.<br>☐ Serialize models. |
| 18 | ML with `smartcore` | ☐ Decision trees & random forests; cross-validation.<br>☐ Feature importance; calibration.<br>☐ Integrate predictions into a toy optimizer. |
| 19 | Deep Learning with `tch-rs` | ☐ Load PyTorch model; run inference from Rust.<br>☐ Measure latency & throughput.<br>☐ Validate outputs against Python. |
| 20 | Pipeline Integration | ☐ Demand forecasting $\rightarrow$ *optimizationpolicy*.<br>☐ Closed-loop sim with ML-in-the-loop.<br>☐ Document architecture. |
| 21 | Parallel Iterators (rayon) | ☐ Parallelize GA population evaluation.<br>☐ Verify determinism vs. speed tradeoffs. |

| Week | Topics | Activities Deliverables (check when complete) |
|---|---|---|
| | | ☐ Benchmark scaling. |
| 22 | Async (tokio) | ☐ Async task runner for events; backpressure.<br>☐ Logging and metrics.<br>☐ Fault injection tests. |
| 23 | MPI (rsmpi) | ☐ Distribute Monte Carlo across nodes.<br>☐ Gatherreduce results.<br>☐ Compare cluster vs. single-node. |
| 24 | Profiling & Optimization | ☐ `cargo-flamegraph` hot paths.<br>☐ Remove allocations in hot loops; `[inline]` when helpful.<br>☐ Record beforeafter benchmarks. |
| 25 | Capstone Planning | ☐ Problem definition and KPIs.<br>☐ Architecture diagram & tech stack.<br>☐ Data model and validation plan. |
| 26 | Core Implementation | ☐ Build optimization + simulation engine core.<br>☐ Minimal CLI or API.<br>☐ Unit and integration tests. |
| 27 | ML Integration | ☐ Add prediction module; adaptive heuristics.<br>☐ Scenario experiments and ablations.<br>☐ Performance tuning. |
| 28 | Docs & Release | ☐ Write README, user guide, examples.<br>☐ Produce benchmark report.<br>☐ Publish repo and release tag. |

## Crates and Tools (with Links)

- Rust Standard Docs: std
- The Book: doc.rust-lang.org/book
- Rust by Example: rust-by-example
- Exercism (Rust): exercism.org/tracks/rust
- ndarray: crates.io/crates/ndarray
- nalgebra: nalgebra.org
- ndarray-linalg: crates.io/crates/ndarray-linalg
- statrs: docs.rs/statrs
- rand: crates.io/crates/rand
- good_lp: crates.io/crates/good_lp
- argmin: crates.io/crates/argmin
- linfa: github.com/rust-ml/linfa
- smartcore: crates.io/crates/smartcore
- tch-rs (PyTorch): github.com/LaurentMazare/tch-rs
- burn (DL framework): github.com/burn-rs/burn

- rayon: crates.io/crates/rayon
- tokio: tokio.rs
- rsmpi (MPI): crates.io/crates/rsmpi
- criterion (bench): crates.io/crates/criterion
- cargo-flamegraph: crates.io/crates/flamegraph
- This Week in Rust: this-week-in-rust.org
- Rust Users Forum: users.rust-lang.org

## Recommended Books (Rust, OR, Heuristics, Simulation, ML)

### Rust

- **Programming Rust** (2nd ed.), Jim Blandy, Jason Orendorff, and Leonora F. S. Tindall. (O'Reilly)
- **Rust for Rustaceans**, Jon Gjengset. (No Starch Press)
- **Rust Atomics and Locks**, Mara Bos. (Manning) — for concurrency and performance.

### Optimization and Heuristics

- **Numerical Optimization**, Jorge Nocedal and Stephen Wright. (Springer)
- **Integer and Combinatorial Optimization**, Nemhauser and Wolsey. (Wiley)
- **Handbook of Metaheuristics**, Gendreau and Potvin (eds.). (Springer)

### Simulation

- **Simulation Modeling and Analysis** (5th ed.), Averill M. Law. (McGraw-Hill)
- **Discrete-Event System Simulation**, Banks, Carson, Nelson, Nicol. (Pearson)

### Machine Learning

- **Pattern Recognition and Machine Learning**, Christopher M. Bishop. (Springer)
- **The Elements of Statistical Learning**, Hastie, Tibshirani, Friedman. (Springer)
- **Deep Learning**, Goodfellow, Bengio, Courville. (MIT Press)
- **Reinforcement Learning: An Introduction** (2nd ed.), Sutton and Barto. (MIT Press)

## Weekly Time Guidance

- **Baseline:** 6–8 hrs/week (steady, retention-focused).
- **Aggressive:** 10–12 hrs/week (accelerated, add stretch tasks).
- **Immersive:** 15+ hrs/week (deep dive; add extra projects, open-source contributions).

## Notes and Extensions

- Replace or augment any week with domain-specific tasks (e.g., Navy UnRep scheduling, airline maintenance) without breaking the overall arc.
- Maintain a running `CHANGELOG.md` and `BENCHMARKS.md` in your repo for professional rigor.
- Consider adding CI (GitHub Actions) for `cargo fmt`, `cargo clippy`, `cargo test` gates by Week 6.

*Fair winds and following seas. Now go make the borrow checker your wingman.*