MATLAB & SIMULINK

Model-Based Development

Detailed training syllabus

From system requirements to software requirements, model development, test design and verification.

180 guided hours + 40-60 hours of independent assignments

Suggested delivery: 18 weeks, 10 guided hours per week. Teaching balance: approximately 40% explanation and demonstrations, 60% practical work.

Designed for engineering students and early-career engineers

Electrical, Electronics, Mechanical, Mechatronics, Instrumentation, Computer Science, Civil, Chemical, Aerospace and related branches.

Course approach

A shared foundation teaches the development and verification process. Branch-specific plant models and capstone projects provide domain relevance. A water-tank pump controller serves as the common end-to-end teaching project.

Version 1.0 | 10 October 2026

Proposed independent training curriculum. MAAB/MAB is taught as a modeling-guideline framework; this document does not claim MathWorks accreditation.

01 | Outcomes and course structure

Entry requirements

Basic algebra, engineering units and introductory programming. Offer preparatory exercises in programming, differential equations and control concepts when needed.

- Write MATLAB scripts for engineering analysis and test automation.
- Develop plant, controller and Stateflow models with defined interfaces.
- Derive measurable software requirements from system requirements.
- Apply MAAB/MAB guidelines and review applicable deviations.
- Build requirement-based tests, execute assessments and interpret coverage.
- Maintain traceability, investigate defects and deliver verification evidence.

Module	Subject	Hours
1	Engineering foundations and lifecycle	6
2	MATLAB fundamentals	14
3	Engineering analysis and automation	10
4	Simulink fundamentals	16
5	Dynamic systems and plant modeling	12
6	Controller and software architecture	12
7	Stateflow and supervisory logic	12
8	System to software requirements	14
9	MAAB/MAB guidelines and reviews	10
10	Test design and harness development	14
11	Automated testing and results	12
12	Structural and boundary coverage	18
13	Code generation and execution modes	10
14	Configuration management and quality	6
15	Domain capstone	14
	TOTAL	180

Detailed modules | 1-2

Module 1: Engineering foundations and lifecycle | 6 hours

- MATLAB, Simulink and Stateflow: purpose and relationships.
- Model-Based Design and Model-Based Development; plant versus controller software.
- Continuous and discrete behavior; inputs, outputs, states, parameters and units.
- Verification versus validation; V-model and development review gates.
- MIL, SIL, PIL and HIL overview; lifecycle artifacts and traceability.

Practical: Identify sensors, actuators, plant and controller for a motor, tank or thermal system.

Deliverable: System context diagram and development workflow.

Module 2: MATLAB fundamentals | 14 hours

- Environment, scripts, functions, variables and workspace.
- Vectors, matrices, indexing and arithmetic operations.
- Relational and logical operators; conditionals and loops.
- Data types, conversions, structures, tables and cell arrays.
- Function scope; data import/export; plotting and debugging.
- Readable code, naming, documentation and input validation.

Practical: Analyze sensor data, calculate engineering quantities and write a reusable validated function.

Deliverable: Documented MATLAB script with plots and results.

Detailed modules | 3-4

Module 3: Engineering analysis and automation | 10 hours

- Numerical differentiation, integration and introductory ODE simulation.
- Interpolation, lookup data, filtering and preprocessing.
- Parameter estimation fundamentals and measured-versus-simulated comparison.
- Error metrics, numerical tolerances and parameter sensitivity.
- Parameter sweeps, scripted pass/fail checks and reproducible data processing.

Practical: Simulate a first-order system, sweep parameters and compare responses with reference data.

Deliverable: Reusable analysis script and parameter-sweep report.

Module 4: Simulink fundamentals | 16 hours

- Model creation, simulation settings, sources, sinks and logging.
- Gain, Sum, Product, Integrator, Transfer Function and State-Space blocks.
- Unit Delay, Memory and discrete-time behavior.
- Relational, logical and Switch blocks; saturation and lookup tables.
- Hierarchy; enabled/triggered subsystems; buses and signal routing.
- Sample times, solvers, initialization, parameters and diagnostics.

Practical: Build continuous and discrete models, implement threshold logic and diagnose a sample-time defect.

Deliverable: Structured model with logged signals and simulation results.

Detailed modules | 5-6

Module 5: Dynamic systems and plant modeling | 12 hours

- Physical laws; mass, energy and charge balances.
- Differential equations, initial conditions and parameter units.
- Transfer-function and state-space representations.
- Linear/nonlinear behavior, disturbances, noise and actuator limits.
- Assumptions, validity range, sensitivity and plant validation.
- Optional introduction to Simscape physical modeling.

Practical: Choose a DC motor, RLC circuit, mass-spring-damper, water tank, thermal process or attitude model.

Deliverable: Plant model, parameter table and assumptions register.

Module 6: Controller and software architecture | 12 hours

- Feedback/feedforward control and P, PI, PID fundamentals.
- Rise time, overshoot, settling time and steady-state error.
- Discrete implementation, saturation and anti-windup.
- Signal conditioning, command limiting, reset and initialization.
- Separate plant, controller and supervisory logic.
- Model references, interfaces, data dictionaries, enumerations and calibration.
- Fixed-point concepts and numerical limitations.

Practical: Develop a controller with command limits, reset behavior and disturbance rejection.

Deliverable: Controller architecture, interface definition and performance results.

Detailed modules | 7-8

Module 7: Stateflow and supervisory logic | 12 hours

- States, default transitions and entry/during/exit actions.
- Transition conditions, priorities and deterministic execution.
- Hierarchical/parallel states; events and data.
- Temporal logic, counters, timers, debouncing and hysteresis.
- Operating modes, arbitration, fault latching and recovery.
- Simulink integration; ambiguous transitions and unintended behavior.

Practical: Implement OFF, READY, RUN and FAULT modes with timeout and recovery rules.

Deliverable: Stateflow chart, transition table and scenario tests.

Module 8: System to software requirements | 14 hours

- Stakeholder needs, scenarios, boundaries and assumptions.
- Functional, performance, interface, timing and fault-response requirements.
- Allocation to plant, hardware and software.
- Atomic, unambiguous, feasible and verifiable statements.
- IDs, rationale, source, acceptance criteria and requirement reviews.
- Software ranges, units, types, update rates, startup and invalid-input behavior.
- Requirement-model-test traceability and change-impact analysis.
- Requirements Toolbox authoring and linking when available.

Practical: Rewrite vague statements and derive software requirements for the shared tank controller.

Deliverable: System/software specifications, interface specification and traceability matrix.

Detailed modules | 9-10

Module 9: MAAB/MAB guidelines and model reviews | 10 hours

- Guideline purpose, selected version, applicability and project tailoring.
- Naming, layout, signal labeling, routing and subsystem organization.
- Block usage, modeling patterns, data types and initialization.
- Stateflow readability and MATLAB Function practices.
- Manual review and Model Advisor checks.
- Deviation rationale and review records; compliance versus functional correctness.

Practical: Correct a deliberately poor model and document justified deviations.

Deliverable: Model-review checklist and guideline-compliance report.

Module 10: Test design and harness development | 14 hours

- Verification planning; component, integration and system tests.
- Requirement-based testing, equivalence partitioning and boundary analysis.
- Positive, negative, robustness, decision-table and transition tests.
- Startup, reset, faults, timeout and persistence scenarios.
- Harness architecture, stimuli, initial states and parameter settings.
- Independent expected results, numerical/timing tolerances and pass/fail rules.
- Test case review and requirement links.

Practical: Create tests and a harness before completing controller implementation.

Deliverable: Test specification and executable harness.

Detailed modules | 11-12

Module 11: Automated testing and results analysis | 12 hours

- Test Manager and simulation, baseline and equivalence tests.
- Test Sequence, assessments, parameter overrides and iterations.
- MATLAB-driven execution, output comparison and regression testing.
- Diagnose requirement, model and test defects.
- Defect logging, correction, retest and reporting.

Practical: Inject an incorrect comparator, delayed fault response and wrong reset state; detect and fix them.

Deliverable: Automated suite, results report and defect log.

Module 12: Structural and relational boundary coverage | 18 hours

- Execution, decision, condition and modified condition/decision coverage.
- Atomic conditions, decision outcomes and MC/DC independence pairs.
- AND/OR expressions, masking/unique-cause concepts and short-circuit behavior.
- Coupled conditions, infeasible combinations and Stateflow reports.
- Integer, fixed-point and floating-point relational boundaries.
- Absolute/relative tolerances and operator-specific objectives.
- Cumulative coverage, gaps, dead logic and justified exclusions.
- Requirement coverage versus model coverage versus code coverage.

Practical: Build decision/condition examples with MC/DC gaps; add independence pairs and boundary tests.

Deliverable: Coverage reports, gap analysis, added tests and exclusion rationale.

Detailed modules | 13-14

Module 13: Code generation and execution modes | 10 hours

- Code-generation-compatible modeling, interfaces and generated-code structure.
- Data types, parameters and execution scheduling.
- MIL and SIL workflows; normal-mode versus SIL equivalence.
- PIL concepts and HIL architecture.
- Numerical differences, tolerances and model versus code coverage.

Practical: Generate a small controller and compare normal simulation with SIL where supported.

Deliverable: Code-generation report and equivalence results. PIL/HIL may be demonstrations.

Module 14: Configuration management and quality | 6 hours

- Reproducible setup, project structure and version control.
- Model comparison; requirement, model and test baselines.
- Change requests, reviews and impact analysis.
- Regression automation and continuous-integration concepts.
- Release notes, unresolved issues and final evidence package.

Practical: Change one requirement, update affected artifacts and execute regression.

Deliverable: Reviewed change package and regression report.

Detailed modules | 15-15

Module 15: Domain capstone | 14 hours

- Select a branch-relevant system and document assumptions.
- Complete requirements, architecture, plant and controller models.
- Apply guidelines and conduct a peer/instructor review.
- Develop and execute tests; analyze coverage and defects.
- Present traceability, results and remaining limitations.

Practical: Complete the project over several weeks with additional independent effort.

Deliverable: Complete project evidence package and technical presentation.

Coverage workshop | Worked examples

Decision and condition coverage

For D = A AND B, tests TT and FF exercise both decision outcomes and both outcomes of each condition. They do not demonstrate the independent influence of either condition.

MC/DC independence pairs

Test	A	B	D = A AND B
T1	True	True	True
T2	False	True	False
T3	True	False	False

T1/T2 demonstrates A independently changing the outcome. T1/T3 demonstrates B independently changing it. Inspect supported objectives and the selected MC/DC definition in the tool report.

Relational boundary exercise

Integer input	Temperature > 80	Purpose
79	False	Just below threshold
80	False	Equal to threshold
81	True	Just above threshold

Integer boundary tolerance is 1. Fixed-point tolerance depends on representable precision. Floating-point tests use the configured absolute and relative tolerance; inspect operator-specific intervals. Equality is not a separate required floating-point objective in the same way as integer/fixed-point equality.

Coverage closure checklist

- Start with tests derived from requirements and independently defined expected results.
- Collect and accumulate coverage; investigate every uncovered objective.
- Add reachable scenarios, repair defects or explain infeasible objectives.
- Review exclusions, repeat regression after changes and retain evidence.
- High coverage demonstrates exercised logic; it does not establish correctness by itself.

Shared project | Requirements to evidence

Water-tank pump controller

System goal: maintain level between 30% and 80% under specified supply/discharge conditions. Physical feasibility depends on pump capacity, tank size and disturbances.

ID	Illustrative software requirement
SWR-001	At each 100 ms execution step, in AUTO mode with a valid sensor, set pump ON when level is at or below 30%.
SWR-002	Under the same conditions, set pump OFF when level is at or above 80%.
SWR-003	Between 30% and 80%, retain the previous pump command.
SWR-004	On entry to AUTO mode, initialize the retained command to OFF.
SWR-005	For an invalid sensor, command pump OFF within one execution step and activate the sensor-fault indication.

Development sequence

Needs and scenarios -> system requirements -> allocation -> software/interface requirements -> plant/controller models -> guideline review -> test design -> execution -> coverage analysis -> defect correction -> regression -> evidence package.

Test specification template

Field	Content
Test ID and links	Unique ID; requirement(s) being verified
Preconditions	Initial state, operating mode and parameters
Stimulus	Input values, transitions and timing
Expected result	Output values, states and response deadlines
Acceptance criteria	Explicit pass/fail rules and tolerances
Evidence	Results, plots, defect references and observations

Domain projects and tool setup

Branch	Suggested capstone
Electrical	DC motor speed controller with protection
Electronics / ECE	Sensor acquisition and alarm controller
Mechanical	Cruise control or thermal control
Mechatronics	Conveyor sorting with interlocks and timeouts
Instrumentation	Process monitoring, calibration and alarms
CSE / IT	Embedded supervisory controller and test automation
Civil	Water-tank pumping and sensor-fault management
Chemical	Mixing-tank temperature controller
Aerospace	Simplified attitude or actuator controller
Biomedical	Simulated equipment-temperature controller

Software products

Activity	Relevant products
Core development	MATLAB, Simulink and Stateflow
Managed requirements	Requirements Toolbox
Automated MAB checks	Simulink Check
Harnesses and managed tests	Simulink Test
Coverage reporting	Simulink Coverage
Optional advanced analysis	Simulink Design Verifier
Code generation	Simulink Coder / Embedded Coder, according to workflow
Optional physical/fixed-point work	Simscape products / Fixed-Point Designer

Declare license requirements before enrollment. Without testing/coverage products, students can practice manual test design and analysis but cannot complete equivalent tool-generated reporting. Confirm release and toolchain compatibility before code-generation labs.

Assessment, delivery and references

Assessment	Weight
MATLAB and Simulink practicals	20%
Requirements and architecture	15%
Stateflow and guideline review	15%
Testing and coverage analysis	25%
Capstone and technical presentation	25%

Final submission

- System/software requirements, interfaces and plant assumptions.

- Simulink/Stateflow models and modeling-guideline review.

- Test specification, executable suite and test results.

- Requirement-model-test traceability and coverage-gap analysis.

- Defect log, reviewed exclusions and final report.

Completion standard

For small teaching controllers, aim for 100% of applicable, feasible coverage objectives, with reviewed rationale for remaining objectives. Grade requirement quality, expected results and engineering explanations alongside percentages.

Progressive course packaging

Foundation: Modules 1-5, 58 hours. Development: Modules 6-9, 48 hours. Verification and project: Modules 10-15, 74 hours. Capstone work continues outside guided sessions.

Official technical references

MAB Modeling Guidelines
https://www.mathworks.com/help/simulink/mab-modeling-guidelines.html

MAB Model Advisor checks
https://www.mathworks.com/help/slcheck/ref/using-model-advisor-checks-for-mab-modeling-guidelines.html

Types of model coverage and relational boundaries
https://www.mathworks.com/help/slcoverage/ug/types-of-model-coverage.html

Structural coverage terminology
https://www.mathworks.com/help/slcoverage/gs/coverage-terminology.html

Model objects that receive coverage
https://www.mathworks.com/help/slcoverage/ug/model-objects-that-receive-coverage.html

Get started with Simulink Test
https://www.mathworks.com/help/sltest/getting-started-with-simulink-test.html