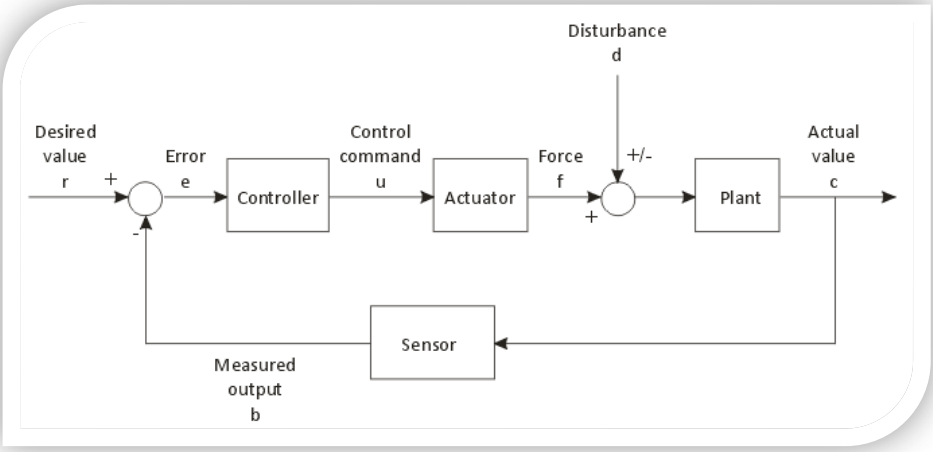
Experiment One

# Introduction to Control Systems Design


**Control Systems Laboratory**

**Dr. Zaer Abo Hammour**

## 1.1 Control System Design

The design of control systems is a specific example of engineering design. The goal of control engineering design is to obtain the configuration, specifications, and identification of the key parameters of a proposed system to meet an actual need.

The control system design process is illustrated in Figure 1.1. The design process consists of seven main building blocks, which we arrange into three groups:

1. Establishment of goals and variables to be controlled, and definition of specifications (metrics) against which to measure performance
1. System definition and modeling
2. Control system design and integrated system simulation and analysis

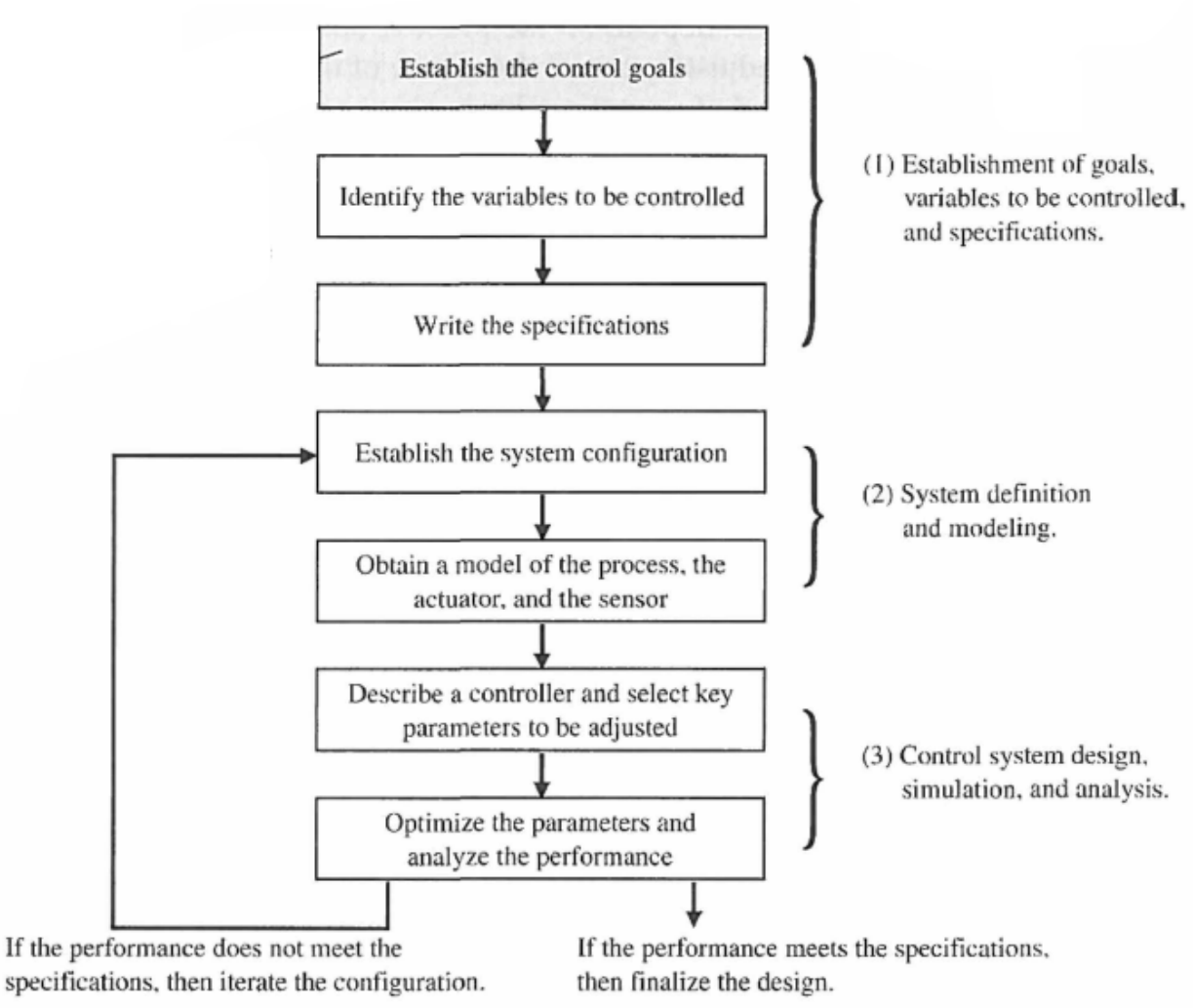



**Figure 1.1**: The Control System Design Process

## 1. Establishing The System Goals

For example, we may state that our goal is to control the velocity of a motor accurately. The second step is to identify the variables that we desire to control (for example, the velocity of the motor). The third step is to write the specifications in terms of the accuracy we must attain. This required accuracy of control will then lead to the identification of a sensor to measure the controlled variable. The performance specifications will describe how the closed-loop system should perform and will include (1) good regulation against disturbances, (2) desirable responses to commands, (3) realistic actuator signals, (4) low sensitivities, and (5) robustness.

## 2. System Definition and Modeling

As designers, we proceed to the first attempt to configure a system that will result in the desired control performance. This system configuration will normally consist of a sensor, the process under control, an actuator, and a controller, as shown in Figure 1.2. The next step consists of identifying a candidate for the actuator. This will, of course, depend on the process, but the actuation chosen must be capable of effectively adjusting the performance of the process. For example, if we wish to control the speed of a rotating flywheel, we will select a motor as the actuator. The sensor, in this case, must be capable of accurately measuring the speed. We then obtain a model for each of these elements.

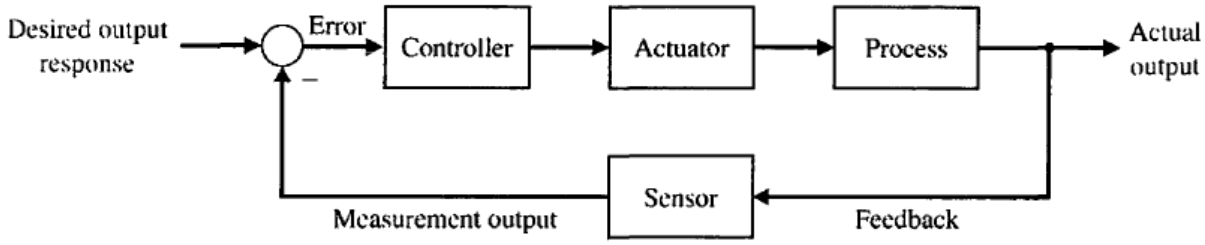



**Figure 1.2**: Closed Loop feedback Control System

## 3. Control System Design, Simulation, and Analysis

The next step is the selection of a controller, which often consists of a summing amplifier that will compare the desired response and the actual response and then forward this error-measurement signal to an amplifier. The final step in the design process is the adjustment of the parameters of the system to achieve the desired performance. If we can achieve the desired performance by adjusting the parameters, we will finalize the design and proceed to document the results. If not, we will need to establish an improved system configuration and perhaps select an enhanced actuator and sensor. Then we will repeat the design steps until we are able to meet the specifications, or until we decide the specifications are too demanding and should be relaxed.

In summary, the controller design problem is as follows: Given a model of the system to be controlled (including its sensors and actuators) and a set of design goals, find a suitable controller, or determine that none exists. As with most of engineering design, the design of a feedback control system is an iterative and nonlinear process. A successful designer must consider the underlying physics of the plant under control, the control design strategy, the controller design architecture (that is, what type of controller will be employed), and effective controller tuning strategies. In addition, once the design is completed, the controller is often implemented in hardware, and hence issues of interfacing with

hardware can appear. When taken together, these different phases of control system design make the task of designing and implementing a control system quite challenging.

## 1.2 Mechatronics Systems

A natural stage in the evolutionary process of modern engineering design is encompassed in the area known as **Mechatronics.** The term mechatronics was coined in Japan in the 1970s. Mechatronics is the synergistic integration of mechanical, electrical, and computer systems and has evolved over the past 30 years, leading to a new breed of intelligent products. Feedback control is an integral aspect of modern mechatronic systems. One can understand the extent that mechatronics reaches into various disciplines by considering the components that make up mechatronics. The key elements of mechatronics are (1) physical systems modeling, (2) sensors and actuators, (3) signals and systems, (4) computers and logic systems, and (5) software and data acquisition. Feedback control encompasses of all five key elements of mechatronics, but is associated primarily with the element of signals and systems, as illustrated in Figure 1.3.

Advances in computer hardware and software technology coupled with the desire to increase the performance-to-cost ratio has revolutionized engineering design. New products are being developed at the intersection of traditional disciplines of engineering, computer science, and the natural sciences. Advancements in traditional disciplines are fueling the growth of mechatronics systems by providing "enabling technologies." A critical enabling technology was the microprocessor which has had a profound effect on the design of consumer products. We should expect continued advancements in cost-effective microprocessors and microcontrollers, novel sensors and actuators enabled by advancements in applications of microelectromechanical systems (MEMS), advanced control methodologies and real-time programming methods, networking and wireless technologies, and mature computer-aided engineering (CAE) technologies for advanced system modeling, virtual prototyping, and testing. The continued rapid development in these areas will only accelerate the pace of smart (that is, actively controlled) products.

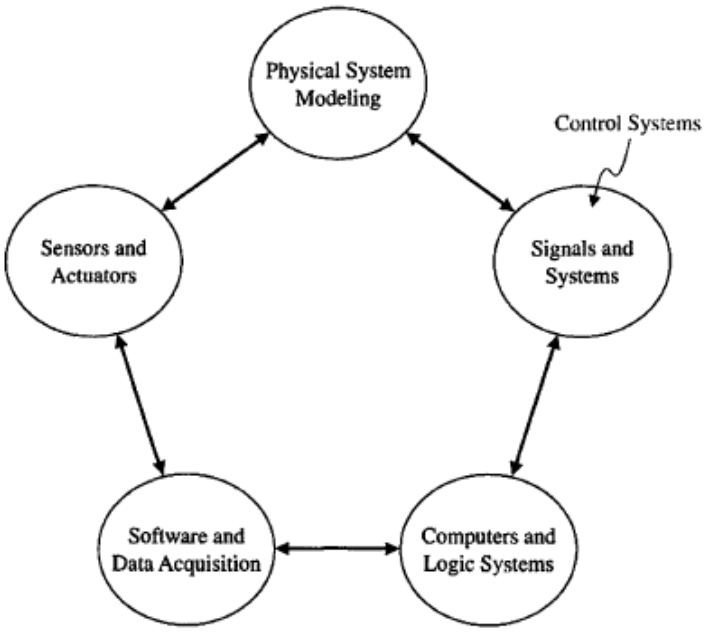



**Figure 1.3**: The Key Elements of Mechatronics

An exciting area of future mechatronic system development in which control systems will play a significant role is the area of alternative energy production and consumption. Hybrid fuel automobiles and efficient wind power generation are two of systems that can benefit from mechatronic design methods. In fact, the mechatronic design philosophy can be effectively illustrated by the example of the evolution of the modern automobile. Before the 1960s, the radio was the only significant electronic device in an automobile. Today, many automobiles have 30-60, up to 100 electric motors, about 200 pounds of wiring, a multitude of sensors, and thousands of lines of software code. A modern automobile can no longer be classified as a strictly mechanical machine—it has been transformed into a comprehensive mechatronic system.

## 1.3 Control System Design MATLAB – SIMULINK

MathWorks tools for control system design support each stage of the development process, from plant modeling to deployment through automatic code generation. Their widespread adoption among control engineers around the world comes from the flexibility of the tools to accommodate different types of control problems. If your control problem is unique, you can create a custom tool or algorithm using MATLAB®.

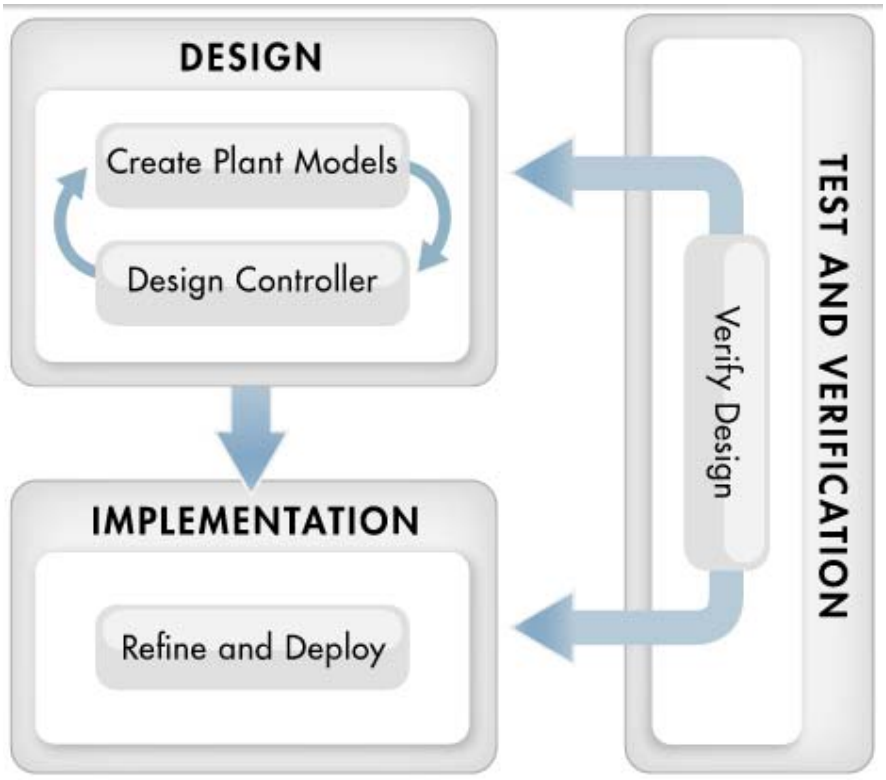



**Figure 1.4** :Control System in MATLAB

## 1. Creating Accurate Plant Models

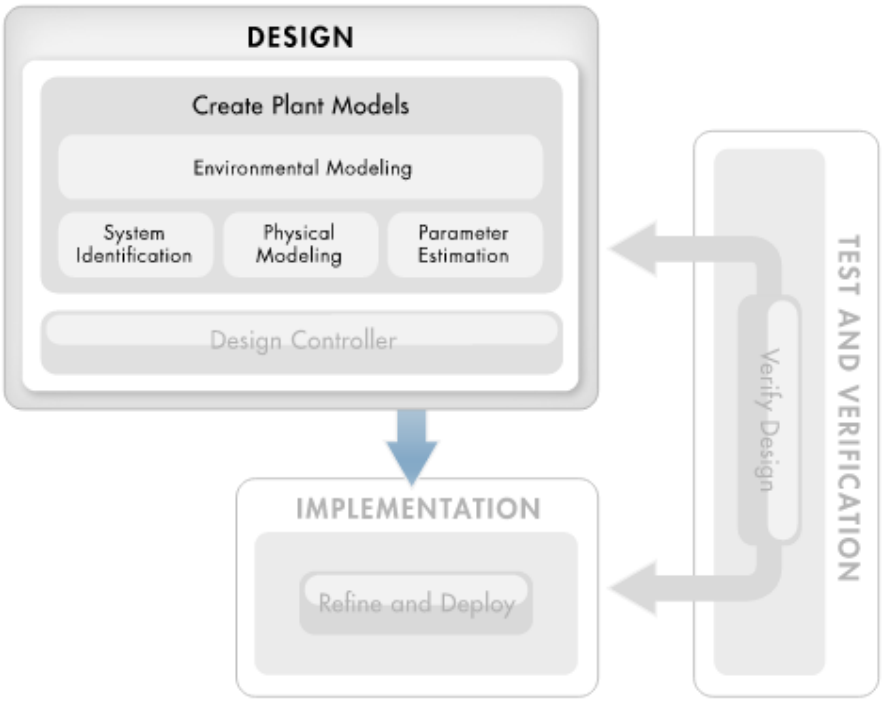



**Figure 1.5:** Plant Models

Control system design starts with an accurate plant model. You can describe the complex dynamics of your plant using a variety of modeling approaches, all supported by MathWorks tools. Because these tools work together in the Simulink® modeling environment, you can use the most appropriate modeling approach for each component in your plant to create the system-level plant model.

- **Create Models from Test Data**

When you don't know the physics of your system, you can employ system identification, a common data-driven modeling technique that uses input-output data to create a plant model. This approach lets you rapidly develop accurate plant models without knowing the detailed structure of the model.

- **Develop Complex Multidomain Plant Models Using Physical Modeling**

MathWorks physical modeling tools let you construct plant models by using blocks that represent mechanical, electrical, magnetic, hydraulic, pneumatic, and thermal components to map the component topography and physical connections of your system. With this approach, you can efficiently create complex multidomain plant models without having to derive the underlying first-principles equations. If you know the plant equations explicitly, you can implement them by using Simulink blocks or express them in the SimScape™ language. First-principles models—those built with physical modeling tools or by implementing equations explicitly—can be developed without access to physical prototypes or plant hardware.

- **Improve Model Accuracy Through Parameter Estimation**

First-principles models contain parameters that correspond to physical properties of the plant, such as mass, electrical resistance, and flow area, that may not be known. Simulink Design Optimization uses experimental data to improve model accuracy by calibrating model parameters with input-output test data.

- **Build Environmental Effects into Your Plant Models**

Model environmental effects, such as atmosphere, gravity, wind turbulence, and the Earth's magnetic fields, with Simulink and Aerospace Blockset™. You can create custom environmental models from Simulink blocks, such as lookup tables

## 2. Design Feedback Compensators and Control Logic

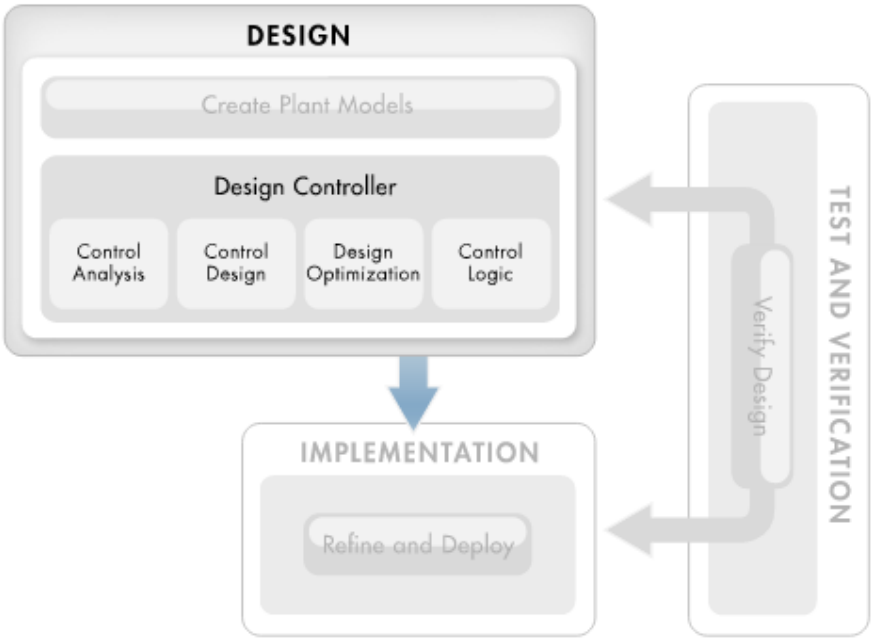



**Figure 1.6**:Design Feedback Compensators and Control Logic

Once an accurate plant model has been created, control engineers analyze and develop closed-loop compensators and open-loop supervisory strategies using a range of MathWorks tools.

- **Analyze Control Systems**

Perform a critical analysis of your control system within MATLAB or Simulink through the following tasks:

1) Assess key performance parameters, such as overshoot, rise time, and stability
2) Trim, linearize, and compute the frequency response of nonlinear Simulink models
3) Model and analyze the effects of uncertainty on the performance and stability of your models

- **Systematically Apply a Control Design Technique**

Simulink Control Design™ and Robust Control Toolbox™ eliminate the need for trial and error by providing systematic methods for tuning single-loop and multiloop control systems entirely within Simulink. You can:

1) Automatically tune PID controllers
2) Apply linear control design techniques using interactive root locus, Bode, and Nichols diagrams
3) Automatically tune decentralized multivariable controllers
4) Leverage advanced control strategies, such as model predictive control and fuzzy logic

All of these control techniques can be easily evaluated and verified in MATLAB and Simulink.

- **Optimize System Performance**

Improve system performance and reduce system cost by automatically tuning design parameters in your Simulink model with Simulink Design Optimization™. You can optimize controller gains to meet rise-time and overshoot constraints, or jointly optimize physical and algorithmic parameters to maximize overall system performance.

- **Design and Simulate Supervisory Logic**

Stateflow® enables you to model, design, and simulate the supervisory logic in your control system, which schedules the operation of the controller, controls the operational mode of the system, and performs fault detection, isolation, and recovery (FDIR).

## 3. Deploying Designs to Embedded Controllers Through Automatic Code Generation

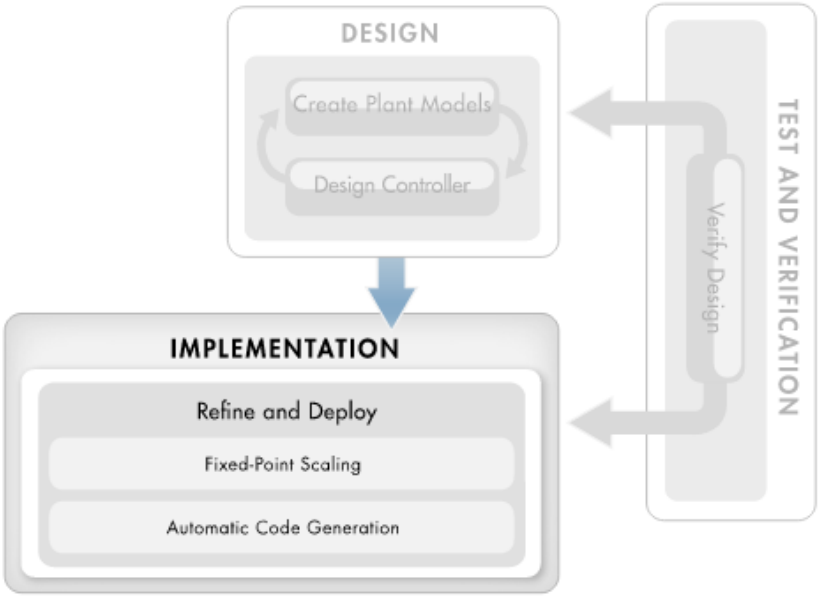



**Figure 1.7:** Implementation of Controller

After the control system has been designed and tested, you refine it for implementation. For example, you can specify all the fixed-point data type properties of your design to prepare it for implementation with fixed-point arithmetic.

- **Use Automatic Code Generation to Deploy Your Design**

You can deploy your design onto an embedded controller through automatic code generation. With this approach you generate highly efficient code for your controller and avoid errors that can happen when your design must be reinterpreted during manual coding. You can generate:

1) ANSI/ISO C and C++ code for targeting microprocessors and microcontrollers
2) IEC 61131 structured text for targeting programmable logic controllers (PLCs) and programmable automation controllers (PACs)
3) Verilog and VHDL code for targeting FPGAs for applications such as AC motor control.

## 4. Verifying Control Design Through Simulation and Formal Verification Methods

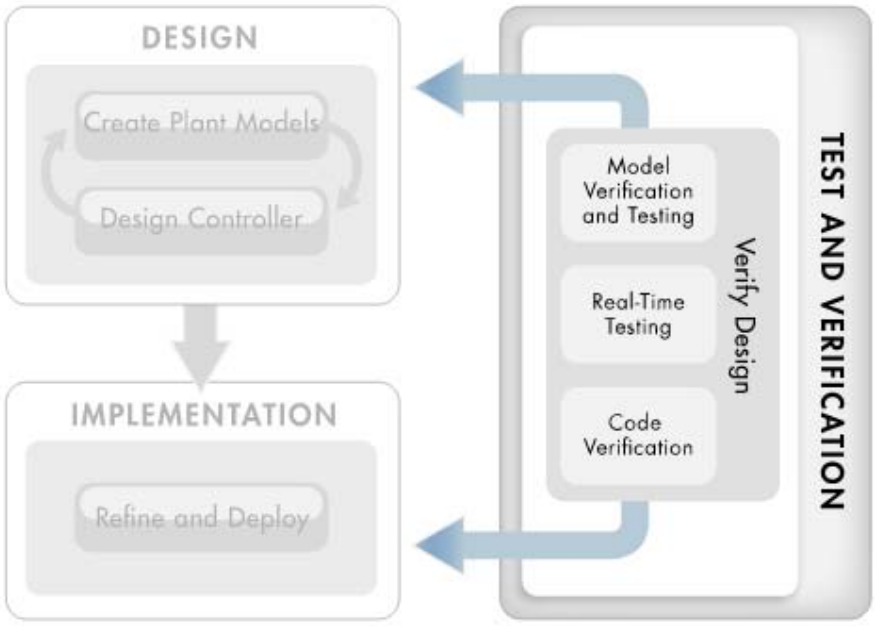



**Figure 1.8:** Test and Verification

MathWorks provides tools for comprehensive testing of your control systems, using desktop simulation, real-time testing, and formal verification methods.

- **Test Control Systems Using Desktop Simulation**

Test control algorithms through desktop simulation to find design errors before building hardware and creating production software. Such early-verification testing reduces the need for expensive and difficult-to-access prototypes. To perform early-verification testing, you combine the plant model and control algorithm into one model and simulate the closed-loop system behavior. For this simulation, utilize the plant model used for control design or develop one with higher fidelity.

One option for creating a higher fidelity plant model is to use a third-party product for your specific application or industry from the MathWorks Connections Program.

- **Test Control Algorithms in Real Time**

If a real-time simulation or physical prototype of the system is available, you can perform real-time testing of your control algorithm by:

1) Implementing it on a hardware target and connecting the target to a prototype (often called rapid prototyping)
2) Simulating it in real time through hardware-in-the-loop (HIL) testing

You can reuse the plant model that you developed for compensator design for HIL testing.

- **Test Using Formal Verification Methods**

MathWorks provides additional tools for verifying, validating, and testing your design that enable you to:

1) Run test cases through your controller and check model coverage, including Modified condition/decision coverage MC/DC coverage
2) Use formal verification methods to automatically create test cases that provide 100% coverage of your control logic and prove specific model properties
3) Check the real-time code for certain run-time errors and trace the origins of errors in the model

- **Real Time Testing**

**1) Rapid Control Prototyping**

Is a process which lets the engineer quickly test and iterate their control strategies on a real-time computer with real input/output devices. Rapid control prototyping figure 1.9 differs from HIL in that the control strategy is simulated in real-time and the "plant," or system under control, is real. Rapid control prototyping is now the typical method used by engineers to develop and test their control strategies. Rapid control prototyping was first used for developing powertrain control strategies. The simple reason is that the control software, which is in the engine and transmission control units, is difficult and time-consuming to modify. It has since been adopted industry wide in applications such as anti-lock braking, anti-roll, vehicle stability, active cruise control, and torque distribution.

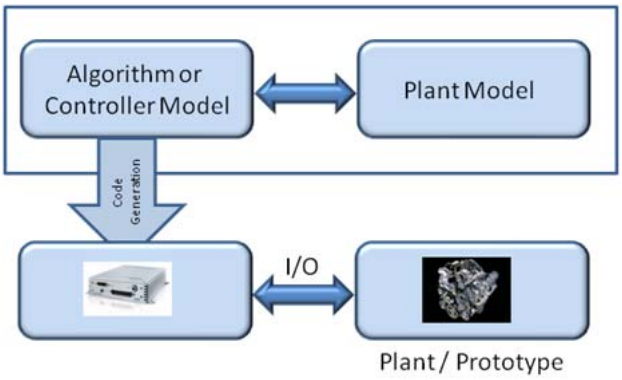



**Figure 1.9:** Rapid Control Prototyping

### 2) Hardware in the Loop

Hardware-In-the-Loop (HIL) Figure 1.10 is a technique for combining a mathematical simulation model of a system with actual physical hardware, such that the hardware performs as though it were integrated into the real system. For testing and development of embedded electronic controllers, the hardware controller and associated software are connected to a mathematical simulation of the system plant, which is executed on a computer in real time. To connect the real time model to the Hardware Controller, the real time computer receives electrical signals from the controller as actuator commands to drive the plant, and converts these signals into the physical variables connected to the plant model. The plant model calculates the physical variables that represent the outputs of the plant, which are converted into electrical signals that represent the voltages produced by the sensors that feed the controller.

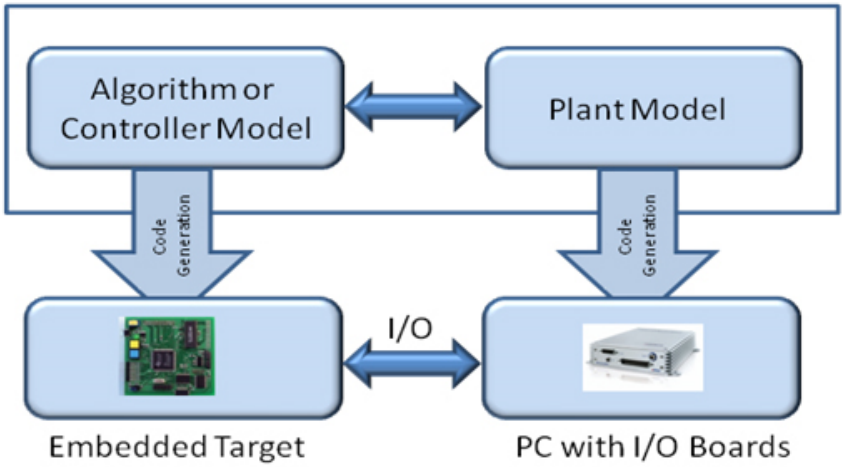



**Figure 1.10**: Hardware in the Loop

Real-time simulation and testing enables you to:

1) Refine and verify the functional operation of control system designs with your hardware
2) Continuously explore and test new ideas using a flexible, scalable platform
3) Test control system hardware even when a physical plant or system is unavailable
4) Investigate scenarios and hardware interactions that are complex, expensive, or dangerous to perform with production hardware
5) Avoid costly design flaws by detecting errors early when they are still cost-effective to correct

## 1.4 Model-Based Design:

### 1 What Is Model-Based Design?

Model-Based Design is a process that enables faster, more cost-effective development of dynamic systems, including control systems, signal processing, and communications systems. In Model-Based Design, a system model is at the center of the development process, from requirements development, through design, implementation, and testing. The model is an executable specification that you continually refine throughout the development process. After model development, simulation shows whether the model works correctly.

When software and hardware implementation requirements are included, such as fixed-point and timing behavior, you can automatically generate code for embedded deployment and create test benches for system verification, saving time and avoiding the introduction of manually coded errors.

Model-Based Design allows you to improve efficiency by:

- Using a common design environment across project teams
- Linking designs directly to requirements
- Integrating testing with design to continuously identify and correct errors
- Refining algorithms through multi-domain simulation
- Automatically generating embedded software code
- Developing and reusing test suites
- Automatically generating documentation
- Reusing designs to deploy systems across multiple processors and hardware targets

### 2 Model-Based Design Process

There are six steps to modeling any system:

1. Defining the System
2. Identifying System Components
3. Modeling the System with Equations
4. Building the Simulink® Block Diagram
5. Running the Simulation
6. Validating the Simulation Results

You perform the first three steps of this process outside of the Simulink software environment before you begin building your model.

**1. Defining the System**

The first step in modeling a dynamic system is to fully define the system. If you are modeling a large system that can be broken into parts, you should model each subcomponent on its own. Then, after building each component, you can integrate them into a complete model of the system.

The most effective way to build a model of this system is to consider each of these subsystems independently.

### 2. Identifying System Components

The second step in the modeling process is to identify the system components. Three types of components define a system:

- **Parameters** : System values that remain constant unless you change them
- **States** :Variables in the system that change over time
- **Signals:** Input and output values that change dynamically during a simulation

In Simulink, parameters and states are represented by blocks, while signals are represented by the lines that connect blocks. For each subsystem that you identified, ask yourself the following questions:

- How many input signals does the subsystem have?
- How many output signals does the subsystem have?
- How many states (variables) does the subsystem have?
- What are the parameters (constants) in the subsystem?
- Are there any intermediate (internal) signals in the subsystem?

Once you have answered these questions, you should have a comprehensive list of system components, and you are ready to begin modeling the system.

### 3. Modeling the System with Equations

The third step in modeling a system is to formulate the mathematical equations that describe the system. For each subsystem, use the list of system components that you identified to describe the system mathematically.

Your model may include:

- Algebraic equations
- Logical equations
- Differential equations, for continuous systems
- Difference equations, for discrete systems

You use these equations to create the block diagram in Simulink.

### 4. Building the Simulink Block Diagram

After you have defined the mathematical equations that describe each subsystem, you can begin building a block diagram of your model in Simulink.

Build the block diagram for each of your subcomponents separately. After you have modeled each subcomponent, you can then integrate them into a complete model of the system.

### 5. Running the Simulation

After you build the Simulink block diagram, you can simulate the model and analyze the results.

Simulink allows you to interactively define system inputs, simulate the model, and observe changes in behavior. This allows you to quickly evaluate your model.

### 6. Validating the Simulation Results

Finally, you must validate that your model accurately represents the physical characteristics of the dynamic system.